



Magazine

Testy penetracyjne aplikacji webowych

Autor: inż. Daniel Brzozowski

O autorze:

Daniel jest programistą o dużym doświadczeniu w tworzeniu aplikacji internetowych w różnych technologiach od ASP .NET poprzez frameworki oparte na Pythonie na PHP/MySQL kończąc. Ostatnie dwa lata pracował jako Front End Developer biorąc udział w tworzeniu aplikacji dla głównych instytucji finansowych w Polsce takich jak PKO BP, Inteligo czy Raiffaisen.

W chwili obecnej mieszka w Wielkiej Brytanii w Londynie, gdzie zamierza pracować przy bezpieczeństwie aplikacji webowych.

Dodatkowo jest studentem Wydziału Matematyki i Nauk Informatycznych i aktualnie pisze pracę magisterską pod tytułem „Testy penetracyjne aplikacji webowych”.

W kwietniu tego roku zdobył tytuł **MCPD** (Microsoft Certified Professional Developer), który jest ostatnim certyfikatem w ścieżce .NET Framework 2.0 Web Applications. Dodatkowo jest aktywnym członkiem Open Web Application Security Project.

W wolnych chwilach trenuje sporty walki (Krav Maga – stopień P1), słucha dużych ilości muzyki oraz uważnie śledzi nowinki ze świata aplikacji webowych.



Basic

Level

Testowanie oprogramowania

Section in the magazine

Wprowadzenie

W dzisiejszych czasach trudno przejść obojętnie obok problemu zabezpieczeń aplikacji webowych. Co jakiś czas z mediów możemy dowiedzieć się nowej luce czy w systemach bankowych, portalach społecznościowych czy to innych tego rodzaju aplikacjach. Największym problemem po ujawnieniu takich dziur jest przede wszystkim spadek reputacji i utrata zaufania klientów/użytkowników, co na ogół skutkuje też stratą dużych pieniędzy.



Znaczna część aplikacji webowych nie jest implementowana z zachowaniem odpowiednich środków ostrożności, co powoduje wszechobecność tego rodzaju podatności. Tego typu programy, często ogólnodostępne, umożliwiają użytkownikom wprowadzanie danych, które w przypadku niezachowania odpowiednich środków ostrożności mogą prowadzić do użycia owego oprogramowania w inny sposób niż przeznaczony.

Braki w wiedzy z zakresu bezpieczeństwa i goniące terminy to podstawowe przyczyny powstawania niezabezpieczonego kodu. Jednak można również zauważyć, że powoduje to wzrost zainteresowania ze strony firm zagadnieniem testów penetracyjnych, dzięki czemu ta tematyka staje się coraz bardziej popularna.

Programiści, kierownicy zespołów jak i osoby zajmujące wyższe stanowiska w firmach są coraz bardziej świadomi, że bez odpowiednich testów bezpieczeństwa nie jest możliwe tworzenie bezpiecznych aplikacji.

OWASP

OWASP (Open Web Application Project)¹⁾ – międzynarodowa organizacja non-profit skupiona na bezpieczeństwie aplikacji internetowych. Jej misją jest uczynienie zagrożeń aplikacji widocznymi, umożliwiając ludziom oraz organizacjom podejmowanie świadomych decyzji dotyczących bezpieczeństwa.

Najbardziej znane projekty:

- OWASP Top 10;
- Application Security Verification Standard;
- Przewodniki: OWASP Testing Guide, OWASP Development Guide i inne;
- Narzędzia: WeScarab, WebGoat, ESAPI i inne.



Rysunek 1: Logo OWASP

Test penetracyjny

Test penetracyjny definiujemy jako proces mający na celu praktyczną ocenę bieżącego stanu bezpieczeństwa systemu teleinformatycznego, w szczególności obecności znanych podatności i odporności na ataki hackerskie. Test penetracyjny ma potwierdzać podatności systemu na takie ataki oraz skuteczność zabezpieczeń w środowisku produkcyjnym lub możliwie najbardziej zbliżonym do środowiska rzeczywistego. Istotnym faktem jest to, że testy penetracyjne są one zawsze przeprowadzane na gotowej aplikacji.

1.1. Podział testów penetracyjnych

Testy penetracyjne możemy podzielić ze względu na:

- znajomość sprawdzanego systemu;
- sposób ich przeprowadzenia.

