JCSI 3 (2017) 1-5
Journal of Computer Sciences Institute

Wysłane: 2016-12-08
Przyjęte: 2016-12-14

Bezpieczeństwo aplikacji internetowych

Michał Furtak*
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. W artykule poruszono temat bezpieczeństwa aplikacji internetowych. Opisane zostały najpopularniejsze rodzaje ataków. Analizie poddana została autorska aplikacja Internetowy Notatnik. Przedstawiono rozwiązania mające na celu poprawę bezpieczeństwa.

Słowa kluczowe: ataki na serwisy internetowe; bezpieczeństwo; sql injection; xss

*Corresponding author.

Adres e-mail: michal.furtak1@pollub.edu.pl

Security of Web Applications

Michał Furtak*
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. This article is about web application security. Describes the most common types of web attacks. The analysis was subjected to authoring application Internetowy Notatnik. Provides solutions to improve safety.

Keywords: web attack; sql injection, xss

*Corresponding author.

E-mail address: mfurtak@gmail.com

1. Wstęp

Początki Internetu sięgają końca lat sześćdziesiątych ubiegłego wieku. W 1969 roku, na Uniwersytecie Kalifornijskim rozpoczęły się prace nad połączeniem pierwszych węzłów sieci ARPANET (ang. Advanced Research Projects Agency Network). Sieć ta obecnie uznawana jest za bezpośredniego przodka współczesnego Internetu [1]. Projekt ten pierwotnie wykorzystywany miał być do celów naukowych oraz wojskowych. Prawdopodobnie niewielu z ówcześnie żyjących zdawało sobie sprawę jak wielki wpływ wywarty zostanie przez ten projekt na życie i funkcjonowanie późniejszych pokoleń.

Obecnie niewielu z nas wyobraża sobie życie bez dostępu do światowej sieci. Według badań Netcraft liczba serwisów internetowych wynosi 863 miliardy [2]. Dla porównania warto dodać że w roku 2000 było to tylko 17 miliardów. Przyczyn tak szybkiego rozwoju można dopatrywać się w upowszechnieniu dostępu do sieci. Obecnie, dzięki postępowi technicznemu, jest on możliwy z prawie każdego miejsca na ziemi przy pomocy smartphonu. Szacuje się, że 43% ludności Ziemi jest użytkownikami Internetu.

Patrząc na historie Internetu nie sposób nie dostrzec zmian w udostępnianych treściach. Przejście z ery Web 1.0 do Web 2.0 wiąże się z licznymi zmianami. Duża rolę zaczęły odgrywać serwisy, w których to użytkownicy są twórcami treści [3]. Witryny internetowe przestały być jedynie bazą wiedzy, którą można przeczytać i przejść dalej. Autorzy stron sprawili, że użytkownicy stali się częścią Internetu. Dobrze znanymi przykładami serwisów Web 2.0 są Facebook, czy YouTube.

Internet ułatwia życie współczesnemu człowiekowi: umożliwia robienie zakupów, wykonywanie przelewów bankowych, szybką komunikację wideo-głosową oraz inne. Pomimo dobrodziejstw jakie ze sobą niesie nie należy zapominać o potencjalnymi niebezpieczeństwie związanym z jego wykorzystaniem.

Serwisy internetowe narażone są na ataki, które przyczynić się mogą do naruszenie integralności oraz poufności danych użytkowników. Skutecznie przeprowadzony DDOS (ang. distributed denial of service) może unieruchomić serwer, co skutkuje przerwą w dostępie do udostępnianiach przez niego usług [4].

W niniejszym artykule opisane zostały najczęściej wykorzystywane metody ataków. Na przykładzie analizy przepadku aplikacji *Internetowy Notatnik* wskazano przyczyny luk w zakresie bezpieczeństwa oraz metody obrony przed atakami.

2. Klasyfikacja ataków

Dwie organizacje podjęły się problemu klasyfikacji ataków: OWASP [5] oraz WASC [6]. Porównanie klasyfikacji zostało przedstawione w tabeli 1.

Istnieje wiele artykułów naukowych poruszających tematykę bezpieczeństwa serwisów internetowych. Po wpisaniu pojęcia „web attack” w wyszukiwarce serwisu IEEE Xplore uzyskano niespełna trzy tysiące rezultatów. Próbę usystematyzowania tych informacji podjęli Jeongseok Seo i inni w swoim artykule [7]. Dowiedli oni, że do pełnej

charakterystyki ataku powinna zostać wykorzystana główna przyczyna ataku oraz jego lokalizacja.

