Atakowanie zarządzania sesją

Mechanizm zarządzania sesją jest podstawowym elementem bezpieczeństwa w większości aplikacji internetowych. To właśnie umożliwia aplikacji unikalną identyfikację danego użytkownika w ramach wielu różnych żądań i obsługę gromadzonych danych na temat stanu interakcji tego użytkownika z aplikacją. Tam, gdzie aplikacja implementuje funkcję logowania, zarządzanie sesją ma szczególne znaczenie, ponieważ to ona umożliwia aplikacji utrzymanie pewności co do tożsamości dowolnego użytkownika poza żądaniem, w którym dostarcza on swoje dane uwierzytelniające. Ze względu na kluczową rolę, jaką odgrywają mechanizmy zarządzania sesją, są one głównym celem złośliwych ataków na aplikację. Jeśli atakujący może złamać zarządzanie sesją aplikacji, może skutecznie ominąć kontrole uwierzytelniania i udawać innych użytkowników aplikacji bez znajomości ich danych uwierzytelniających. Jeśli atakujący narazi w ten sposób użytkownika administracyjnego, może przejąć całą aplikację. Podobnie jak w przypadku mechanizmów uwierzytelniania, w funkcjach zarządzania sesją często można znaleźć wiele różnych defektów. W najbardziej wrażliwych przypadkach atakujący musi po prostu zwiększyć wartość tokena wydanego mu przez aplikację, aby przełączyć jego kontekst na kontekst innego użytkownika. W tej sytuacji aplikacja jest szeroko otwarta dla każdego, aby uzyskać dostęp do wszystkich obszarów. Na drugim końcu spektrum atakujący może być zmuszony do bardzo ciężkiej pracy, rozszyfrowania kilku warstw zaciemniania i obmyślenia wyrafinowanego zautomatyzowanego ataku, zanim znajdzie lukę w zbroi aplikacji. W tej części przyjrzymy się wszystkim typom słabości, jakie autorzy napotkali w rzeczywistych aplikacjach internetowych. Szczegółowo określa praktyczne kroki, które należy podjąć, aby znaleźć i wykorzystać te wady. Na koniec opisuje środki obronne, które aplikacje powinny podjąć, aby chronić się przed tymi atakami.

POWSZECHNY MIT

„Używamy kart inteligentnych do uwierzytelniania, a bez nich nie można naruszyć sesji użytkowników".

Bez względu na to, jak solidny jest mechanizm uwierzytelniania aplikacji, kolejne żądania użytkowników są powiązane z tym uwierzytelnieniem tylko za pośrednictwem wynikowej sesji. Jeśli zarządzanie sesją aplikacji jest wadliwe, osoba atakująca może ominąć solidne uwierzytelnianie i nadal narażać użytkowników.

Potrzeba stanu

Protokół HTTP jest zasadniczo bezstanowy. Opiera się na prostym modelu żądanie-odpowiedź, w którym każda para komunikatów reprezentuje niezależną transakcję. Sam protokół nie zawiera mechanizmu łączenia serii żądań wysyłanych przez konkretnego użytkownika i odróżniania ich od wszystkich innych żądań otrzymanych przez serwer WWW. We wczesnych latach Internetu żaden taki mechanizm nie był potrzebny: strony internetowe były używane do publikowania statycznych stron HTML, które każdy mógł przeglądać. Dziś sprawy mają się zupełnie inaczej. Większość „witryn" internetowych to w rzeczywistości aplikacje internetowe. Umożliwiają rejestrację i logowanie. Pozwalają kupować i sprzedawać towary. Zapamiętują Twoje preferencje przy następnej wizycie. Zapewniają bogate wrażenia multimedialne z zawartością tworzoną dynamicznie na podstawie tego, co klikasz i wpisujesz. Aby zaimplementować którąkolwiek z tych funkcji, aplikacje internetowe muszą korzystać z koncepcji sesji. Najbardziej oczywiste zastosowanie sesji występuje w aplikacjach obsługujących logowanie. Po wprowadzeniu nazwy użytkownika i hasła możesz korzystać z aplikacji jako użytkownik, którego dane uwierzytelniające podałeś, aż do wylogowania lub wygaśnięcia sesji z powodu braku aktywności. Bez sesji użytkownik musiałby na każdej stronie aplikacji ponownie wpisywać swoje hasło. Stąd po jednokrotnym uwierzytelnieniu użytkownika aplikacja tworzy dla niego sesję i traktuje wszystkie żądania należące do tej sesji jako pochodzące od tego użytkownika. Aplikacje, które nie mają funkcji logowania, zwykle również muszą korzystać z sesji. Wiele witryn sprzedających

towary nie wymaga od klientów tworzenia kont. Pozwalają jednak użytkownikom przeglądać katalog, dodawać pozycje do koszyka, podawać szczegóły dostawy oraz dokonywać płatności. W tym scenariuszu nie ma potrzeby uwierzytelniania tożsamości użytkownika: przez większość jego wizyty aplikacja nie wie i nie dba o to, kim jest użytkownik. Jednak aby robić z nim interesy, musi wiedzieć, które serie otrzymywanych żądań pochodzą od tego samego użytkownika.

Najprostszym i wciąż najpopularniejszym sposobem realizacji sesji jest nadanie każdemu użytkownikowi unikalnego tokena lub identyfikatora sesji. Przy każdym kolejnym żądaniu do aplikacji użytkownik ponownie przesyła ten token, umożliwiając aplikacji określenie, której sekwencji wcześniejszych żądań dotyczy bieżące żądanie. W większości przypadków aplikacje używają plików cookie HTTP jako mechanizmu transmisji do przekazywania tych tokenów sesji między serwerem a klientem. Pierwsza odpowiedź serwera do nowego klienta zawiera nagłówek HTTP podobny do następującego:

Set-Cookie: ASP.NET_SessionId=mza2ji454s04cwbgwb2ttj55

i kolejne żądania od klienta zawierają ten nagłówek:

Cookie: ASP.NET_SessionId=mza2ji454s04cwbgwb2ttj55

Ten standardowy mechanizm zarządzania sesjami jest z natury podatny na różne kategorie ataków. Głównym celem atakującego przy atakowaniu mechanizmu jest przejęcie w jakiś sposób sesji legalnego użytkownika, a tym samym podszywanie się pod tę osobę. Jeżeli użytkownik został uwierzytelniony w aplikacji, atakujący może uzyskać dostęp do prywatnych danych należących do użytkownika lub wykonać nieautoryzowane działania w jego imieniu. Jeśli użytkownik nie jest uwierzytelniony, osoba atakująca może nadal przeglądać poufne informacje przesłane przez użytkownika podczas jego sesji. Podobnie jak w poprzednim przykładzie serwera Microsoft IIS z uruchomionym środowiskiem ASP.NET, większość komercyjnych serwerów WWW i platform aplikacji internetowych wdrażają własne gotowe rozwiązania do zarządzania sesją oparte na plikach cookie HTTP. Udostępniają one interfejsy API, za pomocą których programiści aplikacji internetowych mogą integrować z tym rozwiązaniem własne funkcje zależne od sesji. Stwierdzono, że niektóre gotowe implementacje zarządzania sesjami są podatne na różne ataki, co skutkuje naruszeniem bezpieczeństwa sesji użytkowników (zostanie to omówione w dalszej części tego rozdziału). Ponadto niektórzy programiści uznają, że potrzebują bardziej precyzyjnej kontroli nad zachowaniem sesji, niż zapewniają im wbudowane rozwiązania, lub chcą uniknąć pewnych luk właściwych dla rozwiązań opartych na plikach cookie. Z tych powodów dość często spotyka się niestandardowe i/lub nieoparte na plikach cookie mechanizmy zarządzania sesją w aplikacjach o krytycznym znaczeniu dla bezpieczeństwa, takich jak bankowość internetowa. Luki występujące w mechanizmach zarządzania sesjami można podzielić na dwie kategorie:

*Słabości w generowaniu tokenów sesji

* Niedociągnięcia w obsłudze tokenów sesji przez cały cykl ich życia

Przyjrzymy się kolejno każdemu z tych obszarów, opisując różne rodzaje defektów, które są powszechnie spotykane w rzeczywistych mechanizmach zarządzania sesjami, oraz praktyczne techniki ich wykrywania i wykorzystywania. Na koniec opiszemy środki, które aplikacje mogą podjąć, aby bronić się przed tymi atakami.

KROKI HACKOWNIA

W wielu aplikacjach, które wykorzystują standardowy mechanizm plików cookie do przesyłania tokenów sesji, łatwo jest zidentyfikować, który element danych zawiera token. Jednak w innych przypadkach może to wymagać pewnej pracy detektywistycznej.

1. Aplikacja może często wykorzystywać kilka różnych elementów danych razem jako token, w tym pliki cookie, parametry adresów URL i ukryte pola formularzy. Niektóre z tych elementów mogą być używane do utrzymywania stanu sesji w różnych komponentach zaplecza. Nie zakładaj, że konkretny parametr jest tokenem sesji bez udowodnienia tego, ani że sesje są śledzone przy użyciu tylko jednego elementu.

2. Czasami elementy, które wydają się być tokenem sesji aplikacji, mogą nim nie być. W szczególności standardowe pliki cookie sesji generowane przez serwer WWW lub platformę aplikacji mogą być obecne, ale w rzeczywistości nie są wykorzystywane przez aplikację.

3. Obserwuj, jakie nowe elementy są przekazywane do przeglądarki po uwierzytelnieniu. Często nowe tokeny sesji są tworzone po uwierzytelnieniu się użytkownika.

4. Aby sprawdzić, które elementy są faktycznie używane jako tokeny, znajdź stronę, która jest zdecydowanie zależna od sesji (np. strona „moje dane") specyficzna dla użytkownika. Złóż kilka próśb o to, systematycznie usuwając każdy przedmiot, który podejrzewasz, że jest używany jako token. Jeśli usunięcie elementu spowoduje, że strona zależna od sesji nie zostanie zwrócona, może to potwierdzić, że element jest tokenem sesji. Burp Repeater jest przydatnym narzędziem do wykonywania tych testów.

Alternatywy dla sesji

Nie każda aplikacja internetowa korzysta z sesji, a niektóre aplikacje o znaczeniu krytycznym dla bezpieczeństwa, zawierające mechanizmy uwierzytelniania i złożone funkcje, wybierają inne techniki zarządzania stanem. Prawdopodobnie napotkasz dwie możliwe alternatywy:

* Uwierzytelnianie HTTP — aplikacje korzystające z różnych technologii uwierzytelniania opartych na protokole HTTP (podstawowe, szyfrowane, NTLM) czasami unikają potrzeby korzystania z sesji. W przypadku uwierzytelniania HTTP komponent kliencki wchodzi w interakcję z mechanizmem uwierzytelniającym bezpośrednio przez przeglądarkę, przy użyciu nagłówków HTTP, a nie poprzez kod specyficzny dla aplikacji zawarty na każdej pojedynczej stronie. Gdy użytkownik wprowadzi swoje dane uwierzytelniające w oknie dialogowym przeglądarki, przeglądarka skutecznie ponownie przesyła te dane uwierzytelniające (lub ponownie wykonuje wymagane uzgadnianie) przy każdym kolejnym żądaniu do tego samego serwera. Jest to odpowiednik aplikacji, która korzysta z uwierzytelniania opartego na formularzach HTML i umieszcza formularz logowania na każdej stronie aplikacji, wymagając od użytkowników ponownego uwierzytelniania się przy każdej wykonywanej czynności. W związku z tym, gdy używane jest uwierzytelnianie oparte na protokole HTTP, aplikacja może bez niego ponownie identyfikować użytkownika w wielu żądaniach za pomocą sesji. Jednak uwierzytelnianie HTTP jest rzadko stosowane w aplikacjach internetowych o dowolnej złożoności, a inne wszechstronne korzyści, jakie oferują w pełni rozwinięte mechanizmy sesyjne, oznaczają, że praktycznie wszystkie aplikacje internetowe faktycznie wykorzystują te mechanizmy.

* Mechanizmy stanu bez sesji — niektóre aplikacje nie wydają tokenów sesji w celu zarządzania stanem interakcji użytkownika z aplikacją. Zamiast tego przesyłają wszystkie dane wymagane do zarządzania tym stanem za pośrednictwem klienta, zwykle w pliku cookie lub ukrytym polu formularza. W efekcie ten mechanizm używa stanu bez sesji, podobnie jak robi to ASP.NET ViewState. Aby tego typu mechanizm był bezpieczny, dane przesyłane za pośrednictwem klienta muszą być odpowiednio

zabezpieczone. Zwykle obejmuje to skonstruowanie binarnego obiektu blob zawierającego wszystkie informacje o stanie i zaszyfrowanie go lub podpisanie za pomocą uznanego algorytmu. Dane muszą zawierać wystarczający kontekst, aby uniemożliwić osobie atakującej zebranie obiektu stanu w jednym miejscu w aplikacji i przesłanie go do innego miejsca w celu spowodowania niepożądanego zachowania. Aplikacja może również zawierać czas wygaśnięcia w danych obiektu, aby wykonać ekwiwalent limitów czasu sesji.

KROKI HACKOWANIA

1. Jeśli używana jest autoryzacja HTTP, możliwe, że nie jest zaimplementowany żaden mechanizm zarządzania sesją. Użyj metod opisanych wcześniej, aby zbadać rolę odgrywaną przez elementy danych podobne do tokenów.

2. Jeśli aplikacja korzysta z mechanizmu stanu bez sesji, przesyłając wszystkie dane wymagane do utrzymania stanu za pośrednictwem klienta, czasami może to być trudne do wykrycia z całą pewnością, ale następujące mocne wskaźniki wskazują, że ten rodzaj mechanizmu jest używany:

* Elementy danych podobne do tokenów wydawane klientowi są dość długie (100 lub więcej bajtów).

* Aplikacja wydaje nowy przedmiot podobny do tokena w odpowiedzi na każde żądanie.

* Dane w elemencie wydają się być zaszyfrowane (a zatem nie mają rozpoznawalnej struktury) lub podpisane (a zatem mają znaczącą strukturę, której towarzyszy kilka bajtów bezsensownych danych binarnych).

* Aplikacja może odrzucić próby złożenia tej samej pozycji z więcej niż jednym żądaniem.

3. Jeśli dowody wyraźnie sugerują, że aplikacja nie używa tokenów sesyjnych do zarządzania stanem, jest mało prawdopodobne, aby którykolwiek z ataków opisanych w tym rozdziale przyniósł cokolwiek. Prawdopodobnie lepiej byłoby poświęcić czas na szukanie innych poważnych problemów, takich jak zepsuta kontrola dostępu lub wstrzyknięcie kodu.

Słabości w generowaniu tokenów

Mechanizmy zarządzania sesją są często podatne na ataki, ponieważ tokeny są generowane w niebezpieczny sposób, który umożliwia atakującemu zidentyfikowanie wartości tokenów, które zostały wystawione innym użytkownikom.

UWAGA: Istnieje wiele miejsc, w których bezpieczeństwo aplikacji zależy od nieprzewidywalności generowanych przez nią tokenów. Oto kilka przykładów:

* Tokeny odzyskiwania hasła wysyłane na zarejestrowany adres e-mail użytkownika

* Tokeny umieszczone w ukrytych polach formularzy, aby zapobiec atakom fałszowania żądań między witrynami

* Tokeny używane do jednorazowego dostępu do chronionych zasobów

* Trwałe tokeny używane w funkcjach „zapamiętaj mnie”.

* Tokeny umożliwiające klientom aplikacji zakupowej, która nie korzysta z uwierzytelniania, pobranie aktualnego statusu istniejącego zamówienia

Rozważania dotyczące słabości w generowaniu tokenów mają zastosowanie do wszystkich tych przypadków. W rzeczywistości, ponieważ wiele dzisiejszych aplikacji opiera się na mechanizmach

dojrzałej platformy do generowania tokenów sesyjnych, często w tych innych obszarach funkcjonalności znajdują się możliwe do wykorzystania słabości w generowaniu tokenów.

Znaczące tokeny

Niektóre tokeny sesyjne są tworzone przy użyciu przekształcenia nazwy użytkownika lub adresu e-mail lub innych informacji powiązanych z tą osobą. Informacje te mogą być w jakiś sposób zakodowane lub zaciemnione i mogą być łączone z innymi danymi. Na przykład następujący token może początkowo wydawać się długim ciągiem losowym:

757365723d6461663b6170703d61646d696e3b646174653d30312f31322f3131

Jednak po bliższym przyjrzeniu się widać, że zawiera on tylko znaki szesnastkowe. Zgadując, że ciąg znaków może w rzeczywistości być kodowaniem szesnastkowym ciągu znaków ASCII, możesz przepuścić go przez dekoder, aby uzyskać następujące informacje:

użytkownik=daf;app=admin;data=10/09/11

Atakujący mogą wykorzystać znaczenie tego tokena sesji, aby spróbować odgadnąć bieżące sesje innych użytkowników aplikacji. Korzystając z listy wyliczonych lub typowych nazw użytkowników, mogą szybko wygenerować dużą liczbę potencjalnie ważnych tokenów i przetestować je, aby potwierdzić, które są prawidłowe. Tokeny zawierające znaczące dane często mają strukturę. Innymi słowy, zawierają one kilka komponentów, często oddzielonych ogranicznikiem, które można wyodrębnić i przeanalizować oddzielnie, aby umożliwić atakującemu zrozumienie ich funkcji i sposobów generowania. Oto niektóre komponenty, które można napotkać w tokenach strukturalnych:

* Nazwa użytkownika konta

* Identyfikator numeryczny używany przez aplikację do rozróżniania kont

* Imię i nazwisko użytkownika

* Adres e-mail użytkownika

* Grupa użytkownika lub rola w aplikacji

* Znacznik daty/czasu

* Rosnąca lub przewidywalna liczba

* Adres IP klienta

Każdy inny komponent w ustrukturyzowanym tokenie, a nawet cały token, może być zakodowany na różne sposoby. Może to być celowy środek mający na celu zaciemnienie ich treści lub po prostu zapewnienie bezpiecznego transportu danych binarnych przez HTTP. Powszechnie spotykane schematy kodowania obejmują XOR, Base64 i reprezentację szesnastkową przy użyciu znaków ASCII. Może być konieczne przetestowanie różnych dekodowań na każdym komponencie tokena strukturalnego, aby rozpakować go do pierwotnej postaci.

