

Bezpieczeństwo systemów komputerowych

Zagrożenia dla aplikacji internetowych

Marcin Peczarski

Instytut Informatyki Uniwersytetu Warszawskiego

17 października 2011

Lista zagadnień

- ▶ Wstrzykiwanie kodu SQL (ang. *SQL injection*)
- ▶ Skrośny atak skryptowy, XSS (ang. *cross-site scripting*)
- ▶ Skrośne fałszowanie żądania, XSRF lub CSRF (ang. *cross-site request forgery*, wym. *sea-surf*)
- ▶ Porywanie kliknięć (ang. *clickjacking*)

Zarys problemu

- ▶ Wszystkie dane wprowadzane przez użytkownika należy traktować jako niebezpieczne.
- ▶ W aplikacji działającej w oparciu o model klient-serwer:
 - ▶ nie wolno polegać na sprawdzaniu poprawności danych po stronie klienta;
 - ▶ agresor może wprowadzić złośliwe dane;
 - ▶ agresor może napisać własnego klienta.

Wstrzykiwanie kodu SQL

- ▶ Po stronie klienta zaimplementowano formatkę do uwierzytelniania, zwracającą dwie zmienne:
 - ▶ `login` – identyfikator użytkownika,
 - ▶ `password` – hasło użytkownika.
- ▶ Identyfikatory i hasła użytkowników przechowywane są w bazie danych w tablicy `client`, która zawiera kolumny `name` i `pwd`.
- ▶ Po stronie serwera zaimplementowano skrypt, generujący zapytanie SQL:

```
SELECT count(*) FROM client  
WHERE name='$login' and pwd='$password';
```

- ▶ Zapytanie zwraca wartość 1, gdy nazwa użytkownika i hasło są poprawne, a 0 w p.p.

Atak

- ▶ Jeśli agresor wprowadzi

```
login=d' or '1'='1
```

```
password=d' or '1'='1
```

zostanie wygenerowane zapytanie

```
SELECT count(*) FROM client WHERE  
name='d' or '1'='1' and pwd='d' or '1'='1';
```

które zawsze zwraca wartość różną od zera.

- ▶ Agresor może znać identyfikator użytkownika i wprowadzić

```
login=root'; #
```

```
password=niepotrzebne
```

wtedy zostanie wygenerowane zapytanie

```
SELECT count(*) FROM client WHERE name='root';
```

które zwróci wartość 1.

Inny atak

- Agresor może też modyfikować bazę danych, wprowadzając

```
login=d' INSERT INTO client  
VALUES ('hacker', 'alamakota'); #  
  
password=niepotrzebne
```

Wygenerowane zostanie zapytanie

```
SELECT count(*) FROM client WHERE name='d'  
INSERT INTO client  
VALUES ('hacker', 'alamakota');
```

które doda nowego użytkownika do systemu.

Remedium

- ▶ Dane wejściowe powinny podlegać ścisłej weryfikacji po stronie serwera, np. za pomocą wyrażeń regularnych:
 - ▶ łatwo jest napisać wyrażenie regularne dla identyfikatora użytkownika, uniemożliwiające atak polegający na wstrzykiwaniu kodu SQL;
 - ▶ trudno jest wymyślić rozsądne wyrażenie regularne dla hasła.
- ▶ Hasło powinno być przechowywane w postaci binarnego skrótu kryptograficznego:
 - ▶ przed wykonaniem skrótu należy hasło posolić (ang. *salt*), czyli rozszerzyć go o wartość losową;
 - ▶ traktowanie hasła jako wartości binarnej umożliwia stosowanie w nim dowolnych znaków i uniemożliwia atak polegający na wstrzykiwaniu kodu SQL.
- ▶ Zamiast skryptu generującego zapytanie SQL lepiej jest zastosować procedurę składowaną.
- ▶ Dostęp do bazy danych powinny odbywać się z jak najmniejszymi uprawnieniami.