Tabela 1. Porównanie klasyfikacji OWASP oraz WASC

	OWASP	**WASC**
Nazwa	OWASP Top 10	The WASC Threat Classification
Autor	OWASP Foundation	Web Appliaction Security Consortium
Forma reportu	Strona www	Strona www Plik pdf
Częstotliwość publikacji	Ukazały się 3 edycje, co 3 lata	Jednorazowa publikacja
Ilość opisanych zagrożeń	10 najpopularniejszych	49 zagrożeń
Informacje dodatkowe	• przykłady • metody zabezpieczeń	• przykłady • metody zabezpieczeń

2.1. Injection

Jednym z najpopularniejszych atakiem z grupy Injection jest atak SQL Injection. Polega on na wstrzyknięciu w zapytanie SQL (ang. Structured Query Language) kodu mogącego wywołać nieoczekiwane działanie [8].

Przykładowym celem może stać się formularz logowania, w którym użytkownik podając hasło i login zyskuje dostęp do chronionych zasobów. Aplikacja w celu weryfikacji poprawności hasła odwołuje się do bazy danych i pobiera dodatkowe informacje o użytkowniku. Wykonane zostaje zapytanie bazodanowe z Przykładu 1.

Przykład 1. Zapytanie pobierające dane o użytkowniku

```
SELECT * from t_user where login = '[f_login]' and passwod = '[f_password]'
```

gdzie f_login i f_password to wartość pola login i hasło z formularza logowania. Dla uproszczenia przykładu pominięto kwestię hashowania hasła.

Jeśli zapytanie zwróci niepusty rezultat aplikacja może uznać, że hasło jest poprawne i użytkownik zostanie zalogowany. W przeciwnym wypadku wyświetlony zostanie komunikat błędu. Atak SQL Injection polegać może na wpisaniu do formularza w polu hasło wartości: ' or '1'='1. Poprzez zastosowanie tautologii możliwe jest manipulowanie wynikami zwracanymi przez aplikację. Innym sposobem wykorzystywanym przez atakujących może być złączenie rezultatu zapytania z wynikiem innego zapytania (operator UNION). Kluczem do powodzenia jest wtedy dopasowanie ilości oraz typu kolumn zwracanych przez obydwa zapytania. W niektórych przypadkach niemożliwe jest uzyskanie wyniku zapytania. Jeśli rezultatem ataku jest informacja binarna mamy do czynienia z atakiem SQL Blind Injection.

Przyczyn niebezpieczeństwa tego typu należy doszukiwać się w niewystarczającej filtracji danych przekazywanych do zapytań SQL [9].

Do grupy Injection zaliczamy także Command i Log Injection [10]. Z pierwszym z tech zagrożeń mamy do czynienia podczas wykonywania polecań systemu operacyjnego przez aplikację. Drugi dotyczy mechanizmu logowania informacji. Zmodyfikowane logi aplikacji mogą utrudnić zidentyfikowanie problemów związanych z jej działaniem.

2.2. XSS

Atak XSS (ang. Cross-Site-Scripting) nierozłącznie związany jest z wykorzystaniem języków skryptowych. Obecnie trudno wyobrazić sobie stronę internetową niewykorzystującą dobrodziejstw technologii JavaScript. Zagrożenie to znalazło się na 3 miejscu na liście OWASP. Polega ono na wykonaniu niepożądanego kodu skryptowego w przeglądarce klienta. Rezultatem działania takiego kodu może być wykradzenie ciasteczek (w tym identyfikatorów sesji) oraz modyfikacja zawartości treści prezentowanych na stronie [11]. Istnieje klasyfikacja tego ataku pod względem miejsca zlokalizowania niebezpiecznego kodu. Rodzaje ataków:

- stored XSS – niebezpieczny kod przechowywany jest przez aplikację (np. w bazie danych),
- reflected XSS – niebezpieczny kod przekazywany jest poprzez parametry GET w spreparowanym przez atakującego adresie URL.

Ponownie zagrożenie zostanie zobrazowane przykładem. Popularnym elementem współczesnych serwisów internetowych są wyszukiwarki. Dane wpisywane w pola wyszukiwania po zatwierdzeniu kryteriów wyszukiwania znajdują się w parametrach GET. Strona prezentująca rezultat wyszukiwania odczytuje te dane oraz prezentuje wartość parametru. W standardowym scenariuszu nie może tu dojść do niebezpieczeństwa. Problem pojawić się może w wypadku gdy atakujący przygotuje URL zawierający szkodliwy kod JavaScript. Kod taki może przesłać dane o atakowanym bezpośrednio na serwer atakującego (Przykład 2).