UWAGA: Gdy aplikacja obsługuje żądanie zawierające token strukturalny, może w rzeczywistości nie przetworzyć każdego komponentu z tokenem lub wszystkich danych zawartych w każdym komponencie. W poprzednim przykładzie aplikacja może dekodować token w formacie Base64, a następnie przetwarzać tylko komponenty „użytkownik” i „data”. W przypadkach, gdy token zawiera blob danych binarnych, wiele z tych danych może być dopełnieniem. Tylko niewielka część tego może faktycznie mieć znaczenie dla sprawdzania poprawności, które serwer wykonuje na tokenie. Zawężenie

podczęści tokena, które są faktycznie wymagane, może często znacznie zmniejszyć ilość pozornej entropii i złożoności, które zawiera token.

KROKI HACKOWANIA

1. Uzyskaj pojedynczy token z aplikacji i zmodyfikuj go w sposób systematyczny, aby określić, czy cały token jest sprawdzany, czy też niektóre jego podkomponenty są ignorowane. Spróbuj zmienić wartość tokena po jednym bajcie (lub nawet po jednym bicie) i ponownie prześlij zmodyfikowany token do aplikacji, aby określić, czy nadal jest akceptowany. Jeśli okaże się, że niektóre części tokena nie muszą być poprawne, możesz je wykluczyć z dalszej analizy, potencjalnie zmniejszając ilość pracy, którą musisz wykonać. Możesz użyć typu ładunku „char frobber" w Burp Intruder, aby zmodyfikować wartość tokena w jednej pozycji znaku na raz, aby pomóc w tym zadaniu.

2. Zaloguj się jako kilku różnych użytkowników w różnym czasie i zapisz tokeny otrzymane z serwera. Jeśli samodzielna rejestracja jest dostępna i możesz wybrać swoją nazwę użytkownika, zaloguj się za pomocą serii podobnych nazw użytkownika zawierających niewielkie różnice między nimi, takie jak A, AA, AAA, AAAA, AAAB, AAAC, AABA i tak dalej. Jeśli podczas logowania lub przechowywania w profilach użytkowników podawane są inne dane specyficzne dla użytkownika (takie jak adres e-mail), wykonaj podobne ćwiczenie, aby systematycznie zmieniać te dane i rejestrować tokeny otrzymane po zalogowaniu.

3. Przeanalizuj tokeny pod kątem wszelkich korelacji, które wydają się być powiązane z nazwą użytkownika i innymi danymi kontrolowanymi przez użytkownika.

4. Przeanalizuj tokeny pod kątem wykrywalnego kodowania lub zaciemniania. Tam, gdzie nazwa użytkownika zawiera sekwencję tego samego znaku, poszukaj odpowiedniej sekwencji znaków w tokenie, co może wskazywać na użycie zaciemniania XOR. Szukaj w tokenie sekwencji zawierających tylko znaki szesnastkowe, które mogą wskazywać na szesnastkowe kodowanie łańcucha ASCII lub inne informacje. Szukaj sekwencji, które kończą się znakiem równości i/lub zawierają tylko inne prawidłowe znaki Base64: od a do z, od A do Z, od 0 do 9, + i /.

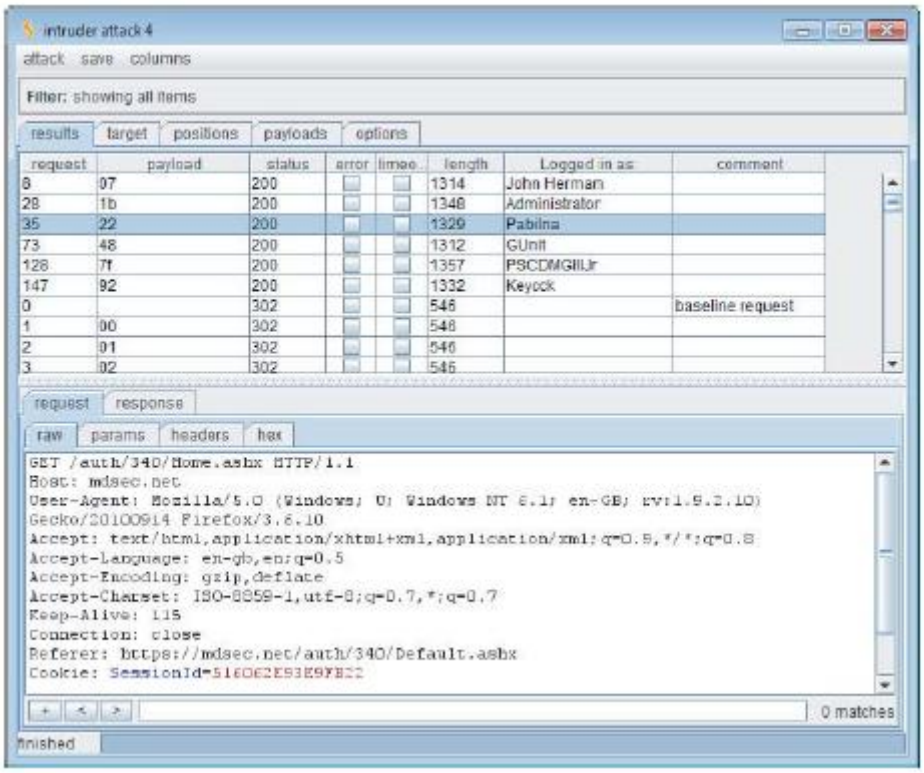
5. Jeśli z próbki tokenów sesyjnych można odtworzyć jakiekolwiek znaczenie, zastanów się, czy masz wystarczające informacje, aby spróbować odgadnąć tokeny ostatnio wydane innym użytkownikom aplikacji. Znajdź stronę aplikacji, która jest zależna od sesji, na przykład stronę, która zwraca komunikat o błędzie lub przekierowanie w inne miejsce, jeśli dostęp do niej odbywa się bez ważnej sesji. Następnie użyj narzędzia takiego jak Burp Intruder, aby wysłać dużą liczbę żądań do tej strony przy użyciu odgadniętych tokenów. Monitoruj wyniki pod kątem wszystkich przypadków, w których strona ładuje się poprawnie, wskazując prawidłowy token sesji.

Przewidywalne tokeny

Niektóre tokeny sesyjne nie zawierają żadnych znaczących danych kojarzących je z konkretnym użytkownikiem. Niemniej jednak można je odgadnąć, ponieważ zawierają sekwencje lub wzorce, które umożliwiają atakującemu ekstrapolację z próbki tokenów w celu znalezienia innych ważnych tokenów ostatnio wyemitowanych przez aplikację. Nawet jeśli ekstrapolacja wymaga pewnych prób i błędów (na przykład jedno prawidłowe odgadnięcie na 1000 prób), nadal umożliwiłoby to zautomatyzowanemu atakowi zidentyfikowanie dużej liczby ważnych tokenów w stosunkowo krótkim czasie. Luki związane z przewidywalnym generowaniem tokenów mogą być znacznie łatwiejsze do wykrycia w komercyjnych implementacjach zarządzania sesją, takich jak serwery WWW lub platformy aplikacji internetowych, niż w aplikacjach na zamówienie. Gdy zdalnie kierujesz się do niestandardowego mechanizmu zarządzania sesjami, próbka wydanych tokenów może być

ograniczona przez pojemność serwera, aktywność innych użytkowników, przepustowość, opóźnienie sieci i tak dalej. Jednak w środowisku laboratoryjnym można szybko utworzyć miliony tokenów próbek, wszystkie precyzyjnie ułożone w kolejności i opatrzone znacznikiem czasu, a także można wyeliminować zakłócenia powodowane przez innych użytkowników.

W najprostszych i najbardziej bezczelnie wrażliwych przypadkach aplikacja może użyć prostego numeru sekwencyjnego jako tokena sesji. W takim przypadku wystarczy uzyskać próbkę dwóch lub trzech tokenów przed rozpoczęciem ataku, który szybko przechwyci 100% aktualnie ważnych sesji. Rysunek pokazuje, jak Burp Intruder jest używany do cyklicznego przeglądania ostatnich dwóch cyfr tokena sesji sekwencyjnej w celu znalezienia wartości, w których sesja jest nadal aktywna i może zostać przejęta.



Tutaj długość odpowiedzi serwera jest wiarygodnym wskaźnikiem, że znaleziono prawidłową sesję. Funkcja wyodrębniania grep została również wykorzystana do wyświetlenia nazwy zalogowanego użytkownika dla każdej sesji. W innych przypadkach tokeny aplikacji mogą zawierać bardziej rozbudowane sekwencje, których odkrycie wymaga pewnego wysiłku. Rodzaje potencjalnych odmian, które możesz tu napotkać, są nieograniczone, ale doświadczenie autorów w tej dziedzinie wskazuje, że przewidywalne tokeny sesyjne zwykle powstają z trzech różnych źródeł:

* Ukryte sekwencje

* Zależność czasowa

* Słabe generowanie liczb losowych

Przyjrzymy się po kolei każdemu z tych obszarów.

Ukryte sekwencje

Często spotyka się tokeny sesji, których nie można łatwo przewidzieć podczas analizy w ich surowej postaci, ale które zawierają sekwencje, które ujawniają się, gdy tokeny zostaną odpowiednio zdekodowane lub rozpakowane. Rozważ następującą serię wartości, które tworzą jeden składnik ustrukturyzowanego tokena sesji:

lwjVJA

Ls3Ajg

xpKr+A

XleXYg

9hyCzA

jefung

JaZoA

Nie można dostrzec żadnego bezpośredniego wzoru; jednak pobieżna inspekcja wskazuje, że tokeny mogą zawierać dane zakodowane w standardzie Base64. Oprócz mieszanych znaków alfabetycznych i numerycznych istnieje znak +, który jest również prawidłowy w ciągu zakodowanym w formacie Base64. Przeprowadzenie tokenów przez dekoder Base64 ujawnia następujące informacje:

--$

.Ž

Æ'«ø

^ W-b

ÌÌ

?an6

%¦Y

Te łańcuchy wydają się bełkotliwe i zawierają również znaki niedrukowalne. Zwykle oznacza to, że masz do czynienia z danymi binarnymi, a nie tekstem ASCII. Renderowanie zdekodowanych danych jako liczb szesnastkowych daje:

9708D524

2ECDC08E

C692ABF8

5E579762

F61C82CC

8DE16E36

25A659A0

Nadal nie ma widocznego wzoru. Jeśli jednak odejmiesz każdą liczbę od poprzedniej, otrzymasz:

FF97C4EB6A

97C4EB6A

FF97C4EB6A

97C4EB6A

FF97C4EB6A

FF97C4EB6A

co natychmiast ujawnia ukryty wzór. Algorytm używany do generowania tokenów dodaje 0x97C4EB6A do poprzedniej wartości, obcina wynik do liczby 32-bitowej, a dane binarne koduje Base64, aby umożliwić ich transport za pomocą protokołu tekstowego HTTP. Korzystając z tej wiedzy, możesz łatwo napisać skrypt do tworzenia serii tokenów, które serwer wytworzy w następnej kolejności, oraz serii, które wytworzył przed przechwyceniem próbki.

Zależność od czasu

Niektóre serwery i aplikacje internetowe wykorzystują algorytmy do generowania tokenów sesji, które wykorzystują czas wygenerowania jako dane wejściowe do wartości tokena. Jeśli do algorytmu zostanie włączona niewystarczająca inna entropia, możesz przewidzieć tokeny innych użytkowników. Chociaż dowolna sekwencja tokenów sama w sobie może wydawać się przypadkowa, ta sama sekwencja w połączeniu z informacją o czasie wygenerowania każdego tokena może zawierać dostrzegalny wzór. W obciążonej aplikacji z dużą liczbą sesji tworzonych w każdej sekundzie atak skryptowy może z powodzeniem zidentyfikować dużą liczbę tokenów innych użytkowników. Podczas testowania aplikacji internetowej sprzedawcy internetowego autorzy napotkali następującą sekwencję tokenów sesji:

3124538-1172764258718

3124539-1172764259062

3124540-1172764259281

3124541-1172764259734

3124542-1172764260046

3124543-1172764260156

3124544-1172764260296

3124545-1172764260421

3124546-1172764260812

3124547-1172764260890

Każdy token wyraźnie składa się z dwóch oddzielnych elementów numerycznych. Pierwsza liczba ma prostą sekwencję rosnącą i jest łatwa do przewidzenia. Druga liczba zwiększa się za każdym razem o różną wartość. Obliczenie różnic między jego wartością w każdym kolejnym żetonie ujawnia, co następuje:

344

219

453

312

110

140

125

391

78

Wydaje się, że sekwencja nie zawiera wiarygodnego przewidywalnego wzoru. Oczywiście możliwe byłoby brutalne użycie odpowiedniego zakresu liczb w zautomatyzowanym ataku w celu wykrycia prawidłowych wartości w sekwencji. Przed przystąpieniem do tego ataku odczekujemy jednak kilka minut i zbieramy kolejną sekwencję żetonów:

3124553-1172764800468

3124554-1172764800609

3124555-1172764801109

3124556-1172764801406

3124557-1172764801703

3124558-1172764802125

3124559-1172764802500

3124560-1172764802656

3124561-1172764803125

3124562-1172764803562

Porównując tę drugą sekwencję żetonów z pierwszą, od razu rzucają się w oczy dwa punkty:

* Pierwsza sekwencja numeryczna rozwija się stopniowo; jednakże pięć wartości zostało pominiętych od końca pierwszej sekwencji. Jest to prawdopodobnie spowodowane tym, że brakujące wartości zostały wydane innym użytkownikom, którzy zalogowali się do aplikacji w oknie pomiędzy dwoma testami.

* Druga sekwencja numeryczna kontynuuje postęp w podobnych odstępach jak poprzednio; jednak pierwsza otrzymana wartość jest o 539 578 większa od poprzedniej wartości.

Ta druga obserwacja natychmiast ostrzega nas o roli czasu w generowaniu tokenów sesji. Najwyraźniej tylko pięć tokenów zostało wydanych między dwoma ćwiczeniami z chwytaniem tokenów. Jednak upłynął okres około 10 minut. Najbardziej prawdopodobnym wyjaśnieniem jest to, że druga liczba jest zależna od czasu i jest prawdopodobnie prostą liczbą milisekund. Rzeczywiście, nasze przeczucie jest słuszne. W kolejnej fazie naszych testów przeprowadzamy przegląd kodu, który ujawnia następujący algorytm generowania tokenów:

```
String sessId = Integer.toString(s_SessionIndex++) +
```

“-” +

System.currentTimeMillis();

Biorąc pod uwagę naszą analizę sposobu tworzenia tokenów, łatwo jest skonstruować atak skryptowy w celu zebrania tokenów sesji, które aplikacja wydaje innym użytkownikom:

* Kontynuujemy sondowanie serwera w celu uzyskania nowych tokenów sesji w krótkim odstępie czasu.

* Monitorujemy przyrosty pierwszej liczby. Gdy wzrośnie o więcej niż 1, wiemy, że token został wydany innemu użytkownikowi.

* Kiedy token został wydany innemu użytkownikowi, znamy górną i dolną granicę drugiego numeru, który został wydany tej osobie, ponieważ posiadamy tokeny, które zostały wydane bezpośrednio przed i po jego. Ponieważ często uzyskujemy nowe tokeny sesji, zakres między tymi granicami będzie zazwyczaj składał się tylko z kilkuset wartości.

* Za każdym razem, gdy token jest wydawany innemu użytkownikowi, przeprowadzamy atak siłowy w celu iteracji każdej liczby w zakresie, dołączając ją do brakującej liczby przyrostowej, o której wiemy, że została wydana innemu użytkownikowi. Próbujemy uzyskać dostęp do chronionej strony za pomocą każdego skonstruowanego przez nas tokena, dopóki próba się nie powiedzie i nie naruszymy sesji użytkownika.

* Ciągłe przeprowadzanie tego ataku skryptowego umożliwi nam przechwycenie tokena sesji każdego innego użytkownika aplikacji. Gdy zaloguje się użytkownik administracyjny, w pełni skompromitujemy całą aplikację.

Słabe generowanie liczb losowych

Bardzo niewiele z tego, co dzieje się w komputerze, jest przypadkowe. Dlatego, gdy do jakiegoś celu wymagana jest losowość, oprogramowanie wykorzystuje różne techniki do generowania liczb w sposób pseudolosowy. Niektóre z zastosowanych algorytmów tworzą sekwencje, które wydają się być stochastyczne i wykazują równomierny rozkład w całym zakresie możliwych wartości. Niemniej jednak każdy, kto uzyska małą próbkę wartości, może je ekstrapolować do przodu lub do tyłu z doskonałą dokładnością. Kiedy przewidywalny generator liczb pseudolosowych jest używany do tworzenia tokenów sesji, powstałe tokeny są podatne na sekwencjonowanie przez atakującego. Jetty to popularny serwer WWW napisany w 100% w Javie, który zapewnia mechanizm zarządzania sesją do użytku przez uruchomione na nim aplikacje. W 2006 roku Chris Anley z NGSSoftware odkrył, że mechanizm był podatny na atak polegający na przewidywaniu tokena sesji. Serwer użył interfejsu Java API java.util.Random do wygenerowania tokenów sesji. To implementuje „liniowy generator kongruencji”, który generuje następną liczbę w sekwencji w następujący sposób:

```
synchronized protected int next(int bits) {

seed = (seed * 0x5DEECE66DL + 0xBL) & ((1L << 48) - 1);

return (int)(seed >>> (48 - bits));

}
```

Algorytm ten bierze ostatnią wygenerowaną liczbę, mnoży ją przez stałą i dodaje kolejną stałą, aby uzyskać następną liczbę. Liczba jest obcinana do 48 bitów, a algorytm przesuwa wynik, aby zwrócić określoną liczbę bitów żądaną przez dzwoniącego. Znając ten algorytm i jedną wygenerowaną przez

niego liczbę, możemy łatwo wyprowadzić ciąg liczb, które algorytm wygeneruje w następnej kolejności. Przy odrobinie teorii liczb możemy również wyprowadzić sekwencję, którą wygenerowała wcześniej. Oznacza to, że atakujący, który uzyska pojedynczy token sesji z serwera, może uzyskać tokeny wszystkich bieżących i przyszłych sesji.