1.1.1. Podział testów penetracyjnych ze względu na znajomość systemu

Biorąc pod uwagę posiadaną wiedzę na temat systemu możemy wyróżnić trzy metodyki przeprowadzania testu penetracyjnego:

- **testy czarno skrzynkowe (Black box)** – metodologia ta zakłada całkowity brak znajomości systemu. Podstawową jej zaletą jest maksymalne odtworzenie sytuacji, w jakiej pracuje potencjalny intruz;
- **testy biało skrzynkowe (White box)** – w metodologii tej zakładamy pełną znajomość testowanego systemu włącznie z jego architekturą, użytymi technologiami i kodem źródłowym;
- **testy szaro skrzynkowe (Grey box)** – jest to podejście hybrydowe łączące elementy z obu powyższych podejść np. poprzez dokładną znajomość architektury, ale bez dostępu do kodu źródłowego aplikacji.

1.1.2. Podział testów penetracyjnych ze względu na sposób ich przeprowadzenia

Jak większość testów tak i testy penetracyjne możemy sklasyfikować biorąc pod uwagę sposób ich wykonania:

- **Testy automatyczne** – w pełni przeprowadzane przy użyciu specjalnych narzędzi. Nie wymagają one specjalistycznej wiedzy eksperckiej, są szybkie, jednak ich skuteczność jest ograniczona do niewielu obszarów.
- **Testy manualne** – wykonywane przez testera „ręcznie” wymagają one dużej wiedzy eksperckiej z zakresu testów penetracyjnych jak i biznesowej danego systemu, pochłaniają dużo czasu i są wrażliwe na ludzkie błędy, co zmniejsza ich wiarygodność.

W praktyce najlepsze rezultaty daje połączenie obu tych metodyk tzn. automatyzację w obszarach, w których jest to możliwe, i testy manualne wykorzystujące wiedzę ekspercką w pozostałych.

Narzędzia

Wśród narzędzi dedykowanych do testów penetracyjnych możemy wyróżnić następujące grupy funkcjonalne:

- **Proxy** – aplikacje pełniące rolę kolejnej warstwy pośredniczącej między przeglądarką a serwerem. Do jej najbardziej pożądanых cech można zaliczyć:
 - automatyczne przechwytywanie i modyfikacja wybranych na bazie stworzonych przez użytkownika reguł,
 - automatyczna modyfikacja nagłówek,
 - automatyczne powtarzanie.;
- **Analizatory tokenów sesji** – narzędzia służące do odkrywania prawidłowości w ID sesji użytkownika z następującą funkcjonalnością:
 - generowanie i zbieranie jak największej ilości ID sesji,
 - analiza prawidłowości i prezentacja wyników np. w graficznej postaci.;
- **Skanery automatyczne** – grupa narzędzi działających automatycznie, które przechodząc po kolejnych podstronach aplikacji starają się wykryć jak największą liczbę znanych luk;
- **Fuzzery** – programy starające się wykryć anomalie używając do tego celu losowych danych;
- **Spidery** – aplikacje zbierające informacje o strukturze danej aplikacji internetowej na podstawie ścieżek i linków w niej obecnych, używane w procesie mapowania serwisu;
- **Narzędzia do statycznej analizy** – programy używane do sprawdzenia kodu źródłowego pod względem bezpieczeństwa.



- Burp Suite – zintegrowana platforma do ataków na aplikacje internetowe, wieloplatformowa, dostępna w dwóch wersjach – darmowej i płatnej;
- W3AF – darmowe, napisane w Pythonie narzędzie z funkcją do testów automatycznych z predefiniowanymi profilami, posiada dwa interfejsy – graficzny i konsolowy;
- Nikto 2 – OpenSource’owy automatyczny skaner podatności serwerów i ich oprogramowania, posiada informacje o ponad 6000 luk na 260 serwerach;
- Grendel-Scan – darmowe, stworzone w Javie narzędzie do testów penetracyjnych z rozbudowanym modułem do skanowania automatycznego;
- Skipfish – narzędzie do testów automatycznych stworzone przez polskiego Hakera, proste w użyciu o wysokiej wydajności;
- WebScarab – rozbudowany framework o budowie pluginowej do testów penetracyjnych stworzony pod skrzydłami OWASP z możliwością rozszerzania.

