Technologie aplikacji internetowych

Aplikacje internetowe wykorzystują niezliczone technologie do implementacji swojej funkcjonalności. Ten rozdział jest krótkim wprowadzeniem do kluczowych technologii, z którymi możesz się spotkać podczas atakowania aplikacji internetowych. Przyjrzymy się protokołowi HTTP, technologiom powszechnie stosowanym po stronie serwera i klienta oraz schematom kodowania używanym do reprezentacji danych w różnych sytuacjach. Technologie te są na ogół łatwe do zrozumienia, a zrozumienie ich odpowiednich funkcji jest kluczem do przeprowadzania skutecznych ataków na aplikacje internetowe. Jeśli jesteś już zaznajomiony z kluczowymi technologiami używanymi w aplikacjach internetowych, możesz przejrzeć ten rozdział, aby upewnić się, że nie oferuje on nic nowego. Jeśli dopiero uczysz się, jak działają aplikacje internetowe, powinieneś przeczytać tę część, zanim przejdziesz do dalszych rozdziałów poświęconych konkretnym lukom w zabezpieczeniach.

Protokół HTTP

Protokół przesyłania hipertekstu (HTTP) jest podstawowym protokołem komunikacyjnym używanym do uzyskiwania dostępu do sieci World Wide Web i jest używany przez wszystkie dzisiejsze aplikacje internetowe. Jest to prosty protokół, który został pierwotnie opracowany do pobierania statycznych zasobów tekstowych. Od tego czasu został rozszerzony i wykorzystany na różne sposoby, aby umożliwić obsługę złożonych aplikacji rozproszonych, które są obecnie powszechne. Protokół HTTP wykorzystuje model oparty na komunikatach, w którym klient wysyła komunikat żądania, a serwer zwraca komunikat odpowiedzi. Protokół jest zasadniczo bezpołączeniowy: chociaż protokół HTTP wykorzystuje stanowy protokół TCP jako mechanizm transportowy, każda wymiana żądania i odpowiedzi jest transakcją autonomiczną i może wykorzystywać inne połączenie TCP.

Żądania HTTP

Wszystkie komunikaty HTTP (żądania i odpowiedzi) składają się z co najmniej jednego nagłówka, każdy w osobnej linii, po którym następuje obowiązkowa pusta linia i opcjonalna treść wiadomości. Typowe żądanie HTTP wygląda następująco:

GET /auth/488/YourDetails.ashx?uid=129 HTTP/1.1

Accept: application/x-ms-application, image/jpeg, application/xaml+xml,

image/gif, image/pjpeg, application/x-ms-xbap, application/x-shockwaveflash,

/

Referer: https://mdsec.net/auth/488/Home.ashx

Accept-Language: en-GB

User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64;

Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR

3.0.30729; .NET4.0C; InfoPath.3; .NET4.0E; FDM; .NET CLR 1.1.4322)

Accept-Encoding: gzip, deflate

Host: mdsec.net

Connection: Keep-Alive

Cookie: SessionId=5B70C71F3FD4968935CDB6682E545476

Pierwsza linia każdego żądania HTTP składa się z trzech elementów oddzielonych spacjami:

* Czasownik wskazujący metodę HTTP. Najczęściej stosowaną metodą jest GET, której funkcją jest pobranie zasobu z serwera WWW. Żądania GET nie mają treści wiadomości, więc żadne dalsze dane nie pojawiają się po pustym wierszu po nagłówkach wiadomości.

* Żądany adres URL. Adres URL zwykle działa jako nazwa żądanego zasobu wraz z opcjonalnym ciągiem zapytania zawierającym parametry, które klient przekazuje do tego zasobu. Ciąg zapytania jest wskazywany przez ? znak w adresie URL. Przykład zawiera pojedynczy parametr o nazwie uid i wartości 129.

* Używana wersja HTTP. Jedynymi powszechnie używanymi wersjami HTTP w Internecie są 1.0 i 1.1, a większość przeglądarek domyślnie używa wersji 1.1. Istnieje kilka różnic między specyfikacjami tych dwóch wersji; jednak jedyną różnicą, którą prawdopodobnie napotkasz podczas atakowania aplikacji internetowych, jest to, że w wersji 1.1 nagłówek żądania hosta jest obowiązkowy.

Oto kilka innych interesujących punktów w żądaniu próbki:

* Nagłówek Referer służy do wskazania adresu URL, z którego pochodzi żądanie (na przykład dlatego, że użytkownik kliknął link na tej stronie). Należy zauważyć, że ten nagłówek został błędnie napisany w oryginalnej specyfikacji HTTP, a błędna wersja została zachowana do dziś.

* Nagłówek User-Agent służy do dostarczania informacji o przeglądarce lub innym oprogramowaniu klienckim, które wygenerowało żądanie. Należy pamiętać, że większość przeglądarek zawiera przedrostek Mozilla ze względów historycznych. Był to ciąg User-Agent używany przez pierwotnie dominującą przeglądarkę Netscape, a inne przeglądarki chciały potwierdzić na stronach internetowych, że są zgodne z tym standardem. Podobnie jak w przypadku wielu dziwactw z historii komputerów, stało się to tak ugruntowane, że nadal jest zachowywane, nawet w aktualnej wersji Internet Explorera, który wykonał żądanie pokazane w przykładzie.

* Nagłówek hosta określa nazwę hosta, która pojawiła się w pełnym adresie URL, do którego uzyskuje się dostęp. Jest to konieczne, gdy wiele witryn jest hostowanych na tym samym serwerze, ponieważ adres URL wysyłany w pierwszym wierszu żądania zwykle nie zawiera nazwy hosta. (Zobacz rozdział 17, aby uzyskać więcej informacji o wirtualnych witrynach internetowych).

* Nagłówek Cookie służy do przesyłania dodatkowych parametrów, które serwer wydał klientowi.

Odpowiedzi HTTP

Typowa odpowiedź HTTP wygląda następująco:

HTTP/1.1 200 OK

Date: Tue, 19 Apr 2011 09:23:32 GMT

Server: Microsoft-IIS/6.0

X-Powered-By: ASP.NET

Set-Cookie: tracking=tI8rk7joMx44S2Uu85nSWc

X-AspNet-Version: 2.0.50727

Cache-Control: no-cache

Pragma: no-cache

Expires: Thu, 01 Jan 1970 00:00:00 GMT

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

Content-Length: 1067

<!DOCTYPE html PUBLIC “-//W3C//DTD XHTML 1.0 Transitional//EN” “http://

www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd”><html xmlns=”http://

www.w3.org/1999/xhtml” ><head><title>Your details</title>

Pierwszy wiersz każdej odpowiedzi HTTP składa się z trzech elementów oddzielonych spacjami:

* Używana wersja HTTP.

* Numeryczny kod stanu wskazujący wynik żądania. 200 to najpopularniejszy kod stanu; oznacza to, że żądanie powiodło się i że żądany zasób jest zwracany.

* Tekstowe „wyrażenie uzasadnienia” dokładniej opisujące stan odpowiedzi. Może to mieć dowolną wartość i nie jest wykorzystywane w żadnym celu przez obecne przeglądarki. Oto kilka innych interesujących punktów w odpowiedzi:

* Nagłówek serwera zawiera baner wskazujący używane oprogramowanie serwera WWW, a czasami inne szczegóły, takie jak zainstalowane moduły i system operacyjny serwera. Zawarte informacje mogą, ale nie muszą być dokładne.

* Nagłówek Set-Cookie wysyła do przeglądarki kolejny plik cookie; jest to przesyłane z powrotem w nagłówku Cookie kolejnych żądań do tego serwera.

* Nagłówek Pragma instruuje przeglądarkę, aby nie zapisywała odpowiedzi w swojej pamięci podręcznej. Nagłówek Expires wskazuje, że treść odpowiedzi wygasła w przeszłości i dlatego nie powinna być buforowana. Instrukcje te są często wydawane podczas zwracania zawartości dynamicznej, aby zapewnić, że przeglądarki uzyskają nową wersję tej zawartości przy kolejnych okazjach.

* Prawie wszystkie odpowiedzi HTTP zawierają treść wiadomości po pustym wierszu po nagłówkach. Nagłówek Content-Type wskazuje, że treść tej wiadomości zawiera dokument HTML.

* Nagłówek Content-Length wskazuje długość treści wiadomości w bajtach.

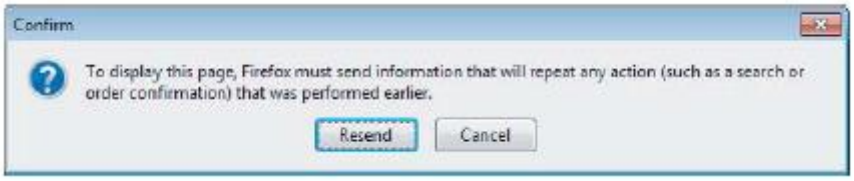
Metody HTTP

Podczas atakowania aplikacji internetowych będziesz miał do czynienia prawie wyłącznie z najczęściej używanymi metodami: GET i POST. Musisz zdawać sobie sprawę z kilku ważnych różnic między tymi metodami, ponieważ mogą one wpłynąć na bezpieczeństwo aplikacji, jeśli zostaną przeoczone.

Metoda GET jest przeznaczona do pobierania zasobów. Można go użyć do wysłania parametrów do żądanego zasobu w ciągu zapytania adresu URL. Dzięki temu użytkownicy mogą dodawać zakładki do adresu URL zasobu dynamicznego, którego mogą ponownie użyć. Lub inni użytkownicy mogą odzyskać równoważny zasób przy kolejnej okazji (jak w zapytaniu z zakładkami). Adresy URL są wyświetlane na ekranie i są rejestrowane w różnych miejscach, takich jak historia przeglądarki i dzienniki dostępu do serwera WWW. Są one również przesyłane w nagłówku strony odsyłającej do innych witryn, gdy następuje kliknięcie linków zewnętrznych. Z tych powodów ciąg zapytania nie powinien być używany do przesyłania żadnych poufnych informacji. Metoda POST jest przeznaczona do wykonywania akcji.