Przykład 2. Zapytanie pobierające dane o użytkowniku

```
document.write('<img src="http://strona_atakujacego.pl/?stolen_cookie=' + document.cookie + '" />')
```

W powyższym przykładzie zaprezentowano kod, który umożliwia przeprowadzenie ataku. Gdy kod wykona się w przeglądarce zostanie wysłany request GET na zewnętrzny serwer, a w parametrach zostaną przekazane wrażliwe dane.

Zapobieganie atakowi możliwe jest w dwojaki sposób. Pierwszy polega na filtrowaniu danych mających zostać zapisane do bazy danych. Filtracja powinna uniemożliwić (lub zneutralizować) zapis kodu skryptowego. Inną możliwością jest filtrowanie danych na poziomie wyświetlania. W tym celu mogą zostać wykorzystane dodatkowe biblioteki (np. ESAPI [12]) lub mechanizmy oferowane w standardowej bibliotece języka programowania (np. funkcja htmlspecialchars w php).

Najpopularniejszą udaną próbą wykonania ataku tego typu jest historia Samyego Kamkara [13]. Przygotowany przez niego skrypt został umieszczony w serwisie Myspace. Powodował on wysłanie zaproszenia do znajomych w imieniu zaatakowanego użytkownika. Atakujący w krótkim czasie zyskał ponad milin znajomych w serwisie. Luka w zabezpieczeniach została szybko zauważona i naprawiona.

2.3. Uwierzytelnianie

Jest to proces polegający na potwierdzeniu przez serwer tożsamości klienta. W przypadku aplikacji internetowych odbywa się zazwyczaj przy pomocy hasła i loginu. Niekiedy wykorzystywane są także dodatkowe mechanizmy:

- tokeny,
- weryfikacja SMS,
- listy haseł jednorazowych.

Strona OWASP opisuje szereg dobrych praktyk związanych z tym procesem. Obejmują one zagadnienia przechowywania, częstotliwości zmian, zasad przypominania oraz złożoności haseł. Autorzy zwracają uwagę na ataki związane z sesją, czyli z mechanizmem odpowiedzialnym za zapamiętanie stanu aplikacji w ramach każdego z klientów. Jednym z zagrożeń jest Session fixation. Scenariusz takiego ataku może polegać na zmuszeniu użytkownika do wykorzystania znanego atakującemu identyfikatora sesji, poprzez przekazanie go w adresie URL. Nie wszystkie serwery aplikacyjne przechowują id sesji w parametrach GET. W momencie gdy ofiara ataku wejdzie na stronę poprzez taki link i zaloguje się atakujący zyska dostęp do jego uprawnień. Rozwiązanie tego ataku zostało wprowadzone w niektórych serwerach aplikacyjnych (np. Apache Tomcat 6.0.23). Polega ono na automatycznym tworzeniu nowej sesji w momencie logowania.

Odrębną kwestę stanowi sposób przechowywania haseł użytkowników. Niedopuszczalne jest przechowywanie plain textu. Obecnie standardem jest wykorzystanie algorytmu bcrypt. Od wersji 5.5.0 języka php istnieją funkcje dające możliwość hashowania oraz weryfikacji poprawności hasła przy pomocy wspomnianego algorytmu (password_hash, password_verify [14]).

2.4. Niezabezpieczone bezpośrednie odniesienia do obiektów

Zazwyczaj zasoby przechowywane w tabelach bazy danych identyfikowane są przez klucze główne. Stanowią one ich identyfikatory. W przypadku gdy część zasobów nie powinna być dostępna dla wszystkich użytkowników serwisu może dojść do ataku tego typu. Polega on na nieautoryzowanym dostępie do treści. Identyfikatory zasobów mogą być przekazywane do aplikacji poprzez parametry GET adresu. Np.:

zasob.php?id=12

Atakujący może zmienić wartość parametru id z zamiarem uzyskania dostępu do zasobów innych użytkowników. Aplikacja powinna wykryć próbę takiej manipulacji. Weryfikacja uprawnień do odczytu danego zasobu może odbyć się poprzez zaimplementowanie mechanizmu ACL (ang. access control list).

3. Badania

Przedmiotem badań była aplikacja *Internetowy Notatnik* autorstwa Michała Furtaka. Zaimplementowana została ona w języku php. Daje możliwość rejestracji użytkowników oraz tworzenia notatek. Zapisywane są one w bazie MySql. Notatki domyślnie widoczne są tylko dla ich autorów. Dodatkowo udostępniona została możliwość dzielenia się z innymi swoimi notatkami poprzez wysłanie linków prowadzących do nich.

