appsec

Report

Webový penetrační test www.example.com

Obsah

1 Úvod

1.1 Disclaimer

Penetrační test je obvykle popisován jako přesná a kompletní simulace útoku na danou službu či aplikaci. Ačkoliv penetrační test a reálný útok mají mnoho společného, například znalosti testera a útočníka, používané nástroje a další, tak zde existuje i několik podstatných rozdílů, které je potřeba brát v úvahu. Jedná se především o omezení penetračního testu penězi či časem.

V případě reálného útoku může útočník plánovat útok i několik měsíců předem. Může si tedy dlouhodobě shromažďovat informace potřebné k útoku. V případě penetračního testu si takovýto luxus dovolit nelze, jelikož takový penetrační test by byl finančně neúnosný a zároveň by nepřinesl požadovaný výsledek v rozumném čase pro případná protiopatření.

Z tohoto důvodu je občas potřeba při penetračním testu určitou součinnost od testovaného subjektu, aby v čase vymezeném pro penetrační test bylo možno otestovat systémy co nejvíce.

Jedná se o anonymizovaný report, který oproti skutečnému reportu neobsahuje screenshoty nálezů.

1.2 Nástroje použité při testování

- Burp Suite Pro
- sqlmap
- Nmap
- Firefox - {Cookies Manager, FireBug, FoxyProxy}
- Kali Linux
- APPSEC Toolkit
- Metasploit Framework

1.3 Postup testování

Testy probíhaly na základě zkušeností a schopností testerů, podílejících se na penetračním testu, v souladu s OWASP Testing Guide a standardu PTES.

Jednotlivé fáze penetračního testování byly 1. identifikace cíle, 2. identifikace aktivních služeb, 3. identifikace možných zranitelností, 4. ověření nalezených zranitelností. V průběhu testu byly využívány nástroje, schopnosti a informace tak, aby bylo co nejvěrněji dosaženo simulace útoku takovým způsobem jakým by ho prováděl případný skutečný útočník.

2 Manažerské shrnutí

Předmětem testování bylo provedení komplexního penetračního testu webové aplikace nacházející se na adrese www.example.com. Testování probíhalo v souladu s OWASP testovací příručkou a v souladu se znalostmi a zkušenostmi penetračních testerů, kteří se na testování podíleli.

Během penetračního testu byly testovány ty oblasti bezpečnosti webových aplikací, které jsou označeny v kapitole Testované oblasti podle OWASP.

2.1 Cíle testu

Cílem tohoto penetračního testu bylo detekovat bezpečnostní nedostatky ve webové aplikaci www.example.com. Identifikace systému:

Webový server: Apache
Operační systém: Linux

Předmětem testování bylo provedení komplexního penetračního testu webové aplikace nacházející se na dané adrese. Testování probíhalo v souladu s OWASP testovací příručkou a v souladu se znalostmi a zkušenostmi penetračních testerů, kteří se na testování podíleli.

2.2 Identifikované zranitelnosti

Během penetračního testu byly odhaleny dvě zranitelnosti se závažností hodnocenou jako kritickou. Jedná se o nálezy, které představují bezprostřední hrozbu pro testovaný systém a data, která jsou na něm uložena. Jedná se o nálezy SQL Injection a únik SQL databáze a Účty získané bruteforce útokem.

Byl nalezen i nález ohodnocený vysokou závažností, který je také závažným problémem v případě zneužití motivovaným či schopným útočníkem.

Další nálezy, kterým je potřeba věnovat pozornost jsou nálezy se střední závažností. Jsou to nálezy Cachování HTTPS odpovědí, Chybějící timeout na odhlášení, Dostupné PHPInfo, Možnost hádání hesel, Náchylnost na CSRF a Nepoužití HTTPS. Nálezy ohodnocené nízkou závažností jsou spíše best practices a nepředstavují samy o sobě významné riziko pro bezpečnost webové aplikace.

2.3 Závěr

Během penetračního testu byly odhaleny zranitelnosti, které mohou být útočníkem použity ke kompromitaci testovaného systému.

Z tohoto důvodu doporučujeme věnovat zvýšenou pozornost nejen jejich opravě, ale i důkladnému prozkoumání systému pro zjištění, zda se dané typy nálezů nenacházejí i v dalších částech aplikace.

Po realizaci náprav identifikovaných nedostatků doporučujeme provést retest aplikace, aby bylo ověřeno, že nálezy byly skutečně opraveny. Doporučujeme provádět testování aplikace minimálně jednou ročně, aby bylo minimalizováno riziko výskytu případných nových zranitelností, které mohly být objeveny nebo zaneseny do aplikace od posledního testování.

3 Klasifikace zranitelností

V této kapitole je popsána klasifikace jednotlivých nalezených zranitelností. Každé zranitelnosti je přiřazena míra závažnosti podle dopadů, které by zneužití dané chyby mělo na testovaný systém.

3.1 Podle míry závažnosti

Závažnost jednotlivých nálezů vychází z jejich dopadů na celkové zabezpečení systému. Během hodnocení závažnosti jednotlivých nálezů je zohledněno případné využití společně s ostatními nalezenými zranitelnostmi.

- Kritická ■

  Závažná zranitelnost, která má přímé dopady na celkovou bezpečnost testovaného systému.

- Vysoká ■

  Zranitelnost, která může mít přímé dopady na bezpečnost testovaného systému v případě schopného či motivovaného útočníka.

- Střední ■

  Zranitelnost, která sama o sobě nemá dopad na celkovou bezpečnost testovaného systému, nicméně společně s dalšími zranitelnostmi může představovat určité riziko.

- Nízká ■

  Nález, který má nízký dopad na celkovou bezpečnost testovaného systému, jedná se většinou spíše o best practices.

- Informativní ■

  Nález, který je čistě informativní povahy a nepředstavuje bezpečnostní problém.

