Wykorzystanie ujawnienia informacji

W części 4 opisano różne techniki, których można użyć do zmapowania aplikacji docelowej i uzyskania wstępnego zrozumienia jej działania. Metodologia ta obejmowała interakcję z aplikacją w bardzo łagodny sposób w celu skatalogowania jej zawartości i funkcjonalności, określenia używanych technologii oraz zidentyfikowania kluczowej powierzchni ataku. W tym rozdziale opisano sposoby uzyskiwania dalszych informacji z aplikacji podczas rzeczywistego ataku. Wiąże się to głównie z interakcją z aplikacją w nieoczekiwany i złośliwy sposób oraz wykorzystywaniem anomalii w zachowaniu aplikacji w celu wyodrębnienia wartościowych dla użytkownika informacji. Jeśli się powiedzie, taki atak może umożliwić odzyskanie poufnych danych, takich jak dane uwierzytelniające użytkownika, głębsze zrozumienie warunku błędu w celu dostrojenia ataku, odkrycie większej ilości szczegółów na temat używanych technologii i zmapowania wewnętrznej struktury i funkcjonalności aplikacji.

Wykorzystanie komunikatów o błędach

Wiele aplikacji internetowych zwraca informacyjne komunikaty o błędach w przypadku wystąpienia nieoczekiwanych zdarzeń. Mogą to być zarówno proste wbudowane komunikaty, które ujawniają tylko kategorię błędu, jak i pełne informacje debugowania, które ujawniają wiele szczegółów na temat stanu aplikacji. Większość aplikacji przed wdrożeniem podlega różnego rodzaju testom użyteczności. Testy te zwykle identyfikują większość błędów, które mogą wystąpić podczas normalnego użytkowania aplikacji. W związku z tym warunki te są zwykle obsługiwane w sposób łagodny, który nie wiąże się z zwracaniem użytkownikowi żadnych komunikatów technicznych. Jednak gdy aplikacja jest przedmiotem aktywnego ataku, prawdopodobnie pojawi się znacznie szerszy zakres błędów, co może skutkować zwróceniem użytkownikowi bardziej szczegółowych informacji. Stwierdzono, że nawet najbardziej krytyczne dla bezpieczeństwa aplikacje, takie jak te używane przez banki internetowe, zwracają bardzo szczegółowe dane wyjściowe debugowania, gdy zostanie wygenerowany wystarczająco nietypowy błąd.

Komunikaty o błędach skryptu

Kiedy pojawia się błąd w interpretowanym języku skryptów WWW, takim jak VBScript, aplikacja zazwyczaj zwraca prosty komunikat ujawniający naturę błędu i ewentualnie numer wiersza pliku, w którym wystąpił błąd. Na przykład:

Microsoft VBScript runtime error 800a0009

Subscript out of range: [number -1]

/register.asp, line 821

Tego rodzaju wiadomość zazwyczaj nie zawiera żadnych wrażliwych informacji o stanie aplikacji ani przetwarzanych danych. Może to jednak pomóc zawęzić cel ataku. Na przykład podczas wstawiania różnych łańcuchów ataku do określonego parametru w celu wykrycia typowych luk w zabezpieczeniach może pojawić się następujący komunikat:

Microsoft VBScript runtime error ‘800a000d’

Type mismatch: ‘ [string: “’”]’

/scripts/confirmOrder.asp, line 715

Ten komunikat wskazuje, że zmodyfikowana wartość jest prawdopodobnie przypisywana do zmiennej numerycznej i podano dane wejściowe, których nie można tak przypisać, ponieważ zawierają znaki

nienumeryczne. W tej sytuacji jest wysoce prawdopodobne, że nic nie można zyskać, podając nieliczbowe ciągi ataku jako ten parametr. Dlatego w przypadku wielu kategorii błędów lepiej kierować się innymi parametrami. Innym sposobem, w jaki ten typ komunikatu o błędzie może pomóc, jest lepsze zrozumienie logiki zaimplementowanej w aplikacji po stronie serwera. Ponieważ komunikat ujawnia numer wiersza, w którym wystąpił błąd, możesz być w stanie potwierdzić, czy dwa różne źle sformułowane żądania wyzwalają ten sam błąd, czy różne błędy. Możesz także określić kolejność, w jakiej przetwarzane są różne parametry, przesyłając błędne dane wejściowe w ramach wielu parametrów i identyfikując lokalizację, w której występuje błąd. Poprzez systematyczne manipulowanie różnymi parametrami można mapować różne ścieżki kodu wykonywane na serwerze.

Ślady stosu

Większość aplikacji internetowych jest napisana w językach, które są bardziej złożone niż proste skrypty, ale nadal działają w zarządzanym środowisku wykonawczym, takim jak Java, C# lub Visual Basic .NET. Gdy w tych językach wystąpi nieobsługiwany błąd, do przeglądarki często zwracane są pełne dane śledzenia stosu. Ślad stosu to ustrukturyzowany komunikat o błędzie, który zaczyna się od opisu rzeczywistego błędu. Następnie następuje seria wierszy opisujących stan stosu wywołań wykonania w momencie wystąpienia błędu. Górny wiersz stosu wywołań pokazuje funkcję, która wygenerowała błąd, następny wiersz pokazuje funkcję, która wywołała poprzednią funkcję, i tak dalej w dół stosu wywołań, aż do wyczerpania hierarchii wywołań funkcji i. Poniżej przedstawiono przykład śledzenia stosu wygenerowanego przez aplikację ASP.NET:

[HttpException (0x80004005): Cannot use a leading .. to exit above the

top directory.]