Dzięki tej metodzie parametry żądania mogą być wysyłane zarówno w ciągu zapytania URL, jak iw treści wiadomości. Chociaż adres URL nadal można dodać do zakładek, wszelkie przesłane parametry

treść wiadomości zostanie usunięta z zakładki. Parametry te zostaną również wykluczone z różnych lokalizacji, w których przechowywane są dzienniki adresów URL oraz z nagłówka strony odsyłającej. Ponieważ metoda POST jest przeznaczona do wykonywania działań, jeśli użytkownik kliknie przycisk Wstecz w przeglądarce, aby powrócić do strony, do której uzyskano dostęp za pomocą tej metody, przeglądarka nie wysyła automatycznie żądania ponownie. Zamiast tego ostrzega użytkownika o tym, co zamierza zrobić, jak pokazano na rysunku.



Zapobiega to nieświadomemu wykonywaniu akcji więcej niż jeden raz. Z tego powodu żądania POST powinny być zawsze używane podczas wykonywania akcji.

Oprócz metod GET i POST protokół HTTP obsługuje wiele innych metod, które zostały stworzone do określonych celów. Oto inne, o których najprawdopodobniej będziesz potrzebować wiedzy:

* HEAD działa w taki sam sposób jak żądanie GET, z tą różnicą, że serwer nie powinien zwracać treści wiadomości w swojej odpowiedzi. Serwer powinien zwrócić te same nagłówki, które zwróciłby w odpowiednim żądaniu GET. Dlatego ta metoda może być użyta do sprawdzenia, czy zasób jest obecny przed wysłaniem żądania GET.

* TRACE jest przeznaczony do celów diagnostycznych. Serwer powinien zwrócić w treści odpowiedzi dokładną treść otrzymanego komunikatu żądania. Można to wykorzystać do wykrycia wpływu dowolnych serwerów proxy między klientem a serwerem, które mogą manipulować żądaniem.

* OPTIONS prosi serwer o podanie metod HTTP dostępnych dla określonego zasobu. Serwer zazwyczaj zwraca odpowiedź zawierającą nagłówek Allow, który zawiera listę dostępnych metod.

* PUT próbuje przesłać określony zasób na serwer, korzystając z treści zawartej w treści żądania. Jeśli ta metoda jest włączona, możesz wykorzystać ją do ataku na aplikację, na przykład przesyłając dowolny skrypt i uruchamiając go na serwerze.

Istnieje wiele innych metod HTTP, które nie są bezpośrednio związane z atakowaniem aplikacji internetowych. Jednak serwer WWW może narazić się na atak, jeśli dostępne są pewne niebezpieczne metody.

Adresy URL

Jednolity lokalizator zasobów (URL) to unikalny identyfikator zasobu sieciowego, za pomocą którego można go pobrać. Format większości adresów URL jest następujący:

protocol://hostname[:port]/[path/]file[?param=value]

Kilka elementów w tym schemacie jest opcjonalnych. Numer portu jest zwykle dołączany tylko wtedy, gdy różni się od domyślnego używanego przez odpowiedni protokół. Adres URL użyty do wygenerowania żądania HTTP pokazanego wcześniej jest następujący:

https://mdsec.net/auth/488/YourDetails.ashx?uid=129

Oprócz tej bezwzględnej postaci, adresy URL mogą być określone względem określonego hosta lub względem określonej ścieżki na tym hoście. Na przykład:

/auth/488/YourDetails.ashx?uid=129

YourDetails.ashx?uid=129

Te względne formy są często używane na stronach internetowych do opisania nawigacji w witrynie lub samej aplikacji.

UWAGA: Możesz spotkać się z terminem URI (lub jednolitym identyfikatorem zasobów) używanym zamiast adresu URL, ale tak naprawdę jest on używany tylko w formalnych specyfikacjach i przez tych, którzy chcą pokazać swoją pedanterię.

REST

Representational state transfer (REST) to styl architektury systemów rozproszonych, w którym żądania i odpowiedzi zawierają reprezentacje bieżącego stanu zasobów systemu. Podstawowe technologie stosowane w sieci World Wide Web, w tym protokół HTTP i format adresów URL, są zgodne ze stylem architektonicznym REST. Chociaż adresy URL zawierające parametry w ciągu zapytania same w sobie są zgodne z ograniczeniami REST, termin „URL w stylu REST" jest często używany do oznaczenia adresu URL, który zawiera parametry w ścieżce pliku URL, a nie w ciągu zapytania. Na przykład następujący adres URL zawierający ciąg zapytania:

http://wahh-app.com/search?make=ford&model=pinto

odpowiada następującemu adresowi URL zawierającemu parametry „w stylu REST":

http://wahh-app.com/search/ford/pinto

Część 4 opisuje, w jaki sposób należy wziąć pod uwagę te różne style parametrów podczas mapowania zawartości i funkcjonalności aplikacji oraz identyfikowania jej kluczowej powierzchni ataku.

Nagłówki HTTP

HTTP obsługuje dużą liczbę nagłówków, z których niektóre są przeznaczone do określonych nietypowych celów. Niektóre nagłówki mogą być używane zarówno dla żądań, jak i odpowiedzi, a inne są specyficzne dla jednego z tych typów wiadomości. W poniższych sekcjach opisano nagłówki, które można napotkać podczas atakowania aplikacji internetowych.

Nagłówki ogólne

* Połączenie mówi drugiemu końcowi komunikacji, czy powinien zamknąć połączenie TCP po zakończeniu transmisji HTTP, czy też pozostawić je otwarte dla dalszych komunikatów.

* Content-Encoding określa, jaki rodzaj kodowania jest używany dla treści zawartej w treści wiadomości, na przykład gzip, który jest używany przez niektóre aplikacje do kompresji odpowiedzi w celu szybszej transmisji.

* Content-Length określa długość treści wiadomości w bajtach (z wyjątkiem odpowiedzi na żądania HEAD, kiedy wskazuje długość treści w odpowiedzi na odpowiednie żądanie GET).

* Content-Type określa typ treści zawartej w treści wiadomości, np. text/html dla dokumentów HTML.

* Transfer-Encoding określa kodowanie, które zostało wykonane w treści wiadomości w celu ułatwienia jej przesyłania przez HTTP. Zwykle jest używany do określania kodowania fragmentarycznego, gdy jest ono stosowane.

Nagłówki żądań

* Akceptuj mówi serwerowi, jakie rodzaje treści klient jest skłonny zaakceptować, takie jak typy obrazów, formaty dokumentów biurowych i tak dalej.

* Accept-Encoding mówi serwerowi, jakie rodzaje kodowania treści klient jest skłonny zaakceptować.

* Autoryzacja przesyła poświadczenia do serwera dla jednego z wbudowanych typów uwierzytelniania HTTP.

* Cookie przesyła do serwera pliki cookie, które serwer wcześniej wydał.

* Host określa nazwę hosta, która pojawiła się w żądanym pełnym adresie URL.

* If-Modified-Since określa, kiedy przeglądarka ostatnio otrzymała żądany zasób. Jeśli zasób nie zmienił się od tego czasu, serwer może poinstruować klienta, aby użył jego kopii z pamięci podręcznej, używając odpowiedzi z kodem statusu 304.

* If-None-Match określa znacznik encji, który jest identyfikatorem oznaczającym zawartość treści wiadomości. Przeglądarka przesyła tag podmiotu, który serwer wydał z żądanym zasobem, kiedy został on ostatnio odebrany. Serwer może użyć znacznika encji, aby określić, czy przeglądarka może używać kopii zasobu zapisanej w pamięci podręcznej.

* Pochodzenie jest używane w międzydomenowych żądaniach Ajax do wskazania domeny, z której pochodzi żądanie.

* Referer określa adres URL, z którego pochodzi bieżące żądanie.

* User-Agent dostarcza informacji o przeglądarce lub innym oprogramowaniu klienckim, które wygenerowało żądanie.

Nagłówki odpowiedzi

* Access-Control-Allow-Origin wskazuje, czy zasób może zostać pobrany za pośrednictwem międzydomenowych żądań Ajax.

* Kontrola pamięci podręcznej przekazuje do przeglądarki dyrektywy buforowania (na przykład no-cache).

* ETag określa tag podmiotu. Klienci mogą przesyłać ten identyfikator w przyszłych żądaniach dotyczących tego samego zasobu w nagłówku If-None-Match, aby powiadomić serwer, która wersja zasobu aktualnie znajduje się w pamięci podręcznej przeglądarki.

* Expires informuje przeglądarkę, jak długo treść wiadomości jest ważna. Do tego czasu przeglądarka może korzystać z buforowanej kopii tego zasobu.

* Lokalizacja jest używana w odpowiedziach przekierowania (tych, które mają kod stanu zaczynający się od 3) w celu określenia celu przekierowania.

* Pragma przekazuje do przeglądarki dyrektywy buforowania (na przykład no-cache).

* Serwer dostarcza informacji o używanym oprogramowaniu serwera WWW.

* Set-Cookie wysyła do przeglądarki pliki cookie, które przesyła z powrotem do serwera w kolejnych żądaniach.

* WWW-Authenticate jest używany w odpowiedziach, które mają kod stanu 401, aby podać szczegółowe informacje na temat typów uwierzytelniania obsługiwanych przez serwer.