NOTATKA : Czasami, gdy tokeny są tworzone na podstawie danych wyjściowych generatora liczb pseudolosowych, programiści decydują się na skonstruowanie każdego tokena poprzez połączenie kilku kolejnych wyjść z generatora. Postrzeganym uzasadnieniem tego jest to, że tworzy to dłuższy, a zatem „silniejszy" token. Jednak taka taktyka jest zwykle błędem. Jeśli atakującemu uda się uzyskać kilka kolejnych wyjść z generatora, może to umożliwić mu wywnioskowanie pewnych informacji o jego stanie wewnętrznym. W rzeczywistości atakującemu może być łatwiej ekstrapolować sekwencję wyjść generatora w przód lub w tył.

Inne gotowe struktury aplikacji wykorzystują zaskakująco proste lub przewidywalne źródła entropii w generowaniu tokenów sesji, z których większość jest deterministyczna. Na przykład w frameworkach PHP 5.3.2 i wcześniejszych token sesji jest generowany na podstawie adresu IP klienta, czasu epoki podczas tworzenia tokena, mikrosekund podczas tworzenia tokena i liniowego generatora kongruencji. Chociaż jest tu kilka nieznanych wartości, niektóre aplikacje mogą ujawniać informacje, które pozwalają je wywnioskować. Serwis społecznościowy może ujawnić czas logowania oraz adres IP użytkowników serwisu. Dodatkowo, ziarno użyte w tym generatorze to czas rozpoczęcia procesu PHP, który można określić jako mieszczący się w niewielkim zakresie wartości, jeśli atakujący monitoruje serwer.

UWAGA: Jest to rozwijający się obszar badań. Słabości w generowaniu tokenów sesyjnych PHP zostały wskazane na liście mailingowej Full Disclosure w 2001 roku, ale nie wykazano, że faktycznie można je wykorzystać. Teoria z 2001 roku została ostatecznie zastosowana w praktyce przez Samy'ego Kamkara za pomocą narzędzia phpwn w 2010 roku.

Testowanie jakości losowości

W niektórych przypadkach można zidentyfikować wzorce w serii tokenów po prostu na podstawie inspekcji wizualnej lub niewielkiej ilości ręcznej analizy. Ogólnie rzecz biorąc, musisz jednak zastosować bardziej rygorystyczne podejście do testowania jakości losowości w tokenach aplikacji. Standardowe podejście do tego zadania stosuje zasady testowania hipotez statystycznych i wykorzystuje różne dobrze udokumentowane testy, które szukają dowodów na nielosowość w próbce tokenów. Etapy wysokiego poziomu w tym procesie są następujące:

1. Zacznij od hipotezy, że tokeny są generowane losowo.

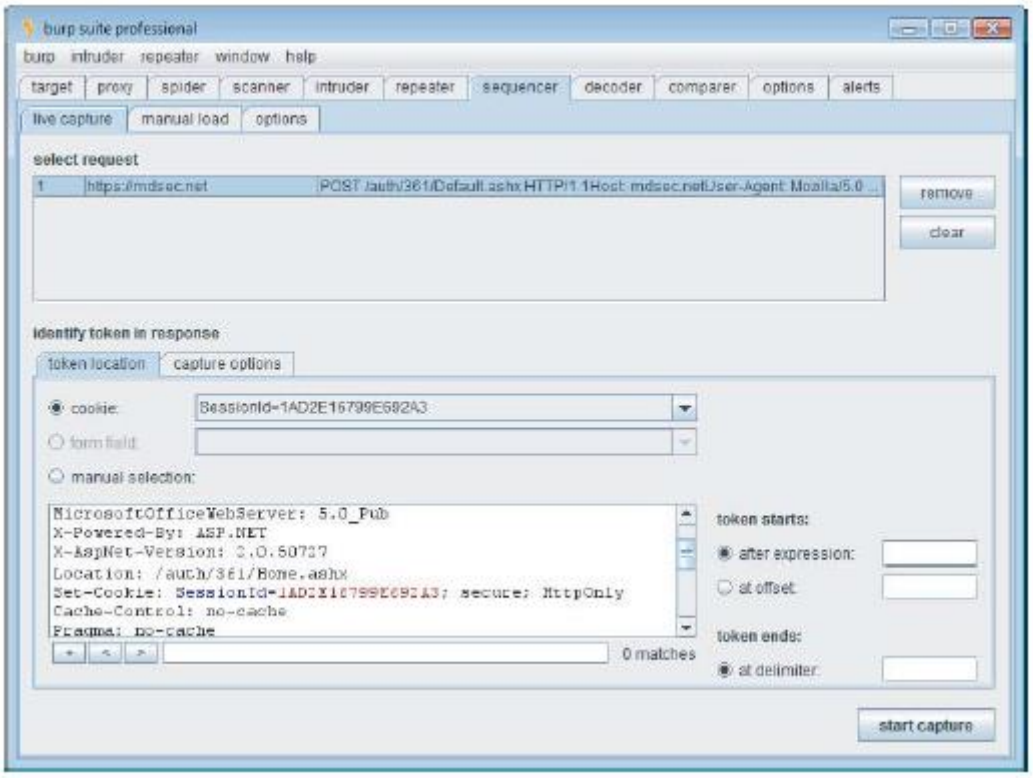
2. Zastosuj serię testów, z których każdy obserwuje określone właściwości próbki, które mogą mieć określone cechy, jeśli tokeny są generowane losowo.

3. Dla każdego testu oblicz prawdopodobieństwo wystąpienia zaobserwowanych cech, opierając się na założeniu, że hipoteza jest prawdziwa.

4. Jeśli to prawdopodobieństwo spadnie poniżej pewnego poziomu („poziom istotności"), odrzuć hipotezę i wyciągnij wniosek, że tokeny nie są generowane losowo.

Dobra wiadomość jest taka, że nie musisz robić tego ręcznie! Najlepszym obecnie dostępnym narzędziem do testowania losowości tokenów aplikacji webowych jest Burp Sequencer. To narzędzie stosuje kilka standardowych testów w elastyczny sposób i daje jasne wyniki, które są łatwe do interpretacji. Aby użyć Burp Sequencer, musisz znaleźć odpowiedź z aplikacji, która wystawia token,

który chcesz przetestować, na przykład odpowiedź na żądanie logowania, które wystawia nowy plik cookie zawierający token sesji. Wybierz opcję „wyślij do sekwensera” z menu kontekstowego Burpa i w konfiguracji Sekwencera ustaw lokalizację tokena w odpowiedzi, jak pokazano na rysunku .



Możesz także skonfigurować różne opcje wpływające na sposób zbierania tokenów, a następnie kliknąć przycisk rozpoczęcia przechwytywania, aby rozpocząć przechwytywanie tokenów. Jeśli uzyskałeś już odpowiednią próbkę tokenów w inny sposób (na przykład zapisując wyniki ataku Burp Intruder), możesz użyć zakładki ładowania ręcznego, aby pominąć przechwytywanie tokenów i przejść od razu do analizy statystycznej.

Po uzyskaniu odpowiedniej próbki żetonów możesz przeprowadzić analizę statystyczną na próbce. Można również przeprowadzać analizy pośrednie, gdy próbka jest nadal pobierana. Ogólnie rzecz biorąc, uzyskanie większej próbki poprawia wiarygodność analizy. Minimalna wielkość próbki, jakiej wymaga Burp, to 100 tokenów, ale najlepiej byłoby uzyskać znacznie większą próbkę. Jeśli analiza kilkuset tokenów jednoznacznie wykaże, że tokeny nie przejdą testów losowości, można rozsądnie uznać, że nie ma potrzeby przejmowania kolejnych tokenów. W przeciwnym razie należy kontynuować przechwytywanie tokenów i okresowo ponownie przeprowadzać analizę. Jeśli zdobędziesz 5000 żetonów, które pomyślnie przejdą testy losowości, możesz zdecydować, że to wystarczy. Aby jednak osiągnąć zgodność z formalnymi testami FIPS na losowość, należy uzyskać próbkę 20 000 tokenów. Jest to największy rozmiar próbki obsługiwany przez Burp. Burp Sequencer przeprowadza testy statystyczne na poziomie znaków i bitów. Wyniki wszystkich testów są sumowane w celu uzyskania ogólnego oszacowania liczby bitów efektywnej entropii w tokenie; jest to kluczowy wynik do rozważenia. Można jednak również przeanalizować wyniki każdego testu, aby dokładnie zrozumieć, w jaki sposób i dlaczego różne części tokenu przeszły lub zakończyły się niepowodzeniem w każdym teście, jak pokazano na rysunku.

Metodologia stosowana dla każdego rodzaju testu jest opisana poniżej wyników testu.

Zauważ, że Burp wykonuje wszystkie testy indywidualnie dla każdego znaku i bitu danych w tokenie. W wielu przypadkach przekonasz się, że duże części tokena strukturalnego nie są przypadkowe; to samo w sobie może nie przedstawiać żadnej słabości. Liczy się to, aby token zawierał wystarczającą liczbę bitów, które przejdą testy losowości. Na przykład, jeśli duży token zawiera 1000 bitów informacji, a tylko 50 z tych bitów przechodzi testy losowości, token jako całość jest nie mniej niezawodny niż token 50-bitowy, który w pełni przechodzi testy.

UWAGA: Podczas przeprowadzania testów statystycznych dotyczących losowości należy pamiętać o dwóch ważnych zastrzeżeniach. Zastrzeżenia te wpływają na poprawną interpretację wyników testów i ich konsekwencje dla stanu bezpieczeństwa aplikacji. Po pierwsze, tokeny generowane w sposób całkowicie deterministyczny mogą przejść testy statystyczne pod kątem losowości. Na przykład liniowy kongruencyjny generator liczb pseudolosowych lub algorytm, który oblicza skrót liczby sekwencyjnej, może generować dane wyjściowe, które pomyślnie przejdą testy. Jednak osoba atakująca, która zna algorytm i wewnętrzny stan generatora, może ekstrapolować jego dane wyjściowe z całkowitą niezawodnością zarówno w kierunku do przodu, jak i do tyłu. Po drugie, tokeny, które nie przejdą testów statystycznych pod kątem losowości, mogą w rzeczywistości nie być przewidywalne w żadnej praktycznej sytuacji. Jeśli dany bit tokena nie przejdzie testów, oznacza to tylko, że sekwencja bitów obserwowana na tej pozycji zawiera cechy, które są mało prawdopodobne, aby wystąpiły w

prawdziwie losowym tokenie. Ale próba przewidzenia wartości tego bitu w następnym żetonie na podstawie zaobserwowanych cech może być niewiele bardziej niezawodna niż zgadywanie na ślepo. Pomnożenie tej zawodności przez dużą liczbę bitów, które należy przewidzieć jednocześnie, może oznaczać, że prawdopodobieństwo wykonania prawidłowej prognozy jest bardzo niskie.

KROKI HACKOWANIA

1. Określ, kiedy i jak wydawane są tokeny sesyjne, przechodząc przez aplikację od pierwszej strony aplikacji przez dowolne funkcje logowania. Powszechne są dwa zachowania:

* Aplikacja tworzy nową sesję za każdym razem, gdy otrzymane zostanie żądanie, które nie przesyła tokena.

* Aplikacja tworzy nową sesję po pomyślnym zalogowaniu. Aby zebrać dużą liczbę tokenów w sposób zautomatyzowany, najlepiej zidentyfikować pojedyncze żądanie (zazwyczaj GET / lub przesłanie loginu), które powoduje wydanie nowego tokena.

2. W Burp Suite wyślij żądanie tworzenia nowej sesji do Burp Sequencer i skonfiguruj lokalizację tokena. Następnie rozpocznij przechwytywanie na żywo, aby zebrać jak najwięcej tokenów. Jeśli używany jest niestandardowy mechanizm zarządzania sesją, a dostęp do aplikacji masz tylko zdalny, zbierz tokeny tak szybko, jak to możliwe, aby zminimalizować utratę tokenów wydanych innym użytkownikom i zmniejszyć wpływ jakiejkolwiek zależności czasowej.

3. Jeśli używany jest komercyjny mechanizm zarządzania sesją i/lub masz lokalny dostęp do aplikacji, możesz w kontrolowanych warunkach pozyskiwać nieskończenie duże sekwencje tokenów sesji.

4. Podczas gdy Burp Sequencer przechwytuje tokeny, włącz ustawienie „automatycznej analizy", aby Burp automatycznie okresowo przeprowadzał analizę statystyczną. Zbierz co najmniej 500 tokenów przed szczegółowym przejrzeniem wyników. Jeśli wystarczająca liczba bitów w tokenie przeszła testy, kontynuuj zbieranie tokenów tak długo, jak to możliwe, przeglądając wyniki analizy w miarę przechwytywania kolejnych tokenów.

5. Jeśli tokeny nie przejdą testów losowości i wydają się zawierać wzorce, które można wykorzystać do przewidywania przyszłych tokenów, wykonaj ponownie ćwiczenie z innego adresu IP i (jeśli dotyczy) innej nazwy użytkownika. Pomoże to określić, czy wykryto ten sam wzorzec i czy tokeny otrzymane w pierwszym ćwiczeniu można ekstrapolować w celu zidentyfikowania tokenów otrzymanych w drugim ćwiczeniu. Czasami sekwencja tokenów przechwyconych przez jednego użytkownika manifestuje wzorzec. Ale to nie pozwoli na prostą ekstrapolację na tokeny wydawane innym użytkownikom, ponieważ informacje takie jak źródłowy adres IP są wykorzystywane jako źródło entropii (takie jak ziarno do generatora liczb losowych)

6. Jeśli uważasz, że masz wystarczający wgląd w algorytm generowania tokenów, aby przeprowadzić zautomatyzowany atak na sesje innych użytkowników, prawdopodobnie najlepszym sposobem na osiągnięcie tego jest użycie dostosowanego skryptu. Może to generować tokeny przy użyciu określonych wzorców, które zaobserwowałeś, i zastosować wszelkie niezbędne kodowanie.

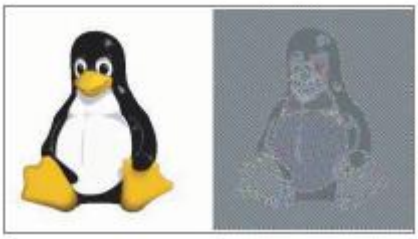
7. Jeśli dostępny jest kod źródłowy, dokładnie przejrzyj kod odpowiedzialny za generowanie tokenów sesji, aby zrozumieć zastosowany mechanizm i określić, czy jest on podatny na przewidywania. Jeśli entropia jest pobierana z danych, które można określić w aplikacji w zakresie brutalnej siły, rozważ praktyczną liczbę żądań, które byłyby potrzebne do brutalnego wymuszenia tokena aplikacji.

Szyfrowane tokeny

Niektóre aplikacje używają tokenów, które zawierają istotne informacje o użytkowniku i starają się uniknąć oczywistych problemów z tym związanych, szyfrując tokeny przed ich wydaniem użytkownikom. Ponieważ tokeny są szyfrowane przy użyciu tajnego klucza, który jest nieznany użytkownikom, wydaje się to solidnym podejściem, ponieważ użytkownicy nie będą mogli odszyfrować tokenów ani manipulować ich zawartością. Jednak w niektórych sytuacjach, w zależności od zastosowanego algorytmu szyfrowania i sposobu, w jaki aplikacja przetwarza tokeny, mimo wszystko użytkownicy mogą manipulować znaczącą zawartością tokenów bez faktycznego ich odszyfrowywania. Choć może to zabrzmieć dziwnie, w rzeczywistości są to wykonalne ataki, które czasami są łatwe do przeprowadzenia, a wiele rzeczywistych aplikacji okazało się na nie podatnych. Rodzaje ataków, które można zastosować, zależą od dokładnego używanego algorytmu kryptograficznego.

Szyfry EBC

Aplikacje wykorzystujące zaszyfrowane tokeny używają algorytmu szyfrowania symetrycznego, dzięki czemu tokeny otrzymane od użytkowników mogą zostać odszyfrowane w celu odzyskania ich znaczącej zawartości. Niektóre algorytmy szyfrowania symetrycznego wykorzystują szyfr „elektronicznej książki kodowej” (ECB). Ten typ szyfru dzieli zwykły tekst na bloki o równej wielkości (na przykład 8 bajtów każdy) i szyfruje każdy blok przy użyciu tajnego klucza. Podczas deszyfrowania każdy blok tekstu zaszyfrowanego jest odszyfrowywany przy użyciu tego samego klucza w celu odzyskania oryginalnego bloku tekstu jawnego. Jedną z cech tej metody jest to, że wzorce w tekście jawnym mogą skutkować wzorcami w tekście zaszyfrowanym, ponieważ identyczne bloki tekstu jawnego zostaną zaszyfrowane w identyczne bloki tekstu zaszyfrowanego. W przypadku niektórych typów danych, takich jak obrazy bitmapowe, oznacza to, że znaczące informacje z tekstu jawnego można rozpoznać w tekście zaszyfrowanym, jak pokazano na rysunku.