Dobre praktyki

Dobre praktyki pomagają nam podejmować decyzje w nowych sytuacjach wykorzystując te same wzorce, do dobrych praktyk z zakresu bezpieczeństwa możemy zaliczyć:

- Wielopoziomowa ochrona – dzisiejsze aplikacje mają budowę wielopoziomową, w przypadku obejścia mechanizmów obronnych na jednym poziomie nadal istnieje szansa na odparcie ataku na kolejnych poziomach.
- Pozytywny model ochrony – model oparty na tzw. białych listach (ang. whitelist) czyli zdefiniowaniu co jest dozwolone i odrzuceniu całej reszty, ich przewaga nad czarnymi listami (ang. blacklist – opis co jest zabronione i zablokowaniu zabronionych elementów) polega na odporności na nowe ataki.
- Unikanie bezpieczeństwa przez ukrywanie – do dziś wiele programistów wierzy, że wkompilowując coś w kod źródłowy, nikt nie będzie w stanie tego odczytać, tego typu założenia są błędne.
- Założenia – nigdy nie powinniśmy ufać założeniom bezpieczeństwa poczynionym przez inne osoby np. „Nasz komponent nigdy nie zawiedzie”.
- Detekcja intruzów – możemy tutaj zastosować skomplikowane systemy detekcji intruzów IDS(ang. Intrusion Detection System) lub po prostu analizować logi aplikacji.
- Najśłabsze ogniwo – najśłabszym ogniwem w przypadku aplikacji webowej może być mechanizm autentykacji czy też płatności, tego typu ogniwa wymagają szczególnej uwagi od początku życia projektu.
- KISS (ang. Keep It Simple Stupid) – powszechna zasada stosowana w IT, polega na unikaniu nadmiarowych komplikacji, nagmatwany kod po prostu ma tendencję do zawierania większej ilości błędów.
- Minimalne uprawnienia – jeśli aplikacja korzysta z fizycznego pliku, upewnijmy się czy przypadkiem nie uzyskała ona dostępu do całego systemu plików, ponieważ stanowi to poważne zagrożenie.
- Bezpieczna awaria – aplikacje często ulegają awarii w niebezpieczny sposób ujawniając ważne informacje na temat samego programu, które mogą zostać wykorzystane przez potencjalnego intruza.

Bezpieczeństwo może zostać zintegrowane z całym cyklem życia projektu na wiele sposobów. Poczynając od analizy możliwych zagrożeń przed rozpoczęciem prac deweloperskich, poprzez tworzenie i analizę kodu źródłowego pod kątem standardów bezpieczeństwa, kończąc na testach penetracyjnych na gotowym produkcie i reakcji na znalezione błędy w aplikacji.



Rysunek 2: Bezpieczeństwo w cyklu życia projektu

Podsumowanie

ze wzrostem ilości aplikacji webowych wzrasta liczba problemów bezpieczeństwa związanych z nimi. Aktualnie obserwujemy namnażanie się aplikacji opartych na technologii AJAX oraz RIA (Rich Internet Application) zwanych Web 2.0. Tego typu aplikacji dostarczają ładny i użyteczny interfejs ich użytkownikom czyniąc je bardzo atrakcyjnymi, w tym samym czasie obserwujemy wzrost liczby nadużyć wobec tych nowych technologii.

Jeśli chodzi o bezpieczeństwo aplikacji internetowych niestety nie istnieje żaden złoty środek, żadne idealne narzędzie. Aby zapewnić zadowalający poziom bezpieczeństwa w organizacji trzeba wdrożyć standardy bezpieczeństwa w wielu obszarach: od edukacji pracowników, stosowania dobrych praktyk na testach penetracyjnych kończąc. Na rynku istnieje wiele dostępnych narzędzi o zróżnicowanej funkcjonalności, należy jednak pamiętać, że są to tylko narzędzia i należy wybrać po prostu te, z którymi nam się najwygodniej pracuje.

Referencje

1. <http://www.owasp.org/> - Open Web Application Security Project
2. The Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws - Dafydd Stuttard, Marcus Pinto, Wiley Publishing, Inc.
3. Professional Pen Testing for Web Applications (Programmer to Programmer), Andres Andreu, Wiley Publishing, Inc.
4. Web Security Testing Cookbook, Paco Hope & Ben Walther, O'Reilly 2008
5. <http://news.cnet.com/2008-1082-276319.html> - Gary McGraw's 10 steps to secure software
6. <http://jeremiahgrossman.blogspot.com/2010/09/website-security-statistics-report-2010.html> - statystyki podatności w 2010