* X-Frame-Options wskazuje, czy i jak bieżąca odpowiedź może zostać załadowana w ramce przeglądarki

Ciasteczka

Pliki cookie to kluczowa część protokołu HTTP, na którym opiera się większość aplikacji internetowych. Często mogą być wykorzystywane jako narzędzie do wykorzystywania luk w zabezpieczeniach. Mechanizm cookies umożliwia serwerowi przesyłanie do klienta danych, które klient przechowuje i ponownie przesyła do serwera. W przeciwieństwie do innych typów parametrów żądania (tych w ciągu zapytania URL lub w treści wiadomości), pliki cookie są nadal przesyłane ponownie w każdym kolejnym żądaniu bez konieczności wykonywania jakichkolwiek działań przez aplikację lub użytkownika. Serwer wysyła plik cookie, używając nagłówka odpowiedzi Set-Cookie, jak widać:

Set-Cookie: tracking=tI8rk7joMx44S2Uu85nSWc

Następnie przeglądarka użytkownika automatycznie dodaje następujący nagłówek do kolejnych żądań z powrotem do tego samego serwera:

Cookie: tracking=tI8rk7joMx44S2Uu85nSWc

Pliki cookie zwykle składają się z pary nazwa/wartość, jak pokazano, ale mogą składać się z dowolnego ciągu znaków niezawierającego spacji. Wiele plików cookie można wystawić, używając wielu nagłówków Set-Cookie w odpowiedzi serwera. Są one przesyłane z powrotem do serwera w tym samym nagłówku plików cookie, ze średnikiem oddzielającym poszczególne pliki cookie. Oprócz rzeczywistej wartości pliku cookie, nagłówek Set-Cookie może zawierać dowolny z poniższych opcjonalnych atrybutów, których można użyć do kontrolowania sposobu, w jaki przeglądarka obsługuje plik cookie:

* wygasa ustawia datę, do której plik cookie jest ważny. Powoduje to, że przeglądarka zapisuje plik cookie w pamięci trwałej i jest on ponownie używany w kolejnych sesjach przeglądarki, aż do osiągnięcia daty wygaśnięcia. Jeśli ten atrybut nie jest ustawiony, plik cookie jest używany tylko w bieżącej sesji przeglądarki.

* domena określa domenę, dla której plik cookie jest ważny. Musi to być ta sama domena lub domena nadrzędna, z której pochodzi plik cookie.

* ścieżka określa ścieżkę adresu URL, dla którego plik cookie jest ważny.

* bezpieczny — jeśli ten atrybut jest ustawiony, plik cookie będzie przesyłany tylko w żądaniach HTTPS.

* HttpOnly — jeśli ten atrybut jest ustawiony, nie można uzyskać bezpośredniego dostępu do pliku cookie za pośrednictwem kodu JavaScript po stronie klienta.

Każdy z tych atrybutów plików cookie może mieć wpływ na bezpieczeństwo aplikacji. Główny wpływ ma na zdolność atakującego do bezpośredniego kierowania ataków na innych użytkowników aplikacji.

Kody stanu

Każdy komunikat odpowiedzi HTTP musi zawierać w pierwszym wierszu kod stanu, wskazujący wynik żądania. Kody stanu dzielą się na pięć grup, zgodnie z pierwszą cyfrą kodu:

* 1xx — Informacyjny.

* 2xx — Żądanie powiodło się.

* 3xx — Klient zostaje przekierowany do innego zasobu.

* 4xx — Żądanie zawiera jakiś błąd.

* 5xx — Serwer napotkał błąd podczas realizacji żądania.

Istnieje wiele określonych kodów statusu, z których wiele jest używanych tylko w szczególnych okolicznościach. Oto kody stanu, które najprawdopodobniej napotkasz podczas atakowania aplikacji internetowej, wraz z typową frazą powodu, która jest z nimi powiązana:

* 100 Kontynuuj jest wysyłane w pewnych okolicznościach, gdy klient przesyła żądanie zawierające treść. Odpowiedź wskazuje, że nagłówki żądania zostały odebrane i że klient powinien kontynuować wysyłanie treści. Serwer zwraca drugą odpowiedź, gdy żądanie zostanie zakończone.

* 200 OK wskazuje, że żądanie zakończyło się pomyślnie i że treść odpowiedzi zawiera wynik żądania.

* 201 Created jest zwracane w odpowiedzi na żądanie PUT, aby wskazać, że żądanie się powiodło.

* 301 Moved Permanently przekierowuje przeglądarkę na stałe do innego adresu URL, który jest określony w nagłówku Lokalizacja. Klient powinien w przyszłości używać nowego adresu URL zamiast oryginalnego.

* 302 Found powoduje tymczasowe przekierowanie przeglądarki do innego adresu URL, który jest określony w nagłówku lokalizacji. Klient powinien powrócić do pierwotnego adresu URL w kolejnych żądaniach.

* 304 Not Modified instruuje przeglądarkę, aby użyła kopii żądanego zasobu z pamięci podręcznej. Serwer używa nagłówków żądań If-Modified-Since i If-None-Match, aby określić, czy klient ma najnowszą wersję zasobu.

* 400 Złe żądanie wskazuje, że klient przesłał nieprawidłowe żądanie HTTP. Prawdopodobnie spotkasz się z tym, gdy zmodyfikujesz żądanie w pewien nieprawidłowy sposób, na przykład umieszczając znak spacji w adresie URL.

* 401 Unauthorized wskazuje, że serwer wymaga uwierzytelnienia HTTP przed przyznaniem żądania. Nagłówek WWW-Authenticate zawiera szczegółowe informacje o obsługiwanych typach uwierzytelniania.

* 403 Forbidden wskazuje, że nikt nie ma dostępu do żądanego zasobu, niezależnie od uwierzytelnienia.

* 404 Not Found oznacza, że żądany zasób nie istnieje.

* 405 Metoda niedozwolona wskazuje, że metoda użyta w żądaniu nie jest obsługiwana dla określonego adresu URL. Na przykład możesz otrzymać ten kod stanu, jeśli spróbujesz użyć metody PUT, gdy nie jest ona obsługiwana.

* 413 Request Entity Too Large — Jeśli szukasz luk w zabezpieczeniach związanych z przepełnieniem bufora w kodzie natywnym i dlatego przesyłasz długie ciągi danych, oznacza to, że treść żądania jest zbyt duża, aby serwer mógł ją obsłużyć.

* Zbyt długi identyfikator żądania 414 jest podobny do odpowiedzi 413. Wskazuje, że adres URL użyty w żądaniu jest zbyt duży, aby serwer mógł go obsłużyć.

* 500 Wewnętrzny błąd serwera wskazuje, że serwer napotkał błąd podczas realizacji żądania. Zwykle dzieje się tak, gdy prześlesz nieoczekiwane dane wejściowe, które spowodowały nieobsługiwany błąd gdzieś w trakcie przetwarzania aplikacji. Należy uważnie przejrzeć pełną treść odpowiedzi serwera, aby znaleźć wszelkie szczegóły wskazujące na charakter błędu.

* 503 Usługa niedostępna zwykle wskazuje, że chociaż web

sam serwer działa i może odpowiadać na żądania, aplikacja, do której uzyskuje się dostęp za pośrednictwem serwera, nie odpowiada. Powinieneś zweryfikować, czy jest to wynikiem jakiejkolwiek wykonanej przez Ciebie czynności.

HTTPS

Protokół HTTP wykorzystuje zwykły protokół TCP jako mechanizm transportowy, który jest niezaszyfrowany i dlatego może zostać przechwycony przez atakującego, który jest odpowiednio umieszczony w sieci. HTTPS jest zasadniczo tym samym protokołem warstwy aplikacji co HTTP, ale jest tunelowany przez mechanizm bezpiecznego transportu, Secure Sockets Layer (SSL). Chroni to prywatność i integralność danych przesyłanych przez sieć, zmniejszając możliwości nieinwazyjnych ataków przechwytujących. Żądania i odpowiedzi HTTP działają dokładnie w ten sam sposób, niezależnie od tego, czy do transportu używany jest protokół SSL.

UWAGA: SSL został całkowicie zastąpiony przez zabezpieczenia warstwy transportowej (TLS), ale ta ostatnia zwykle nadal jest określana przy użyciu starszej nazwy.

Serwery proxy HTTP

Serwer proxy HTTP to serwer, który pośredniczy w dostępie między przeglądarką klienta a docelowym serwerem WWW. Gdy przeglądarka została skonfigurowana do korzystania z serwera proxy, wysyła wszystkie swoje żądania do tego serwera. Serwer proxy przekazuje żądania do odpowiednich serwerów sieciowych i przekazuje ich odpowiedzi z powrotem do przeglądarki. Większość serwerów proxy zapewnia również dodatkowe usługi, w tym buforowanie, uwierzytelnianie,

i kontroli dostępu. Należy zdawać sobie sprawę z dwóch różnic w sposobie działania protokołu HTTP, gdy używany jest serwer proxy:

* Gdy przeglądarka wysyła niezaszyfrowane żądanie HTTP do serwera proxy, umieszcza w żądaniu pełny adres URL, w tym przedrostek protokołu http://, nazwę hosta serwera i numer portu, jeśli jest niestandardowy. Serwer proxy wyodrębnia nazwę hosta i port i używa ich do kierowania żądania do właściwego docelowego serwera WWW.

* Gdy używany jest HTTPS, przeglądarka nie może wykonać uzgadniania SSL z serwerem proxy, ponieważ spowodowałoby to przerwanie bezpiecznego tunelu i narażenie komunikacji na ataki przechwycenia. W związku z tym przeglądarka musi używać proxy jako przekaźnika na poziomie czystego protokołu TCP, który przekazuje wszystkie dane sieciowe w obu kierunkach między przeglądarką a docelowym serwerem internetowym, z którym przeglądarka wykonuje uzgadnianie SSL w normalny sposób. Aby ustanowić ten przekaźnik, przeglądarka wysyła żądanie HTTP do serwera

proxy przy użyciu metody CONNECT i podając nazwę hosta docelowego i numer portu jako adres URL. Jeśli serwer proxy zezwoli na żądanie, zwraca odpowiedź HTTP ze stanem 200, utrzymuje otwarte połączenie TCP i od tego momentu działa jako zwykły przekaźnik na poziomie TCP do docelowego serwera WWW.

W pewnym sensie najbardziej przydatnym elementem zestawu narzędzi podczas atakowania aplikacji internetowych jest wyspecjalizowany rodzaj serwera proxy, który znajduje się między przeglądarką a docelową witryną i umożliwia przechwytywanie i modyfikowanie wszystkich żądań i odpowiedzi, nawet tych korzystających z protokołu HTTPS. Zaczniemy sprawdzać, jak możesz użyć tego rodzaju narzędzia w następnej części.

Uwierzytelnianie HTTP

Protokół HTTP zawiera własne mechanizmy uwierzytelniania użytkowników przy użyciu różnych schematów uwierzytelniania, w tym następujących:

* Basic to prosty mechanizm uwierzytelniania, który wysyła poświadczenia użytkownika jako ciąg zakodowany w formacie Base64 w nagłówku żądania z każdą wiadomością.

* NTLM jest mechanizmem wyzwanie-odpowiedź i wykorzystuje wersję protokołu Windows NTLM.