System.Web.Util.UrlPath.Reduce(String path) +701

System.Web.Util.UrlPath.Combine(String basepath, String relative)+304

System.Web.UI.Control.ResolveUrl(String relativeUrl) +143

PBSApp.StatFunc.Web.MemberAwarePage.Redirect(String url) +130

PBSApp.StatFunc.Web.MemberAwarePage.Process() +201

PBSApp.StatFunc.Web.MemberAwarePage.OnLoad(EventArgs e)

System.Web.UI.Control.LoadRecursive() +35

System.Web.UI.Page.ProcessRequestMain() +750

Version Information: Microsoft .NET Framework Version:1.1.4322.2300;

ASP.NET Version:1.1.4322.2300

Ten rodzaj komunikatu o błędzie zawiera dużą ilość przydatnych informacji, które mogą pomóc w dopracowaniu ataku na aplikację:

* Często opisuje dokładny powód wystąpienia błędu. Może to umożliwić dostosowanie danych wejściowych w celu obejścia warunku błędu i przyspieszenia ataku.

* Stos wywołań zwykle odwołuje się do wielu bibliotek i komponentów kodu stron trzecich, które są używane w aplikacji. Możesz przejrzeć dokumentację tych komponentów, aby zrozumieć ich zamierzone zachowanie i założenia. Możesz także utworzyć własną lokalną implementację i

przetestować ją, aby zrozumieć, w jaki sposób obsługuje ona nieoczekiwane dane wejściowe i potencjalnie zidentyfikować luki w zabezpieczeniach.

*- Stos wywołań zawiera nazwy zastrzeżonych komponentów kodu używanych do przetwarzania żądania. Schemat ich nazewnictwa i wzajemne powiązania między nimi mogą pozwolić ci wywnioskować szczegóły dotyczące wewnętrznej struktury i funkcjonalności aplikacji.

* Ślad stosu często zawiera numery linii. Podobnie jak w przypadku opisanych wcześniej prostych komunikatów o błędach skryptu, mogą one umożliwić zbadanie i zrozumienie wewnętrznej logiki poszczególnych składników aplikacji

* Komunikat o błędzie często zawiera dodatkowe informacje o aplikacji i środowisku, w którym jest uruchomiona. W poprzednim przykładzie można określić dokładną wersję używanej platformy ASP.NET. Umożliwia to zbadanie platformy pod kątem znanych lub nowych luk w zabezpieczeniach, nietypowych zachowań, typowych błędów konfiguracji i tak dalej.

Informacyjne komunikaty debugowania

Niektóre aplikacje generują niestandardowe komunikaty o błędach, które zawierają dużą ilość informacji debugowania. Są one zwykle implementowane w celu ułatwienia debugowania podczas opracowywania i testowania i często zawierają szczegółowe informacje o stanie działania aplikacji. Na przykład:

---

* * * S E S S I O N * * *

---

i5agor2n2pw3gp551pszsb55

SessionUser.Sessions App.FEStructure.Sessions

SessionUser.Auth 1

SessionUser.BranchID 103

SessionUser.CompanyID 76

SessionUser.BrokerRef RRadv0

SessionUser.UserID 229

SessionUser.Training 0

SessionUser.NetworkID 11

SessionUser.BrandingPath FE

LoginURL /Default/fedefault.aspx

ReturnURL ../default/fedefault.aspx

SessionUser.Key f7e50aef8fadd30f31f3aea104cef26ed2ce2be50073c

SessionClient.ID 306

SessionClient.ReviewID 245

UPriv.2100

SessionUser.NetworkLevelUser 0

UPriv.2200

SessionUser.BranchLevelUser 0

SessionDatabase fd219.prod.wahh-bank.com

Następujące elementy są często dołączane do pełnych komunikatów debugowania:

* Wartości kluczowych zmiennych sesyjnych, którymi można manipulować za pomocą danych wprowadzanych przez użytkownika

* Nazwy hostów i poświadczenia dla komponentów zaplecza, takich jak bazy danych

* Nazwy plików i katalogów na serwerze

* Informacje osadzone w znaczących tokenach sesji

* Klucze szyfrujące służące do ochrony danych przesyłanych za pośrednictwem klienta

* Informacje debugowania dotyczące wyjątków pojawiających się w komponentach kodu natywnego, w tym wartości rejestrów procesora, zawartość stosu oraz lista załadowanych bibliotek DLL i ich adresów bazowych

Gdy tego rodzaju funkcja raportowania błędów jest obecna w działającym kodzie produkcyjnym, może to oznaczać krytyczną słabość w zabezpieczeniach aplikacji. Należy go dokładnie przejrzeć, aby zidentyfikować elementy, które można wykorzystać do dalszego ataku, oraz wszelkie sposoby dostarczania spreparowanych danych wejściowych w celu manipulowania stanem aplikacji i kontrolowania pobieranych informacji.