3.2 Podle OWASP

Pro testování webových aplikací vycházíme z celosvětově uznávaného projektu OWASP a jeho OWASP Testing Guide verze 4. Jednotlivé nálezy jsou proto kategorizovány a přiřazeny do dané konkrétní sekce OWASP Testing Guide.

4 Výsledky penetračního testu

V této kapitole uvádíme technické informace k provedenému penetračnímu testu. Jsou zde uvedeny jednotlivé oblasti OWASP a informace zda byla daná oblast testována nebo ne.

4.1 Testované oblasti podle OWASP

Následující kapitola je rozdělena do několika tabulek, které obsahují jednotlivé sekce OWASP, které byly během penetračního testu testovány.

Information Gathering

V této fázi dochází ke sběru a získávání dat o testované webové aplikaci a systému. s využitím různých nástrojů získáváme informace nejen ze samotné aplikace, ale i například z webových vyhledávačů. Information Gathering je prvním krokem penetračního testu.

Information Gathering	Testováno
OTG-INFO-001 - Conduct Search Engine Discovery and Reconnaissance for Information Leakage	✔
OTG-INFO-002 - Fingerprint Web Server	✔
OTG-INFO-003 - Review Webserver Metafiles for Information Leakage	✔
OTG-INFO-004 - Enumerate Applications on Webserver	✔
OTG-INFO-005 - Review Webpage Comments and Metadata for Information Leakage	✔
OTG-INFO-006 - Identify application entry points	✔
OTG-INFO-007 - Map execution paths through application	✔
OTG-INFO-008 - Fingerprint Web Application Framework	✔
OTG-INFO-009 - Fingerprint Web Application	✔
OTG-INFO-010 - Map Application Architecture	✔

Configuration and Deploy Management Testing

Při testování konfigurace systému často dochází k zjištění dalších informací o testované aplikaci. Jedná se například o citlivé soubory, administrační rozhraní, konkrétní nastavení infrastruktury či povolené HTTP metody.

Configuration and Deploy Management Testing	Testováno
OTG-CONFIG-001 - Test Network/Infrastructure Configuration	✔
OTG-CONFIG-002 - Test Application Platform Configuration	✔
OTG-CONFIG-003 - Test File Extensions Handling for Sensitive Information	✔
OTG-CONFIG-004 - Review Old, Backup and Unreferenced Files for Sensitive Information	✔
OTG-CONFIG-005 - Enumerate Infrastructure and Application Admin Interfaces	✔
OTG-CONFIG-006 - Test HTTP Methods	✔
OTG-CONFIG-007 - Test HTTP Strict Transport Security	✔
OTG-CONFIG-008 - Test RIA cross domain policy	✔

Identity Management Testing

V rámci testování správy identit je testován samotný mechanismus pro správu identit. Například zakládání uživatelskách účtů, jejich identifikace, správa rolí či rušení uživatelských účtů.

Identity Management Testing	Testováno
OTG-IDENT-001 - Test Role Definitions	✔
OTG-IDENT-002 - Test User Registration Process	✔
OTG-IDENT-003 - Test Account Provisioning Process	✔
OTG-IDENT-004 - Testing for Account Enumeration and Guessable User Account	✔
OTG-IDENT-005 - Testing for Weak or unenforced username policy	✔

Authentication Testing

V rámci aplikace slouží autentizace k identifikaci uživatele, ať již osoby nebo jiného systému. V této části jsou testovány například možnosti obejití autentizace, nedostatky v práci s hesly či nedostatky při používání kontrolních otázek.

Authentication Testing	Testováno
OTG-AUTHN-001 - Testing for Credentials Transported over an Encrypted Channel	✔
OTG-AUTHN-002 - Testing for default credentials	✔
OTG-AUTHN-003 - Testing for Weak lock out mechanism	✔
OTG-AUTHN-004 - Testing for bypassing authentication schema	✔
OTG-AUTHN-005 - Test remember password functionality	✔
OTG-AUTHN-006 - Testing for Browser cache weakness	✔
OTG-AUTHN-007 - Testing for Weak password policy	✔
OTG-AUTHN-008 - Testing for Weak security question/answer	✔
OTG-AUTHN-009 - Testing for weak password change or reset functionalities	✔
OTG-AUTHN-010 - Testing for Weaker authentication in alternative channel	✔

Authorization Testing

Autorizace znamená povolení přístupu k určitým zdrojům pouze pro oprávněné uživatele. V této části testu jsou zjišťovány například následující nedostatky - možnost obejití autorizace, nepoužití autorizace pro některé zdroje či možnost neoprávněného získání autorizace k určitým zdrojům.

Authorization Testing	Testováno
OTG-AUTHZ-001 - Testing Directory traversal/file include	✔
OTG-AUTHZ-002 - Testing for bypassing authorization schema	✔
OTG-AUTHZ-003 - Testing for Privilege Escalation	✔
OTG-AUTHZ-004 - Testing for Insecure Direct Object References	✔

Session Management Testing

Session management se zabývá stavem komunikace mezi webovou aplikací a uživatelem. V V této části textu jsou testovány například útoky vedoucí k ukradení uživatelského sezení, náchylnost aplikace na man-in-the-middle útoky a podobně.

Session Management Testing	Testováno
OTG-SESS-001 - Testing for Bypassing Session Management Schema	✔
OTG-SESS-002 - Testing for Cookies attributes	✔
OTG-SESS-003 - Testing for Session Fixation	✔
OTG-SESS-004 - Testing for Exposed Session Variables	✔
OTG-SESS-005 - Testing for Cross Site Request Forgery	✔
OTG-SESS-006 - Testing for logout functionality	✔
OTG-SESS-007 - Test Session Timeout	✔
OTG-SESS-008 - Testing for Session puzzling	✔

Data Validation Testing

Častým problémem webových aplikací je nedostatečná kontrola uživatelského vstupu. Aplikace očekává zadání pouze validních vstupů, což otevírá prostor k manipulaci se vstupy a provedení některého z injection útoků. V této části testů jsou testovány například SQL injection, cross site scripting či možnost spuštění příkazu operačního systému.