* Digest jest mechanizmem challenge-response i wykorzystuje sumy kontrolne MD5 jednorazowego użytku z poświadczeniami użytkownika. Stosunkowo rzadko można spotkać te protokoły uwierzytelniania używane przez aplikacje internetowe wdrożone w Internecie. Są one częściej używane w organizacjach w celu uzyskania dostępu do usług opartych na intranecie.

POWSZECHNY MIT

„Uwierzytelnianie podstawowe nie jest bezpieczne”.

Ponieważ uwierzytelnianie podstawowe umieszcza poświadczenia w postaci niezaszyfrowanej w żądaniu HTTP, często stwierdza się, że protokół jest niezabezpieczony i nie powinien być używany. Ale uwierzytelnianie oparte na formularzach, stosowane przez wiele banków, również umieszcza dane uwierzytelniające w niezaszyfrowanej formie w żądaniu HTTP. Każda wiadomość HTTP może być chroniona przed atakami podsłuchiwania za pomocą HTTPS jako mechanizmu transportowego, co powinno być wykonywane przez każdą aplikację świadomą bezpieczeństwa. Przynajmniej jeśli chodzi o podsłuchiwanie, podstawowe uwierzytelnianie samo w sobie nie jest gorsze od metod stosowanych przez większość dzisiejszych aplikacji internetowych.

Funkcjonalność internetowa

Oprócz podstawowego protokołu komunikacyjnego używanego do przesyłania wiadomości między klientem a serwerem, aplikacje internetowe wykorzystują liczne technologie w celu zapewnienia swojej funkcjonalności. Każda w miarę funkcjonalna aplikacja może wykorzystywać dziesiątki różnych technologii w komponentach serwera i klienta. Zanim przystąpisz do poważnego ataku na aplikację internetową, potrzebujesz podstawowej wiedzy na temat implementacji jej funkcjonalności, zachowania zastosowanych technologii oraz ich słabych punktów.

Funkcjonalność po stronie serwera

Wczesna sieć World Wide Web zawierała całkowicie statyczne treści. Witryny internetowe składały się z różnych zasobów, takich jak strony HTML i obrazy, które były po prostu ładowane na serwer WWW i dostarczane każdemu użytkownikowi, który ich zażądał. Za każdym razem, gdy żądano określonego zasobu, serwer odpowiadał tą samą treścią. Dzisiejsze aplikacje internetowe nadal zazwyczaj

wykorzystują znaczną liczbę zasobów statycznych. Jednak duża ilość treści, które prezentują użytkownikom, jest generowana dynamicznie. Kiedy użytkownik żąda dynamicznego zasobu, odpowiedź serwera jest tworzona w locie, a każdy użytkownik może otrzymać zawartość, która jest specjalnie dla niego dostosowana. Zawartość dynamiczna jest generowana przez skrypty lub inny kod wykonywany na serwerze. Skrypty te same w sobie są podobne do programów komputerowych. Mają różne dane wejściowe, wykonują na nich przetwarzanie i zwracają dane wyjściowe użytkownikowi.

Gdy przeglądarka użytkownika żąda zasobu dynamicznego, zwykle nie prosi po prostu o kopię tego zasobu. Ogólnie rzecz biorąc, przesyła również różne parametry wraz z żądaniem. To właśnie te parametry umożliwiają aplikacji serwerowej generowanie treści dostosowanych do indywidualnego użytkownika. Żądania HTTP mogą służyć do wysyłania parametrów do aplikacji na trzy główne sposoby:

* W ciągu zapytania adresu URL

* W ścieżce pliku adresów URL w stylu REST

* W plikach cookie HTTP

* W treści żądań przy użyciu metody POST

Oprócz tych głównych źródeł danych wejściowych aplikacja po stronie serwera może w zasadzie wykorzystywać dowolną część żądania HTTP jako dane wejściowe do swojego przetwarzania. Na przykład aplikacja może przetwarzać nagłówek User-Agent w celu wygenerowania treści zoptymalizowanej pod kątem typu używanej przeglądarki. Podobnie jak ogólnie oprogramowanie komputerowe, aplikacje internetowe wykorzystują szeroki zakres technologii po stronie serwera, aby zapewnić swoją funkcjonalność:

* Języki skryptowe, takie jak PHP, VBScript i Perl

* Platformy aplikacji internetowych, takie jak ASP.NET i Java

* Serwery sieci Web, takie jak Apache, IIS i Netscape Enterprise

* Bazy danych, takie jak MS-SQL, Oracle i MySQL

* Inne komponenty zaplecza, takie jak systemy plików, usługi sieciowe oparte na protokole SOAP i usługi katalogowe

Wszystkie te technologie i rodzaje luk, które mogą się z nimi pojawić, są szczegółowo omawiane w tej książce. W poniższych sekcjach opisano niektóre z najczęściej spotykanych platform i technologii aplikacji internetowych.

WSPÓLNY MIT

„Nasze aplikacje wymagają jedynie pobieżnej oceny bezpieczeństwa, ponieważ wykorzystują dobrze wykorzystywaną strukturę”. Korzystanie z dobrze używanego frameworka jest często przyczyną samozadowolenia w tworzeniu aplikacji internetowych, przy założeniu, że typowe luki w zabezpieczeniach, takie jak iniekcja SQL, są automatycznie unikane. Założenie to jest błędne z dwóch powodów. Po pierwsze, duża liczba luk w zabezpieczeniach aplikacji internetowych wynika z projektu aplikacji, a nie z jej implementacji, i jest niezależna od wybranego środowiska programistycznego lub języka. Po drugie, ponieważ framework zwykle wykorzystuje wtyczki i pakiety z najnowocześniejszych najnowszych repozytoriów, jest prawdopodobne, że te pakiety nie przeszły kontroli bezpieczeństwa. Co ciekawe, jeśli później w aplikacji zostanie wykryta luka, ci sami zwolennicy mitu chętnie zamienią się stronami i obwinią swój framework lub zewnętrzny pakiet!

Platforma Javy

Przez wiele lat Java Platform Enterprise Edition (wcześniej znana jako J2EE) była de facto standardem dla dużych aplikacji korporacyjnych. Pierwotnie opracowany przez Sun Microsystems, a obecnie należący do Oracle, nadaje się do wielowarstwowych architektur z równoważeniem obciążenia i dobrze nadaje się do modułowego programowania i ponownego wykorzystania kodu. Ze względu na długą historię i szerokie zastosowanie, dostępnych jest wiele wysokiej jakości narzędzi programistycznych, serwerów aplikacji i struktur, które mogą pomóc programistom. Platformę Java można uruchomić na kilku bazowych systemach operacyjnych, w tym Windows, Linux i Solaris. W opisach aplikacji internetowych opartych na języku Java często występuje kilka potencjalnie mylących terminów, o których warto wiedzieć:

* Enterprise Java Bean (EJB) jest stosunkowo ciężkim komponentem oprogramowania, który hermetyzuje logikę określonej funkcji biznesowej w aplikacji. EJB są przeznaczone do radzenia sobie z różnymi wyzwaniami technicznymi, z którymi muszą się zmierzyć twórcy aplikacji, takimi jak integralność transakcji.

* Zwykły stary obiekt Java (POJO) jest zwykłym obiektem Java, w odróżnieniu od obiektu specjalnego, takiego jak EJB. POJO zwykle jest używany do oznaczania obiektów, które są zdefiniowane przez użytkownika i są znacznie prostsze i lżejsze niż EJB i te używane w innych frameworkach.

* AJava Servlet to obiekt, który znajduje się na serwerze aplikacji i odbiera żądania HTTP od klientów oraz zwraca odpowiedzi HTTP. Implementacje serwletów mogą wykorzystywać liczne interfejsy ułatwiające tworzenie użytecznych aplikacji.

* Kontener internetowy Java to platforma lub silnik, który zapewnia środowisko wykonawcze dla aplikacji internetowych opartych na Javie. Przykładami kontenerów internetowych Java są Apache Tomcat, BEA WebLogic i JBoss.

Wiele aplikacji internetowych Java wykorzystuje komponenty innych firm i open source wraz z niestandardowym kodem. Jest to atrakcyjna opcja, ponieważ zmniejsza wysiłek programistyczny, a Java dobrze nadaje się do tego modułowego podejścia. Oto kilka przykładów komponentów powszechnie używanych do kluczowych funkcji aplikacji:

* Uwierzytelnianie — JAAS, ACEGI

* Warstwa prezentacji — SiteMesh, Tapestry

* Relacyjne mapowanie obiektów bazy danych — Hibernacja

* Logowanie — Log4J

Jeśli możesz określić, które pakiety open source są używane w aplikacji, którą atakujesz, możesz je pobrać i przeprowadzić przegląd kodu lub zainstalować, aby poeksperymentować. Luka w zabezpieczeniach którejkolwiek z nich może zostać wykorzystana do naruszenia szerszej aplikacji.

ASP.NET

ASP.NET to platforma aplikacji internetowych firmy Microsoft, która jest bezpośrednim konkurentem platformy Java. ASP.NET jest kilka lat młodszy od swojego odpowiednika, ale poczynił znaczące postępy na terytorium Javy. ASP.NET korzysta z platformy .NET Framework firmy Microsoft, która zapewnia maszynę wirtualną (Środowisko uruchomieniowe języka wspólnego) oraz zestaw potężnych interfejsów API. Dzięki temu aplikacje ASP.NET można pisać w dowolnym języku .NET, takim jak C# czy VB.NET. ASP.NET nadaje się do paradygmatu programowania sterowanego zdarzeniami, który jest

zwykle używany w konwencjonalnym oprogramowaniu komputerowym, zamiast podejścia opartego na skryptach stosowanego w większości wcześniejszych platform aplikacji internetowych. To, wraz z potężnymi narzędziami programistycznymi dostarczonymi z Visual Studio, sprawia, że tworzenie funkcjonalnej aplikacji internetowej jest niezwykle łatwe dla każdego, kto ma minimalne umiejętności programistyczne. Struktura ASP.NET pomaga chronić przed niektórymi typowymi lukami w zabezpieczeniach aplikacji sieci Web, takimi jak skrypty między witrynami, bez konieczności podejmowania jakichkolwiek działań przez programistę. Jednak jedną praktyczną wadą jego pozornej prostoty jest to, że wiele małych aplikacji ASP.NET jest w rzeczywistości tworzonych przez początkujących, którzy nie są świadomi podstawowych problemów bezpieczeństwa, z którymi borykają się aplikacje internetowe.