Komunikaty serwera i bazy danych

Informacyjne komunikaty o błędach są często zwracane nie przez samą aplikację, ale przez niektóre komponenty zaplecza, takie jak baza danych, serwer pocztowy lub serwer SOAP. Jeśli wystąpi całkowicie nieobsługiwany błąd, aplikacja zazwyczaj odpowiada kodem stanu HTTP 500, a treść odpowiedzi może zawierać dodatkowe informacje o błędzie. W innych przypadkach aplikacja może sprawnie obsłużyć błąd i zwrócić użytkownikowi dostosowany komunikat, czasem zawierający informacje o błędzie wygenerowane przez komponent zaplecza. W niektórych sytuacjach samo ujawnienie informacji może posłużyć jako kanał do ataku. Informacje ujawnione przez aplikację w komunikacie debugowania lub wyjątku są często niezamierzone, w wyniku czego procedury bezpieczeństwa organizacji mogą całkowicie przeoczyć istnienie ujawnienia. Zwrócony błąd może umożliwić szereg dalszych ataków, jak opisano w poniższych sekcjach.

Wykorzystanie ujawnienia informacji do przeprowadzenia ataku

Kiedy przeprowadzany jest określony atak na komponent zaplecza serwera, ten komponent często przekazuje bezpośrednią informację zwrotną na temat wszelkich napotkanych błędów. Może to pomóc w dostrojeniu ataku. Komunikaty o błędach bazy danych często zawierają przydatne informacje. Na przykład często ujawniają zapytanie, które wygenerowało błąd, umożliwiając dostrojenie ataku typu SQL injection:

Failed to retrieve row with statement - SELECT object_data FROM

deftr.tblobject WHERE object_id = ‘FDJE00012’ AND project_id = ‘FOO’

and 1=2--’

Ataki typu Cross-Site Scripting w ramach komunikatów o błędach

Jak opisano w części 12, zabezpieczenie przed atakami typu cross-site scripting jest żmudnym zadaniem, wymagającym identyfikacji każdej lokalizacji wyjściowej danych dostarczonych przez użytkownika. Chociaż większość platform automatycznie koduje dane w formacie HTML podczas zgłaszania błędów, nie jest to bynajmniej uniwersalne. Komunikaty o błędach mogą pojawiać się w wielu, często nietypowych miejscach odpowiedzi HTTP. W wywołaniu HttpServletResponse .sendError() używanym przez Tomcat dane błędu są również częścią nagłówka odpowiedzi:

HTTP/1.1 500 General Error Accessing Doc10083011

Server: Apache-Coyote/1.1

Content-Type: text/html;charset=ISO-8859-1

Content-Length: 1105

Date: Sat, 23 Apr 2011 08:52:15 GMT

Connection: close

Osoba atakująca, która ma kontrolę nad ciągiem wejściowym Doc10083011, może podać znaki powrotu karetki i przeprowadzić atak polegający na wstrzyknięciu nagłówka HTTP lub atak typu cross-site scripting w ramach odpowiedzi HTTP. Więcej szczegółów można znaleźć tutaj:

Osoba atakująca, która ma kontrolę nad ciągiem wejściowym Doc10083011, może podać znaki powrotu karetki i przeprowadzić atak polegający na wstrzyknięciu nagłówka HTTP lub atak typu cross-site scripting w ramach odpowiedzi HTTP. Często dostosowywane komunikaty o błędach są przeznaczone dla miejsca docelowego innego niż HTML, takiego jak konsola, ale są błędnie zgłaszane użytkownikowi w odpowiedzi HTTP. W takich sytuacjach skrypty między witrynami są często łatwe do wykorzystania.

Deszyfrowanie Oracle w ujawnianiu informacji

W części 11 podano przykład, w jaki sposób niezamierzona „wyrocznia szyfrująca” może zostać wykorzystana do odszyfrowania łańcuchów prezentowanych użytkownikowi w zaszyfrowanym formacie. Ta sama kwestia może dotyczyć ujawniania informacji. W części 7 podano przykład aplikacji, która zapewniała zaszyfrowane łącze do pobrania w celu uzyskania dostępu do pliku. Jeśli od tego czasu plik został przeniesiony lub usunięty, aplikacja zgłaszała, że nie można go pobrać. Oczywiście komunikat o błędzie zawierał odszyfrowaną wartość pliku, więc każda zaszyfrowana „nazwa pliku” mogła zostać dostarczona do łącza pobierania, co spowodowało błąd. W takich przypadkach ujawnienie informacji wynikało z nadużycia celowej informacji zwrotnej. Możliwe jest również, że ujawnienie informacji będzie bardziej przypadkowe, jeśli parametry zostaną odszyfrowane, a następnie wykorzystane w różnych funkcjach, z których każda może rejestrować dane lub generować komunikaty o błędach. Przykładem napotkanym przez autorów była złożona aplikacja workflow, wykorzystująca zaszyfrowane parametry przesyłane przez klienta. Zamieniając domyślne wartości używane dla dbid i grouphome, aplikacja odpowiedziała błędem:

java.sql.SQLException: Listener refused the connection with the

following error: ORA-12505, TNS:listener does not currently know

of SID given in connect descriptor The Connection descriptor used

by the client was: 172.16.214.154:1521:docs/londonoffice/2010/general

Dało to znaczny wgląd. W szczególności dbid był w rzeczywistości zaszyfrowanym identyfikatorem SID dla połączenia z bazą danych Oracle (deskryptor połączenia ma postać Serwer:Port:SID), a strona główna grupy była zaszyfrowaną ścieżką do pliku. W ataku analogicznym do wielu innych ataków polegających na ujawnieniu informacji, znajomość ścieżki pliku dostarczyła informacji niezbędnych do przeprowadzenia ataku polegającego na manipulacji ścieżką pliku. Podając dokładnie trzy znaki przemierzania ścieżki w nazwie pliku i poruszając się po podobnej strukturze katalogów, można było przesyłać pliki zawierające szkodliwy skrypt bezpośrednio do przestrzeni roboczej innej grupy:

POST /dashboard/utils/fileupload HTTP/1.1

Accept: text/html, application/xhtml+xml, */*

Referer: http://wahh/dashboard/common/newnote

Accept-Language: en-GB

Content-Type: multipart/form-data; boundary=------7db3d439b04c0

Accept-Encoding: gzip, deflate

Host: wahh

Content-Length: 8088

Proxy-Connection: Keep-Alive

--------7db3d439b04c0

Content-Disposition: form-data; name=”MAX_FILE_SIZE”

100000

--------7db3d439b04c0

Content-Disposition: form-data; name=”uploadedfile”; filename=”../../../

newportoffice/2010/general/xss.html”

Content-Type: text/html

<html><body><script>...

...

KROKI HACKOWANIA

1. Gdy sondujesz aplikację pod kątem typowych luk w zabezpieczeniach, przesyłając spreparowane ciągi ataków w różnych parametrach, zawsze monitoruj odpowiedzi aplikacji, aby zidentyfikować wszelkie komunikaty o błędach, które mogą zawierać przydatne informacje. Spróbuj wymusić odpowiedź na błąd z aplikacji, dostarczając zaszyfrowane ciągi danych w niewłaściwym kontekście lub wykonując działania na zasobach, które nie są w odpowiednim stanie do obsługi działania.

2. Należy pamiętać, że informacje o błędach zwracane przez serwer

odpowiedź może nie zostać wyświetlona na ekranie w przeglądarce. Skutecznym sposobem identyfikacji wielu błędów jest przeszukiwanie każdej nieprzetworzonej odpowiedzi pod kątem słów kluczowych, które często występują w komunikatach o błędach. Na przykład:

*błąd

* wyjątek

* nielegalny

* nieważny

* ponieść porażkę

* stos

* dostęp

* katalog

* plik

* nie znaleziono

* varchar

* ODBC

* SQL

* SELECT

3. Gdy wysyłasz serię żądań modyfikujących parametry w ramach żądania podstawowego, sprawdź, czy pierwotna odpowiedź zawiera już któreś ze słów kluczowych, których szukasz, aby uniknąć fałszywych trafień.

4. Możesz użyć funkcji Grep w Burp Intruder, aby szybko zidentyfikować wszelkie wystąpienia interesujących słów kluczowych w odpowiedziach generowanych przez dany atak. W przypadku znalezienia dopasowań przejrzyj ręcznie odpowiednie odpowiedzi, aby określić, czy zwrócono przydatne informacje o błędzie.

WSKAZÓWKA Jeśli przeglądasz odpowiedzi serwera w przeglądarce, pamiętaj, że Internet Explorer domyślnie ukrywa wiele komunikatów o błędach i zastępuje je ogólną stroną. Możesz wyłączyć to zachowanie, wybierając Narzędzia ➢ Opcje internetowe, a następnie wybierając kartę Zaawansowane.

Korzystanie z informacji publicznej

Ze względu na ogromną różnorodność powszechnie używanych technologii i składników aplikacji sieci Web należy często spodziewać się nietypowych komunikatów, których wcześniej nie widziano i które mogą nie od razu wskazywać na naturę błędu, który wystąpił w aplikacji. W takiej sytuacji często możesz uzyskać dodatkowe informacje na temat znaczenia wiadomości z różnych źródeł publicznych. Często nietypowy komunikat o błędzie jest wynikiem awarii określonego interfejsu API. Wyszukiwanie tekstu wiadomości może prowadzić do dokumentacji tego interfejsu API lub do forów programistów i innych miejsc, w których omawiany jest ten sam problem. Wiele aplikacji wykorzystuje komponenty innych firm do wykonywania określonych typowych zadań, takich jak wyszukiwanie, koszyki na zakupy i funkcje przesyłania opinii o witrynach. Wszelkie komunikaty o błędach generowane przez te

komponenty prawdopodobnie pojawiły się w innych aplikacjach i prawdopodobnie zostały omówione w innym miejscu. Niektóre aplikacje zawierają publicznie dostępny kod źródłowy. Wyszukując określone wyrażenia pojawiające się w nietypowych komunikatach o błędach, można znaleźć kod źródłowy implementujący odpowiednią funkcję. Następnie możesz to przejrzeć, aby dokładnie zrozumieć, jakie przetwarzanie jest wykonywane na danych wejściowych i jak możesz manipulować aplikacją w celu wykorzystania luki w zabezpieczeniach.