Data Validation Testing	Testováno
OTG-INPVAL-001 - Testing for Reflected Cross Site Scripting	✔
OTG-INPVAL-002 - Testing for Stored Cross Site Scripting	✔
OTG-INPVAL-003 - Testing for HTTP Verb Tampering	✔
OTG-INPVAL-004 - Testing for HTTP Parameter pollution	✔
OTG-INPVAL-005 - Testing for SQL Injection	✔
OTG-INPVAL-006 - Testing for LDAP Injection	✔
OTG-INPVAL-007 - Testing for ORM Injection	✔
OTG-INPVAL-008 - Testing for XML Injection	✔
OTG-INPVAL-009 - Testing for SSI Injection	✔
OTG-INPVAL-010 - Testing for XPath Injection	✔
OTG-INPVAL-011 - IMAP/SMTP Injection	✔
OTG-INPVAL-012 - Testing for Code Injection	✔
OTG-INPVAL-013 - Testing for Command Injection	✔
OTG-INPVAL-014 - Testing for Buffer overflow	✔
OTG-INPVAL-015 - Testing for incubated vulnerabilities	✔
OTG-INPVAL-016 - Testing for HTTP Splitting/Smuggling	✔

Error Handling Testing

K úniku citlivých informací často dochází kvůli nesprávné práci s chybovými stavy. Tato část testů slouží k otestování právě těchto problémů.

Error Handling Testing	Testováno
OTG-ERR-001 - Analysis of Error Codes	✔
OTG-ERR-002 - Analysis of Stack Traces	✔

Cryptography Testing

Vzhledem k určité složitosti kryptografických schémat často dochází k jejich nesprávnému použití v rámci webových aplikací. V této části testů jsou ověřovány nedostatky související s SSL/TLS či obecně nedostatky v kryptografii použité pro zabezpečení webové aplikaci.

Cryptography Testing	Testováno
OTG-CRYPST-001 - Testing for Weak SSL/TSL Ciphers, Insufficient Transport Layer Protection	✔
OTG-CRYPST-002 - Testing for Padding Oracle	✔
OTG-CRYPST-003 - Testing for Sensitive information sent via unencrypted channels	✔

Business Logic Testing

Vedle technických nedostatků aplikace často obsahují i nedostatky, které nejsou čistě technické povahy, ale vznikly například v důsledku špatného pochopení fungování aplikace či nějakého použitého schématu. V této části testů je potřeba, aby se penetrační tester seznámil s aplikací, pochopil ji a mohl poté odhalit nedostatky v principu práce s ní.

Business Logic Testing	Testováno
OTG-BUSLOGIC-001 - Test Business Logic Data Validation	✔
OTG-BUSLOGIC-002 - Test Ability to Forge Requests	✔
OTG-BUSLOGIC-003 - Test Integrity Checks	✔
OTG-BUSLOGIC-004 - Test for Process Timing	✔
OTG-BUSLOGIC-005 - Test Number of Times a Function Can be Used Limits	✔
OTG-BUSLOGIC-006 - Testing for the Circumvention of Work Flows	✔
OTG-BUSLOGIC-007 - Test Defenses Against Application Mis-use	✔
OTG-BUSLOGIC-008 - Test Upload of Unexpected File Types	✔
OTG-BUSLOGIC-009 - Test Upload of Malicious Files	✔

Client Side Testing

V této kategorii jsou testovány mechanismy, které aplikace používá pro ochranu uživatelů před útoky směřujícími přímo na uživatele a jeho webový prohlížeč. Testovány jsou různé injekční útoky pomocí klientských skriptovacích jazyků či modifikace různých parametrů spravovaných prohlížečem.

Client Side Testing	Testováno
OTG-CLIENT-001 - Testing for DOM based Cross Site Scripting	✔
OTG-CLIENT-002 - Testing for JavaScript Execution	✔
OTG-CLIENT-003 - Testing for HTML Injection	✔
OTG-CLIENT-004 - Testing for Client Side URL Redirect	✔
OTG-CLIENT-005 - Testing for CSS Injection	✔
OTG-CLIENT-006 - Testing for Client Side Resource Manipulation	✔
OTG-CLIENT-007 - Test Cross Origin Resource Sharing	✔
OTG-CLIENT-008 - Testing for Cross Site Flashing	✔
OTG-CLIENT-009 - Testing for Clickjacking	✔
OTG-CLIENT-010 - Testing WebSockets	✔
OTG-CLIENT-011 - Test Web Messaging	✔
OTG-CLIENT-012 - Test Local Storage	✔

4.2 Identifikované zranitelnosti

V této kapitole jsou uvedeny jednotlivé zranitelnosti a nálezy seřazené podle závažnosti od nejzávažnějších po nejméně závažné. Jedná se o nálezy, které byly identifikovány během penetračního testování. Jsou zde uvedeny společně s případnými dopady a doporučením na eliminaci rizika. Vzhledem k povaze penetračního testu nemusí být jejich seznam kompletní. V případě nalezených zranitelností tedy doporučujeme věnovat danému problému pozornost a opravit daný typ zranitelnosti v celém systému.
Z důvodu provádění neautentizovaného i autentizovaného penetračního testu webové aplikace obsahuje tato kapitola dvě podkapitoly. Jedná se o nálezu, které byly identifikovány ve fázi neautentizovaného penetračního testu a ve fázi autentizovaného penetračního testu. Rozdělení jednotlivých zranitelností je provedeno podle toho, ve které fázi byly tyto nálezy identifikovány, nikoliv podle toho, zda mohou nebo nemohou být zneužity autentizovaným / neautentizovaným útočníkem. Zjednodušeně řečeno je možné, že některé nálezy, které byly identifikovány ve fázi autentizovaného testu mohou být za určitých okolností zneužity neautentizovaným útočníkem a ve fázi neautentizovaného penetračního testu nebyly nalezeny z důvodu časových či jiných omezení.