PHP

Język PHP wyłonił się z projektu hobbystycznego (skrót pierwotnie oznaczał „osobistą stronę główną"). Od tego czasu ewoluował prawie nie do poznania w bardzo potężny i bogaty framework do tworzenia aplikacji internetowych. Jest często używany w połączeniu z innymi wolnymi technologiami w tak zwanym stosie LAMP (składającym się z Linuksa jako systemu operacyjnego, Apache jako serwera WWW, MySQL jako serwera bazy danych i PHP jako języka programowania aplikacji internetowej) . Liczne aplikacje i komponenty open source zostały opracowane przy użyciu PHP. Wiele z nich zapewnia gotowe rozwiązania dla typowych funkcji aplikacji, które często są włączane do szerszych, niestandardowych aplikacji:

* Tablice ogłoszeń — PHPBB, PHP-Nuke

* Interfejs administracyjny — PHPMyAdmin

* Poczta internetowa — SquirrelMail, IlohaMail

* Galerie zdjęć — Galeria

* Wózki sklepowe — osCommerce, ECW-Shop

* Wiki — MediaWiki, WakkaWikki

Ponieważ PHP jest darmowy i łatwy w użyciu, często był językiem wybieranym przez wielu początkujących piszących aplikacje internetowe. Co więcej, projekt i domyślna konfiguracja frameworka PHP w przeszłości ułatwiały programistom nieświadome wprowadzanie błędów bezpieczeństwa do ich kodu. Czynniki te sprawiły, że aplikacje napisane w PHP są narażone na nieproporcjonalną liczbę luk w zabezpieczeniach. Ponadto w samej platformie PHP istniało kilka defektów, które często można było wykorzystać za pośrednictwem działających na niej aplikacji.

Ruby on Rails

Rails 1.0 został wydany w 2005 roku, z silnym naciskiem na architekturę Model-View-Controller. Kluczową siłą Rails jest zawrotna prędkość, z jaką można tworzyć w pełni rozwinięte aplikacje oparte na danych. Jeśli programista przestrzega stylu kodowania i konwencji nazewnictwa Railsów, Railsy mogą automatycznie wygenerować model dla zawartości bazy danych, działań kontrolera w celu jej modyfikacji oraz domyślnych widoków dla użytkownika aplikacji. Podobnie jak w przypadku każdej wysoce funkcjonalnej nowej technologii, w Ruby on Rails wykryto kilka luk, w tym możliwość ominięcia „trybu bezpiecznego", analogicznego do tego, który można znaleźć w PHP. Więcej informacji na temat ostatnich luk w zabezpieczeniach można znaleźć tutaj:

www.ruby-lang.org/en/security/

SQL

Structured Query Language (SQL) służy do uzyskiwania dostępu do danych w relacyjnych bazach danych, takich jak Oracle, serwer MS-SQL i MySQL. Zdecydowana większość dzisiejszych aplikacji internetowych wykorzystuje bazy danych oparte na SQL jako magazyn danych zaplecza, a prawie wszystkie funkcje aplikacji wymagają w jakiś sposób interakcji z tymi magazynami danych.

Relacyjne bazy danych przechowują dane w tabelach, z których każda zawiera określoną liczbę wierszy i kolumn. Każda kolumna reprezentuje pole danych, takie jak „nazwisko" lub „adres e-mail", a każdy wiersz reprezentuje element z wartościami przypisanymi do niektórych lub wszystkich tych pól. SQL używa zapytań do wykonywania typowych zadań, takich jak odczytywanie, dodawanie, aktualizowanie i usuwanie danych. Na przykład, aby pobrać adres e-mail użytkownika o określonej nazwie, aplikacja może wykonać następujące zapytanie:

select email from users where name = 'daf'

Aby zaimplementować potrzebną funkcjonalność, aplikacje internetowe mogą zawierać dane wejściowe dostarczone przez użytkownika w zapytaniach SQL, które są wykonywane przez bazę danych zaplecza. Jeśli ten proces nie zostanie przeprowadzony w bezpieczny sposób, osoby atakujące mogą wprowadzić złośliwe dane wejściowe, aby zakłócić działanie bazy danych i potencjalnie odczytać i zapisać poufne dane.

XML

Extensible Markup Language (XML) to specyfikacja kodowania danych w formie czytelnej dla maszyn. Jak każdy język znaczników, format XML dzieli dokument na treść (która jest danymi) i znaczniki (które opisują dane). Znaczniki są reprezentowane głównie za pomocą znaczników, którymi mogą być znaczniki początkowe, znaczniki końcowe lub znaczniki pustych elementów:

<tagname>

</tagname>

<tagname />

Znaczniki początkowe i końcowe są parowane w elementy i mogą hermetyzować treść dokumentu lub elementy podrzędne:

<pet>imbir</pet>

<pets><dog>miejsce</dog><cat>łapy</cat></pets>

Tagi mogą zawierać atrybuty będące parami nazwa/wartość:

<data version="2.1"><pets>...</pets></data>

XML jest rozszerzalny, ponieważ pozwala na dowolne nazwy znaczników i atrybutów. Dokumenty XML często zawierają definicję typu dokumentu (DTD), która definiuje znaczniki i atrybuty używane w dokumentach oraz sposoby ich łączenia. XML i wywodzące się z niego technologie są szeroko stosowane w aplikacjach internetowych, zarówno po stronie serwera, jak i klienta.

Usługi internetowe

Chociaż w tej książce omówiono hakowanie aplikacji internetowych, wiele z opisanych luk można w równym stopniu zastosować do usług sieciowych. W rzeczywistości wiele aplikacji jest zasadniczo interfejsem GUI dla zestawu usług sieciowych zaplecza. Usługi internetowe używają protokołu SOAP

(Simple Object Access Protocol) do wymiany danych. SOAP zazwyczaj używa protokołu HTTP do przesyłania komunikatów i reprezentuje dane przy użyciu formatu XML. Typowe żądanie SOAP wygląda następująco:

```
POST /doTransfer.asp HTTP/1.0

Host: mdsec-mgr.int.mdsec.net

Content-Type: application/soap+xml; charset=utf-8

Content-Length: 891

<?xml version=”1.0”?>

<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>
```

W kontekście aplikacji internetowych, do których dostęp uzyskuje się za pomocą przeglądarki, najprawdopodobniej napotkasz SOAP używany przez aplikację po stronie serwera do komunikacji z różnymi systemami zaplecza. Jeśli dane dostarczone przez użytkownika zostaną włączone bezpośrednio do komunikatów SOAP zaplecza, mogą pojawić się podobne luki w zabezpieczeniach, jak w przypadku języka SQL. Jeśli aplikacja internetowa udostępnia bezpośrednio usługi sieciowe, to również warto je zbadać. Nawet jeśli aplikacja front-end jest po prostu napisana na górze usługi internetowej, mogą istnieć różnice w obsłudze danych wejściowych i funkcjonalności udostępnianej przez same usługi. Serwer zwykle publikuje dostępne usługi i parametry przy użyciu formatu Web Services Description Language (WSDL). Narzędzia takie jak soapUI mogą być używane do tworzenia przykładowych żądań na podstawie opublikowanego pliku WSDL w celu wywołania usługi sieciowej uwierzytelniania, uzyskania tokena uwierzytelnienia i wykonania kolejnych żądań usługi sieciowej.

Funkcjonalność po stronie klienta

Aby aplikacja po stronie serwera mogła odbierać dane wejściowe i działania użytkownika oraz przedstawiać użytkownikowi wyniki, musi udostępniać interfejs użytkownika po stronie klienta. Ponieważ wszystkie aplikacje internetowe są dostępne za pośrednictwem przeglądarki internetowej,

wszystkie te interfejsy mają wspólny rdzeń technologii. Zostały one jednak zbudowane na różne, różnorodne sposoby, a sposoby, w jakie aplikacje wykorzystują technologię po stronie klienta, ewoluowały szybko w ostatnich latach.

HTML

Podstawową technologią używaną do budowy interfejsów internetowych jest hipertekstowy język znaczników (HTML). Podobnie jak XML, HTML jest językiem opartym na znacznikach, używanym do opisywania struktury dokumentów wyświetlanych w przeglądarce. Od swoich prostych początków jako sposobu zapewniania podstawowego formatowania dokumentów tekstowych, HTML rozwinął się w bogaty i potężny język, którego można używać do tworzenia bardzo złożonych i funkcjonalnych interfejsów użytkownika. XHTML to rozwinięcie HTML, które jest oparte na XML i ma bardziej rygorystyczną specyfikację niż starsze wersje HTML. Częścią motywacji dla XHTML była potrzeba przejścia w kierunku bardziej sztywnego standardu znaczników HTML, aby uniknąć różnych kompromisów i problemów z bezpieczeństwem, które mogą się pojawić, gdy przeglądarki są zobowiązane do tolerowania mniej rygorystycznych form HTML.

Hiperłącza

Duża część komunikacji między klientem a serwerem jest napędzana przez klikanie przez użytkownika hiperłączy. W aplikacjach internetowych hiperłącza często zawierają wstępnie ustawione parametry żądania. Są to pozycje danych, których użytkownik nigdy nie wprowadza; są przesyłane, ponieważ serwer umieszcza je w docelowym adresie URL hiperłącza, które klika użytkownik. Na przykład aplikacja internetowa może prezentować serię linków do wiadomości, z których każdy ma następującą postać:

What’s happening?

Gdy użytkownik kliknie ten link, przeglądarka wysyła następujące żądanie:

GET /news/8/?redir=/updates/update29.html HTTP/1.1

Host: mdsec.net

...

Serwer otrzymuje parametr redir w ciągu zapytania i używa jego wartości do określenia, jaka treść powinna zostać przedstawiona użytkownikowi.