KROKI HACKOWANIA

1. Wyszukaj tekst nietypowych komunikatów o błędach za pomocą standardowych wyszukiwarek. Możesz użyć różnych zaawansowanych funkcji wyszukiwania, aby zawęzić wyniki. Na przykład:

„nie można pobrać" typ pliku: php

2. Przejrzyj wyniki wyszukiwania, szukając zarówno dyskusji na temat komunikatu o błędzie, jak i innych witryn internetowych, w których pojawił się ten sam komunikat. Inne aplikacje mogą generować ten sam komunikat w bardziej szczegółowym kontekście, co pozwala lepiej zrozumieć, jakie warunki powodują błąd. Użyj pamięci podręcznej wyszukiwarki, aby pobrać przykłady komunikatów o błędach, które nie pojawiają się już w działającej aplikacji.

3. Użyj wyszukiwarki kodów Google, aby zlokalizować publicznie dostępny kod, który może być odpowiedzialny za konkretny komunikat o błędzie. Szukaj fragmentów komunikatów o błędach, które mogą być na stałe zakodowane w kodzie źródłowym aplikacji. Możesz także użyć różnych zaawansowanych funkcji wyszukiwania, aby określić język kodu i inne szczegóły, jeśli są one znane. Na przykład:

nie można\ pobrać\ lang:php package:mail

4. Jeśli uzyskałeś ślady stosu zawierające nazwy bibliotek i komponentów kodu innych firm, wyszukaj te nazwy w obu typach wyszukiwarek.

Informacyjne komunikaty o błędach inżynierskich

W niektórych sytuacjach możliwe może być systematyczne projektowanie warunków błędów w taki sposób, aby uzyskać poufne informacje w samym komunikacie o błędzie. Jedną z typowych sytuacji, w których pojawia się taka możliwość, jest sytuacja, w której można spowodować, że aplikacja podejmie próbę wykonania nieprawidłowej akcji na określonym elemencie danych. Jeśli wynikowy komunikat o błędzie ujawnia wartość tych danych i możesz spowodować przetwarzanie interesujących informacji w ten sposób, możesz wykorzystać to zachowanie do wyodrębnienia dowolnych danych z aplikacji. Pełne komunikaty o błędach otwartej łączności z bazą danych (ODBC) można wykorzystać w ataku typu SQL injection w celu pobrania wyników dowolnych zapytań do bazy danych. Na przykład następujący kod SQL, jeśli zostanie wstrzyknięty do klauzuli WHERE, spowoduje, że baza danych przekształci hasło pierwszego użytkownika w tabeli users na liczbę całkowitą w celu przeprowadzenia oceny:

‘ and 1=(select password from users where uid=1)—

Powoduje to wyświetlenie następującego informacyjnego komunikatu o błędzie:

Error: Conversion failed when converting the varchar value

‘37CE1CCA75308590E4D6A35F288B58FACDBB0841’ to data type int.

Innym sposobem wykorzystania tego rodzaju techniki jest sytuacja, w której błąd aplikacji generuje ślad stosu zawierający opis błędu i można zaprojektować sytuację, w której interesujące informacje

zostaną włączone do opisu błędu. Niektóre bazy danych umożliwiają tworzenie funkcji zdefiniowanych przez użytkownika napisanych w Javie. Wykorzystując lukę wstrzykiwania kodu SQL, możesz stworzyć własną funkcję do wykonywania dowolnych zadań. Jeśli aplikacja zwróci komunikaty o błędach do przeglądarki, z poziomu swojej funkcji możesz zgłosić wyjątek Java zawierający dowolne dane, które chcesz pobrać. Na przykład poniższy kod wykonuje polecenie ls systemu operacyjnego, a następnie generuje wyjątek, który zawiera dane wyjściowe polecenia. Spowoduje to zwrócenie do przeglądarki śladu stosu, którego pierwszy wiersz zawiera listę katalogów:

```
ByteArrayOutputStream baos = new ByteArrayOutputStream();

try

{

Process p = Runtime.getRuntime().exec("ls");

InputStream is = p.getInputStream();

int c;

while (-1 != (c = is.read()))

baos.write((byte) c);

catch (Exception e)

{

}

throw new RuntimeException(new String(baos.toByteArray()));
```

Zbieranie opublikowanych informacji

Oprócz ujawniania przydatnych informacji w komunikatach o błędach, innym podstawowym sposobem, w jaki aplikacje internetowe udostępniają poufne dane, jest faktyczne publikowanie ich bezpośrednio. Istnieją różne powody, dla których aplikacja może publikować informacje, które może wykorzystać osoba atakująca:

* Zgodnie z projektem, jako część podstawowej funkcjonalności aplikacji

* Jako niezamierzony efekt uboczny innej funkcji

* Dzięki funkcji debugowania, która pozostaje obecna w aplikacji na żywo

* Z powodu pewnych luk w zabezpieczeniach, takich jak zepsuta kontrola dostępu Oto kilka przykładów potencjalnie poufnych informacji, które aplikacje często publikują użytkownikom:

* Listy ważnych nazw użytkowników, numerów kont i identyfikatorów dokumentów

* Szczegóły profilu użytkownika, w tym role i uprawnienia użytkownika, data ostatniego logowania i stan konta

* Bieżące hasło użytkownika (zwykle jest zamaskowane na ekranie, ale jest obecne w źródle strony)

* Pliki dziennika zawierające informacje, takie jak nazwy użytkowników, adresy URL, wykonane działania, tokeny sesji i zapytania do bazy danych

* Szczegóły aplikacji w źródle HTML po stronie klienta, takie jak skomentowane łącza lub pola formularzy oraz komentarze dotyczące błędów

KROKI HACKOWANIA

1. Przejrzyj wyniki ćwiczeń z mapowania aplikacji, aby zidentyfikować wszystkie możliwe funkcje po stronie serwera i dane po stronie klienta wykorzystywane do uzyskiwania przydatnych informacji.

2. Zidentyfikuj wszystkie miejsca w aplikacji, w których poufne dane, takie jak hasła lub dane karty kredytowej, są przesyłane z serwera do przeglądarki. Nawet jeśli są one zamaskowane na ekranie, nadal są widoczne w odpowiedzi serwera. Jeśli znalazłeś inną odpowiednią lukę, na przykład w ramach kontroli dostępu lub obsługi sesji, to zachowanie może zostać wykorzystane do uzyskania informacji należących do innych użytkowników aplikacji.

3. Jeśli zidentyfikujesz jakiekolwiek sposoby wydobywania poufnych informacji, użyj technik opisanych w części 14, aby zautomatyzować ten proces.

Korzystanie z wnioskowania

W niektórych sytuacjach aplikacja może nie ujawniać bezpośrednio użytkownikowi żadnych danych, ale może zachowywać się w sposób umożliwiający wiarygodne wywnioskowanie przydatnych informacji. Zetknęliśmy się już z wieloma przypadkami tego zjawiska w trakcie badania innych kategorii powszechnej podatności. Na przykład:

* Funkcja rejestracji, która umożliwia wyliczanie zarejestrowanych nazw użytkowników na podstawie komunikatu o błędzie po wybraniu istniejącej nazwy użytkownika.

* Wyszukiwarka, która pozwala wywnioskować zawartość zindeksowanych dokumentów, do których nie masz uprawnień do bezpośredniego przeglądania.

* Luka umożliwiająca ślepe wstrzyknięcie kodu SQL, w której można zmienić zachowanie aplikacji, dodając warunek binarny do istniejącego zapytania, co umożliwia wyodrębnianie informacji bit po bicie.

* Atak „dopełnienie wyroczni” w .NET, w którym osoba atakująca może odszyfrować dowolny ciąg, wysyłając serię żądań do serwera i obserwując, które z nich powodują błąd podczas odszyfrowywania.

Inny sposób, w jaki subtelne różnice w zachowaniu aplikacji mogą ujawniać informacje, występuje, gdy wykonanie różnych operacji zajmuje różny czas, w zależności od pewnego faktu, który jest interesujący dla atakującego. Ta rozbieżność może wynikać z różnych powodów:

* Wiele dużych i złożonych aplikacji pobiera dane z wielu systemów zaplecza, takich jak bazy danych, kolejki komunikatów i komputery mainframe. Aby poprawić wydajność, niektóre aplikacje przechowują często używane informacje. Podobnie, niektóre aplikacje stosują podejście leniwego ładowania, w którym obiekty i dane są ładowane tylko wtedy, gdy są potrzebne. W tej sytuacji dane, do których ostatnio uzyskano dostęp, są szybko pobierane z lokalnej kopii w pamięci podręcznej serwera, podczas gdy inne dane są pobierane wolniej z odpowiedniego źródła zaplecza. Takie zachowanie zaobserwowano w aplikacjach bankowości internetowej. Żądanie dostępu do konta trwa dłużej, jeśli konto jest uśpione, niż gdy jest aktywne, co umożliwia wykwalifikowanemu atakującemu wyliczenie kont, do których ostatnio korzystali inni użytkownicy.

* W niektórych sytuacjach ilość przetwarzania, które aplikacja wykonuje w związku z konkretnym żądaniem, może zależeć od tego, czy przesłany element danych jest ważny. Na przykład, gdy do mechanizmu logowania zostanie dostarczona poprawna nazwa użytkownika, aplikacja może wykonać

różne zapytania do bazy danych w celu pobrania informacji o koncie i zaktualizowania dziennika kontroli. Może również wykonywać operacje wymagające dużej mocy obliczeniowej w celu sprawdzenia poprawności dostarczonego hasła względem przechowywanego skrótu. Jeśli osoba atakująca może wykryć tę różnicę czasu, może wykorzystać ją do wyliczenia prawidłowych nazw użytkowników.