4.2.1 SQL Injection a únik SQL databáze

Závažnost:

Nález

Během penetračního testu bylo zjištěno, že aplikace je náchylná na zranitelnost známou jako SQL injection. Jedná se o velice známou a rozšířenou zranitelnost, která je způsobena chybějícím nebo špatným filtrováním nedůvěryhodných vstupů v aplikaci (většinou se jedná o uživatelské vstupy jako jsou formuláře, identifikace prohlížeče, cookies a další). Dané nefiltrované vstupy jsou pomocí SQL požadavku předány přímo do databáze, což může mít nepředpokládané následky.

Zranitelné jsou následující stránky a parametry:

- /main/login - parametry: password, username

Požadavek odeslaný na server.

```
POST /main/login HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (X11; Fedora; Linux x86\_64; rv:44.0) \
  Gecko/20100101 Firefox/44.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Referer: https://example.com
Content-Length: 40
Cookie: PHPSESSID=6tbd9hlc9782qkc5d6f704ha46
Connection: close

username=anon%40attacker.com\&password='
```

Odpověď ze serveru, která obsahuje informace o chybě přímo z SQL databáze.

```
HTTP/1.1 200 OK
Server: nginx
Date: Mon, 24 Oct 2016 09:53:09 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 591
Connection: close
X-Powered-By: PHP/5.4.42
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
```

```
<pre>SELECT u.recid, CASE WHEN u.admin\_server IS NOT NULL AND LENGTH
(u.admin\_server) >= 4 THEN u.admin\_server ELSE p.admin\_server END
AS admin\_server, CASE WHEN u.user\_server IS NOT NULL AND LENGTH
(u.user\_server) >= 4 THEN u.user\_server ELSE p.user\_server END AS
server\_url
  FROM user u
  JOIN project p ON (u.id\_project = p.recid)
  WHERE u.password = f\_hash\_password(u.recid, ''') AND u.email
  = 'anon@attacker.com'</pre>
You have an error in your SQL syntax; check the manual that corresponds
to your MySQL server version for the right syntax to use near
'@attacker.com'' at line 4
```

Postup exploitace / ověření zranitelnosti

1. We webovém prohlížeči otevřete webovou aplikaci www.example.com/main/login.
2. Do formuláře pro přihlášení vložte libovolné uživatelské jméno a místo hesla vložte znak '.
3. Po odeslání formuláře bude zobrazena chybová hláška MySQL indikující SQL injection.
4. Získaná data jsou přiložena, pro jejich rychlé a jednoduché získání doporučujeme použití nástroje sqlmap.

Riziko

Provedením útoku za pomoci SQL injection může dojít k úniku citlivých dat, může dojít k jejich poškození či neautorizované změně. Pokročilým využitím SQL injection je možné provést celkovou kompromitaci databáze i hostitelského systému. Uvedená zranitelnost je přímým ohrožením citlivých dat i hostitelského systému.

Doporučení

Doporučujeme zkontrolovat všechny funkce, které pracují s databází a případně upravit jejich funkcionalitu tak, aby obsahovaly filtrování veškerých nedůvěryhodných vstupů, ať již od uživatele (fomuláře, identifikace prohlížeče, cookies, ...) nebo jiného potenciálně nedůvěryhodného subsystému. Vstupy doporučujeme filtrovat principem whitelist, kdy jsou přijaty pouze řetězce obsahující pouze povolené znaky. Ty by měly být například a-zA-Z0-9. Dále doporučujeme vždy jasně specifikovat odchozí kódování dat.

4.2.2 Účty získané bruteforce útokem

Závažnost:

Nález

Během penetračního testu bylo zjištěno, že některé z dříve enumerovaných uživatelských účtů (nález zabývající se enumerací uživatelů následuje) používají velice slabá hesla a jsou tedy náchylné na bruteforce hádání hesla.

Jedná se konkrétně o následující uživatelské účty ke kterým byla zjištěna následující hesla:

- ucet1 / heslo1
- ucet2 / heslo2

V případě uživatelského účtu siskova se jedná o vysoce privilegovaný účet, který má přístup ke všem informacím dostupným v aplikaci, včetně přihlašovacích jmen a hesel ostatních uživatelů.

Ke zjišťování hesel byl použit slovník 1000 nejpoužívanějších hesel rozšířený o přihlašovací jména zjištěna během enumerace. Celý slovník je přiložen jako slovnik_hesla_1000_nejbeznejsich.txt. V žádném případě se nejedná o kompletní seznam všech uživatelů se slabými hesly.

Postup exploitace / ověření zranitelnosti

1. We webovém prohlížeči otevřete webovou aplikaci www.example.com/main/login.
2. Do formuláře pro přihlášení vložte získané přihlašovací údaje.
3. V případě validních přihlašovacích údajů dojde k přihlášení.
4. Použité slovníky jsou přiloženy, k vlastnímu testování doporučujeme použití nástroje hydra.

Riziko

V případě, že útočník získá přístup k jakýmkoliv uživatelským účtům ke kterým přístup mít nemá, jedná se o závažný problém pro bezpečnost systému. Problém je tím větší, čím větší jsou oprávnění daného uživatelského účtu v rámci aplikace. V tomto případě se jedná o kritický problém, jelikož uživatelský účet má jedny z nejvyšších práv. Vzhledem ke slabé politice hesel předpokládáme, že účtů se slabým heslem se v systému nachází více.

Doporučení

Po zavedení vhodné politiky hesel důrazně doporučujeme všem účtům se slabým (tedy politice hesel nevyhovujícím) heslem toto heslo nechat expirovat a přimět uživatele si heslo změnit na silné.

