OWASP

The Open Web Application Security Project

OWASP Top 10 – 2010 rc1

Dziesięć najbardziej krytycznych zagrożeń w web aplikacjach

Release Candidate 1

(tłum.+ zmiany: Michał Wiczyński, http://thinklikeninja.blogspot.com)

1. **Zmiany w najnowszym dokumencie:**

Dodano	Usunięto	Zmiana
A6 – Security Misconfiguration		(2004) A10 – Insecure Configuration Management
A8 – Unvalidated Redirects and Forwards		
	A3 – Malicious File Execution	
	A6 – Information Leakage and Improper Error Handling	

2. **T10. Poszczególne punkty:**

2.1 A1- Injection: 'błędy wstrzyknięcia',

zaliaczamy do nich błędy: SQL, OS shell, LDAP, XPath Injection. Błędy te powstają podczas przesłania specjalnie spreparowanych danych do interpretera, jako część zapytania lub polecenia(prościej mówiąc: wartości traktowane są jak polecenia). Jeżeli dane te, nie są w odpowiedni sposób filtrowane, osoba atakująca ma wówczas możliwość wywołania własnych poleceń/ zapytań. Może to spowodować dostęp do poufnych informacji, lub nieautoryzowanych działań w systemie.

Przykładowy atak:

Błąd ten jest nadal bardzo popularny, a jego ofiarą padają potężne firmy. Przykładem może być całkiem niedawny atak na Yahoo (http://news.softpedia.com/news/Yahoo-Local-Hacked-120044.shtml).

Warte uwagi:

http://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

http://www.owasp.org/index.php/Command_Injection

http://www.owasp.org/index.php/ASVS

http://www.owasp.org/index.php/Reviewing_Code_for_OS_Injection

http://owasp-esapi-java.googlecode.com/svn/trunk_doc/latest/org/owasp/esapi/Validator.html

http://cwe.mitre.org/data/definitions/89.html

2.2 A2- Cross Site Scripting (XSS): ‘skrypty międzyserwisowe’,

błędy tego typu powstają podczas próby wyświetlenia w przeglądarce internetowej danych pochodzących od użytkownika, bez ich wcześniejszej walidacji. Umożliwia to atakującemu, wykonanie skryptu na prawach ofiary. Efektem może być przechwycenie sesji zalogowanego użytkownika, podmiana zawartości serwisu lub przekierowanie użytkownika na stronę zawierającą inne szkodliwe skrypty lub oprogramowanie.

Przykładowy atak:

XSS polega na wykonaniu kodu HTML, który został wprowadzony przez osobę trzecia a jest wykonywany przez dana aplikacje. Typowy przykład JavaScript użytego podczas ataku typu XSS, powodujący wyświetlenie okienka z zawartością pliku cookie:

<script>alert(document.cookie);</script>

Na podstawie informacji z ‘ciasteczka’ można określić m.in. login, hasło ofiary.

Oczywiście bardziej wyrafinowane metody wykorzystania tego typu błędów, umożliwiają np. obserwowanie, jakie strony przegląda ofiara, lub tworzą z komputera ofiary komputer zombie.

Warte uwagi:

http://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

http://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

http://www.owasp.org/index.php/ESAPI

http://www.owasp.org/index.php/ASVS

http://www.owasp.org/index.php/ASVS

http://www.owasp.org/index.php/Testing_for_Cross_site_scripting

http://cwe.mitre.org/data/definitions/79.html

http://ha.ckers.org/xss.html

2.3 A3- Broken Authentication and Session Management: 'niepoprawna obsługa uwierzytelniania i sesji',

Bardzo często funkcje związane z autentykacją, oraz zarządzaniem sesjami nie są odpowiednio zaimplementowane. Umożliwia to osobom atakującym ujawnienie haseł (w zależności od struktury serwisu), kluczy, tokenów sesji, lub wykonania poleceń na prawach zalogowanego użytkownika.