* Niektóre funkcje aplikacji mogą wykonywać działania na podstawie danych wprowadzonych przez użytkownika, które wygasają, jeśli przesłane dane są nieprawidłowe. Na przykład aplikacja może wykorzystywać plik cookie do przechowywania adresu hosta znajdującego się za frontowym modułem równoważenia obciążenia. Osoba atakująca może być w stanie manipulować tym adresem w celu wyszukania serwerów WWW w wewnętrznej sieci organizacji. W przypadku podania adresu rzeczywistego serwera, który nie jest częścią infrastruktury aplikacji, aplikacja może natychmiast zwrócić błąd. W przypadku podania nieistniejącego adresu aplikacja może przekroczyć limit czasu próby skontaktowania się z tym adresem przed zwróceniem tego samego błędu ogólnego. Możesz użyć liczników czasu reakcji w tabeli wyników Burp Intruder, aby ułatwić to testowanie. Pamiętaj, że te kolumny są domyślnie ukryte, ale można je wyświetlić za pomocą menu Kolumny.

Zapobieganie wyciekom informacji

Chociaż zapobieganie ujawnieniu absolutnie jakichkolwiek informacji, które mogą być przydatne dla osoby atakującej, może nie być wykonalne lub pożądane, można podjąć różne stosunkowo proste środki w celu ograniczenia wycieku informacji do minimum i wstrzymania najbardziej wrażliwych danych, które mogą krytycznie zagrozić bezpieczeństwa aplikacji, jeśli zostaną ujawnione atakującemu.

Użyj ogólnych komunikatów o błędach

Aplikacja nigdy nie powinna zwracać pełnych komunikatów o błędach ani informacji debugowania do przeglądarki użytkownika. W przypadku wystąpienia nieoczekiwanego zdarzenia (takiego jak błąd w zapytaniu do bazy danych, niepowodzenie odczytu pliku z dysku lub wyjątek w wywołaniu zewnętrznego API) aplikacja powinna zwrócić ten sam ogólny komunikat informujący użytkownika o wystąpieniu błędu . Jeśli konieczne jest zapisanie informacji debugowania do celów wsparcia lub diagnostyki, powinno to być przechowywane w dzienniku po stronie serwera, który nie jest publicznie dostępny. Numer indeksu do odpowiedniego wpisu dziennika może zostać zwrócony użytkownikowi, co umożliwi mu zgłoszenie tego podczas kontaktu z działem pomocy, jeśli zajdzie taka potrzeba. Większość platform aplikacji i serwerów WWW można skonfigurować tak, aby maskowały błędy informacji przed zwróceniem do przeglądarki:

* W ASP.NET można pominąć szczegółowe komunikaty o błędach za pomocą elementu customErrors pliku Web.config, ustawiając atrybut mode na On lub RemoteOnly i określając niestandardową stronę błędu w węźle defaultRedirect.

* W platformie Java można konfigurować niestandardowe komunikaty o błędach za pomocą elementu error-page w pliku web.xml. Możesz użyć węzła typu wyjątku, aby określić typ wyjątku Java, lub możesz użyć węzła kodu błędu, aby określić kod stanu HTTP. Możesz użyć węzła lokalizacji, aby ustawić niestandardową stronę, która ma być wyświetlana w przypadku określonego błędu.

* W usługach Microsoft IIS można określić niestandardowe strony błędów dla różnych kodów stanu HTTP za pomocą karty Błędy niestandardowe na karcie Właściwości witryny internetowej. Dla każdego kodu stanu można ustawić inną stronę niestandardową, aw razie potrzeby dla poszczególnych katalogów.

* W Apache niestandardowe strony błędów można skonfigurować za pomocą dyrektywy ErrorDocument w httpd.conf:

ErrorDocument 500 /generalerror.html

Chroń poufne informacje

W miarę możliwości aplikacja nie powinna publikować informacji, które mogą być przydatne dla atakującego, w tym nazw użytkowników, wpisów w dzienniku i szczegółów profilu użytkownika. Jeżeli niektórzy użytkownicy potrzebują dostępu do tych informacji, należy je chronić za pomocą skutecznych kontroli dostępu i udostępniać tylko wtedy, gdy jest to bezwzględnie konieczne. W przypadkach, gdy poufne informacje muszą zostać ujawnione upoważnionemu użytkownikowi (na przykład, gdy użytkownicy mogą aktualizować własne informacje o koncie), istniejące dane nie powinny być ujawniane, jeśli nie jest to konieczne. Na przykład przechowywane numery kart kredytowych powinny być wyświetlane w skróconej formie, a pola hasła nigdy nie powinny być wstępnie wypełniane, nawet jeśli są zamaskowane na ekranie. Te środki obronne pomagają złagodzić wpływ wszelkich poważnych luk w zabezpieczeniach, które mogą istnieć w programie podstawowego mechanizm bezpieczeństwa aplikacji, takie jak uwierzytelnianie, zarządzanie sesjami i kontrola dostępu.