4.2.3 Možnost manipulace s cenou

Závažnost:

Nález

Bylo zjištěno, že v e-shopu na umístění /e-shop při aktualizaci licence a na umístění /e-shop/aktualizace při koupi nové licence je možné manipulovat s cenou. Lze toho docílit nastavením počtu objednávaných kusů licencí na zápornou hodnotu. Tím sice dojde ke snížení celkového počtu objednaných / aktualizovaných licencí, nicméně vhodnou manipulací je možné v konečném důsledku dosáhnout snížení ceny až k ceně blízké nule.

Celý proces probíhá automatizovaně a výsledná cena se poté promítne i do faktury.

Postup exploitace / ověření zranitelnosti

1. Nastavte webový prohlížeč tak, aby používal HTTP proxy (doporučujeme Burp Suite) pro zachytávání HTTP požadavků a odpovědí.
2. We webovém prohlížeči otevřete webovou aplikaci www.example.com/e-shop/aktualizace.
3. Přidejte do košíku požadovanou položku.
4. Nastavte HTTP proxy na odchytávání požadavků a přidejte do košíku jinou položku.
5. V HTTP proxy v zachyceném požadavku změňte parametr "pocetlicenci"na -1 a požadavek odešlete.
6. Ve webovém prohlížeči můžete vidět, že v košíku jsou nyní dvě položky, první s normální cenou, druhá se svojí cenou, ale opačným znaménkem.
7. Celková cena je tedy snížena o cenu druhé položky.

Riziko

Rizikem tohoto počínání je možnost, že uživatelé můžou být schopni přidáním záporného počtu licencí (na produkt, který nepotřebují) snížit cenu, kterou v konečném důsledku zaplatí.

Rizikem tohoto počínání je, že uživatelé budou modifikovat cenu jednotlivých produktů čímž mohou způsobit finanční škodu provozovateli e-shopu. Riziko zneužití vzrůstá v případě, kdyby zpracování požadavků probíhalo automatizovaně. V případě masového zneužití může mít tato zranitelnost dalekosáhlé finanční následky.

Doporučení

Doporučujeme důkladné filtrování nedůvěryhodných uživatelských vstupů. Je důležité kontrolovat počet licencí, zda se jedná o nezáporné celé číslo, aby nemohla být výsledná cena tímto způsobem modifikována.

4.2.4 Na serveru se nacházejí škodlivé aplikace (malware)

Závažnost:

Nález

Během penetračního testu bylo zjištěno, že na adrese /downloads se nachází aplikace, které jsou pro uživatele potenciálně nebezpečné. Jedna z aplikací například nabádá uživatele k zadání svého hesla k seznam.cz účtu (a pravděpodobně jej zaznamenává), další aplikace se snaží o instalaci falešné aktualizace k aplikaci Adobe Reader. Aplikace had zase simuluje okno Mozilla Firefox pro zmatení uživatele s cílem přimět ho k zadání citlivých informací.

Postup exploitace / ověření zranitelnosti

1. Použijte testovací počítač s nainstalovaným a aktualizovaným antivirovým nástrojem.
2. Ve webovém prohlížeči otevřete stránku www.example.com/downloads a stáhněte některý z malware.
3. Antivirový nástroj malware detekuje a zabrání jeho stažení.

Riziko

Největším rizikem pro uživatele je potenciální zneužití jejich hesel a pracovních stanic (pomocí potenciálně nainstalovaného malware). Uživatel totiž snáze podlehne dojmu, že se jedná o legitimní aplikaci v případě, že je hostována na důvěryhodném serveru. Tento nález také může sloužit jako upozornění pro administrátora, že webový server mohl být zneužit k šíření malware.

Doporučení

Doporučujeme dané stránky z webového serveru odstranit, aby nemohly být dále využívány. Dále doporučujeme zjistit jak se dané informace na server dostaly a provést taková opatření, aby tomu bylo do budoucna zamezeno.

4.2.5 Neošetřený upload

Závažnost:

Nález

Během penetračního testu bylo zjištěno, že z umístění http://www.example.com/nastaveni.php je možné nahrávat soubory do aplikace. Formulář pro nahrání potrétu však není dostatečně ošetřen a nepovoluje pouze nahrání souborů s příponou .php. Díky tomu je útočník schopen nahrát například soubor s příponou .php5, který filtrem bez problémů projde a je poté uložen a zpřístupněn pro spuštění libovolnému uživateli.

Postup exploitace / ověření zranitelnosti

1. Vytvořte soubor upload.php5 s obsahem <?php phpinfo(); ?>.
2. Otevřete stránku www.example.com/nastaveni.php ve webovém prohlížeči.
3. Ve formuláři pro upload souborů nahrajte soubor upload.php5, dojde k jeho úspěšnému nahrání.
4. Otevřete stránku www.example.com/uploads/upload.php5.
5. Vidíte phpinfo stránku, důkaz, že došlo k nahrání a provedení PHP kódu.

Riziko

Útočník může tímto způsobem nahrát na server libovolný php kód, který se poté spustí v kontextu dané webové aplikace. Tímto způsobem může libovolně manipulovat nejen s aplikací, ale i se serverem na kterém běží (může manipulovat se soubory ke kterým má přístup uživatel pod kterým aplikace běží).

Doporučení

Doporučujeme důkladnější kontrolu nahrávaných souborů:

- - nahrávané soubory doporučujeme filtrovat oproti whitelistu, který bude obsahovat pouze specificky povolené typy souborů
- - nenechávat nahrávanému souboru původní příponu

Dále doporučujeme kontrolovat přímo typ daného souboru bez ohledu na použitou příponu.

4.2.6 Reflected Cross Site Scripting (XSS)

Závažnost:

Nález

Během penetračního testu bylo zjištěno, že při uploadu Certificate Signing Request (CSR) na adrese https://www.example.com/evs.php nedochází při výpisu adresy v kroku 2 k filtrování dat získaných z nedůvěryhodného vstupu (obsahu CSR). Zadaná data jsou poté vypsána zpět do kódu webové stránky, což může vést k útoku zvanému Cross Site Scripting. Konkrétně se jedná o položku Common Name v CSR.