3.1. Środowisko testowe

W badaniach wykorzystano aplikację uruchomioną na środowisku lokalnym. Zastosowano serwer aplikacji Apache 2.4 oraz serwer baz danych MySql 5.6.25. Do przygotowania aplikacji testowych użyto IDE NetBeans. Wykorzystano dodatkowe aplikacje:

- sqlmap,
- Fiddler 4,
- przeglądarka Firefox z pluginem Firebug.

Całość pracowała pod kontrolą systemu Windows 7 Professional.

3.2. Przebieg badań

Przeprowadzono szereg testów mający na celu wykluczyć lub potwierdzić podatność aplikacji na ataki.

Testy SQL Injection

Przeprowadzono próbę ataku przy pomocy narzędzia sqlmap. Następnie na podstawie badań literaturowych określono cztery rodzaje wstrzyknięć, które spowodować mogą naruszenie bezpieczeństwa aplikacji:

- tautologie,
- usuwanie zawartości tabeli (polecenia delete, truncate),
- operator UNION,
- test na Blind SQL Injection.

Dodatkowo przygotowane zostało pięć aplikacji pomocniczych, które pozwoliły na przetestowanie mechanizmów obrony, których nie wykorzystano w *Internetowym Notatniku*. Przetestowano skuteczność:

- mechanizmu Prepared Statements (w wersjach: poprawnie zaimplemetowanej (I) i błędnej (II)),
- frameworka Hibernate (Przykład 3),
- frameworka Medoo,
- biblioteki ESAPI.

Przykład 3. Program testujący framework Hibernate

```
public static void main(String[] args) {

Session session =
   SessionFactory.getSessionFactory().openSession();
```

```
    doTest(session, "17 OR true=true --");
    doTest(session, "17 OR 'test' = 'test'");
    doTest(session, "17 ; drop table test");
    doTest(session, "17 ; truncate table test");
    doTest(session, "20 and VERSION( ) = '5.6.25'");
    doTest(session, "17 UNION select id, null, name as
'title', PASSWORD as 'content', null, null, null from
user");
    session.close();
}

public static void doTest(Session session, Object param) {
    System.out.println("=====  Test: " + param + "
=====");
    try {
        List<Note> notes = session.createQuery("FROM
Note where id = :id ")
                .setParameter("id", param).list();
        for (Note note : notes) {
            System.out.println(note);
        }
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}
```

Testy XSS

Wytypowane zostały miejsca potencjalnego ataku: strona tworzenia nowego użytkownika oraz dodawania notatki. Przeprowadzono test przy pomocy udostępnionych skryptów JavaScript. Dodatkowo wykonano test mechanizmu blokującego ataki XSS w przeglądarkach internetowych. Do tego celu przygotowano stronę testową wyświetlającą zawartość parametru GET. Dokonano próby przekazania w parametrze kodu JavaScript.

Testy mechanizmu uwierzytelniania

Metodą eksperymentalną sprawdzono w jaki sposób aplikacja weryfikuje złożoność haseł podczas rejestracji. Dokonano próby przeprowadzenia ataku Session Fixation. Zweryfikowano w jaki sposób przechowywane są hasła w bazie danych.

Testy podatności na ataki związane z bezpośrednim odwołaniem do obiektu

Przygotowano aplikację testową wysyłającą dużą ilość requestów do aplikacji. Celem było odnalezienie identyfikatorów notatek innych użytkowników.

3.3. Prezentacja rezultatów badań

Testy SQL Injection

Testy wykazały ze mechanizmy obronne wykorzystane w aplikacji Internetowy Notatnik spełniły swoje cele. Nie udało się przeprowadzić skutecznego ataku na tej aplikacji.

Wyniki dotyczące porównania mechanizmów obrony zostały przedstawione w tabeli 2.

Tabela 2. Porównanie mechanizmów obrony przed SQL Injection

	Prepared Statements (wersja I)	Prepared Statements (bwersja II)	Hibernate	Medoo	ESAPI
tautologie	☐	☐	☐	☐	☐
usuwanie zawartości tabeli	☐	☐	☐	☐	☐
operator UNION	☐	☐	☐	☐	☐
Blind SQL Injection	☐	☐	☐	☐	☐

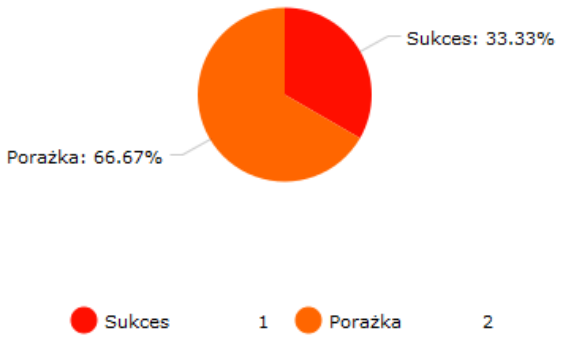
Wszystkie poprawnie wykorzystane metody dają gwarancję bezpieczeństwa. Nieprawdą jest, że sama metoda przesądza o tym, że aplikacja jest bezpieczna. W przypadku testu aplikacji gdzie w niepoprawny sposób wykorzystano Prepered Statemants przyczyny potencjalnego ataku należy dopatrywać się w błędach programisty implementującego serwis.