Remedium

- ▶ Można pracować wyszukiwać w danych od użytkownika potencjalnie „niebezpiecznych” znaków i je „eskejpować” (ang. *escape*).
- ▶ Jeśli postać wszystkich zapytań jest znana, można zastosować mechanizm łączenia (ang. *binding*):

```
$login = $_POST['login'];  
$password = $_POST['password'];  
$cnt = dbQueryParams("SELECT count(*) FROM client  
                      WHERE name=$1 and pwd=$2",  
                      array($login, $password));
```

- ▶ Treść zapytania i parametry są przekazywane osobno, co eliminuje potencjalne problemy powstające przy sklejeniu zapytania z fragmentów napisów.

Skrośny atak skryptowy

- ▶ Wykorzystuje zaufanie użytkownika do odwiedzanej strony.
- ▶ Wiele serwisów umożliwia dodawanie treści pochodzących od użytkowników, celem wyświetlenia innym użytkownikom.
- ▶ Agresor może umieścić na stronie jakiegoś serwisu odwołanie do innej strony lub skrypt, który wykona się w przeglądarce użytkownika, gdy ten będzie korzystał z tego serwisu.
- ▶ Umożliwia to wykonanie niepożądanego akcja lub kradzież ciasteczka (ang. *cookie*).

Zalecenia

- ▶ Należy filtrować treści pochodzące od użytkowników.
- ▶ Ciasteczka nie powinny być dostępne dla skryptów (znacznik *httponly*).
- ▶ Wymuszanie, aby jedynym dopuszczalnym źródłem skryptów, obrazków i stylów była konkretna domena.
- ▶ Blokowanie wykonywania skryptów, które są zagnieżdżone w treści strony.
- ▶ Blokowanie niebezpiecznych konstrukcji, które umożliwiają dynamiczne tworzenie kodu wykonywalnego z napisu.

Skrośne fałszowanie żądania

- ▶ Wykorzystuje zaufanie strony do użytkownika, który ją odwiedza.
- ▶ Użytkownik zalogowany do serwisu uruchamia spreparowany odnośnik pochodzący z innego źródła (np. ukryty w obrazku, zagnieżdżonej ramce), co powoduje wykonanie (przez przeglądarkę bez wiedzy użytkownika) w tym serwisie akcji, której użytkownik nigdy by nie wykonał.

Zalecenia

- ▶ Będąc zalogowanym do serwisu internetowego, należy unikać otwierania wszelkich innych stron.
- ▶ Należy wylogować się z serwisu, gdy skończy się z niego korzystać oraz unikać zapamiętywania sesji.
- ▶ Wymóg podawania jednorazowego hasła przy każdej newralgicznej akcji uniemożliwia agresorowi spreparowanie poprawnego żądania do serwera.
- ▶ Okres ważności zalogowania i dopuszczalny czas bezczynności powinny być możliwie krótkie.
- ▶ Do każdego formularza można dodawać ukryte pole z pseudolosową liczbą, która jest powiązana z sesją użytkownika i musi zostać przekazana wraz z żądaniem.
- ▶ Należy stosować metodę POST do żądań powodujących skutki uboczne, co jest jednym z zaleceń protokołu HTTP.

Porywanie kliknięć

- ▶ Agresor wysyła użytkownikowi stronę, która zawiera pozornie niegroźną treść, a w niewidocznej ramce znajdującej się na pierwszym planie ładuje atakowany serwis, do którego użytkownik ma większe uprawnienia niż agresor.
- ▶ Użytkownik musi być zalogowany do tego serwisu.
- ▶ Wszystkie kliknięcia wykonują się w serwisie, którego użytkownik nie widzi.
- ▶ Inna możliwość to podsuniecie użytkownikowi niewidocznej strony, która, gdy się na nią klika, wykonuje złośliwe operacje na komputerze użytkownika.

Zalecenia

- ▶ Serwisy internetowe powinny blokować możliwość wczytania ich w ramce i odpowiednio reagować (ang. *frame busting*).