Postup exploitace / ověření zranitelnosti

1. Vygenerujte Certificate Signing Request (CSR) s položkou Common Name ve tvaru «script>alert('XSS');</script>".
2. Nastavte webový prohlížeč tak, aby používal HTTP proxy (doporučujeme Burp Suite) pro zachytávání HTTP požadavků a odpovědí.
3. We webovém prohlížeči otevřete webovou aplikaci https://www.example.com/evs.php.
4. Uploadujte CSR soubor.
5. Na další stránce dojde k vykonání JavaScriptového příkazu a zobrazení "alert"okna s textem XSS.

Riziko

Možnost ovlivnění informací na webu pomocí útoku XSS může vést například k poškození pověsti provozovatele webové aplikace. Může také dojít k jinému typu škody v závislosti na charakteru podvržené informace. Dalším rizikem je odcizení session ID, které udržuje uživatelskou relaci v dané webové aplikaci. Pomocí útoku XSS může dojít k jeho odcizení a s ním i k odcizení uživatelské relace. Útočník také může spouštět skripty v kontextu napadené aplikace či například zneužít klientský systém jako XSS Proxy.

Doporučení

Doporučujeme zkontrolovat všechny funkce, které pracují s jakýmikoliv uživateli ovlivnitelnými vstupy a případně upravit jejich funkcionalitu tak, aby obsahovaly filtrování veškerých nedůvěryhodných vstupů, ať již od uživatele (fomuláře, identifikace prohlížeče, cookies, ...) nebo jiného potenciálně nedůvěryhodného subsystému. Vstupy doporučujeme filtrovat principem whitelist, kdy jsou přijaty pouze řetězce obsahující pouze povolené znaky. Ty by měly být například a-zA-Z0-9. Dále doporučujeme vždy jasně specifikovat odchozí kódování dat.

4.2.7 Cachování HTTPS odpovědí

Závažnost:

Nález

Během penetračního testu bylo zjištěno, že webovému prohlížeči není zakázáno cachování odpovědí. Některé prohlížeče cachují i stránky přijaté přes HTTPS. V takovém případě veškeré informace dostupné na stránkách webové aplikace mohou být lokálně uloženy v počítači (či na proxy serveru, je-li použit).

Postup exploitace / ověření zranitelnosti

1. Nastavte webový prohlížeč tak, aby používal HTTP proxy (doporučujeme Burp Suite) pro zachytávání HTTP požadavků a odpovědí.
2. Ve webovém prohlížeči otevřete webovou aplikaci.
3. V HTTP proxy v hlavičkách HTTP odpovědi je vidět, že chybí hlavičky Cache-control a Pragma, které se starají o cachování odpovědí.

Riziko

V případě, že útočník získá přístup k obsahu cache prohlížeče, může si přečíst odpovědi webového serveru tak jak byly přijaty uživatelem včetně všech potenciálně citlivých informací. V tomto konkrétním případě je tento nález hodnocen s vyšší závažností z důvodu dostupnosti hesel v čitelné podobě ve webové aplikaci.

Doporučení

Doporučujeme, aby aplikace instruovala webový prohlížeč, aby neukládal odpovědi přijaté přes HTTPS a to použitím následujících HTTP hlaviček v každé odpovědi webového serveru:

```
Cache-control: no-store
Pragma: no-cache
```

4.2.8 Chybějící timeout na odhlášení

Závažnost:

Nález

Během penetračního testu bylo zjištěno, že ani po 20 minutách neaktivity není přihlášený uživatel automaticky odhlášen. Jedná se o bezpečnostní opatření používané v aplikacích, které pracují s citlivými daty. Jeho cílem je omezit útoky vyplývající z toho, že uživatel opustí webovou aplikaci bez odhlášení.

Postup exploitace / ověření zranitelnosti

1. Přihlašte se do webové aplikace se svými přihlašovacími údaji.
2. Po 20 minut neprovádějte žádnou činnost.
3. Ani po 20 minutách nedojde k odhlášení neaktivního uživatele.

Riziko

Rizikem chybějícího automatického odhlášení po určité době je to, že uživatel může na chvíli opustit počítač i s aplikací ve které je přihlášen. V té chvíli může útočník, přistoupit k počítači a aplikaci s příhlášeným uživatelem po celou dobu, kdy je uživatel pryč. V případě zavedení timeoutu na odhlášení, toto okno ve kterém je možné k aplikaci přistoupit je výrazně zredukováno.

Doporučení

Doporučujeme zavedení automatického odhlášení po určité době, třeba po 5 - 10 minutách tak, aby bylo zamezeno zneužití citlivých dat v situacích, kdy uživatel na chvíli opustí počítač, ale zůstane v aplikaci přihlášen.

4.2.9 Dostupné PHPInfo

Závažnost:

Nález

Během penetračního testu bylo zjištěno, že na adrese http://www.example.com/info.php se nachází tzv. phpinfo stránka s informacemi o použitém php, webovém serveru a další informace.

Z dané stránky byly zjištěny mimo jiné tyto potenciálně citlivé údaje:

```
Verze použitého PHP - PHP Version 5.5.9
Verze použitého webového prohlížeče - Apache/2.4.7
Kořenový adresář pro webovou aplikaci - /home/example/data/www/
Verze Linux jádra - 3.13.0
```

Postup exploitace / ověření zranitelnosti

1. Ve webovém prohlížeči otevřete stránku www.example.com/info.php.
2. Otevřená stránka obsahuje citlivé informace.

Riziko

Rizika tohoto nálezu souvisí především s možností přesnějšího zaměření útoku ze strany útočníka. Díky phpinfo je útočník schopen s velkou přesností zjistit přesné verze použitého software. Stejně tak je útočník schopen zjistit informace o nastavených direktivách PHP, například allow_url_fopen, a další.