Pomimo tego niedociągnięcia w przypadku EBC, szyfry te są często używane do szyfrowania informacji w aplikacjach internetowych. Nawet w sytuacjach, w których nie pojawia się problem wzorców w tekście jawnym, nadal mogą istnieć luki w zabezpieczeniach. Wynika to z zachowania szyfru polegającego na szyfrowaniu identycznych bloków tekstu jawnego w identyczne bloki tekstu zaszyfrowanego. Rozważmy aplikację, której tokeny zawierają kilka różnych znaczących komponentów, w tym numeryczny identyfikator użytkownika:

rnd=2458992;app=iTradeEUR_1;uid=218;username=dafydd;time=634430423694715

000;

Kiedy ten token jest zaszyfrowany, najwyraźniej nie ma on znaczenia i prawdopodobnie przejdzie wszystkie standardowe testy statystyczne pod kątem losowości:

68BAC980742B9EF80A27CBBBC0618E3876FF3D6C6E6A7B9CB8FCA486F9E11922776F0307329140AA
BD2223F003A8309DDB6B970C47BA2E249A067059140AABD223F003A8309A5C2EC

Stosowany szyfr EBC działa na 8-bajtowych blokach danych, a bloki tekstu jawnego są odwzorowywane na odpowiadające im bloki tekstu zaszyfrowanego w następujący sposób:

rnd=2458 68BAC980742B9EF8

992;app= 0A27CBBBC0618E38

iTradeEU 76FF3D6C6E6A7B9C

R_1;uid= B8FCA486F9E11922

218;user 776F0307329140AA

name=daf BD223F003A8309DD

ydd;time B6B970C47BA2E249

=6344304 A0670592D74BCD07

23694715 D51A3E150EFC2E69

000; 885A5C8131E4210F

Teraz, ponieważ każdy blok tekstu zaszyfrowanego będzie zawsze deszyfrowany do tego samego bloku tekstu jawnego, osoba atakująca może manipulować sekwencją bloków tekstu zaszyfrowanego, aby zmodyfikować odpowiedni tekst jawny w znaczący sposób. W zależności od tego, jak dokładnie aplikacja przetwarza wynikowy odszyfrowany token, może to umożliwić atakującemu przełączenie się na innego użytkownika lub zwiększenie uprawnień. Na przykład, jeśli drugi blok zostanie zduplikowany po czwartym bloku, sekwencja bloków będzie następująca:

rnd=2458 68BAC980742B9EF8

992;app= 0A27CBBBC0618E38

iTradeEU 76FF3D6C6E6A7B9C

R_1;uid= B8FCA486F9E11922

992;app= 0A27CBBBC0618E38

218;user 776F0307329140AA

name=daf BD223F003A8309DD

ydd;time B6B970C47BA2E249

=6344304 A0670592D74BCD07

23694715 D51A3E150EFC2E69

000; 885A5C8131E4210F

Odszyfrowany token zawiera teraz zmodyfikowaną wartość UID, a także zduplikowaną wartość aplikacji. Dokładnie to, co się stanie, zależy od tego, jak aplikacja przetwarza odszyfrowany token. Często aplikacje wykorzystujące tokeny w ten sposób sprawdzają tylko niektóre części odszyfrowanego tokena, takie jak identyfikator użytkownika. Jeśli aplikacja zachowuje się w ten sposób, to przetworzy żądanie w kontekście użytkownika, który ma identyfikator użytkownika 992, a nie oryginalny 218. Opisany właśnie atak polegałby na nadaniu odpowiedniej wartości rnd, która odpowiada poprawna wartość uid podczas manipulowania blokami. Alternatywnym i bardziej niezawodnym atakiem byłoby zarejestrowanie nazwy użytkownika zawierającej wartość liczbową z odpowiednim przesunięciem i

zduplikowanie tego bloku, aby zastąpić istniejącą wartość uid. Załóżmy, że rejestrujesz nazwę użytkownika daf1 i otrzymujesz następujący token:

9A5A47BF9B3B6603708F9DEAD67C7F4C76FF3D6C6E6A7B9CB8FCA486F9E11922A5BC430A

73B38C14BD223F003A8309DDF29A5A6F0DC06C53905B5366F5F4684C0D2BBBB08BD834BB ADEBC07FFE87819D

Bloki tekstu jawnego i zaszyfrowanego dla tego tokena są następujące:

rnd=9224 9A5A47BF9B3B6603

856;app= 708F9DEAD67C7F4C

iTradeEU 76FF3D6C6E6A7B9C

R_1;uid= B8FCA486F9E11922

219;user A5BC430A73B38C14

name=daf BD223F003A8309DD

1;time=6 F29A5A6F0DC06C53

34430503 905B5366F5F4684C

61065250 0D2BBBB08BD834BB

0; ADEBC07FFE87819D

Jeśli następnie zduplikujesz siódmy blok po czwartym bloku, odszyfrowany token będzie zawierał wartość uid równą 1:

rnd=9224 9A5A47BF9B3B6603

856;app= 708F9DEAD67C7F4C

iTradeEU 76FF3D6C6E6A7B9C

R_1;uid= B8FCA486F9E11922

1;time=6 F29A5A6F0DC06C53

219;user A5BC430A73B38C14

name=daf BD223F003A8309DD

1;time=6 F29A5A6F0DC06C53

34430503 905B5366F5F4684C

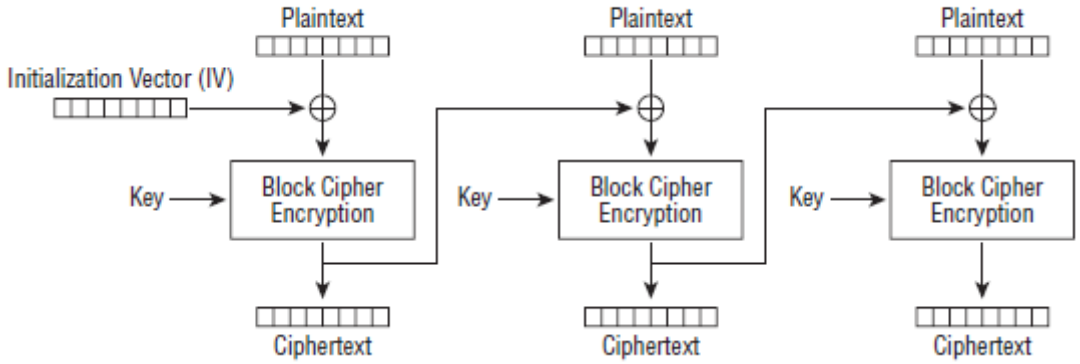
61065250 0D2BBBB08BD834BB

0; ADEBC07FFE87819D

Rejestrując odpowiedni zakres nazw użytkowników i ponownie przeprowadzając ten atak, możesz potencjalnie przechodzić przez cały zakres prawidłowych wartości UID i w ten sposób udawać każdego użytkownika aplikacji.

Szyfry CBC

Niedociągnięcia w szyfrach EBC doprowadziły do rozwoju szyfrów opartych na łańcuchu bloków szyfrów (CBC). W przypadku szyfru CBC, zanim każdy blok tekstu jawnego zostanie zaszyfrowany, jest on poddawany operacji XOR względem poprzedniego bloku tekstu zaszyfrowanego, jak pokazano na rysunku .



Zapobiega to szyfrowaniu identycznych bloków tekstu jawnego w identyczne bloki tekstu zaszyfrowanego. Podczas deszyfrowania operacja XOR jest stosowana w odwrotnej kolejności, a każdy odszyfrowany blok jest poddawany operacji XOR względem poprzedniego bloku tekstu zaszyfrowanego w celu odzyskania oryginalnego tekstu jawnego.

Ponieważ szyfry CBC pozwalają uniknąć niektórych problemów z szyframi EBC, w trybie CBC często używane są standardowe algorytmy szyfrowania symetrycznego, takie jak DES i AES. Jednak sposób, w jaki zaszyfrowane tokeny CBC są często wykorzystywane w aplikacjach internetowych, oznacza, że osoba atakująca może być w stanie manipulować częściami odszyfrowanych tokenów bez znajomości tajnego klucza. Rozważ odmianę poprzedniej aplikacji, której tokeny zawierają kilka różnych znaczących komponentów, w tym numeryczny identyfikator użytkownika:

rnd=191432758301;app=eBankProdTC;uid=216;czas=6343303;

Tak jak poprzednio, po zaszyfrowaniu tych informacji powstaje pozornie bezsensowny token:

0FB1F1AFB4C874E695AAFC9AA4C2269D3E8E66BBA9B2829B173F255D447C51321586257C
6E459A93635636F45D7B1A43163201477

Ponieważ ten token jest szyfrowany przy użyciu szyfru CBC, podczas odszyfrowywania tokena każdy blok tekstu zaszyfrowanego jest poddawany operacji XOR względem następnego bloku odszyfrowanego tekstu w celu uzyskania tekstu jawnego. Teraz, jeśli atakujący zmodyfikuje części tekstu zaszyfrowanego (token, który otrzymał), spowoduje to odszyfrowanie tego konkretnego bloku do śmieci. Jednak powoduje to również XORowanie następującego bloku odszyfrowanego tekstu względem innej wartości, co skutkuje zmodyfikowanym, ale nadal znaczącym tekstem jawnym. Innymi słowy, manipulując pojedynczym pojedynczym blokiem tokena, osoba atakująca może systematycznie modyfikować odszyfrowaną zawartość następującego po nim bloku. W zależności od tego, w jaki sposób aplikacja przetwarza wynikowy odszyfrowany token, może to umożliwić atakującemu przełączenie się na innego użytkownika lub zwiększenie uprawnień. Zobaczmy jak. W opisanym przykładzie atakujący działa przez zaszyfrowany token, zmieniając jeden znak na raz w dowolny sposób i wysyłając każdy zmodyfikowany token do aplikacji. Wiąże się to z dużą liczbą wniosków. Poniżej przedstawiono wybrane wartości, które pojawiają się, gdy aplikacja odszyfrowuje każdy zmodyfikowany token:

????????32858301;app=eBankProdTC;uid=216;time=6343303;

????????32758321;app=eBankProdTC;uid=216;time=6343303;

rnd=1914????????;aqp=eBankProdTC;uid=216;time=6343303;

rnd=1914????????;app=eAankProdTC;uid=216;time=6343303;

rnd=191432758301????????nkPqodTC;uid=216;time=6343303;

rnd=191432758301????????nkProdUC;uid=216;time=6343303;

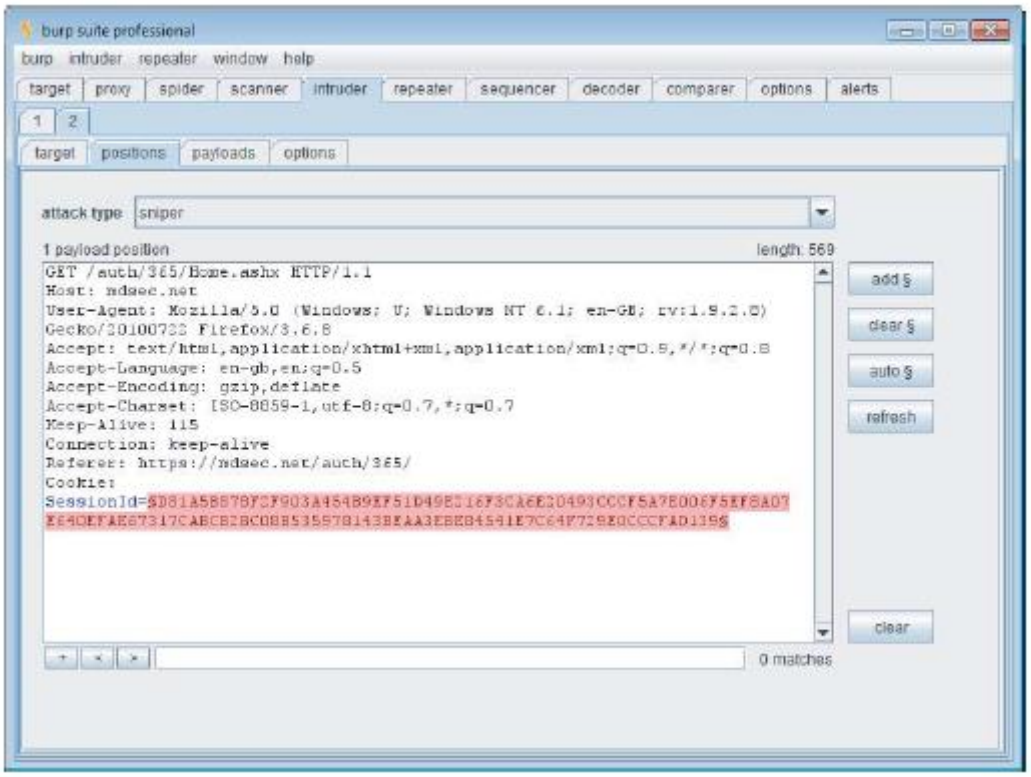
rnd=191432758301;app=eBa????????;uie=216;time=6343303;

rnd=191432758301;app=eBa????????;uid=226;time=6343303;

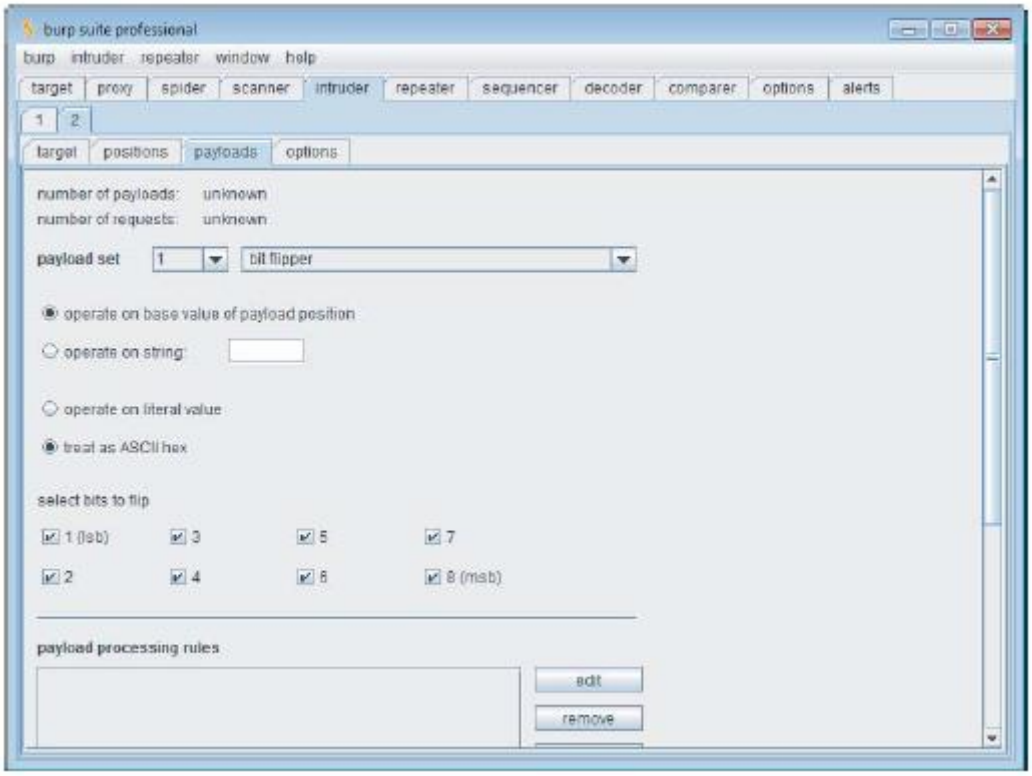
rnd=191432758301;app=eBankProdTC????????;timd=6343303;

rnd=191432758301;app=eBankProdTC????????;time=6343503;

W każdym przypadku blok zmodyfikowany przez atakującego jest zgodnie z oczekiwaniami odszyfrowywany do śmieci (oznaczony jako ????????). Jednak poniższy blok jest odszyfrowywany do zrozumiałego tekstu, który różni się nieco od oryginalnego tokena. Jak już opisano, różnica ta występuje, ponieważ odszyfrowany tekst jest poddawany operacji XOR względem poprzedniego bloku tekstu zaszyfrowanego, który atakujący nieznacznie zmodyfikował. Chociaż atakujący nie widzi odszyfrowanych wartości, aplikacja próbuje je przetworzyć, a atakujący widzi wyniki w odpowiedziach aplikacji. Dokładnie to, co się stanie, zależy od tego, jak aplikacja obsłuży część odszyfrowanego tokena, która została uszkodzona. Jeśli aplikacja odrzuci tokeny zawierające nieprawidłowe dane, atak się nie powiedzie. Często jednak aplikacje wykorzystujące tokeny w ten sposób sprawdzają tylko niektóre części odszyfrowanego tokena, takie jak identyfikator użytkownika. Jeśli aplikacja zachowuje się w ten sposób, to ósmy przykład pokazany na powyższej liście powiedzie się, a aplikacja przetwarza żądanie w kontekście użytkownika, który ma identyfikator użytkownika równy 226, a nie oryginalny 216. Możesz łatwo przetestować aplikacje pod tym kątem luka w zabezpieczeniach wykorzystująca typ ładunku „bit flipper” w Burp Intruder. Najpierw należy zalogować się do aplikacji przy użyciu własnego konta. Następnie znajdujesz stronę aplikacji, która zależy od zalogowanej sesji i pokazuje tożsamość zalogowanego użytkownika w odpowiedzi. Zazwyczaj temu służy główna strona docelowa użytkownika lub strona szczegółów konta. Rysunek przedstawia Burp Intruder ustawionego na kierowanie na stronę główną użytkownika, z zaszyfrowanym tokenem sesji oznaczonym jako pozycja ładunku.

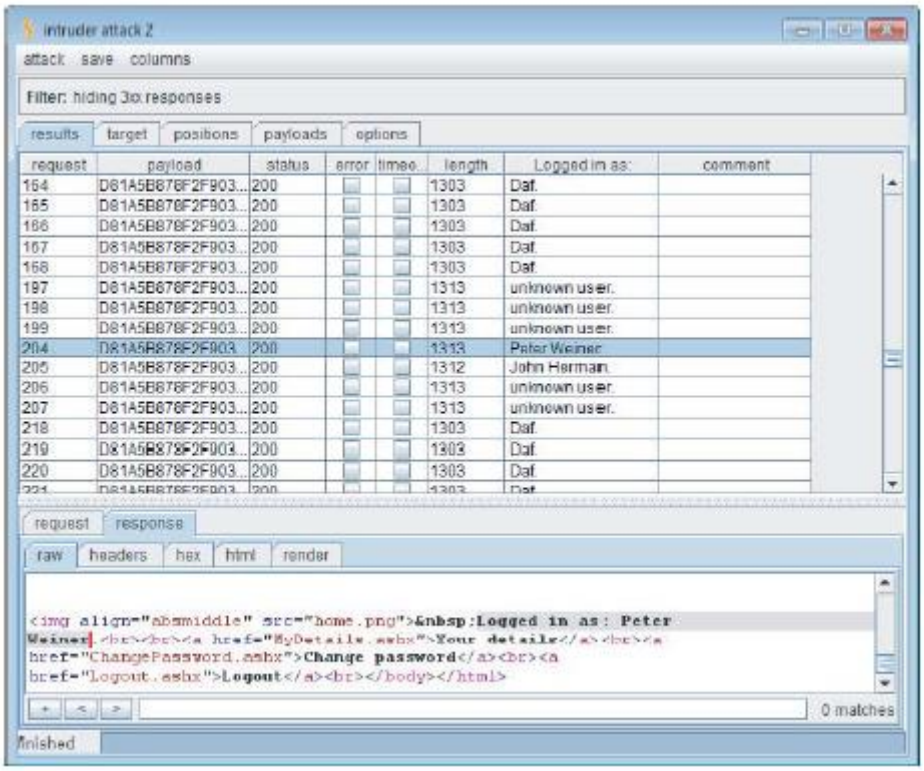


Rysunek przedstawia wymaganą konfigurację ładunku.