Dlaczego pojawiają się tego typu błędy? HTTP jest protokołem typu 'stateless', co oznacza, że dane odnośnie zalogowanego użytkownika muszą być wysyłane przy każdym odwołaniu się do strony. Wszelkie dane (przy braku aktywnego SSL) pojawiają się w postaci niezaszyfrowanej, umożliwiając podsłuchanie tych danych w dowolnym momencie. Najczęstszym problemem są dane zawarte w tzw. sesjach (SessionID). Odwołując się do innego serwisu niż tego na którym jesteśmy zalogowani, jest możliwość przesłania tego typu zmiennych (np. via referrer). W takim przypadku, osoba atakująca ustawia w swojej przeglądarce sesję odwiedzającego, co w wielu przypadkach umożliwia poprawne zalogowanie się do obcego serwisu.

Przykładowy atak:

Jesteśmy zalogowani w popularnym serwisie randkowym i chcemy pokazać znajomej osobie ciekawe zdjęcie odnalezione w wyszukiwarce tego serwisu. Link wygląda tak:

http://jakisserwisrandkowy/zdjecie-10-10/?sessionid=34923592752

Przekazując w/w link, przekazujemy również dane odnośnie naszej sesji. Klikając w link, jest możliwość, że zostaniemy automatycznie zalogowani jako osoba która wysłała nam link.

Warte uwagi:

http://cwe.mitre.org/data/definitions/287.html

http://www.owasp.org/index.php/Testing_for_authentication

http://www.owasp.org/index.php/Guide_to_Authentication

http://owasp-esapi-java.googlecode.com/svn/trunk_doc/latest/org/owasp/esapi/User.html

http://owasp-esapi-java.googlecode.com/svn/trunk_doc/latest/org/owasp/esapi/Authenticator.html

http://www.owasp.org/index.php/ASVS

2.4 A4- Insecure Direct Object References: 'niezabezpieczone bezpośrednie odwołanie do obiektu',

pojawia się kiedy zostaje ujawniona referencja do wewnętrznego obiektu. Może być to plik, katalog lub wartość klucza z bazy danych. Bez kontroli dostępu, lub innych kontroli, atakujący może manipulować referencjami w celu uzyskania dostępu do poufnych informacji.

Punkt ten może, a nawet powinien, służyć jako wprowadzenie do punktu 7. Powody występowania błędu:

- 'ukrywanie' wrażliwych informacji w polach typu: hidden

- nadmierne ufanie danym wprowadzanym przez użytkownika

- brak walidacji tego typu danych po stornie serwera.

Przykładowy atak:

Omówię dosyć częsty błąd, umożliwiający przeglądanie danych innych osób, np. faktur, lub notatek. Załóżmy, że istnieje aplikacja umożliwiająca dokonywanie płatności za internet przez internet. Po zalogowaniu się można przeglądać listę dokonanych opłat oraz własnych danych osobowych przez link:

http://platnoscizainternet.lan/?abonent=5654

Jeżeli występuje tego typu błąd, to po zamianie wartości 5654 na 1337, najprawdopodobniej będziemy w stanie przejrzeć dane abonenta o id 1337. Możliwości jest wiele, w zależności od struktury serwisu. Co się stanie, jeżeli dane te nie są brane z bazy danych ale z pliku w katalogu /var/www/users/abonenci/5654? Wtedy wystarczy zmienić link na:

http://platnoscizainternet.lan/?abonent=../../../../../../../etc/passwd

i zamiast danych abonenta widzimy zawartość pliku /etc/passwd.

Warte uwagi:

http://www.owasp.org/index.php/Top_10_2007-Insecure_Direct_Object_Reference

http://owasp-esapi-java.googlecode.com/svn/trunk_doc/org/owasp/esapi/AccessController.html

http://owasp-esapi-java.googlecode.com/svn/trunk_doc/latest/org/owasp/esapi/AccessController.html

http://www.owasp.org/index.php/ASVS

http://cwe.mitre.org/data/definitions/639.html

http://cwe.mitre.org/data/definitions/22.html

2.5 A5- Cross Site Request Forgery: 'Fałszowanie żądań',

nieautoryzowany użytkownik wabiąc ofiarę na swoją stronę, może wykonywać zapytania jako zalogowana ofiara. Atak ma na celu skłonić użytkownika zalogowanego do serwisu internetowego do uruchomienia odnośnika, który wykona w wybranym serwisie akcję, do której atakujący nie miałby dostępu (np brak cookie).