Doporučení

Doporučujeme danou stránku úplně odstranit, pokud není důvod aby byla na produkčním serveru přístupná. V případě, že taková potřeba existuje a je oprávněná, doporučujeme nasazení nějakého autentizačního mechanismu k přístupu k ní.

4.2.10 Možnost hádání hesel

Závažnost:

Nález

Bylo zjištěno, že aplikace nepoužívá pro přihlašování mechanismus CAPTCHA, a to ani po několika neúspěšných pokusech o přihlášení. Stejně tak nebylo detekováno žádné zamčení účtu či omezení počtu přihlašování a to ani po několika neúspěšných pokusech.

Postup exploitace / ověření zranitelnosti

1. Ve webovém prohlížeči otevřete přihlašovací stránku aplikace.
2. Vyzkoušejte přihlásit se zhruba 3-5 krát na existující účet s nesprávným heslem.
3. Poté se zkuste přihlásit s existujícím účtem a správným heslem.
4. V průběhu testu nedojde k žádnému zobrazení CAPTCHA či omezení možnosti hádaní hesla

Riziko

Takovéto nastavení umožňuje útočníkovi fakticky neomezené hádání hesel do uživatelských účtů. Toto může vést až ke kompromitaci uživatelského účtu. Riziko nálezu je zvýšeno faktem, že v aplikaci chybí bezpečná politika hesel a tedy je pravděpodobné, že jednoduchých uhodnutelných hesel se v aplikaci nachází nemalé množství.

Doporučení

Doporučujeme zvážit možnost implementace mechanismu CAPTCHA a blokování, a to například takovým způsobem, že po třech neúspěšných pokusech o přihlášení bude uživateli zobrazena CAPTCHA a po dalších například třech neúspěšných pokusech dojde k dočasnému zablokování dané IP adresy.

4.2.11 Náchylnost na CSRF

Závažnost:

Nález

Během penetračního testu bylo zjištěno, že aplikace je náchylná na CSRF (Cross Site Request Forgery). Jedná se o typ útoku, kdy je útočníkem uživateli podstrčen legitimně se tvářící odkaz, na kterém je kód umožňující provedení akce uživatelovým jménem.

Následující požadavky nejsou proti tomuto útoku chráněny:

- /support/send_message
- /banking/money_transfer
- /banking/set_email

Následujícím kódem, který uživatel otevře ve svém prohlížeči může útočník odeslat ticket na podporu jménem uživatele:

```
<form id="myform" name="myform" action="https://example.com/set_email" method="post">
<input type='text' name='email' value="hacker@somewhere.com" />
<input type="submit" name="mysubmit" value="Click!" />
</form>
<script type="text/javascript">
document.getElementById("myform").submit();
</script>
```

Příklad daného požadavku:

```
POST /set_email HTTP/1.1
Host: example.com
Referer: https://attacker.com/
Cookie: PHPSESSID=l38dcjuhpjearcah5fm3f9gts2

email=hacker@somewhere.com
```

Postup exploitace / ověření zranitelnosti

1. Vytvořte soubor csrf.html s výše uvedeným obsahem.
2. Přihlaste se ve webovém prohlížeči do webové aplikace.
3. Poté ve stejném webovém prohlážeči otevřete soubor csrf.html.
4. Dojde k provedení daného požadavku, což je možné ověřit ve webové aplikaci, v tomto případě například v odeslaných požadavcích.

Riziko

Rizikem tohoto útoku je, že útočník je schopen s minimálním přispěním uživatele provádět operace v aplikaci jménem legitimního uživatele. V tomto případě se jedná hlavně o kontaktování podporu a vedení útoku například sociálním inženýrstvím.

Doporučení

K eliminaci náchylnosti aplikace k útoku CSRF doporučujeme použití tzv. CSRF tokenu. Jedná se o mechanismus, kdy je pro každého uživatele a každý formulář vygenerován unikátní řetězec, který je poté vložen do formuláře a zároveň zapamatován aplikací. Při odeslání formuláře je poté CSRF token zkontrolován a v případě nerovnosti existuje důvodné podezření, že se jedná o útok cross site request forgery a takový požadavek by měl být zamítnut.

4.2.12 Nepoužití HTTPS

Závažnost:

Nález

Bylo zjištěno, že server na kterém běží webová aplikace nepoužívá šifrovaný přenos dat prostřednictvím HTTPS. Informace, které jsou přenášeny mezi klientem a serverem můžou být zachyceny a přečteny útočníkem. Závažnost nálezu je zvýšena faktem, že e-mailový účet je v dnešní době středobodem internetové identity uživatele a jeho kompromitací by došlo k výrazným problémům uživatele z hlediska identity a soukromí.

Postup exploitace / ověření zranitelnosti

1. Otevřete webovou stránku http://www.example.com, pozor na http na začátku.
2. Po otevření webové stránky nedojde k přesměrování na zabezpečenou variantu protokolu HTTPS.

Riziko

V případě, že se útočníkovi podaří získat přístup mezi klienta a server, může být schopen odposlouchávat veškerou komunikaci mezi nimi. Útočník může být schopen odposlechnout například přihlašovací údaje, odesílané zprávy, zprávy, které uživatel zobrazuje a další informace. V takovém případě tedy může dojít ke kompromitaci uživatelského účtu či úniku potenciálně citlivých informací.

Doporučení

Doporučujeme nasazení HTTPS a vynucování jeho použití. V dnešní době není problém nasadit například certifikát od Let's Encrypt, který je zadarmo a jeho nasazení společně se zapnutím HTTPS je otázkou několika minut.

4.2.13 Chybějící historie hesel

Závažnost:

Nález

Bylo zjištěno, že aplikace si napamatuje předchozí hesla k uživatelskému účtu. Jedná se o mechanismus, který v případě změny uživatelského hesla zabraňuje jeho opětovnému použití po nějakou dobu a to s cílem zabránění opětovného nastavení některého z dřívějších hesel v případě například vynucené změny hesla z důvodu jeho kompromitace.