Mówi Burpowi, aby operował na oryginalnej wartości tokena, traktując go jako szesnastkowy kod ASCII i aby odwracał każdy bit w każdej pozycji znaku. Takie podejście jest idealne, ponieważ wymaga stosunkowo niewielkiej liczby żądań (osiem żądań na bajt danych w tokenie) i prawie zawsze identyfikuje, czy aplikacja jest podatna na ataki. Pozwala to na użycie bardziej ukierunkowanego ataku

w celu rzeczywistego wykorzystania. Po wykonaniu ataku początkowe żądania nie powodują zauważalnych zmian w odpowiedziach aplikacji, a sesja użytkownika pozostaje nienaruszona. Jest to interesujące samo w sobie, ponieważ wskazuje, że pierwsza część tokena nie jest używana do identyfikacji zalogowanego użytkownika. Wiele żądań późniejszych w ataku powoduje przekierowanie do strony logowania, co wskazuje, że modyfikacja w jakiś sposób unieważniła token. Co najważniejsze, istnieje również seria żądań, w przypadku których odpowiedź wydaje się być częścią prawidłowej sesji, ale nie jest powiązana z oryginalną tożsamością użytkownika. Odpowiada to blokowi tokena, który zawiera wartość uid. W niektórych przypadkach aplikacja po prostu wyświetla komunikat „nieznany użytkownik", wskazując, że zmodyfikowany identyfikator użytkownika nie odpowiada faktycznemu użytkownikowi, w związku z czym atak się nie powiódł. W innych przypadkach pokazuje nazwę innego zarejestrowanego użytkownika aplikacji, dowodząc jednoznacznie, że atak się powiódł. Rysunek przedstawia wyniki ataku.



Tutaj zdefiniowaliśmy kolumnę wyodrębniania grep, aby wyświetlić tożsamość zalogowanego użytkownika i ustawiliśmy filtr, aby ukryć odpowiedzi, które są przekierowaniami do strony logowania. Po zidentyfikowaniu luki w zabezpieczeniach można przystąpić do jej wykorzystania za pomocą bardziej ukierunkowanego ataku. Aby to zrobić, należy określić na podstawie wyników dokładnie, który blok zaszyfrowanego tokena jest modyfikowany, gdy zmienia się kontekst użytkownika. Następnie przeprowadziłbyś atak, który testuje wiele dalszych wartości w tym bloku. Aby to zrobić, możesz użyć typu ładunku liczbowego w Burp Intruder.

UWAGA: Niektóre aplikacje wykorzystują ogólniejszą technikę szyfrowania znaczących danych w parametrach żądania, aby zapobiec manipulowaniu danymi, takimi jak ceny zakupów. W każdym miejscu, w którym widzisz najwyraźniej zaszyfrowane dane, które odgrywają kluczową rolę w funkcjonalności aplikacji, powinieneś wypróbować technikę przerzucania bitów, aby sprawdzić, czy możesz manipulować zaszyfrowanymi informacjami w znaczący sposób, aby zakłócić logikę aplikacji.

Próbując wykorzystać lukę w zabezpieczeniach opisaną w tej sekcji, Twoim celem byłoby oczywiście podszywanie się pod innych użytkowników aplikacji — najlepiej administratora z wyższymi uprawnieniami. Jeśli jesteś ograniczony do ślepego manipulowania częściami zaszyfrowanego tokena, może to wymagać odrobiny szczęścia. Jednak w niektórych przypadkach aplikacja może dać ci więcej pomocy. Gdy aplikacja stosuje szyfrowanie symetryczne w celu ochrony danych przed manipulacją przez użytkowników, często w całej aplikacji używany jest ten sam algorytm szyfrowania i ten sam klucz. W tej sytuacji, jeśli jakakolwiek funkcja aplikacji ujawni użytkownikowi odszyfrowaną wartość dowolnego zaszyfrowanego ciągu, można to wykorzystać do pełnego odszyfrowania dowolnego elementu chronionych informacji. Jedna zaobserwowana przez autorów aplikacja zawierała funkcję wysyłania/pobierania plików. Po przesłaniu pliku użytkownicy otrzymywali link do pobrania zawierający parametr nazwy pliku. Aby zapobiec różnym atakom manipulującym ścieżkami plików, aplikacja zaszyfrowała nazwę pliku w ramach tego parametru. Jeśli jednak użytkownik poprosił o plik, który został usunięty, aplikacja wyświetlała komunikat o błędzie zawierający odszyfrowaną nazwę żądanego pliku. To zachowanie można wykorzystać do znalezienia wartości zwykłego tekstu dowolnego zaszyfrowanego ciągu używanego w aplikacji, w tym wartości tokenów sesji. Wykryto, że tokeny sesji zawierają różne znaczące wartości w ustrukturyzowanym formacie, który był podatny na atak opisany w tej sekcji. Ponieważ wartości te obejmowały tekstowe nazwy użytkowników i role aplikacji, a nie identyfikatory numeryczne, niezwykle trudno byłoby wykonać udany exploit przy użyciu tylko ślepego odwracania bitów. Jednak za pomocą funkcji deszyfratora nazw plików można było systematycznie manipulować bitami tokena podczas przeglądania wyników. Pozwoliło to na zbudowanie tokena, który po odszyfrowaniu określał ważną rolę użytkownika i administratora, umożliwiając pełną kontrolę nad aplikacją.

UWAGA: Inne techniki mogą pozwolić na odszyfrowanie zaszyfrowanych danych używanych przez aplikację. Wyrocznię szyfrującą „ujawniającą” można wykorzystać w celu uzyskania wartości zwykłego tekstu zaszyfrowanego tokena. Chociaż może to stanowić znaczącą lukę w zabezpieczeniach podczas odszyfrowywania hasła, odszyfrowanie tokena sesji nie zapewnia natychmiastowego sposobu na naruszenie bezpieczeństwa sesji innych użytkowników. Niemniej jednak odszyfrowany token zapewnia przydatny wgląd w strukturę zwykłego tekstu, co jest przydatne w przeprowadzaniu ukierunkowanego ataku polegającego na przerzucaniu bitów. Ataki kanału bocznego na dopełniające wyrocznie mogą zostać wykorzystane do naruszenia zaszyfrowanych tokenów.

KROKI HACKOWANIA

W wielu sytuacjach, w których używane są zaszyfrowane tokeny, rzeczywista użyteczność może zależeć od różnych czynników, w tym od przesunięcia granic bloków w stosunku do danych, które mają zostać zaatakowane, oraz tolerancji aplikacji na zmiany, które powodujesz w otaczającej strukturze zwykłego tekstu. Działając całkowicie na ślepo, skonstruowanie skutecznego ataku może wydawać się trudne, jednak w wielu sytuacjach jest to faktycznie możliwe.

1. O ile token sesji nie jest sam w sobie znaczący lub sekwencyjny, zawsze należy wziąć pod uwagę możliwość, że może on zostać zaszyfrowany. Często można stwierdzić, że używany jest szyfr blokowy, rejestrując kilka różnych nazw użytkowników i dodając za każdym razem jeden znak długości. Jeśli znajdziesz punkt, w którym dodanie jednego znaku powoduje skok długości tokena sesji o 8 lub 16 bajtów, prawdopodobnie używany jest szyfr blokowy. Możesz to potwierdzić, kontynuując dodawanie bajtów do swojej nazwy użytkownika i szukając tego samego skoku występującego 8 lub 16 bajtów później.

2. Luki w zabezpieczeniach EBC związane z manipulacją szyframi są zwykle trudne do zidentyfikowania i wykorzystania w kontekście czysto czarnej skrzynki. Możesz spróbować ślepo duplikować i przenosić

bloki tekstu zaszyfrowanego w swoim tokenie i sprawdzać, czy pozostajesz zalogowany do aplikacji w ramach własnego kontekstu użytkownika, kontekstu innego użytkownika, czy też żadnego.

3. Możesz przetestować luki w zabezpieczeniach manipulacji szyfrem CBC, przeprowadzając atak Burp Intruder na cały token, używając źródła ładunku „przerzucania bitów". Jeśli atak z odwracaniem bitów zidentyfikuje sekcję w tokenie, której manipulacja powoduje, że pozostajesz w prawidłowej sesji, ale jako inny lub nieistniejący użytkownik, przeprowadź bardziej skoncentrowany atak tylko na tej sekcji, próbując szerszego zakresu wartości w każda pozycja.

4. Podczas obu ataków monitoruj odpowiedzi aplikacji, aby zidentyfikować użytkownika powiązanego z twoją sesją po każdym żądaniu i spróbuj wykorzystać wszelkie możliwości eskalacji uprawnień, które mogą wynikać.

5. Jeśli twoje ataki nie powiodą się, ale od kroku 1 wynika, że dane wejściowe o zmiennej długości, które kontrolujesz, są włączane do tokena, powinieneś spróbować wygenerować serię tokenów, dodając po jednym znaku na raz, przynajmniej do rozmiaru używane bloki. Dla każdego wynikowego tokena należy ponownie wykonać kroki 2 i 3. Zwiększy to prawdopodobieństwo, że dane, które należy zmodyfikować, zostaną odpowiednio dopasowane do granic bloków, aby atak się powiódł.

Niedociągnięcia w obsłudze tokenów sesji

Bez względu na to, jak skuteczna jest aplikacja w zapewnianiu, że generowane przez nią tokeny sesji nie zawierają żadnych znaczących informacji i nie są podatne na analizę lub przewidywanie, jej mechanizm sesyjny będzie szeroko otwarty na atak, jeśli te tokeny nie będą traktowane ostrożnie po wygenerowaniu. Na przykład, jeśli tokeny zostaną ujawnione atakującemu w jakiś sposób, atakujący może przejąć sesje użytkownika, nawet jeśli przewidzenie tokenów jest niemożliwe. Niebezpieczna obsługa tokenów przez aplikację może narazić ją na atak na kilka sposobów.

POWSZECHNY MIT

„Nasz token jest zabezpieczony przed ujawnieniem stronom trzecim, ponieważ używamy protokołu SSL".

Właściwe użycie SSL z pewnością pomaga chronić tokeny sesji przed przechwyceniem. Jednak różne błędy mogą nadal powodować przesyłanie tokenów w postaci zwykłego tekstu, nawet jeśli obowiązuje protokół SSL. W celu uzyskania tokenów można wykorzystać różne bezpośrednie ataki na użytkowników końcowych.

POWSZECHNY MIT

„Nasz token jest generowany przez platformę przy użyciu dojrzałych, bezpiecznych technologii kryptograficznych, więc nie jest podatny na kompromisy".

Domyślnym zachowaniem serwera aplikacji jest często tworzenie sesyjnego pliku cookie, gdy użytkownik po raz pierwszy odwiedza witrynę, i utrzymywanie go przez cały czas interakcji użytkownika z witryną. Jak opisano w poniższych sekcjach, może to prowadzić do różnych luk w zabezpieczeniach w sposobie obsługi tokenu.

Ujawnianie Tokenów w Sieci

Ten obszar podatności powstaje, gdy token sesji jest przesyłany przez sieć w postaci niezaszyfrowanej, co umożliwia odpowiednio ustawionemu podsłuchującemu uzyskanie tokena, a tym samym podszywanie się pod uprawnionego użytkownika. Odpowiednie miejsca do podsłuchiwania obejmują lokalną sieć użytkownika, w ramach IT użytkownika

w ramach ISP użytkownika, w sieci szkieletowej Internetu, w ramach ISP aplikacji oraz w dziale IT organizacji hostującej aplikację. W każdym przypadku dotyczy to zarówno upoważnionego personelu odpowiedniej organizacji, jak i zewnętrznych napastników, którzy włamali się do danej infrastruktury. W najprostszym przypadku, gdy aplikacja używa do komunikacji niezaszyfrowanego połączenia HTTP, osoba atakująca może przechwycić wszystkie dane przesyłane między klientem a serwerem, w tym dane logowania, dane osobowe, szczegóły płatności itd. W tej sytuacji atak na sesję użytkownika jest często niepotrzebny, ponieważ osoba atakująca może już wyświetlać uprzywilejowane informacje i logować się przy użyciu przechwyconych danych uwierzytelniających w celu wykonywania innych złośliwych działań. Jednak nadal mogą wystąpić przypadki, w których sesja użytkownika jest głównym celem. Na przykład, jeśli przechwycone dane uwierzytelniające są niewystarczające do wykonania drugiego logowania (np. w aplikacji bankowej mogą zawierać numer wyświetlany na zmieniającym się fizycznym tokenie lub określone cyfry z PIN-u użytkownika), atakujący może potrzebować przejąć kontrolę nad podsłuchiwaną sesją w celu wykonania dowolnych działań. Lub jeśli logowanie jest ściśle kontrolowane, a użytkownik jest powiadamiany o każdym udanym logowaniu, osoba atakująca może chcieć uniknąć wykonywania własnego logowania, aby zachować jak największą tajemnicę. W innych przypadkach aplikacja może używać protokołu HTTPS do ochrony kluczowej komunikacji klient-serwer, ale nadal może być narażona na przechwycenie tokenów sesji w sieci. Ta słabość może wystąpić na różne sposoby, z których wiele może wystąpić w szczególności, gdy pliki cookie HTTP są używane jako mechanizm transmisji tokenów sesji:

* Niektóre aplikacje wybierają użycie protokołu HTTPS w celu ochrony danych uwierzytelniających użytkownika podczas logowania, ale następnie powracają do protokołu HTTP na pozostałą część sesji użytkownika. W ten sposób zachowuje się wiele aplikacji poczty internetowej. W tej sytuacji osoba podsłuchująca nie może przechwycić danych uwierzytelniających użytkownika, ale nadal może przechwycić token sesji. Narzędzie Firesheep, wydane jako wtyczka do przeglądarki Firefox, ułatwia ten proces.

* Niektóre aplikacje używają protokołu HTTP do wstępnie uwierzytelnionych obszarów witryny, takich jak strona główna witryny, ale przełączają się na HTTPS począwszy od strony logowania. Jednak w wielu przypadkach użytkownik otrzymuje token sesji na pierwszej odwiedzonej stronie, a token ten nie jest modyfikowany podczas logowania użytkownika. Sesja użytkownika, która pierwotnie nie była uwierzytelniona, jest aktualizowana do sesji uwierzytelnionej po zalogowaniu. W takiej sytuacji podsłuchujący może przechwycić token użytkownika przed zalogowaniem, poczekać, aż komunikacja użytkownika przełączy się na HTTPS, wskazując, że użytkownik się loguje, a następnie spróbować uzyskać dostęp do chronionej strony (takiej jak Moje konto) za pomocą tego tokena.

* Nawet jeśli aplikacja wystawi nowy token po pomyślnym zalogowaniu i użyje protokołu HTTPS od strony logowania, token dla uwierzytelnionej sesji użytkownika może nadal zostać ujawniony. Może się tak zdarzyć, jeśli użytkownik ponownie odwiedzi stronę wstępnego uwierzytelniania (taką jak Pomoc lub Informacje), korzystając z łączy w obszarze uwierzytelniania, używając przycisku Wstecz lub bezpośrednio wpisując adres URL.

*W odmianie poprzedniego przypadku aplikacja może próbować przełączyć się na HTTPS, gdy użytkownik kliknie link Logowanie. Może jednak nadal akceptować logowanie przez HTTP, jeśli użytkownik odpowiednio zmodyfikuje adres URL. W tej sytuacji atakujący o odpowiedniej pozycji może zmodyfikować strony zwracane w obszarach witryny, które zostały wstępnie uwierzytelnione, tak aby łącze logowania wskazywało stronę HTTP. Nawet jeśli aplikacja wystawi nowy token sesji po pomyślnym zalogowaniu, atakujący może nadal przechwycić ten token, jeśli pomyślnie obniży wersję połączenia użytkownika do HTTP.

* Niektóre aplikacje używają protokołu HTTP dla całej zawartości statycznej w aplikacji, takiej jak obrazy, skrypty, arkusze stylów i szablony stron. Takie zachowanie jest często wskazywane przez ostrzeżenie w przeglądarce użytkownika, jak pokazano na rysunku.

Gdy przeglądarka wyświetla to ostrzeżenie, oznacza to, że pobrała już odpowiedni element przez HTTP, więc token sesji został już przesłany. Celem ostrzeżenia przeglądarki jest umożliwienie użytkownikowi odmowy przetwarzania danych odpowiedzi otrzymanych przez HTTP, które mogą zostać skażone. Jak opisano wcześniej, osoba atakująca może przechwycić token sesji użytkownika, gdy przeglądarka użytkownika uzyskuje dostęp do zasobu za pośrednictwem protokołu HTTP, i użyć tego tokena, aby uzyskać dostęp do chronionych, niestatycznych obszarów witryny za pośrednictwem protokołu HTTPS.

* Nawet jeśli aplikacja korzysta z protokołu HTTPS na każdej stronie, w tym na nieuwierzytelnionych obszarach witryny i zawartości statycznej, nadal mogą wystąpić okoliczności, w których tokeny użytkowników są przesyłane przez protokół HTTP. Jeśli osoba atakująca może w jakiś sposób skłonić użytkownika do wysłania żądania przez HTTP (albo do usługi HTTP na tym samym serwerze, jeśli jest uruchomiona, albo do http://serwer:443/w przeciwnym razie), jego token może zostać przesłany. Środki, za pomocą których osoba atakująca może podjąć taką próbę, obejmują wysłanie użytkownikowi adresu URL w wiadomości e-mail lub wiadomości błyskawicznej, umieszczenie linków do automatycznego ładowania w witrynie internetowej kontrolowanej przez osobę atakującą lub użycie klikalnych banerów reklamowych.