Zminimalizuj wyciek informacji po stronie klienta

Tam, gdzie to możliwe, banery serwisowe powinny być usuwane lub modyfikowane, aby zminimalizować ujawnianie określonych wersji oprogramowania i tak dalej. Kroki konieczne do wdrożenia tego środka zależą od stosowanych technologii. Na przykład w usługach Microsoft IIS nagłówek serwera można usunąć za pomocą narzędzia URLScan w narzędziu IISLockDown. W późniejszych wersjach Apache można to osiągnąć za pomocą modułu mod_headers. Ponieważ informacje te mogą ulec zmianie, zaleca się zapoznanie się z dokumentacją serwera przed dokonaniem jakichkolwiek modyfikacji. Wszystkie komentarze powinny zostać usunięte z kodu po stronie klienta, który jest wdrażany w działającym środowisku produkcyjnym, w tym z całego kodu HTML i JavaScript. Należy zwrócić szczególną uwagę na wszelkie komponenty rozszerzeń przeglądarki, takie jak aplety Java i formanty ActiveX. W tych komponentach nie należy ukrywać żadnych poufnych informacji. Wykwalifikowany atakujący może zdekompilować lub poddać inżynierii wstecznej te komponenty, aby skutecznie odzyskać ich kod źródłowy.

Streszczenie

Wyciek zbędnych informacji często nie stanowi istotnej wady bezpieczeństwa aplikacji. Nawet bardzo rozwlekłe ślady stosu i inne komunikaty debugowania mogą czasami zapewniać niewielki wpływ na próbę zaatakowania aplikacji. Jednak w innych przypadkach możesz odkryć źródła informacji, które mają wielką wartość przy opracowywaniu ataku. Na przykład możesz znaleźć listy nazw użytkowników, dokładne wersje składników oprogramowania lub wewnętrzną strukturę i funkcjonalność logiki aplikacji po stronie serwera. Ze względu na tę możliwość każdy poważny atak na aplikację powinien obejmować badanie kryminalistyczne zarówno samej aplikacji, jak i publicznie dostępnych zasobów, aby można było zebrać wszelkie informacje, które mogą być przydatne przy formułowaniu ataków na nią. W niektórych przypadkach informacje zebrane w ten sposób mogą stanowić podstawę do całkowitego skompromitowania aplikacji, która je ujawniła.

Pytania

1. Podczas sondowania luk w zabezpieczeniach związanych z iniekcją SQL żądasz następującego adresu URL:

https://wahh-app.com/list.aspx?artist=foo'+having+1%3d1--

Pojawia się następujący komunikat o błędzie:

Server: Msg 170, Level 15, State 1, Line 1

Line 1: Incorrect syntax near 'having1'.

Co możesz z tego wywnioskować? Czy aplikacja zawiera warunek, który można wykorzystać?

2. Podczas przeprowadzania testów rozmytych różnych parametrów aplikacja zwraca następujący komunikat o błędzie:

Warning: mysql_connect() [function.mysql-connect]: Access denied for user 'premiumdde'@'localhost' (using password: YES) in

/home/doau/public_html/premiumdde/directory on line 15

Warning: mysql_select_db() [function.mysql-select-db]: Access denied for user 'nobody'@'localhost' (using password: NO) in

/home/doau/public_html/premiumdde/directory on line 16

Warning: mysql_select_db() [function.mysql-select-db]: A link to

the server could not be established in

/home/doau/public_html/premiumdde/directory on line 16

Warning: mysql_query() [function.mysql-query]: Access denied for

user 'nobody'@'localhost' (using password: NO) in

/home/doau/public_html/premiumdde/directory on line 448

Jakie przydatne informacje możesz z tego wyciągnąć?

3. Podczas mapowania aplikacji odkrywasz ukryty katalog na serwerze, który ma włączone wyświetlanie katalogów i wydaje się, że zawiera pewną liczbę starych skryptów. Żądanie jednego z tych skryptów zwraca następujący komunikat o błędzie:

CGIWrap Docs: http://wahh-app.com/cgiwrap-docs/

Contact EMail: helpdesk@wahh-app.com

Server Data:

Server Administrator/Contact: helpdesk@wahh-app.com

Server Name: wahh-app.com

Server Port: 80

Server Protocol: HTTP/1.1

Request Data:

User Agent/Browser: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT

5.1; .NET CLR 2.0.50727; FDM; InfoPath.1; .NET CLR 1.1.4322)

Request Method: GET

Remote Address: 192.168.201.19

Remote Port: 57961

Referring Page: http://wahh-app.com/cgi-bin/cgiwrap/fodd

Co spowodowało ten błąd i jakie typowe luki w zabezpieczeniach aplikacji internetowych należy szybko sprawdzić?

4. Próbujesz sprawdzić funkcję parametru żądania w celu określenia celu w aplikacji. Żądasz adresu URL:

https://wahh-app.com/agents/checkcfg.php?name=admin&id=13&log=1

Aplikacja zwraca następujący komunikat o błędzie:

Warning: mysql_connect() [function.mysql-connect]: Can't connect to MySQL server on 'admin' (10013) in

/var/local/www/include/dbconfig.php on line 23

Co spowodowało ten komunikat o błędzie i jakie luki w zabezpieczeniach należy sprawdzić w związku z tym?

5. Zafałszowując zapytanie dotyczące różnych kategorii luk w zabezpieczeniach, umieszczasz po kolei pojedynczy cudzysłów w ramach każdego parametru żądania. Jeden z wyników zawiera kod stanu HTTP 500, wskazujący na potencjalną iniekcję SQL. Sprawdzasz pełną treść wiadomości, która wygląda następująco:

Microsoft VBScript runtime error '800a000d'

Type mismatch: ' [string: "'"]'

/scripts/confirmOrder.asp, line 715

Is the application vulnerable?