Przykładowy atak:

Użytkowniczka Małgosia, na stałe zalogowana do forum internetowego, może w pewnym momencie otworzyć spreparowany przez Jasia link:

http://forum/changedata.php?login=jasio&new_pass=123&mail=pwn3d@domena

Link ten zmieni jej dane kontaktowe albo usunie wątek dyskusji. Jako link może również posłużyć obrazek, którego adres został odpowiednio przygotowany, a konsekwencje wykonanej akcji mogą być znacznie poważniejsze.

Warte uwagi:

http://www.owasp.org/index.php/CSRF

http://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet

http://www.owasp.org/index.php/CSRFGuard

http://www.owasp.org/index.php/ESAPI

http://owasp-esapi-java.googlecode.com/svn/trunk_doc/latest/org/owasp/esapi/HTTPUtilities.html

http://www.owasp.org/index.php/Testing_for_CSRF_(OWASP-SM-005)

http://www.owasp.org/index.php/CSRFTester

http://cwe.mitre.org/data/definitions/352.html

2.6 A6- Security Misconfiguration: ‘niepoprawne ustawienia’,

dana aplikacja, Framework, web Server, Server aplikacji, platforma; powinny posiadać odpowiednie ustawienia mające wpływ na bezpieczeństwo. Wszelkie ustawienia powinny zostać zdefiniowane, zaimplementowane, oraz utrzymywane przez cały czas, ponieważ często domyślne ustawienia w/ w części nie zapewniają żadnego bezpieczeństwa. Do tego typu problemów możemy zaliczyć zostawianie domyślnych haseł oraz konfiguracji, możliwości listowania katalogów, brak podejmowania akcji w przypadku plików typu *.inc, włączone informacje odnośnie błędów aplikacji i wiele innych.

Warte uwagi:

http://www.owasp.org/index.php/Configuration

http://www.owasp.org/index.php/Testing_for_configuration_management

http://www.owasp.org/index.php/A10_2004_Insecure_Configuration_Management

http://www.owasp.org/index.php/ASVS

http://www.pcmag.com/article2/0,2817,11525,00.asp

http://cwe.mitre.org/data/definitions/2.html

2.7 A7- Failure to Restrict URL Access: 'Brak zabezpieczeń dostępu przez URL',

wielokrotnie dostęp do poufnych danych, lub narzędzi administracyjnych. Aplikacje powinny wykonywać dodatkowe sprawdzanie czy dana osoba ma dostęp do tego typu danych, nawet jeżeli zna 'ukryte' URL.

Przykładowy atak:

Najczęściej popełniany błąd, 'ukrywanie' panelu administracyjnego pod URL:

http://serwis/adminsitracja (oczywiście może to być: /admin/, /adm/, /panel/ itp....)

W tym momencie programista stwierdził: 'skoro nie ma bezpośredniego odwołania do panelu administracyjnego na stronie głównej serwisu, to nikt nie wpadnie na pomysł żeby wpisać /administracja/...więc po co zakładać logowanie na taki panel'. Jak się okazuje, zawód niektórych ludzi polega właśnie na takim 'wpadaniu na pomysł' ;)

Warte uwagi:

http://www.owasp.org/index.php/Top_10_2007-Failure_to_Restrict_URL_Access

http://owasp-esapi-java.googlecode.com/svn/trunk_doc/latest/org/owasp/esapi/AccessController.html

http://www.owasp.org/index.php/Guide_to_Authorization

http://www.owasp.org/index.php/Testing_for_Path_Traversal

http://www.owasp.org/index.php/Forced_browsing

http://www.owasp.org/index.php/ASVS

http://cwe.mitre.org/data/definitions/285.html

2.8 A8- Unvalidated Redirects and Forwards: 'Brak walidacji Przekierowań',

aplikacje wielokrotnie przekierowują użytkownika na inne strony lub serwisy na podstawie przesłanych danych. Bez odpowiedniej walidacji tych danych, użytkownik może zostać skierowany w zupełnie inne miejsce, np. na stronę phishingową lub ze szkodliwym oprogramowaniem.