Formularze

Chociaż nawigacja oparta na hiperłączach jest odpowiedzialna za dużą ilość komunikacji klient-serwer, większość aplikacji internetowych potrzebuje bardziej elastycznych sposobów gromadzenia danych wejściowych i otrzymywania działań od użytkowników. Formularze HTML są zwykle mechanizmem umożliwiającym użytkownikom wprowadzanie dowolnych danych za pośrednictwem przeglądarki. Typowa forma jest następująca:

<form action=”/secure/login.php?app=quotations” method=”post”>

username: <input type=”text” name=”username”>

password: <input type=”password” name=”password”>

<input type=”hidden” name=”redir” value=”/secure/home.php”>

<input type=”submit” name=”submit” value=”log in”>

</form>

Gdy użytkownik wprowadza wartości do formularza i klika przycisk Prześlij, przeglądarka wysyła żądanie w następujący sposób:

POST /secure/login.php?app=quotations HTTP/1.1

Host: wahh-app.com

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

Content-Length: 39

Cookie: SESS=GTnrpx2ss2tSWSnhXJGyG0LJ47MXRsjcFM6Bd

username=daf&password=foo&redir=/secure/home.php&submit=log+in

W tym żądaniu kilka interesujących punktów odzwierciedla sposób, w jaki różne aspekty żądania są wykorzystywane do sterowania przetwarzaniem po stronie serwera:

n Ponieważ znacznik formularza HTML zawiera atrybut określający metodę POST, przeglądarka używa tej metody do wysłania formularza i umieszcza dane z formularza w treści komunikatu żądania.

* Oprócz dwóch pozycji danych, które wprowadza użytkownik, formularz zawiera parametr ukryty (redir) i parametr przesyłania (submit). Oba są przesyłane w żądaniu i mogą być używane przez aplikację po stronie serwera do sterowania jej logiką.

* Docelowy adres URL do przesłania formularza zawiera wstępnie ustawiony parametr (aplikację), tak jak w przykładzie hiperłącza pokazanym wcześniej. Ten parametr może być używany do sterowania przetwarzaniem po stronie serwera.

* Żądanie zawiera parametr cookie (SESS), który został wysłany do przeglądarki we wcześniejszej odpowiedzi z serwera. Ten parametr może być używany do sterowania przetwarzaniem po stronie serwera.

Poprzednie żądanie zawiera nagłówek określający, że typ treści w treści wiadomości to x-www-form-urlencoded. Oznacza to, że parametry są reprezentowane w treści wiadomości jako pary nazwa/wartość w taki sam sposób, jak w ciągu zapytania adresu URL. Innym typem zawartości, który prawdopodobnie napotkasz podczas przesyłania danych formularza, jest multipart/form-data. Aplikacja może zażądać, aby przeglądarki używały kodowania wieloczęściowego, określając to w atrybucie enctype w znaczniku formularza. Przy tej formie kodowania nagłówek Content-Type w żądaniu określa również losowy ciąg, który jest używany jako separator parametrów zawartych w treści żądania. Na przykład, jeśli w formularzu określono kodowanie wieloczęściowe, wynikowe żądanie wyglądałoby następująco:

POST /secure/login.php?app=quotations HTTP/1.1

Host: wahh-app.com

Content-Type: multipart/form-data; boundary=----------7d71385d0a1a

Content-Length: 369

Cookie: SESS=GTnrpx2ss2tSWSnhXJGyG0LJ47MXRsjcFM6Bd

----------7d71385d0a1a

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

daf

----------7d71385d0a1a

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

foo

----------7d71385d0a1a

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

/secure/home.php

----------7d71385d0a1a

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

log in

----------7d71385d0a1a—

CSS

Kaskadowe arkusze stylów (CSS) to język używany do opisywania prezentacji dokumentu napisanego w języku znaczników. W aplikacjach internetowych służy do określania, w jaki sposób treść HTML powinna być renderowana na ekranie (oraz w innych mediach, takich jak wydrukowana strona). Nowoczesne standardy internetowe mają na celu jak największe oddzielenie treści dokumentu od jego prezentacji. Ta separacja ma wiele zalet, w tym prostsze i mniejsze strony HTML, łatwiejszą aktualizację formatowania w całej witrynie oraz lepszą dostępność. CSS opiera się na regułach formatowania, które można definiować na różnych poziomach szczegółowości. Gdy wiele reguł pasuje do pojedynczego elementu dokumentu, różne atrybuty zdefiniowane w tych regułach mogą „kaskadować” te reguły, dzięki czemu do elementu zostanie zastosowana odpowiednia kombinacja atrybutów stylu. Składnia CSS wykorzystuje selektory do zdefiniowania klasy elementów znaczników, do których należy zastosować dany zestaw atrybutów. Na przykład następujące

Reguła CSS definiuje kolor pierwszego planu dla nagłówków oznaczonych za pomocą

<h2> tags:

h2 { color: red; }

W pierwszych dniach bezpieczeństwa aplikacji internetowych CSS był w dużej mierze pomijany i uważano, że nie ma wpływu na bezpieczeństwo. Obecnie CSS ma coraz większe znaczenie zarówno jako samo w sobie źródło luk w zabezpieczeniach, jak i jako sposób dostarczania skutecznych exploitów dla innych kategorii luk w zabezpieczeniach.

JavaScript

Hiperłącza i formularze mogą służyć do tworzenia bogatego interfejsu użytkownika, który może łatwo gromadzić większość rodzajów danych wejściowych wymaganych przez aplikacje internetowe. Jednak większość aplikacji wykorzystuje bardziej rozproszony model, w którym strona kliencka służy nie tylko do przesyłania danych i działań użytkownika, ale także do rzeczywistego przetwarzania danych. Odbywa się to z dwóch podstawowych powodów:

* Może poprawić wydajność aplikacji, ponieważ niektóre zadania mogą być wykonywane wyłącznie na komponencie klienckim, bez konieczności wysyłania żądania i odpowiedzi do serwera.

* Może zwiększyć użyteczność, ponieważ części interfejsu użytkownika mogą być dynamicznie aktualizowane w odpowiedzi na działania użytkownika, bez konieczności ładowania całkowicie nowej strony HTML dostarczanej przez serwer. JavaScript to stosunkowo prosty, ale potężny język programowania, którego można łatwo używać do rozszerzania interfejsów sieciowych w sposób niemożliwy do osiągnięcia przy użyciu samego HTML. Jest powszechnie używany do wykonywania następujących zadań:

* Sprawdzanie poprawności danych wprowadzonych przez użytkownika przed przesłaniem ich do serwera, aby uniknąć niepotrzebnych żądań, jeśli dane zawierają błędy

* Dynamiczne modyfikowanie interfejsu użytkownika w odpowiedzi na działania użytkownika — na przykład w celu wdrożenia menu rozwijanych i innych elementów sterujących znanych z interfejsów innych niż sieciowe

* Wyszukiwanie i aktualizowanie modelu obiektowego dokumentu (DOM) w przeglądarce w celu kontrolowania zachowania przeglądarki (opis modelu DOM przeglądarki za chwilę)

VBScript

VBScript to alternatywa dla JavaScript, która jest obsługiwana tylko w przeglądarce Internet Explorer. Jest wzorowany na Visual Basic i umożliwia interakcję z DOM przeglądarki. Ale generalnie jest nieco słabszy i mniej rozwinięty niż JavaScript. Ze względu na swoją specyficzną dla przeglądarki naturę, VBScript jest rzadko używany w dzisiejszych aplikacjach internetowych. Jego głównym celem z punktu widzenia bezpieczeństwa jest dostarczanie exploitów dla luk w zabezpieczeniach, takich jak cross-site scripting, w sporadycznych sytuacjach, w których exploit przy użyciu JavaScript nie jest możliwy.

Obiektowy model dokumentu

Document Object Model (DOM) to abstrakcyjna reprezentacja dokumentu HTML, którą można przeszukiwać i manipulować za pomocą jej interfejsu API. DOM umożliwia skryptom działającym po stronie klienta dostęp do poszczególnych elementów HTML na podstawie ich identyfikatora i programowe przechodzenie przez strukturę elementów. Dane, takie jak aktualny adres URL i pliki cookie, można również odczytywać i aktualizować. DOM zawiera również model zdarzeń, umożliwiający kodowi przechwytywanie zdarzeń, takich jak przesyłanie formularzy, nawigacja za pomocą linków i naciśnięcia klawiszy. Manipulowanie DOM przeglądarki jest kluczową techniką stosowaną w aplikacjach opartych na technologii Ajax, co opisano w następnej sekcji.

Ajax

Ajax to zbiór technik programistycznych używanych po stronie klienta do tworzenia interfejsów użytkownika, których celem jest naśladowanie płynnej interakcji i dynamicznego zachowania tradycyjnych aplikacji komputerowych. Pierwotnie nazwa była skrótem od „Asynchroniczny JavaScript i XML”, chociaż w dzisiejszych internetowych żądaniach Ajax nie muszą być asynchroniczne i nie muszą wykorzystywać XML. Najwcześniejsze aplikacje internetowe opierały się na pełnych stronach. Każde działanie użytkownika, takie jak kliknięcie łącza lub wysłanie formularza, inicjowało zdarzenie nawigacji na poziomie okna, powodując załadowanie nowej strony z serwera. Takie podejście skutkowało niespójnym doświadczeniem użytkownika, z zauważalnymi opóźnieniami podczas odbierania dużych odpowiedzi z serwera i ponownego renderowania całej strony. W Ajax niektóre działania użytkownika są obsługiwane w kodzie skryptu po stronie klienta i nie powodują pełnego przeładowania strony.