KROKI HACKOWANIA

1. Przejdź przez aplikację w normalny sposób od pierwszego wejścia (adres URL „początkowy”), przez proces logowania, a następnie przez wszystkie funkcje aplikacji. Prowadź rejestr każdego odwiedzanego adresu URL i notuj każdy przypadek otrzymania nowego tokena sesji. Zwróć szczególną uwagę na funkcje logowania i przejścia między komunikacją HTTP i HTTPS. Można to osiągnąć ręcznie za pomocą sniffera sieciowego, takiego jak Wireshark, lub częściowo zautomatyzować, korzystając z funkcji rejestrowania przechwytującego serwera proxy.

2. Jeśli jako mechanizm transmisji tokenów sesyjnych używane są pliki cookie HTTP, sprawdź, czy ustawiona jest bezpieczna flaga, która uniemożliwia ich przesyłanie przez niezaszyfrowane połączenia.

3. Ustal, czy podczas normalnego użytkowania aplikacji tokeny sesyjne są kiedykolwiek przesyłane przez połączenie nieszyfrowane. Jeśli tak, należy je uznać za podatne na przechwycenie.

4. Jeśli strona startowa korzysta z protokołu HTTP, a aplikacja przełącza się na HTTPS dla obszarów logowania i uwierzytelniania witryny, sprawdź, czy po zalogowaniu jest wydawany nowy token lub czy token przesłany na etapie HTTP jest nadal używany do śledzenia uwierzytelnioną sesję użytkownika. Sprawdź również, czy aplikacja będzie akceptować logowanie przez HTTP, jeśli adres URL logowania zostanie odpowiednio zmodyfikowany.

5. Nawet jeśli aplikacja korzysta z protokołu HTTPS dla każdej strony, sprawdź, czy serwer również nasłuchuje na porcie 80, uruchamiając jakąkolwiek usługę lub treść. Jeśli tak, odwiedź dowolny adres URL HTTP bezpośrednio z uwierzytelnionej sesji i sprawdź, czy token sesji jest przesyłany.

6. W przypadkach, gdy token uwierzytelnionej sesji jest przesyłany do serwera przez HTTP, sprawdź, czy token nadal jest ważny, czy też jest natychmiast przerywany przez serwer.

Ujawnianie Tokenów w Logach

Poza transmisją zwykłego tekstu tokenów sesji w komunikacji sieciowej, najczęstszym miejscem, w którym tokeny są po prostu ujawniane nieautoryzowanemu widokowi, są różnego rodzaju dzienniki systemowe. Chociaż jest to rzadsze zjawisko, konsekwencje tego rodzaju ujawnienia są zwykle poważniejsze. Te dzienniki mogą być przeglądane przez znacznie szerszy krąg potencjalnych atakujących, a nie tylko przez kogoś, kto ma odpowiednią pozycję do podsłuchiwania sieci. Wiele aplikacji zapewnia funkcjonalność dla administratorów i innego personelu pomocniczego do monitorowania i kontrolowania aspektów stanu działania aplikacji, w tym sesje użytkowników. Na przykład pracownik pomocy technicznej pomagający użytkownikowi, który ma problemy, może poprosić o jego nazwę użytkownika, zlokalizować jego bieżącą sesję za pomocą listy lub funkcji wyszukiwania oraz wyświetlić odpowiednie szczegóły dotyczące sesji. Lub administrator może przeglądać dziennik ostatnich sesji w trakcie badania naruszenia bezpieczeństwa. Często tego rodzaju funkcje monitorowania i kontroli ujawniają rzeczywisty token sesji powiązany z każdą sesją. I często funkcjonalność jest słabo chroniona, umożliwiając nieautoryzowanym użytkownikom dostęp do listy bieżących tokenów sesji, a tym samym przejmowanie sesji wszystkich użytkowników aplikacji. Inną główną przyczyną pojawiania się tokenów sesji w dziennikach systemowych jest sytuacja, w której aplikacja używa ciągu zapytania adresu URL jako mechanizmu przesyłania tokenów, w przeciwieństwie do używania plików cookie HTTP lub treści żądań POST. Na przykład Googling inurl:jsessionid identyfikuje tysiące aplikacji, które przesyłają token sesji platformy Java (o nazwie jsessionid) w adresie URL:

http://www.webjunction.org/do/Navigation;jsessionid=

F27ED2A6AAE4C6DA409A3044E79B8B48?kategoria=327

Kiedy aplikacje przesyłają swoje tokeny sesji w ten sposób, prawdopodobnie ich tokeny sesji pojawią się w różnych dziennikach systemowych, do których dostęp mogą mieć osoby nieupoważnione:

* Dzienniki przeglądarki użytkowników

* Dzienniki serwera WWW

* Dzienniki serwerów proxy korporacyjnych lub ISP

* Dzienniki wszelkich odwrotnych serwerów proxy używanych w środowisku hostingowym aplikacji

* Dzienniki Referer wszelkich serwerów, które odwiedzają użytkownicy aplikacji, korzystając z linków poza witryną

Niektóre z tych luk pojawiają się nawet wtedy, gdy w całej aplikacji używany jest protokół HTTPS.

Ostatni opisany przypadek przedstawia atakującemu wysoce skuteczny sposób przechwytywania tokenów sesji w niektórych aplikacjach. Na przykład, jeśli aplikacja poczty internetowej przesyła tokeny sesji w adresie URL, osoba atakująca może wysłać do użytkowników aplikacji wiadomość e-mail zawierającą łącze do kontrolowanego przez siebie serwera WWW. Jeśli jakikolwiek użytkownik uzyska dostęp do łącza (ponieważ je kliknie lub ponieważ jej przeglądarka załaduje obrazy zawarte w

wiadomości e-mail w formacie HTML), osoba atakująca otrzyma w czasie rzeczywistym token sesji użytkownika. Atakujący może uruchomić prosty skrypt na swoim serwerze, aby przejąć sesję każdego otrzymanego tokena i wykonać złośliwe działania, takie jak wysyłanie spamu, zbieranie danych osobowych lub zmiana haseł.

NOTATKA : Bieżące wersje przeglądarki Internet Explorer nie zawierają nagłówka Referer podczas korzystania z łączy poza witryną zawartych na stronie, do której uzyskano dostęp za pośrednictwem protokołu HTTPS. W tej sytuacji Firefox zawiera nagłówek Referer, pod warunkiem, że łącze poza witryną jest również dostępne przez HTTPS, nawet jeśli należy do innej domeny. W związku z tym wrażliwe dane umieszczone w adresach URL są narażone na wyciek w dziennikach odsyłaczy, nawet jeśli używany jest protokół SSL.

KROKI HACKOWANIA

1. Zidentyfikuj wszystkie funkcje w aplikacji i zlokalizuj wszelkie funkcje rejestrowania lub monitorowania, w których można przeglądać tokeny sesji. Sprawdź, kto może uzyskać dostęp do tej funkcji — na przykład administratorzy, każdy uwierzytelniony użytkownik lub dowolny użytkownik anonimowy.

2. Zidentyfikuj wszystkie instancje w aplikacji, w których tokeny sesji są przesyłane w adresie URL. Może się zdarzyć, że tokeny są generalnie przesyłane w bardziej bezpieczny sposób, ale programiści używali adresu URL w określonych przypadkach, aby obejść określone trudności. Na przykład takie zachowanie jest często obserwowane, gdy aplikacja internetowa łączy się z systemem zewnętrznym.

3. Jeśli tokeny sesji są przesyłane w adresach URL, spróbuj znaleźć jakąkolwiek funkcjonalność aplikacji, która umożliwia wstrzykiwanie dowolnych linków poza witrynę do stron przeglądanych przez innych użytkowników. Przykłady obejmują funkcjonalność implementującą tablicę ogłoszeń, opinie o witrynie, pytania i odpowiedzi i tak dalej. Jeśli tak, prześlij łącza do kontrolowanego przez siebie serwera internetowego i poczekaj, aby zobaczyć, czy w dziennikach osób odsyłających są odbierane tokeny sesji użytkowników.

4. Jeśli zostaną przechwycone jakiekolwiek tokeny sesji, spróbuj przejąć sesje użytkownika, używając aplikacji w normalny sposób, ale zastępując przechwycony token własnym. Możesz to zrobić, przechwytując następną odpowiedź z serwera i dodając własny nagłówek Set-Cookie z przechwyconą wartością pliku cookie. W Burp możesz zastosować pojedynczą konfigurację obejmującą cały pakiet, która ustawia określony plik cookie we wszystkich żądaniach do aplikacji docelowej, aby umożliwić łatwe przełączanie między różnymi kontekstami sesji podczas testowania.

6. Jeśli przechwycona zostanie duża liczba tokenów, a przechwycenie sesji umożliwi Ci dostęp do poufnych danych, takich jak dane osobowe, informacje o płatnościach lub hasła użytkownika, możesz użyć zautomatyzowanych technik opisanych w Części 14 w celu zebrania wszystkich pożądanych danych należących do innych użytkowników aplikacji.

Wrażliwe mapowanie tokenów na sesje

Różne typowe luki w mechanizmach zarządzania sesjami wynikają ze słabości w sposobie, w jaki aplikacja odwzorowuje tworzenie i przetwarzanie tokenów sesji na sesje poszczególnych użytkowników. Najprostszą słabością jest umożliwienie jednoczesnego przypisania wielu ważnych tokenów do tego samego konta użytkownika. W praktycznie każdej aplikacji nie ma uzasadnionego powodu, dla którego jakikolwiek użytkownik miałby mieć aktywną więcej niż jedną sesję w tym samym czasie. Oczywiście dość często zdarza się, że użytkownik porzuca aktywną sesję i rozpoczyna nową — na przykład dlatego, że zamyka okno przeglądarki lub przenosi się na inny komputer. Ale jeśli wydaje

się, że użytkownik korzysta jednocześnie z dwóch różnych sesji, zwykle oznacza to, że nastąpiło naruszenie bezpieczeństwa: albo użytkownik ujawnił swoje dane uwierzytelniające innej stronie, albo osoba atakująca uzyskała swoje dane uwierzytelniające w inny sposób. W obu przypadkach zezwalanie na sesje równoczesne jest niepożądane, ponieważ umożliwia użytkownikom kontynuowanie niepożądanych praktyk bez niedogodności, a atakującemu umożliwia wykorzystanie przechwyconych danych uwierzytelniających bez ryzyka wykrycia. Powiązaną, ale wyraźną słabością jest używanie przez aplikacje „statycznych" tokenów. Wyglądają jak tokeny sesyjne i początkowo mogą wydawać się działać tak, jak one, ale w rzeczywistości nimi nie są. W tych aplikacjach każdemu użytkownikowi przypisywany jest token i ten sam token jest ponownie wydawany użytkownikowi przy każdym logowaniu. Aplikacja zawsze akceptuje token jako ważny, niezależnie od tego, czy użytkownik był ostatnio zalogowany i czy został mu wydany. Aplikacje takie jak ta naprawdę wiążą się z nieporozumieniem co do całej koncepcji sesji i korzyści, jakie zapewnia w zakresie zarządzania i kontrolowania dostępu do aplikacji. Czasami aplikacje działają w ten sposób, aby wdrożyć źle zaprojektowaną funkcję „zapamiętaj mnie", a token statyczny jest odpowiednio przechowywany w trwałym pliku cookie. Czasami same tokeny są podatne na ataki predykcyjne, przez co luka jest znacznie poważniejsza. Zamiast narażać sesje aktualnie zalogowanych użytkowników, udany atak na zawsze narusza konta wszystkich zarejestrowanych użytkowników. Sporadycznie obserwowane są również inne rodzaje dziwnego zachowania aplikacji, które wskazują na fundamentalny defekt w relacji między tokenami a sesjami. Jednym z przykładów jest sytuacja, w której znaczący token jest konstruowany na podstawie nazwy użytkownika i losowego składnika. Weźmy na przykład token:

dXNlcj1kYWY7cjE9MTMwOTQxODEyMTM0NTkwMTI=

które dekoduje Base64 do:

użytkownik=daf;r1=13094181213459012

Po obszernej analizie składowej r1 możemy stwierdzić, że nie da się tego przewidzieć na podstawie próby wartości. Jeśli jednak logika przetwarzania sesji w aplikacji jest nieprawidłowa, może się zdarzyć, że osoba atakująca musi po prostu przesłać dowolną prawidłową wartość jako r1 i dowolną prawidłową wartość jako użytkownik, aby uzyskać dostęp do sesji w kontekście bezpieczeństwa określonego użytkownika. Zasadniczo jest to luka w zabezpieczeniach kontroli dostępu, ponieważ decyzje dotyczące dostępu są podejmowane na podstawie danych dostarczonych przez użytkownika poza sesją. Powstaje, ponieważ aplikacja skutecznie wykorzystuje tokeny sesji, aby wskazać, że osoba żądająca nawiązała jakąś ważną sesję z aplikacją. Jednak kontekst użytkownika, w którym ta sesja jest przetwarzana, nie jest integralną właściwością samej sesji, ale jest określany na żądanie za pomocą innych środków. W tym przypadku środki te mogą być bezpośrednio kontrolowane przez wnioskodawcę.

KROKI HACKOWANIA

1. Zaloguj się dwukrotnie do aplikacji na to samo konto użytkownika, z różnych procesów przeglądarki lub z różnych komputerów. Określ, czy obie sesje pozostają aktywne jednocześnie. Jeśli tak, aplikacja obsługuje sesje równoległe, umożliwiając atakującemu, który naruszył dane uwierzytelniające innego użytkownika, wykorzystanie ich bez ryzyka wykrycia.

2. Zaloguj się i wyloguj kilka razy przy użyciu tego samego konta użytkownika, z różnych procesów przeglądarki lub z różnych komputerów. Określ, czy przy każdym logowaniu jest wydawany nowy token sesji, czy też przy każdym logowaniu jest wydawany ten sam token. Jeśli tak się dzieje, aplikacja nie korzysta z odpowiednich sesji.

3. Jeśli tokeny wydają się zawierać jakąkolwiek strukturę i znaczenie, spróbuj oddzielić elementy, które mogą identyfikować użytkownika, od tych, które wydają się nieodgadnione. Spróbuj zmodyfikować wszelkie komponenty tokena związane z użytkownikiem, aby odnosiły się do innych znanych użytkowników aplikacji i sprawdź, czy otrzymany token jest akceptowany przez aplikację i umożliwia podszywanie się pod tego użytkownika.

Wrażliwe zakończenie sesji

Właściwe zakończenie sesji jest ważne z dwóch powodów. Po pierwsze, utrzymywanie tak krótkiego czasu życia sesji, jak to konieczne, zmniejsza okno możliwości, w którym osoba atakująca może przechwycić, odgadnąć lub niewłaściwie użyć ważnego tokena sesji. Po drugie, zapewnia użytkownikom możliwość unieważnienia istniejącej sesji, gdy już jej nie potrzebują. Umożliwia im to dalsze skrócenie tego okna i wzięcie części odpowiedzialności za zabezpieczenie sesji we współdzielonym środowisku komputerowym. Główne słabości funkcji zakończenia sesji polegają na niespełnianiu tych dwóch kluczowych celów. Niektóre aplikacje nie wymuszają efektywnego wygaśnięcia sesji. Raz utworzona sesja może pozostać ważna przez wiele dni od otrzymania ostatniego żądania, zanim serwer ostatecznie wygaśnie sesję. Jeśli tokeny są podatne na jakąś lukę w sekwencjonowaniu, która jest szczególnie trudna do wykorzystania (na przykład 100 000 zgadnięć dla każdego zidentyfikowanego ważnego tokena), osoba atakująca może nadal być w stanie przechwycić tokeny każdego użytkownika, który uzyskał dostęp do aplikacji w niedawnej przeszłości. Niektóre aplikacje nie zapewniają skutecznej funkcji wylogowania:

* W niektórych przypadkach funkcja wylogowania po prostu nie jest zaimplementowana. Użytkownicy nie mają możliwości spowodowania przez aplikację unieważnienia ich sesji.

* W niektórych przypadkach funkcja wylogowania w rzeczywistości nie powoduje unieważnienia sesji przez serwer. Serwer usuwa token z przeglądarki użytkownika (na przykład wydając instrukcję Set-Cookie, aby wyczyścić token). Jeśli jednak użytkownik nadal przesyła token, serwer nadal go akceptuje.

* W najgorszym przypadku, gdy użytkownik kliknie Wyloguj, serwer nie przekazuje tego faktu, więc serwer nie wykonuje żadnej akcji. Zamiast tego wykonywany jest skrypt po stronie klienta, który usuwa plik cookie użytkownika, co oznacza, że kolejne żądania powodują powrót użytkownika do strony logowania. Osoba atakująca, która uzyska dostęp do tego pliku cookie, może wykorzystać sesję tak, jakby użytkownik nigdy się nie wylogował.

Niektóre aplikacje, które nie używają uwierzytelniania, nadal zawierają funkcje, które umożliwiają użytkownikom gromadzenie poufnych danych w ramach ich sesji (na przykład aplikacja zakupowa). Jednak zazwyczaj nie zapewniają one żadnego odpowiednika funkcji wylogowania umożliwiającej użytkownikom zakończenie sesji

KROKI HACKOWANIA

1. Nie wpadnij w pułapkę sprawdzania działań, które aplikacja wykonuje na tokenie po stronie klienta (takich jak unieważnianie ciasteczek przez nową instrukcję Set-Cookie, skrypt po stronie klienta czy atrybut czasu wygaśnięcia). Jeśli chodzi o zakończenie sesji, niewiele zależy od tego, co stanie się z tokenem w przeglądarce klienta. Zamiast tego sprawdź, czy wygaśnięcie sesji jest zaimplementowane po stronie serwera:

A. Zaloguj się do aplikacji, aby uzyskać ważny token sesji.

B. Poczekaj przez pewien czas bez użycia tego tokena, a następnie prześlij prośbę o dostęp do chronionej strony (np. „moje dane”) za pomocą tokena.

C. Jeśli strona wyświetla się normalnie, token jest nadal aktywny.

D. Skorzystaj z metody prób i błędów, aby określić, jak długi jest limit czasu wygaśnięcia sesji lub czy token może być nadal używany kilka dni po ostatnim żądaniu jego użycia. Burp Intruder można skonfigurować tak, aby zwiększał odstęp czasu między kolejnymi żądaniami automatyzacji tego zadania.