Przykładowy atak:

Serwis po poprawnym zalogowaniu użytkownika, wywołuje następujący adres:

http://serwis/redirect.php?strona=main.php

Wartość zmiennej 'strona', określa na jaką nazwę strony/ pliku ma zostać przekierowany użytkownik. Wykorzystując błędną obsługę wartości zmiennej, możemy wpisać:

http://serwis/redirect.php?strona=http://evil.com/malware.exe

co w rezultacie przekieruje użytkownika na stronę ze szkodliwym oprogramowaniem.

Warte uwagi:

http://www.owasp.org/index.php/Open_redirect

http://owasp-esapi-java.googlecode.com/svn/trunk_doc/latest/org/owasp/esapi/filters/SecurityWrapperResponse.html

http://cwe.mitre.org/data/definitions/601.html

http://cwe.mitre.org/data/definitions/601.html

http://googlewebmastercentral.blogspot.com/2009/01/open-redirect-urls-is-your-site-being.html

2.9 A9- Insecure Cryptographic Storage: ‘Błędy szyfrowania danych’,

aplikacje w wielu przypadkach przechowują dane poufne w niezaszyfrowanej postaci. Mogą to być dane typu SSN (Social Security Number), dane do autentykacji, dane osobowe.
Poufne dane należy przechowywać w zaszyfrowanej postaci, jednocześnie zapewniając dostateczny poziom szyfrowania.

Przykładowy atak/ błąd:

Załóżmy, że wszystkie dane odnośnie użytkowników danego serwisu trzymane są w bazie w postaci niezaszyfrowanej, loginy hasła dane osobowe itp. Wykorzystując np. błąd typu Injection, osoba atakująca jest w posiadaniu tych wszystkich danych w niezakodowanej postaci, dzięki czemu może je niezwłocznie wykorzystać do dalszych ataków. Wystarczyłoby zastosować najprostsze szyfrowanie/ hashowanie danych (np. haseł) z odpowiednim ‘salt’em’, co ograniczyłoby w pewnym stopniu możliwości wykorzystania tych danych.

Warte uwagi:

http://www.owasp.org/index.php/ASVS

http://www.owasp.org/index.php/Top_10_2007-Insecure_Cryptographic_Storage

http://www.owasp.org/index.php/Guide_to_Cryptography

http://www.owasp.org/index.php/Codereview-Cryptography

http://cwe.mitre.org/data/definitions/312.html

http://cwe.mitre.org/data/definitions/326.html

2.10 A10- Insufficient Transport Layer Protection: 'Niedostateczne zabezpieczenia wymiany danych',

aplikacje często nie posiadają odpowiednio zabezpieczonych kanałów przesyłu danych. Może to być brak stosowania SSL podczas logowania, wygasłe certyfikaty, lub zbyt słabe szyfrowanie połączenia.

Przykładowy atak:

Jak już było wspomniane wyżej, brak użycia SSL podczas logowania się na serwis pocztowy, lub do banku, umożliwia podsłuchanie przesyłanych danych, bez informowania o tym użytkownika (tak, jestem świadom ostatnich błędów związanych z '*' w certyfikatach;)).

Warte uwagi:

http://www.owasp.org/index.php/ASVS

http://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet

http://www.owasp.org/index.php/Top_10_2007-Insecure_Communications

http://www.owasp.org/index.php/Guide_to_Cryptography

http://www.owasp.org/index.php/Testing_for_SSL-TLS

http://cwe.mitre.org/data/definitions/319.html

https://www.ssllabs.com/ssldb/index.html

http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

Na sam koniec chciałbym dodać od siebie jeszcze jedno zdanie, niezwykle ważne przy wykrywaniu, oraz zapobieganiu błędom w aplikacjach:

„nigdy nie ufaj danym od użytkownika”