Zamiast tego skrypt wykonuje żądanie „w tle” i zwykle otrzymuje znacznie mniejszą odpowiedź, która jest używana do dynamicznej aktualizacji tylko części interfejsu użytkownika. Na przykład w aplikacji zakupowej opartej na technologii Ajax kliknięcie przycisku Dodaj do koszyka może spowodować powstanie tła żądanie, które aktualizuje rekord koszyka użytkownika po stronie serwera i lekką odpowiedź, która aktualizuje liczbę elementów koszyka wyświetlanych na ekranie użytkownika. Praktycznie cała istniejąca strona pozostaje niezmodyfikowana w przeglądarce, zapewniając użytkownikowi znacznie szybsze i bardziej satysfakcjonujące doświadczenie. Podstawową technologią używaną w Ajax jest XMLHttpRequest. Po pewnej konsolidacji standardów jest to teraz natywny obiekt JavaScript, którego skrypty po stronie klienta mogą używać do wysyłania żądań „w tle” bez wymagania zdarzenia nawigacji na poziomie okna. Pomimo swojej nazwy, XMLHttpRequest umożliwia wysyłanie dowolnych treści w żądaniach i odbieranie ich w odpowiedziach. Chociaż wiele aplikacji Ajax używa XML do formatowania danych komunikatów, coraz więcej decyduje się na wymianę danych przy użyciu innych metod reprezentacji. (Jeden przykład można znaleźć w następnej sekcji.) Zauważ, że chociaż większość aplikacji Ajax korzysta z komunikacji asynchronicznej z serwerem, nie jest to konieczne. W niektórych sytuacjach bardziej sensowne może być odtworzenie interakcji użytkownika z aplikacją podczas wykonywania określonej czynności. W takich sytuacjach Ajax jest nadal korzystny, zapewniając bardziej płynne działanie, unikając konieczności ponownego ładowania całej strony. Historycznie rzecz biorąc, użycie technologii Ajax wprowadziło kilka nowych rodzajów luk w zabezpieczeniach aplikacji internetowych. Mówiąc szerzej, zwiększa również powierzchnię ataku typowej aplikacji, wprowadzając więcej potencjalnych celów ataku zarówno po stronie serwera, jak i klienta. Techniki Ajax są również dostępne do wykorzystania przez atakujących, gdy opracowują skuteczniejsze exploity dla innych luk w zabezpieczeniach.

JSON

JavaScript Object Notation (JSON) to prosty format przesyłania danych, którego można użyć do serializacji dowolnych danych. Może być przetwarzany bezpośrednio przez interpretery JavaScript. Jest powszechnie stosowany w aplikacjach Ajax jako alternatywa dla formatu XML pierwotnie używanego do transmisji danych. W typowej sytuacji, gdy użytkownik wykonuje akcję, JavaScript po stronie klienta używa XMLHttpRequest do przekazania tej czynności serwerowi. Serwer zwraca lekką odpowiedź zawierającą dane w formacie JSON. Następnie skrypt po stronie klienta przetwarza te dane i odpowiednio aktualizuje interfejs użytkownika. Na przykład aplikacja poczty internetowej oparta na technologii Ajax może zawierać funkcję wyświetlania szczegółów wybranego kontaktu. Gdy użytkownik kliknie kontakt, przeglądarka używa XMLHttpRequest do pobrania szczegółów wybranego kontaktu, które są zwracane za pomocą JSON:

{

“name”: “Mike Kemp”,

“id”: “8041148671”,

“email”: “fkwitt@layerone.com”

}

Skrypt po stronie klienta wykorzystuje interpreter JavaScript do wykorzystania odpowiedzi JSON i aktualizuje odpowiednią część interfejsu użytkownika na podstawie jej zawartości. Kolejnym miejscem, w którym można napotkać dane JSON we współczesnych aplikacjach, jest sposób hermetyzacji danych w konwencjonalnych parametrach żądania. Na przykład, gdy użytkownik aktualizuje szczegóły kontaktu, nowe informacje mogą zostać przesłane do serwera przy użyciu następującego żądania:

POST /contacts HTTP/1.0

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

Content-Length: 89

Contact={“name”:”Mike Kemp”,”id”:”8041148671”,”email”:”pikey@

clappymonkey.com”}

&submit=update

Polityka tego samego źródła

Polityka tego samego pochodzenia to kluczowy mechanizm zaimplementowany w przeglądarkach, który ma na celu zapobieganie wzajemnemu zakłócaniu się treści pochodzących z różnych źródeł. Zasadniczo treści otrzymane z jednej witryny internetowej mogą odczytywać i modyfikować inne treści otrzymane z tej samej witryny, ale nie mają dostępu do treści otrzymanych z innych witryn. Jeśli nie istniałaby zasada tego samego pochodzenia, a nieświadomy użytkownik przeglądał złośliwą witrynę internetową, kod skryptu działający w tej witrynie mógłby uzyskać dostęp do danych i funkcji dowolnej innej witryny internetowej również odwiedzanej przez użytkownika. Może to umożliwić złośliwej witrynie wykonywanie przelewów środków z internetowego banku użytkownika, odczytywanie jego poczty internetowej lub przechwytywanie danych karty kredytowej, gdy użytkownik robi zakupy online. Z tego powodu przeglądarki wprowadzają ograniczenia, aby umożliwić tego typu interakcję tylko z treściami otrzymanymi z tego samego źródła. W praktyce stosowanie tej koncepcji do szczegółów różnych funkcji i technologii internetowych prowadzi do różnych komplikacji i kompromisów. Oto niektóre kluczowe cechy polityki tego samego pochodzenia, o których musisz wiedzieć:

* Strona znajdująca się w jednej domenie może spowodować wysłanie dowolnego żądania do innej domeny (na przykład przesłanie formularza lub załadowanie obrazu). Ale nie może sam przetwarzać danych zwróconych z tego żądania.

* Strona znajdująca się w jednej domenie może załadować skrypt z innej domeny i wykonać go we własnym kontekście. Dzieje się tak, ponieważ zakłada się, że skrypty zawierają kod, a nie dane, więc dostęp między domenami nie powinien prowadzić do ujawnienia żadnych poufnych informacji.

* Strona znajdująca się w jednej domenie nie może odczytywać ani modyfikować plików cookie ani innych danych DOM należących do innej domeny.

Funkcje te mogą prowadzić do różnych ataków między domenami, takich jak nakłanianie użytkownika do działania i przechwytywanie danych. Dalsze komplikacje pojawiają się w przypadku technologii rozszerzeń przeglądarki, które implementują ograniczenia dotyczące tego samego pochodzenia na różne sposoby.

HTML5

HTML5 to główna aktualizacja standardu HTML. Z punktu widzenia bezpieczeństwa HTML5 jest interesujący przede wszystkim z następujących powodów:

* Wprowadza różne nowe tagi, atrybuty i interfejsy API, które można wykorzystać do przeprowadzania skryptów między witrynami i innych ataków.

* Modyfikuje podstawową technologię Ajax, XMLHttpRequest, aby umożliwić dwukierunkową interakcję między domenami w określonych sytuacjach. Może to prowadzić do nowych ataków międzydomenowych.

* Wprowadza nowe mechanizmy przechowywania danych po stronie klienta, które mogą prowadzić do problemów z prywatnością użytkowników, oraz nowe kategorie ataków, takie jak iniekcja SQL po stronie klienta.

„Sieć 2.0”

To modne słowo stało się modne w ostatnich latach jako dość luźna i mglista nazwa dla szeregu powiązanych trendów w aplikacjach internetowych, w tym następujących:

* Intensywne użycie Ajaksa do wykonywania asynchronicznych, zakulisowych żądań

* Zwiększona integracja między domenami przy użyciu różnych technik

* Wykorzystanie nowych technologii po stronie klienta, w tym XML, JSON, Flex

* Bardziej widoczne funkcje obsługujące treści generowane przez użytkowników, udostępnianie informacji i interakcje

Podobnie jak w przypadku wszystkich zmian w technologii, trendy te stwarzają nowe możliwości pojawienia się luk w zabezpieczeniach. Nie określają one jednak ogólnie jasnego podzbioru problemów związanych z bezpieczeństwem aplikacji internetowych. Luki w zabezpieczeniach, które występują w tych kontekstach, są w dużej mierze takie same lub ściśle wywodzą się z typów luk, które poprzedzały te trendy. Mówiąc ogólnie o „Bezpieczeństwie Web 2.0” zwykle stanowi błąd kategorii, który nie ułatwia jasnego myślenia o sprawach, które mają znaczenie.

Technologie rozszerzeń przeglądarki

Wykraczając poza możliwości JavaScript, niektóre aplikacje internetowe wykorzystują technologie rozszerzeń przeglądarki, które wykorzystują niestandardowy kod w celu rozszerzenia wbudowanych możliwości przeglądarki w dowolny sposób. Komponenty te mogą być wdrażane jako kod bajtowy, który jest wykonywany przez odpowiednią wtyczkę przeglądarki lub mogą obejmować instalowanie natywnych plików wykonywalnych na samym komputerze klienckim. Technologie grubego klienta, które prawdopodobnie napotkasz podczas atakowania aplikacji internetowych, są

* Aplety Javy

* Formanty ActiveX

* Obiekty Flash

* Obiekty Silverlight

Stan i sesje

Opisane dotychczas technologie umożliwiają komponentom serwerowym i klienckim aplikacji internetowej wymianę i przetwarzanie danych na wiele sposobów. Aby jednak zaimplementować większość przydatnych funkcji, aplikacje muszą śledzić stan interakcji każdego użytkownika z aplikacją w ramach wielu żądań. Na przykład aplikacja zakupowa może umożliwiać użytkownikom przeglądanie katalogu produktów, dodawanie pozycji do koszyka, przeglądanie i aktualizowanie zawartości koszyka, przechodzenie do kasy oraz podawanie danych osobowych i danych dotyczących płatności. Aby tego rodzaju funkcjonalność była możliwa, aplikacja musi utrzymywać zestaw danych stanowych

generowanych przez działania użytkownika w ramach kilku żądań. Dane te są zwykle przechowywane w strukturze po stronie serwera zwanej sesją. Gdy użytkownik wykonuje czynność, taką jak dodanie elementu do koszyka, aplikacja serwerowa aktualizuje odpowiednie szczegóły w ramach sesji użytkownika. Gdy użytkownik później przegląda zawartość swojego koszyka, dane z sesji są wykorzystywane do zwracania użytkownikowi prawidłowych informacji. W niektórych aplikacjach informacje o stanie są przechowywane w komponencie klienckim, a nie na serwerze. Bieżący zestaw danych jest przekazywany klientowi w każdej odpowiedzi serwera i odsyłany z powrotem do serwera w każdym żądaniu klienta. Oczywiście, ponieważ użytkownik może modyfikować dowolne dane przesyłane za pośrednictwem komponentu klienckiego, aplikacje muszą chronić się przed atakującymi, którzy mogą zmienić te informacje o stanie, próbując zakłócić logikę aplikacji. Platforma ASP.NET wykorzystuje ukryte pole formularza o nazwie ViewState do przechowywania informacji o stanie interfejsu sieciowego użytkownika, a tym samym zmniejsza obciążenie serwera. Domyślnie zawartość ViewState zawiera skrót z kluczem, aby zapobiec manipulowaniu. Ponieważ protokół HTTP sam w sobie jest bezstanowy, większość aplikacji potrzebuje sposobu na ponowną identyfikację poszczególnych użytkowników w wielu żądaniach, aby prawidłowy zestaw danych stanu był używany do przetwarzania każdego żądania. Zwykle osiąga się to poprzez wydanie każdemu użytkownikowi tokena, który jednoznacznie identyfikuje sesję tego użytkownika. Te tokeny mogą być przesyłane przy użyciu dowolnego typu parametru żądania, ale większość aplikacji korzysta z plików cookie HTTP.