2. Ustal, czy istnieje funkcja wylogowania i czy jest ona dostępna dla użytkowników w widocznym miejscu. W przeciwnym razie użytkownicy są bardziej narażeni, ponieważ nie mają możliwości spowodowania unieważnienia sesji przez aplikację.

3. Jeśli dostępna jest funkcja wylogowania, przetestuj jej skuteczność. Po wylogowaniu spróbuj ponownie użyć starego tokena i sprawdź, czy jest on nadal ważny. Jeśli tak, użytkownicy pozostają narażeni na niektóre ataki polegające na przejęciu sesji nawet po „wylogowaniu". Możesz użyć Burp Suite, aby to przetestować, wybierając ostatnie żądanie zależne od sesji z historii proxy i wysyłając je do Burp Repeater w celu ponownego wysłania po wylogowaniu z aplikacji.

Narażenie klienta na przejęcie tokena

Osoba atakująca może atakować innych użytkowników aplikacji w celu przechwycenia lub niewłaściwego wykorzystania tokena sesji ofiary na różne sposoby:

* Oczywistym celem ataków typu cross-site scripting jest wysłanie zapytania do plików cookie użytkownika w celu uzyskania tokena sesji, który można następnie przesłać na dowolny serwer kontrolowany przez atakującego.

* Różne inne ataki na użytkowników mogą zostać wykorzystane do przejęcia sesji użytkownika na różne sposoby. W przypadku luk związanych z utrwalaniem sesji osoba atakująca przekazuje użytkownikowi znany token sesji, czeka na zalogowanie się, a następnie przejmuje kontrolę nad sesją. W przypadku ataków polegających na fałszowaniu żądań między witrynami osoba atakująca wysyła spreparowane żądanie do aplikacji z kontrolowanej przez siebie witryny internetowej i wykorzystuje fakt, że przeglądarka użytkownika automatycznie przesyła jej bieżący plik cookie wraz z tym żądaniem.

KROKI HACKOWANIA

1. Zidentyfikuj wszelkie luki w zabezpieczeniach aplikacji związane z atakami typu cross-site scripting i określ, czy można je wykorzystać do przechwytywania tokenów sesji innych użytkowników.

2. Jeśli aplikacja wystawia tokeny sesyjne nieuwierzytelnionym użytkownikom, uzyskaj token i wykonaj logowanie. Jeśli aplikacja nie wystawi nowego tokena po pomyślnym zalogowaniu, jest narażona na utrwalanie sesji.

3. Nawet jeśli aplikacja nie wydaje tokenów sesji nieuwierzytelnionym użytkownikom, uzyskaj token logując się, a następnie wróć do strony logowania. Jeśli aplikacja chce zwrócić tę stronę, mimo że jesteś już uwierzytelniony, prześlij kolejny login jako inny użytkownik używający tego samego tokena. Jeśli aplikacja nie wystawi nowego tokena po drugim logowaniu, jest podatna na utrwalanie sesji.

4. Zidentyfikuj format tokenów sesji używanych przez aplikację. Zmodyfikuj swój token na wymyśloną wartość, która jest prawidłowo utworzona, i spróbuj się zalogować. Jeśli aplikacja umożliwia utworzenie uwierzytelnionej sesji przy użyciu wymyślonego tokena, jest podatna na utrwalanie sesji.

5. Jeśli aplikacja nie obsługuje logowania, ale przetwarza poufne dane użytkownika (takie jak dane osobowe i dane dotyczące płatności) i umożliwia wyświetlenie ich po przesłaniu (np. na stronie „zweryfikuj moje zamówienie"), wykonaj trzy poprzednie testy w stosunku do stron wyświetlających

dane wrażliwe. Jeśli token ustawiony podczas anonimowego korzystania z aplikacji może zostać później użyty do pobrania poufnych informacji o użytkowniku, aplikacja jest podatna na utrwalanie sesji.

6. Jeśli aplikacja używa plików cookie HTTP do przesyłania tokenów sesji, może być narażona na fałszerstwo żądań między witrynami (XSRF). Najpierw zaloguj się do aplikacji. Następnie potwierdź, że żądanie skierowane do aplikacji, ale pochodzące ze strony innej aplikacji, skutkuje przesłaniem tokena użytkownika. (Przesłanie to musi zostać wykonane z okna tego samego procesu przeglądarki, który został ustawiony w celu zalogowania się do docelowej aplikacji.) Spróbuj zidentyfikować wszelkie wrażliwe funkcje aplikacji, których parametry osoba atakująca może z góry określić, i wykorzystać to do przeprowadzenia nieautoryzowanych działań w kontekście bezpieczeństwa użytkownika docelowego.

Liberalny zakres plików cookie

Zwykłe proste podsumowanie działania plików cookie polega na tym, że serwer wysyła plik cookie za pomocą nagłówka odpowiedzi HTTP Set-cookie, a następnie przeglądarka ponownie przesyła ten plik cookie w kolejnych żądaniach do tego samego serwera za pomocą nagłówka Cookie. W rzeczywistości sprawy są bardziej subtelne niż to. Mechanizm plików cookie umożliwia serwerowi określenie zarówno domeny, jak i ścieżki adresu URL, do którego każdy plik cookie zostanie przesłany ponownie. W tym celu wykorzystuje atrybuty domeny i ścieżki, które mogą być zawarte w instrukcji Set-cookie.

Ograniczenia domeny plików cookie

Kiedy aplikacja rezydująca pod adresem foo.wahh-app.com ustawia plik cookie, przeglądarka domyślnie ponownie przesyła plik cookie we wszystkich kolejnych żądaniach do foo.wahh-app .com, a także do wszelkich subdomen, takich jak admin.foo.wahh- app.com. Nie przesyła pliku cookie do żadnych innych domen, w tym domeny nadrzędnej wahh-app.com i żadnych innych subdomen rodzica, takich jak bar.wahh-app.com. Serwer może zastąpić to domyślne zachowanie, umieszczając atrybut domeny w instrukcji Set-cookie. Załóżmy na przykład, że aplikacja pod adresem foo.wahh-app.com zwraca następujący nagłówek HTTP:

Set-cookie: sessionId=19284710; domain=wahh-app.com;

Następnie przeglądarka ponownie przesyła ten plik cookie do wszystkich subdomen wahh-app.com, w tym bar.wahh-app.com.

UWAGA: Serwer nie może określić dowolnej domeny za pomocą tego atrybutu. Po pierwsze, określona domena musi być albo tą samą domeną, w której działa aplikacja, albo domeną, która jest jej rodzicem (bezpośrednio lub w pewnym oddaleniu). Po drugie, określona domena nie może być domeną najwyższego poziomu, taką jak .com lub .co.uk, ponieważ umożliwiłoby to złośliwemu serwerowi ustawienie dowolnych plików cookie w dowolnej innej domenie. Jeśli serwer naruszy jedną z tych zasad, przeglądarka po prostu zignoruje instrukcję Set-cookie.

Jeśli aplikacja ustawi zakres domeny pliku cookie jako nadmiernie liberalny, może to narazić aplikację na różne luki w zabezpieczeniach.

Rozważmy na przykład aplikację do blogowania, która umożliwia użytkownikom rejestrację, logowanie, pisanie postów na blogu i czytanie blogów innych osób. Główna aplikacja znajduje się w domenie wahh-blogs.com. Gdy użytkownicy logują się do aplikacji, otrzymują token sesji w pliku cookie, którego zakres obejmuje tę domenę. Każdy użytkownik może tworzyć blogi, do których dostęp uzyskuje za pośrednictwem nowej subdomeny poprzedzonej nazwą użytkownika:

herman.wahh-blogs.com

solero.wahh-blogs.com

Ponieważ pliki cookie są automatycznie przesyłane ponownie do każdej subdomeny w swoim zakresie, gdy zalogowany użytkownik przegląda blogi innych użytkowników, jego token sesji jest wysyłany wraz z jego żądaniami. Jeśli autorzy blogów mogą umieszczać dowolny kod JavaScript w swoich własnych blogach (jak to zwykle ma miejsce w rzeczywistych aplikacjach blogowych), złośliwy bloger może ukraść tokeny sesji innych użytkowników w taki sam sposób, jak ma to miejsce w przechowywanych atak typu site scripting. Problem powstaje, ponieważ blogi tworzone przez użytkowników są tworzone jako subdomeny głównej aplikacji, która obsługuje uwierzytelnianie i zarządzanie sesjami. W plikach cookie HTTP nie ma możliwości, aby aplikacja zapobiegała ponownemu przesyłaniu plików cookie wydanych przez domenę główną do jej subdomen. Rozwiązaniem jest użycie innej nazwy domeny dla głównej aplikacji (na przykład www.wahh-blogs.com) i ograniczenie domeny plików cookie tokenów sesji do tej w pełni kwalifikowanej nazwy. Sesyjny plik cookie nie zostanie przesłany, gdy zalogowany użytkownik przegląda blogi innych użytkowników. Inna wersja tej luki powstaje, gdy aplikacja jawnie ustawia zakres domeny swoich plików cookie na domenę nadrzędną. Załóżmy na przykład, że aplikacja o krytycznym znaczeniu dla bezpieczeństwa znajduje się w domenie wrażliwa aplikacja .wahh-organizacja.com. Kiedy ustawia pliki cookie, wyraźnie liberalizuje ich działanie zakresu domeny, w następujący sposób:

Set-cookie: sessionId=12df098ad809a5219; domain=wahh-organization.com

Konsekwencją tego jest to, że pliki cookie tokenów sesyjnych wrażliwej aplikacji zostaną przesłane, gdy użytkownik odwiedzi każdą subdomenę używaną przez wahh-organization .com, w tym:

www.wahh-organizacja.com

testapp.wahh-organization.com

Chociaż wszystkie te inne aplikacje mogą należeć do tej samej organizacji co poufna aplikacja, niepożądane jest przesyłanie plików cookie poufnej aplikacji do innych aplikacji z kilku powodów:

* Personel odpowiedzialny za inne aplikacje może mieć inny poziom zaufania niż personel odpowiedzialny za poufną aplikację.

* Inne aplikacje mogą zawierać funkcjonalność umożliwiającą stronom trzecim uzyskanie wartości plików cookie przesłanych do aplikacji, jak w poprzednim przykładzie blogowania.

* Inne aplikacje mogły nie zostać poddane tym samym standardom bezpieczeństwa lub testom, co aplikacja wrażliwa (ponieważ są mniej ważne, nie obsługują danych wrażliwych lub zostały stworzone wyłącznie w celach testowych). Wiele rodzajów luk, które mogą występować w tych aplikacjach (na przykład luki w zabezpieczeniach związane z atakami typu cross-site scripting) może nie mieć znaczenia dla poziomu bezpieczeństwa tych aplikacji. Mogą jednak umożliwić zewnętrznemu atakującemu wykorzystanie niezabezpieczonej aplikacji do przechwycenia tokenów sesji utworzonych przez wrażliwą aplikację.

UWAGA: Oparta na domenach segregacja plików cookie nie jest tak ścisła, jak ogólnie polityka sameorigin. Oprócz problemów już opisanych w obsłudze nazw hostów, przeglądarki ignorują zarówno protokół, jak i numer portu podczas określania zakresu plików cookie. Jeśli aplikacja ma wspólną nazwę hosta z niezaufaną aplikacją i polega na różnicy w protokole lub numerze portu w celu oddzielenia się, luźniejsza obsługa plików cookie może podważyć tę segregację. Wszelkie pliki cookie wydawane przez aplikację będą dostępne dla niezaufanej aplikacji, która udostępnia swoją nazwę hosta.

KROKI HACKOWANIA

Przejrzyj wszystkie pliki cookie wysłane przez aplikację i sprawdź, czy atrybuty domeny używane do kontrolowania zakresu plików cookie.

1. Jeśli aplikacja wyraźnie zliberalizuje zakres swoich plików cookie do domeny nadrzędnej, może narazić się na ataki za pośrednictwem innych aplikacji internetowych.

2. Jeśli aplikacja ustawia zakres domeny swoich plików cookie na własną nazwę domeny (lub nie określa atrybutu domeny), może nadal być narażona na aplikacje lub funkcje dostępne za pośrednictwem subdomen. Zidentyfikuj wszystkie możliwe nazwy domen, które będą otrzymywać pliki cookie wysyłane przez aplikację. Ustal, czy za pośrednictwem tych nazw domen jest dostępna jakakolwiek inna aplikacja internetowa lub funkcja, którą możesz wykorzystać do uzyskania plików cookie wydanych użytkownikom aplikacji docelowej.

Ograniczenia ścieżki plików cookie

Gdy aplikacja rezydująca pod adresem /apps/secure/foo-app/index.jsp ustawia plik cookie, przeglądarka domyślnie ponownie przesyła plik cookie we wszystkich kolejnych żądaniach do ścieżki /apps/secure/foo-app/, a także do dowolnych podkatalogów. Nie przesyła pliku cookie do katalogu nadrzędnego ani do żadnych innych ścieżek katalogów istniejących na serwerze. Podobnie jak w przypadku ograniczeń zakresu plików cookie opartych na domenie, serwer może zastąpić to domyślne zachowanie, umieszczając atrybut ścieżki w instrukcji Set-cookie. Na przykład, jeśli aplikacja zwróci następujący nagłówek HTTP:

Set-cookie: sessionId=187ab023e09c00a881a; path=/apps/;

przeglądarka ponownie przesyła ten plik cookie do wszystkich podkatalogów ścieżki /apps/. W przeciwieństwie do określania zakresu plików cookie na podstawie domeny, to ograniczenie oparte na ścieżce jest znacznie bardziej rygorystyczne niż to, co jest narzucone przez zasady tego samego pochodzenia. W związku z tym jest prawie całkowicie nieskuteczny, jeśli jest używany jako mechanizm bezpieczeństwa do obrony przed niezaufanymi aplikacjami hostowanymi w tej samej domenie. Kod po stronie klienta działający w jednej ścieżce może otworzyć okno lub ramkę iframe kierującą inną ścieżkę w tej samej domenie i może odczytywać i zapisywać w tym oknie bez żadnych ograniczeń. W związku z tym uzyskanie pliku cookie, którego zakres obejmuje inną ścieżkę w tej samej domenie, jest stosunkowo proste. Więcej informacji można znaleźć w następującym artykule Amita Kleina:

http://lists.webappsec.org/pipermail/websecurity_lists.webappsec.org/2006-March/000843.html

Zabezpieczanie zarządzania sesją

Środki obronne, które muszą podjąć aplikacje internetowe, aby zapobiec atakom na ich mechanizmy zarządzania sesją, odpowiadają dwóm szerokim kategoriom luk w zabezpieczeniach, które mają wpływ na te mechanizmy. Aby bezpiecznie zarządzać sesjami, aplikacja musi solidnie generować swoje tokeny i chronić je przez cały cykl ich życia, od utworzenia do usunięcia.

Generuj silne tokeny

Tokeny używane do ponownej identyfikacji użytkownika pomiędzy kolejnymi żądaniami powinny być generowane w sposób, który nie daje możliwości atakującemu, który w zwykły sposób uzyskuje dużą próbkę tokenów z aplikacji, do przewidywania lub ekstrapolacji tokenów wydanych innym użytkownikom. Najskuteczniejsze mechanizmy generowania tokenów to te, które:

* Użyj bardzo dużego zestawu możliwych wartości

* Zawierają silne źródło pseudolosowości, zapewniając równomierny i nieprzewidywalny rozkład tokenów w całym zakresie możliwych wartości

Zasadniczo każdy element o dowolnej długości i złożoności można odgadnąć przy użyciu brutalnej siły, mając wystarczająco dużo czasu i zasobów. Celem zaprojektowania mechanizmu do generowania silnych tokenów jest bardzo mało prawdopodobne, aby zdeterminowany atakujący z dużą przepustowością i zasobami przetwarzania odgadł pojedynczy ważny token w okresie jego ważności. Tokeny powinny składać się wyłącznie z identyfikatora używanego przez serwer do zlokalizowania odpowiedniego obiektu sesji, który ma zostać wykorzystany do przetworzenia żądania użytkownika. Token nie powinien zawierać żadnego znaczenia ani struktury, jawnej lub owiniętej warstwami kodowania lub zaciemniania. Wszystkie dane o właścicielu sesji i statusie powinny być przechowywane na serwerze w obiekcie sesji, któremu odpowiada token sesji. Zachowaj ostrożność przy wyborze źródła losowości. Deweloperzy powinni mieć świadomość, że różne dostępne dla nich źródła mogą znacznie różnić się siłą. Niektóre, takie jak java.util.Random, są doskonale przydatne do wielu celów, w których wymagane jest źródło zmiany danych wejściowych. Ale można je ekstrapolować zarówno w przód, jak iw tył z całkowitą pewnością na podstawie pojedynczego produktu wyjściowego. Deweloperzy powinni zbadać właściwości matematyczne rzeczywistych algorytmów używanych w ramach różnych dostępnych źródeł losowości i powinni przeczytać odpowiednią dokumentację dotyczącą zalecanych zastosowań różnych interfejsów API. Ogólnie rzecz biorąc, jeśli algorytm nie jest wyraźnie opisany jako bezpieczny kryptograficznie, należy założyć, że jest przewidywalny.

UWAGA: Niektóre źródła losowości o dużej sile potrzebują trochę czasu, aby zwrócić następną wartość w swojej sekwencji wyjściowej z powodu kroków, które podejmują w celu uzyskania wystarczającej entropii (na przykład ze zdarzeń systemowych). W związku z tym mogą nie dostarczać wartości wystarczająco szybko, aby wygenerować tokeny dla niektórych aplikacji o dużej objętości.

Oprócz wybrania najsolidniejszego możliwego źródła losowości, dobrą praktyką jest wprowadzenie jako źródła entropii pewnych informacji o indywidualnym żądaniu, dla którego generowany jest token. Informacje te mogą nie być unikalne dla tego żądania, ale mogą być skuteczne w łagodzeniu wszelkich słabości używanego podstawowego generatora liczb pseudolosowych. Oto kilka przykładów informacji, które można włączyć:

* Źródłowy adres IP i numer portu, z którego otrzymano żądanie

* Nagłówek User-Agent w żądaniu

* Czas żądania w milisekundach

Wysoce efektywną formułą uwzględnienia tej entropii jest skonstruowanie ciągu, który łączy liczbę pseudolosową, różnorodne dane specyficzne dla żądania, jak podano, oraz tajny ciąg znany tylko serwerowi i generowany od nowa przy każdym ponownym uruchomieniu. Następnie z tego ciągu pobierany jest odpowiedni skrót (przy użyciu na przykład SHA-256 w czasie pisania tego tekstu) w celu utworzenia łatwego do zarządzania ciągu o stałej długości, który może być używany jako token. (Umieszczenie najbardziej zmiennych elementów na początku danych wejściowych skrótu maksymalizuje efekt „lawiny” w algorytmie mieszającym.)