Testy XSS

Testy wykazały, że strona dodawania notatki umożliwia skuteczne przeprowadzenie ataku XSS typu pierwszego. Odpowiednio preparując request możliwe okazało się wyświetlenie komunikatu zawierającego identyfikator sesji użytkownika:

/add.php?title="%2F><script>+alert(document.cookie%3B+<
%2Fscript>&content=&submit=

Łącznie przeprowadzono trzy próby wykonania ataku XSS. Tylko powyższa zakończyła się sukcesem (Rys. 1).



Rys. 1. Skuteczność przeprowadzonych ataków XSS

Testy mechanizmu uwierzytelniania

Analiza kodu pozwoliła stwierdzić, że wykorzystywany jest hash md5. W celu poprawy bezpieczeństwa

przygotowano poprawkę. Obejmuje ona zmianę sposobu hashowania na taką, która wykorzystuje algorytm bcrypt.

Dodatkowo zaproponowano ulepszenie polegające na wprowadzeniu validatora weryfikującego złożoność hasła. Nowy mechanizm uwzględnia nie tylko długość wprowadzonego ciągu ale także zawartość znaków z różnych grup (małe i wielkie litery, cyfry, znaki specjalne).

Atak Session Fixation zakończył się sukcesem. Wykazano, że możliwe jest przejęcie sesji innego użytkownika. Zaproponowano rozwiązanie tworzące nową sesję bezpośrednio przed zalogowaniem użytkownika. Rozwiązanie to przyczyniło się do wyeliminowania tej nieprawidłowości.

Testy podatności na ataki związane z bezpośrednim odwołaniem do obiektu

Testy nie wykazały nieprawidłowości. Aplikacja weryfikuje uprawnienia dostępu do obiektu.

4. Wnioski

Badania wykazały, że nawet tak prosta aplikacja, jaką jest *Internetowy Notatnik* może zawierać błędy. Wykazane zostały nieprawidłowości z zakresu sposobu przechowywania hasła, zarządzania sesją oraz podatności na ataki XSS. Dla wszystkich tym problemów zostało opracowane rozwiązanie, które zostało wdrożone w aplikację.

Przyczyn podatności aplikacji na ataki doszukiwać się można w nieprawidłowym wykorzystaniu narzędzi mającym im zapobiegać (lub ich nie wykorzystaniu). Osobną kwestią, na którą należy zwrócić uwagę jest środowisko produkcyjne aplikacji. Konieczne jest instalowanie aktualizacji serwera aplikacyjnego oraz wszystkich innych komponentów systemu.

Najsłabszą rolę w systemie stanowi użytkownik. Na nic nie zdadzą się zabezpieczenia jeśli ten nie będzie świadomy czekających na niego zagrożeń. Rolą developera jest wymuszenie pewnych zachowań takich jak okresowa zmiana hasła.

Literatura

[1] R. Cohen-Almagor, Internet History, International Journal of Technoethics, 2011
[2] Netcraft, Web Server Survey, 2014
[3] R. Krzyżaniak, Web 2.0 w Polsce, 2007
[4] J. Mirković i inni, Attacking DDoS at the Source, 2002
[5] https://www.owasp.org/ [20.11.2016]
[6] http://projects.webappsec.org/ [20.11.2016]
[7] S. Jeongseok, Web server attack categorization based on root causes and their locations, 2004
[8] K. Navdeep, Modeling a SQL injection attack, 2015
[9] M. Qbea'h, Detecting and Preventing SQL Injection Attacks: A Formal Approach, 2016
[10] http://www.jtmelton.com/2010/09/21/preventing-log-forging-in-java/ [20.11.2016]
[11] M. K. Gupta i inni, Predicting Cross-Site Scripting (XSS) security vulnerabilities in web applications, 2015
[12] A. Singh i inni, A Survey on XSS web-attack and Defense Mechanisms, 2014
[13] L. Franceschi-Bicchierai, The MySpace Worm that Changed the Internet Forever, 2015
[14] http://php.net/manual/en/function.password-hash.php [20.11.2016]