Schematy kodowania

Aplikacje internetowe wykorzystują kilka różnych schematów kodowania swoich danych. Zarówno protokół HTTP, jak i język HTML są historycznie oparte na tekście i opracowano różne schematy kodowania, aby zapewnić, że te mechanizmy mogą bezpiecznie obsługiwać nietypowe znaki i dane binarne. Kiedy atakujesz aplikację internetową, często będziesz musiał zakodować dane przy użyciu odpowiedniego schematu, aby upewnić się, że są one obsługiwane w zamierzony sposób. Ponadto w wielu przypadkach możesz manipulować schematami kodowania używanymi przez aplikację, aby wywołać zachowanie, którego nie zamierzali jej projektanci.

Kodowanie adresów URL

Adresy URL mogą zawierać tylko znaki drukowalne z zestawu znaków US-ASCII — czyli takie, których kod ASCII mieści się w zakresie od 0x20 do 0x7e włącznie. Ponadto niektóre znaki z tego zakresu są ograniczone, ponieważ mają specjalne znaczenie w samym schemacie adresu URL lub w protokole HTTP. Schemat kodowania adresów URL służy do kodowania wszelkich problematycznych znaków w rozszerzonym zestawie znaków ASCII, aby można je było bezpiecznie przesyłać przez HTTP. Forma dowolnego znaku zakodowana w adresie URL to przedrostek %, po którym następuje dwucyfrowy kod ASCII znaku wyrażony w systemie szesnastkowym. Oto niektóre znaki, które są zwykle kodowane w adresach URL:

* %3d — =

* %25 — %

* %20 — Spacja

* %0a — Nowa linia

* %00 — Bajt zerowy

Kolejnym kodowaniem, o którym należy pamiętać, jest znak +, który reprezentuje spację zakodowaną w adresie URL (oprócz reprezentacji spacji w %20). UWAGA W celu atakowania aplikacji sieci Web

należy zakodować w adresie URL dowolne z poniższych znaków, gdy wstawia się je jako dane do żądania HTTP:

space % ? & = ; + #

(Oczywiście podczas modyfikowania żądania często będziesz musiał używać tych znaków w ich specjalnym znaczeniu — na przykład, aby dodać parametr żądania do ciągu zapytania. W takim przypadku należy ich używać w ich dosłownej formie).

Kodowanie Unicode

Unicode to standard kodowania znaków zaprojektowany do obsługi wszystkich systemów pisma na świecie. Wykorzystuje różne schematy kodowania, z których niektóre mogą być używane do reprezentowania nietypowych znaków w aplikacjach internetowych. 16-bitowe kodowanie Unicode działa w podobny sposób jak kodowanie adresów URL. W przypadku transmisji przez HTTP 16-bitową postacią znaku zakodowaną w Unicode jest przedrostek %u, po którym następuje punkt kodowy Unicode znaku wyrażony szesnastkowy:

n %u2215 — /

n %u00e9 — e

UTF-8 to standard kodowania o zmiennej długości, który wykorzystuje jeden lub więcej bajtów do wyrażenia każdego znaku. Do transmisji przez HTTP postać znaku wielobajtowego zakodowana w UTF-8 po prostu wykorzystuje każdy bajt wyrażony w systemie szesnastkowym i poprzedzony prefiksem %:

n %c2%a9 — ©

n %e2%89%a0 — ≠

W celu atakowania aplikacji internetowych kodowanie Unicode jest przede wszystkim interesujące, ponieważ czasami można go użyć do pokonania mechanizmów sprawdzania poprawności danych wejściowych. Jeśli filtr wejściowy blokuje pewne szkodliwe wyrażenia, ale komponent, który następnie przetwarza dane wejściowe, rozumie kodowanie Unicode, możliwe może być ominięcie filtra przy użyciu różnych standardowych i zniekształconych kodowań Unicode.

Kodowanie HTML

Kodowanie HTML służy do reprezentowania problematycznych znaków, aby można je było bezpiecznie włączyć do dokumentu HTML. Różne znaki mają specjalne znaczenie jako metaznaki w HTML i służą do definiowania struktury dokumentu, a nie jego treści. Aby bezpiecznie używać tych znaków jako części treści dokumentu, konieczne jest ich kodowanie HTML. Kodowanie HTML definiuje wiele jednostek HTML reprezentujących określone znaki literalne:

* " — "

* ' — „

* & amp; — &

* < — <

* > — >

Ponadto każdy znak można zakodować w formacie HTML przy użyciu jego kodu ASCII w postaci dziesiętnej:

* " — "

* ' — „

lub używając jego kodu ASCII w postaci szesnastkowej (z prefiksem x):

* " — "

* ' — „

Kiedy atakujesz aplikację internetową, twoim głównym zainteresowaniem kodowaniem HTML będzie prawdopodobnie poszukiwanie luk w zabezpieczeniach związanych ze skryptami między witrynami. Jeśli aplikacja zwraca niezmodyfikowane dane wejściowe użytkownika w swoich odpowiedziach, prawdopodobnie jest podatna na ataki, natomiast jeśli niebezpieczne znaki są zakodowane w HTML, może być bezpieczna.

Kodowanie Base64

Kodowanie Base64 umożliwia bezpieczne przedstawienie dowolnych danych binarnych przy użyciu wyłącznie drukowalnych znaków ASCII. Jest powszechnie używany do kodowania załączników wiadomości e-mail w celu bezpiecznej transmisji przez SMTP. Jest również używany do kodowania poświadczeń użytkownika w podstawowym uwierzytelnianiu HTTP. Kodowanie Base64 przetwarza dane wejściowe w blokach po trzy bajty. Każdy z tych bloków jest podzielony na cztery części po sześć bitów każda. Sześć bitów danych pozwala na 64 różne możliwe permutacje, więc każdy fragment można przedstawić za pomocą zestawu 64 znaków. Kodowanie Base64 wykorzystuje następujący zestaw znaków, który zawiera tylko drukowane znaki ASCII:

ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/

Jeśli ostatni blok danych wejściowych daje mniej niż trzy fragmenty danych wyjściowych, dane wyjściowe są uzupełniane jednym lub dwoma znakami =.

Oto na przykład zakodowana w formacie Base64 forma Podręcznika hakera aplikacji TheWeb:

VGhlIFdlYiBBcHBsaWNhdGlvbiBIYWNrZXIncyBIYW5kYm9vaw==

Wiele aplikacji internetowych używa kodowania Base64 do przesyłania danych binarnych w plikach cookie i innych parametrach, a nawet do zaciemniania (tj. ukrywania) wrażliwych danych, aby zapobiec trywialnym modyfikacjom. Należy zawsze szukać i dekodować wszelkie dane Base64, które są wysyłane do klienta. Łańcuchy zakodowane w formacie Base64 często można łatwo rozpoznać po ich specyficznym zestawie znaków i obecności znaków dopełniających na końcu ciągu.

Kodowanie szesnastkowe

Wiele aplikacji używa prostego kodowania szesnastkowego podczas przesyłania danych binarnych, używając znaków ASCII do reprezentowania bloku szesnastkowego. Na przykład kodowanie szesnastkowe nazwy użytkownika „daf” w pliku cookie spowodowałoby to:

646166

Podobnie jak w przypadku Base64, dane zakodowane szesnastkowo są zwykle łatwe do wykrycia. Zawsze powinieneś próbować odszyfrować wszelkie takie dane, które serwer wysyła do klienta, aby zrozumieć jego funkcję.

Frameworki zdalne i serializacja

W ostatnich latach ewoluowały różne frameworki do tworzenia interfejsów użytkownika, w których kod po stronie klienta może zdalnie uzyskiwać dostęp do różnych programistycznych interfejsów API zaimplementowanych po stronie serwera. Pozwala to programistom na częściowe odejście od rozproszonej natury aplikacji internetowych i pisanie kodu w sposób bliższy paradygmatowi konwencjonalnej aplikacji komputerowej. Te ramy zazwyczaj zapewniają pośredniczące interfejsy API do użytku po stronie klienta. Automatycznie obsługują zarówno zdalne wywołania tych wywołań API do odpowiednich funkcji po stronie serwera, jak i serializację wszelkich danych przekazywanych do tych funkcji. Przykłady tego rodzaju struktur komunikacji zdalnej i serializacji obejmują:

* Flex i AMF

* Silverlight i WCF

* Serializowane obiekty Java

Następne kroki

Do tej pory opisaliśmy obecny stan (nie)bezpieczeństwa aplikacji internetowych, zbadaliśmy podstawowe mechanizmy, za pomocą których aplikacje internetowe mogą się bronić, oraz krótko przyjrzeliśmy się kluczowym technologiom wykorzystywanym we współczesnych aplikacjach. Mając te podstawy, możemy teraz zacząć przyglądać się praktycznym aspektom atakowania aplikacji internetowych. Pierwszym zadaniem każdego ataku jest zmapowanie zawartości i funkcjonalności docelowej aplikacji, aby ustalić, jak ona działa, jak próbuje się bronić i jakich technologii używa. W następnym rozdziale szczegółowo przeanalizujemy ten proces mapowania i pokażemy, jak można go wykorzystać do uzyskania głębokiego zrozumienia obszaru ataku aplikacji. Ta wiedza okaże się kluczowa, jeśli chodzi o znajdowanie i wykorzystywanie luk w zabezpieczeniach twojego celu.