WSKAZÓWKA: Po wybraniu algorytmu generowania tokenów sesji przydatnym „eksperymentem myślowym” jest wyobrażenie sobie, że twoje źródło pseudolosowości jest zepsute i zawsze zwraca tę samą wartość. Czy w takiej sytuacji atakujący, który uzyska dużą próbkę tokenów z aplikacji, będzie w stanie ekstrapolować tokeny wystawione innym użytkownikom? Używając opisanego tutaj wzoru, jest to generalnie wysoce nieprawdopodobne, nawet przy pełnej znajomości zastosowanego algorytmu.

Źródłowy adres IP, numer portu, nagłówek User-Agent i czas żądania razem generują ogromną ilość entropii. A nawet mając pełną wiedzę na ten temat, atakujący nie będzie w stanie wyprodukować odpowiedniego tokena bez znajomości tajnego ciągu używanego przez serwer.

Chroń tokeny przez cały cykl ich życia

Teraz, gdy stworzyłeś solidny token, którego wartości nie można przewidzieć, token ten musi być chroniony przez cały jego cykl życia, od utworzenia do usunięcia, aby upewnić się, że nie zostanie ujawniony nikomu poza użytkownikiem, któremu został wydany:

* Token powinien być przesyłany tylko przez HTTPS. Każdy token przesłany w postaci zwykłego tekstu powinien być traktowany jako skażony – to znaczy nie dający pewności co do tożsamości użytkownika. Jeśli do przesyłania tokenów używane są pliki cookie HTTP, należy je oznaczyć jako bezpieczne, aby uniemożliwić przeglądarce użytkownika przesyłanie ich przez HTTP. Jeśli to możliwe, protokół HTTPS powinien być używany na każdej stronie aplikacji, w tym w treściach statycznych, takich jak strony pomocy, obrazy itd. Jeśli nie jest to pożądane, a usługa HTTP jest nadal zaimplementowana, aplikacja powinna przekierować wszelkie żądania dotyczące wrażliwych treści (w tym strony logowania) do usługi HTTPS. Zasoby statycznetakie jak strony pomocy zwykle nie są poufne i można uzyskać do nich dostęp bez żadnej uwierzytelnionej sesji. W związku z tym użycie bezpiecznych plików cookie można zabezpieczyć za pomocą instrukcji dotyczących zakresu plików cookie, aby zapobiec przesyłaniu tokenów w żądaniach dotyczących tych zasobów.

* Tokeny sesji nigdy nie powinny być przesyłane w adresie URL, ponieważ zapewnia to proste narzędzie do ataków utrwalania sesji i powoduje pojawianie się tokenów w wielu mechanizmach logowania. W niektórych przypadkach programiści wykorzystują tę technikę do realizacji sesji w przeglądarkach, które mają wyłączone pliki cookie. Jednak lepszym sposobem osiągnięcia tego celu jest użycie żądań POST dla wszystkich nawigacji i przechowywanie tokenów w ukrytym polu formularza HTML.

* Należy zaimplementować funkcję wylogowania. Powinno to usunąć wszystkie zasoby sesji przechowywane na serwerze i unieważnić token sesji.

* Wygaśnięcie sesji powinno nastąpić po odpowiednim okresie bezczynności (np. 10 minut). Powinno to spowodować takie samo zachowanie, jak w przypadku jawnego wylogowania użytkownika.

* Należy zapobiegać jednoczesnym logowaniom. Za każdym razem, gdy użytkownik się loguje, powinien zostać wystawiony inny token sesji, a każda istniejąca sesja należąca do użytkownika powinna zostać usunięta tak, jakby się z niej wylogował. W takim przypadku stary token może być przechowywany przez pewien czas. Wszelkie kolejne żądania otrzymane przy użyciu tokena powinny zwrócić użytkownikowi alert bezpieczeństwa informujący, że sesja została zakończona, ponieważ zalogował się z innej lokalizacji.

* Jeśli aplikacja zawiera jakiekolwiek funkcje administracyjne lub diagnostyczne, które umożliwiają przeglądanie tokenów sesji, należy solidnie chronić tę funkcjonalność przed nieautoryzowanym dostępem. W większości przypadków nie ma potrzeby, aby ta funkcja wyświetlała rzeczywisty token sesji. Powinien raczej zawierać wystarczające informacje o właścicielu sesji, aby można było wykonać wszelkie zadania pomocnicze i diagnostyczne, bez ujawniania tokena sesji przesłanego przez użytkownika w celu zidentyfikowania jego sesji.

* Zakres domeny i ścieżki plików cookie sesji aplikacji powinien być ustawiony tak restrykcyjnie, jak to możliwe. Pliki cookie o zbyt liberalnym zakresie są często generowane przez źle skonfigurowane platformy aplikacji internetowych lub serwery sieciowe, a nie przez samych twórców aplikacji. Żadne inne aplikacje internetowe ani niezaufane funkcje nie powinny być dostępne za pośrednictwem nazw

domen lub ścieżek URL, które są objęte zakresem plików cookie aplikacji. Szczególną uwagę należy zwrócić na wszelkie istniejące subdomeny nazwy domeny, która jest używana do uzyskania dostępu do aplikacji. W niektórych przypadkach, aby zapewnić, że ta luka się nie pojawi, może być konieczna modyfikacja schematu nazw domen i ścieżek używanych przez różne aplikacje używane w organizacji.

Należy podjąć szczególne środki w celu obrony mechanizmu zarządzania sesją przed różnymi atakami, których celem mogą być użytkownicy aplikacji:

* Baza kodu aplikacji powinna być rygorystycznie kontrolowana w celu zidentyfikowania i usunięcia wszelkich luk w zabezpieczeniach związanych ze skryptami między witrynami. Większość takich

luki w zabezpieczeniach mogą zostać wykorzystane do ataku na mechanizmy zarządzania sesją. W szczególności przechowywane (lub drugiego rzędu) ataki XSS można zwykle wykorzystać do pokonania każdej możliwej obrony przed niewłaściwym wykorzystaniem sesji i przejęciem.

* Dowolne tokeny przesłane przez użytkowników, których serwer nie rozpoznaje, nie powinny być akceptowane. Token powinien zostać natychmiast anulowany w przeglądarce, a użytkownik powinien wrócić do strony startowej aplikacji.

* Fałszowanie żądań między witrynami i inne ataki na sesje można utrudnić, wymagając dwuetapowego potwierdzenia i/lub ponownego uwierzytelnienia przed wykonaniem krytycznych działań, takich jak transfer środków.

* Ataki polegające na fałszowaniu żądań między witrynami można obronić, nie polegając wyłącznie na plikach cookie HTTP do przesyłania tokenów sesji. Korzystanie z mechanizmu cookies wprowadza podatność, ponieważ cookies są wysyłane automatycznie przez przeglądarkę niezależnie od przyczyny żądania. Jeśli tokeny są zawsze przesyłane w ukrytym polu formularza HTML, atakujący nie może utworzyć formularza, którego przesłanie spowoduje nieautoryzowane działanie, chyba że zna już wartość tokena. W takim przypadku może po prostu wykonać łatwy atak porwania. Tokeny na stronę mogą również pomóc w zapobieganiu tym atakom (zobacz następną sekcję).

* Po pomyślnym uwierzytelnieniu należy zawsze tworzyć nową sesję, aby złagodzić skutki ataków polegających na utrwalaniu sesji. Tam, gdzie aplikacja nie korzysta z uwierzytelniania, ale pozwala na przesyłanie poufnych danych, trudniej jest przeciwdziałać zagrożeniu związanemu z atakami fiksacyjnymi. Jednym z możliwych podejść jest jak najkrótsza kolejność stron, na których przesyłane są dane wrażliwe. Następnie możesz utworzyć nową sesję na pierwszej stronie tej sekwencji (w razie potrzeby kopiując z istniejącej sesji wszelkie wymagane dane, takie jak zawartość koszyka). Możesz też użyć tokenów na stronę (opisanych w następnej sekcji), aby uniemożliwić atakującemu, który zna token użyty na pierwszej stronie, dostęp do kolejnych stron. Z wyjątkiem sytuacji, gdy jest to bezwzględnie konieczne, dane osobowe nie powinny być ponownie wyświetlane użytkownikowi. Nawet tam, gdzie jest to wymagane (np. strona „potwierdzenia zamówienia” zawierająca adresy), poufne elementy, takie jak numery kart kredytowych i hasła, nigdy nie powinny być ponownie wyświetlane użytkownikowi i zawsze powinny być maskowane w źródle odpowiedzi aplikacji.

Rejestruj, monitoruj i ostrzegaj

Funkcjonalność zarządzania sesją aplikacji powinna być ściśle zintegrowana z jej mechanizmami rejestrowania, monitorowania i ostrzegania, aby zapewnić odpowiednie zapisy nietypowej aktywności i umożliwić administratorom podjęcie działań obronnych w razie potrzeby:

* Aplikacja powinna monitorować żądania zawierające nieprawidłowe tokeny. Poza najbardziej przewidywalnymi przypadkami, udany atak polegający na próbie odgadnięcia tokenów wydanych

innym użytkownikom zazwyczaj wiąże się z wysłaniem dużej liczby żądań zawierających nieprawidłowe tokeny, pozostawiając zauważalny ślad w dziennikach aplikacji.

* Ataki typu brute-force na tokeny sesji są trudne do całkowitego zablokowania, ponieważ nie można wyłączyć żadnego konkretnego konta użytkownika ani sesji, aby powstrzymać atak. Jedną z możliwych akcji jest blokowanie źródłowych adresów IP na pewien czas, gdy otrzymano pewną liczbę żądań zawierających nieprawidłowe tokeny. Jednak może to być nieskuteczne, gdy żądania jednego użytkownika pochodzą z wielu adresów IP (na przykład użytkownicy AOL) lub gdy żądania wielu użytkowników pochodzą z tego samego adresu IP (na przykład użytkownicy za serwerem proxy lub zaporą ogniową wykonującą translację adresów sieciowych).

* Nawet jeśli atakom siłowym na sesje nie można skutecznie zapobiegać w czasie rzeczywistym, prowadzenie szczegółowych dzienników i powiadamianie administratorów umożliwia im zbadanie ataku i podjęcie odpowiednich działań tam, gdzie to możliwe.

* Tam, gdzie to możliwe, użytkownicy powinni być powiadamiani o nietypowych zdarzeniach związanych z ich sesją, takich jak równoczesne logowanie lub pozorne przejęcie (wykrywane za pomocą tokenów na stronę). Nawet jeśli kompromitacja mogła już nastąpić, umożliwia to użytkownikowi sprawdzenie, czy nie miały miejsca nieautoryzowane działania, takie jak przelewy środków.

Reaktywne zakończenie sesji

Mechanizm zarządzania sesją może być wykorzystany jako wysoce skuteczna obrona przed wieloma rodzajami innych ataków na aplikację. Niektóre aplikacje o krytycznym znaczeniu dla bezpieczeństwa, takie jak bankowość internetowa, bardzo agresywnie przerywają sesję użytkownika za każdym razem, gdy przesyła on nietypowe żądanie. Przykładami są dowolne żądania zawierające zmodyfikowane ukryte pole formularza HTML lub

Parametr ciągu zapytania adresu URL, każde żądanie zawierające ciągi związane z atakami typu SQL injection lub cross-site scripting oraz wszelkie dane wprowadzane przez użytkownika, które normalnie zostałyby zablokowane przez kontrole po stronie klienta, takie jak ograniczenia długości. Oczywiście wszelkie rzeczywiste luki w zabezpieczeniach, które mogą zostać wykorzystane przy użyciu takich żądań, należy rozwiązać u źródła. Ale zmuszanie użytkowników do ponownego uwierzytelnienia każdego czasu, w którym prześlą nieprawidłowe żądanie, może spowolnić proces badania aplikacji pod kątem luk o wiele rzędów wielkości, nawet w przypadku zastosowania zautomatyzowanych technik. Jeśli nadal istnieją szczątkowe luki w zabezpieczeniach, prawdopodobieństwo ich odkrycia przez kogokolwiek w terenie jest znacznie mniejsze. W przypadku wdrożenia tego rodzaju ochrony zaleca się również, aby można ją było łatwo wyłączyć do celów testowych. Jeśli legalny test penetracyjny aplikacji zostanie spowolniony w taki sam sposób, jak rzeczywisty atakujący, jego skuteczność drastycznie spadnie. Ponadto jest bardzo prawdopodobne, że obecność mechanizmu spowoduje, że w kodzie produkcyjnym pozostanie więcej luk niż w przypadku braku mechanizmu.

KROKI HACKOWANIA

Jeśli aplikacja, którą atakujesz, używa tego rodzaju środka obronnego, może się okazać, że sprawdzanie aplikacji pod kątem wielu typowych luk w zabezpieczeniach jest niezwykle czasochłonne. Oszałamiająca konieczność logowania się po każdym nieudanym teście i ponownego nawigowania do punktu aplikacji, na którą patrzyłeś, szybko skłoniłaby cię do poddania się. W takiej sytuacji często można skorzystać z automatyzacji, aby rozwiązać problem. Używając Burp Intruder do przeprowadzenia ataku, możesz użyć funkcji Obtain Cookie, aby wykonać nowe logowanie przed wysłaniem każdego przypadku testowego i użyć nowego tokena sesji (pod warunkiem, że logowanie

jest jednoetapowe). Podczas ręcznego przeglądania i sondowania aplikacji można korzystać z funkcji rozszerzeń Burp Proxy za pośrednictwem interfejsu IBurpExtender. Możesz utworzyć rozszerzenie, które wykrywa, kiedy aplikacja wykonała wymuszone wylogowanie, automatycznie loguje się ponownie do aplikacji i zwraca nową sesję i stronę do przeglądarki, opcjonalnie z wyskakującym komunikatem informującym o tym, co się stało. Chociaż w żaden sposób nie usuwa to problemu, w niektórych przypadkach może go znacznie złagodzić.

Streszczenie

Mechanizm zarządzania sesją zapewnia bogate źródło potencjalnych luk, na które można zwrócić uwagę podczas formułowania ataku na aplikację. Ze względu na swoją fundamentalną rolę w umożliwianiu aplikacji identyfikowania tego samego użytkownika w wielu żądaniach, zepsuta funkcja zarządzania sesjami zwykle zapewnia klucze do królestwa. Wskakiwanie do sesji innych użytkowników jest dobre. Przejęcie sesji administratora jest jeszcze lepsze; zazwyczaj umożliwia to złamanie zabezpieczeń całej aplikacji. Możesz spodziewać się szerokiego zakresu defektów w rzeczywistej sesji funkcjonalności zarządzania. Gdy stosowane są mechanizmy na zamówienie, możliwe słabości i drogi ataku mogą wydawać się nieskończone. Najważniejszą lekcją, jaką można wyciągnąć z tego tematu, jest cierpliwość i determinacja. Sporo mechanizmów zarządzania sesją, które wydają się solidne przy pierwszej inspekcji, może okazać się niewystarczające po dokładnej analizie. Rozszyfrowanie metody używanej przez aplikację do generowania sekwencji pozornie losowych tokenów może wymagać czasu i pomysłowości. Ale biorąc pod uwagę nagrodę, jest to zwykle inwestycja warta podjęcia.

Pytania

1. Logujesz się do aplikacji, a serwer ustawia następujący plik cookie:

Set-cookie: sessid=amltMjM6MTI0MToxMTk0ODcwODYz;

Godzinę później logujesz się ponownie i otrzymujesz:

Set-cookie: sessid=amltMjM6MTI0MToxMTk0ODc1MTMy;

Co możesz wywnioskować o tych ciasteczkach?

2. Aplikacja wykorzystuje sześcioznakowe alfanumeryczne tokeny sesyjne i pięcioznakowe alfanumeryczne hasła. Oba są generowane losowo zgodnie z nieprzewidywalnym algorytmem. Który z nich może być bardziej wartościowym celem ataku polegającego na zgadywaniu metodą brute-force? Wymień wszystkie czynniki, które mogą mieć znaczenie dla Twojej decyzji.

3. Logujesz się do aplikacji pod następującym adresem URL:

https://foo.wahh-app.com/login/home.php

a serwer ustawia następujący plik cookie:

Set-cookie: sessionId=1498172056438227; domena=foo.wahhapp.com; ścieżka=/logowanie; HttpTylko;

Następnie odwiedzasz szereg innych adresów URL. Do którego z poniższych adresów Twoja przeglądarka prześle plik cookie sessionId? (Wybierz wszystkie pasujące.)

(a) https://foo.wahh-app.com/login/myaccount.php

(b) https://bar.wahh-app.com/login

(c) https://staging.foo.wahh-app.com/login/home.php

(d) http://foo.wahh-app.com/login/myaccount.php

(e) http://foo.wahh-app.com/logintest/login.php

(f) https://foo.wahh-app.com/logout

(g) https://wahh-app.com/login/

(h) https://xfoo.wahh-app.com/login/myaccount.php

4. Aplikacja, na którą celujesz, oprócz podstawowego tokena sesji używa tokenów na stronę. Jeśli token na stronę zostanie odebrany poza kolejnością, cała sesja zostanie unieważniona. Załóżmy, że odkryjesz jakiś defekt, który umożliwia przewidywanie lub przechwytywanie tokenów wystawionych innym użytkownikom, którzy aktualnie uzyskują dostęp do aplikacji. Czy możesz przejąć ich sesje?

5. Logujesz się do aplikacji, a serwer ustawia następujący plik cookie:

Set-cookie: sess=ab11298f7eg14;

Kliknięcie przycisku wylogowania powoduje wykonanie następującego skryptu po stronie klienta:

document.cookie=”sess=”;

dokument.lokalizacja=”/”;

Jaki wniosek wyciągnąłbyś z takiego zachowania?