Postup exploitace / ověření zranitelnosti

1. Ve webové aplikaci změňte heslo na nové (označme si ho jako heslo1).
2. Poté znovu změňte heslo, tentokrát na heslo, které si označíme jako heslo2.
3. Teď opět změňte heslo na heslo1.
4. Aplikace heslo1 přijme, ačkoliv již bylo použito v nedávné minulosti.

Riziko

V případě, že uživatel by si znovu nastavil některé z dřívějších hesel, které bylo odhaleno, může dojít k ovládnutí účtu útočníkem.

Doporučení

Doporučujeme ukládání dřívějších hesel, alespoň tří, a nedovolit uživateli opětovné nastavení takového hesla. Zároveň doporučujeme nastavení minimálního stáří hesla na alespoň jeden den. V opačném případě by si uživatel mohl heslo okamžitě třikrát změnit a poté si opět nastavit heslo původní.

4.2.14 Náchylnost na clickjacking

Závažnost:

Nález

Během testů bylo zjištěno, že aplikaci je možno načíst v rámci a provést tzv. clickjacking. Jedná se o útok, kdy útočník načte webovou aplikaci v rámu (iframe) a překryje ji svým obsahem takovým způsobem, že uživatel při interakci s překrývajícími prvky nevědomky interaguje se samotnou webovou aplikací.

Postup exploitace / ověření zranitelnosti

1. Vytvořte soubor clickjacking.html s obsahem <iframe src=’http://www.example.com’></iframe>.
2. Otevřete soubor clickjacking.html ve webovém prohlížeči.
3. Zobrazí se původní webová aplikace v rámci v nadřazeném kontextu.

Riziko

Riziko spočívá v tom, že útočník může načíst danou aplikaci v rámu, a překryje ji svým obsahem. Když pak uživatel interaguje s útočníkovým obsahem (nějaký atraktivní obsah, například jednoduchá webová hra), nevědomky také interaguje s danou webovou aplikací, která je načtená v rámci. Tímto může útočník uživatele přimět k nevědomé akci s danou aplikací (vytvoření uživatelského účtu, smazání uživatele, atd.).

Doporučení

Doporučujeme použití HTTP hlavičky X-Frame-Options a nastavení její hodnoty na Deny. Další možností je implementovat obranu proti clickjacking v JavaScriptu.

4.2.15 Přihlašovací formulář s Autocomplete

Závažnost:

Nález

Během penetračního testu bylo zjištěno, že na přihlašovacím formuláři není zakázán autocomplete. Moderní webové prohlížeče umožňují uživatelům ukládání přihlašovacích údajů. Tyto uložené přihlašovací údaje mohou být získány útočníkem buď v případě, že útočník získá přístup k danému webovému prohlížeči nebo například pomocí cross site scripting útoku vůči aplikaci, kdy pomocí xss payloadu útočník přečte heslo z políček pro přihlašovací údaje a odešle na server útočníka.

Postup exploitace / ověření zranitelnosti

1. Otevřete přihlašovací stránku ve webovém prohlížeči a zobrazte si zdrojový kód dané stránky.
2. Vyhledejte kód přihlašovacího formuláře.
3. Chybí direktiva autocomplete="off".

Riziko

Je-li v prohlížeči uživatele povoleno automatické ukládání hesel a zároveň není tato funkcionalita zakázána v kódu webové stránky, je zde riziko, že citlivá data budou uložena v počítači a můžou být zpřístupněna útočníkovi, který má k počítači přístup.

Doporučení

Doporučujeme přidání autocomplete="off"k danému poli pro heslo pro zakázání výše popsané funkcionality. Ačkoliv některé webové prohlížeče tuto direktivu ignorují a ukládání hesla i přesto podporují, je dobrou praktikou tuto funkcionalitu zakázat.

4.2.16 Aplikace nedělá rozdíl mezi POST a GET metodami

Závažnost:

Nález

Během penetračního testu bylo zjištěno, že aplikace nedělá rozdíl mezi metodami POST a GET. Požadavky vytvořené jako POST mohou být odeslány jako GET a jsou aplikací přijaty tak jakoby byly validní.

Postup exploitace / ověření zranitelnosti

1. Nastavte webový prohlížeč tak, aby používal HTTP proxy (doporučujeme Burp Suite) pro zachytávání HTTP požadavků a odpovědí.
2. We webovém prohlížeči otevřete webovou aplikaci na přihlašovacím formuláři.
3. Nastavte HTTP proxy na zachytávání požadavků a přihlašte se ve webové aplikaci.
4. V odchyceném požadavku změňte POST požadavek na GET (v Burp Suite pravý klik a "Change request method") a požadavek odešlete.
5. V aplikaci dojde k přihlašení ačkoliv došlo ke změně POST/GET metody.

Riziko

Tento nález je informační povahy a nepředstavuje riziko pro aplikaci.

Doporučení

Tento nález je informační povahy a nepředstavuje riziko pro aplikaci.

4.2.17 Dostupná informace o autorovi aplikace

Závažnost:

Nález

Testovaná aplikace v patičce a ve zdrojovém kódu odhaluje svého autora. Toto samo o sobě není bezpečnostním nedostatkem, nicméně útočník může na základě této informace usuzovat na autorův přístup k bezpečnosti (je-li znám). Útočník také může na základě toho usuzovat na použité technologie při tvorbě aplikace.

Postup exploitace / ověření zranitelnosti

1. Otevřete webovou aplikaci ve webovém prohlížeči.
2. V patičce je vidět jméno a identifikace autora aplikace.

Riziko

Tento nález je informační povahy a nepředstavuje riziko pro aplikaci.

Doporučení

Tento nález je informační povahy a nepředstavuje riziko pro aplikaci.

5 Přílohy

Veškeré soubory jsou součástí dokumentů předaných společně se zprávou.