VYSOKÁ ŠKOLA POLYTECHNICKÁ JIHLAVA

Aplikovaná informatika

Penetrační testování bezpečnosti webových stránek

Bakalářská práce

Autor práce: David Felenda

Vedoucí práce: Mgr. Antonín Přibyl

Jihlava 2024

Abstrakt

Bakalářská práce je zaměřena na zkoumání kyberbezpečnosti webových stránek pomocí penetračních testů. Práce provádí důkladnou analýzu aktuálních kybernetických hrozeb a následně implementuje penetrační testy s využitím rozmanitých metod a technik, včetně SQL injection a cross-site scripting. Hlavním cílem těchto testů je identifikovat potenciální bezpečnostní zranitelnosti webových stránek a kategorizovat je podle jejich závažnosti. Vedle toho práce podrobně navrhuje konkrétní bezpečnostní opatření a řešení pro eliminaci zjištěných zranitelností, přispívající k celkovému zlepšení kyberbezpečnosti webových prostředí.

Klíčová slova

Kyberbezpečnost, Penetrační testování, OWASP, Analýza hrozeb, Metody a techniky

Abstract

This bachelor's thesis is dedicated to exploring the cybersecurity of websites through penetration testing. The study involves a comprehensive analysis of current cyber threats, followed by the implementation of penetration tests utilizing diverse methods and techniques, including SQL injection and cross-site scripting. The primary objective of these tests is to identify potential security vulnerabilities within web pages and categorize them based on their severity. In addition, the thesis provides detailed proposals for specific security measures and solutions to address the identified vulnerabilities, contributing to an overall enhancement of the cybersecurity posture of web environments.

Keywords

Cybersecurity, Penetration Testing, OWASP, Threat Analysis, Methods and Techniques

Prohlašuji, že předložená bakalářská práce je původní a zpracoval/a jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem v práci neporušil/a autorská práva (ve smyslu zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů, v platném znění, dále též „AZ“).

Byl/a jsem seznámen/a s tím, že na mou bakalářskou práci se plně vztahuje AZ, zejména § 60 (školní dílo).

Podle § 47b zákona o vysokých školách souhlasím se zveřejněním své práce podle směrnice prorektora pro studium č. 2/2020, a to bez ohledu na výsledek obhajoby.

Beru na vědomí, že VŠPJ má právo na uzavření licenční smlouvy o užití mé bakalářské práce a prohlašuji, že souhlasím s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom/a toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem VŠPJ, která má právo ode mě požadovat přiměřený příspěvek na úhradu nákladů, vynaložených vysokou školou na vytvoření díla (až do jejich skutečné výše), z výdělku dosaženého v souvislosti s užitím díla či poskytnutím licence.

V Jihlavě dne Klikněte nebo klepněte sem a zadejte datum.

…….

Podpis studenta/ky

Poděkování

Rád bych na tomto místě poděkoval svému vedoucímu práce Mgr. Antonínu Přibylovi za možnost provedení práce, cených návrhů a poskytnutí webových stránek pro otestování.

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

V dnešní digitální éře je kybernetická bezpečnost klíčovým aspektem, neboť pozorujeme neustále se zvyšující počet kybernetických hrozeb a útoků. Útoky mohou mít značný dopad na organizace i jednotlivce, a to vyžaduje aktivní zabezpečení a identifikaci bezpečnostních zranitelností.

Bakalářská práce se zaměřuje na zabezpečení webových stránek prostřednictvím penetračních testů s cílem identifikovat potenciální bezpečnostní zranitelnosti a navrhnout adekvátní bezpečnostní opatření k jejich odstranění. Struktura práce zahrnuje jak teoretickou, tak praktickou část.

V teoretické části práce budou vysvětleny základy kybernetické bezpečnosti a analyzovány současné kybernetické hrozby a zranitelnosti. Důraz bude kladen na význam kybernetické detekce a prevence. Dále v této části práce budeme podrobně rozebírat penetrační testování, včetně jeho metodik, a představíme nástroje, které budeme v průběhu testování využívat.

Praktická část práce se bude věnovat konkrétnímu provádění penetračních testů na cílových webových stránkách. Zde provedeme analýzu identifikovaných zranitelností, klasifikujeme je podle závažnosti a provedeme celkové zhodnocení bezpečnosti webového prostředí. Pro každou identifikovanou zranitelnost navrhneme a implementujeme adekvátní bezpečnostní opatření, která budou sloužit k eliminaci rizik a ochraně cílového systému.

2 Aspekty Kybernetické Bezpečnosti

S rychlým rozvojem digitálního prostředí se stává kybernetická bezpečnost klíčovým prvkem pro udržení integrity a důvěryhodnosti informačních systémů, proto se bakalářská práce se zaměřuje na neustále se rozvíjející oblast penetračního testování, která se stala nedílnou součástí celkové strategie kybernetické bezpečnosti. Představíme komplexní pohled na to, jak penetrační testování hraje klíčovou roli v odhalování kybernetických hrozeb a identifikaci zranitelností v informačních systémech. Předtím, je ale důležité porozumět kontextu, ve kterém penetrační testování operuje. V následující sekcích se zaměříme na základní koncepty kybernetických hrozeb a zranitelností, poskytujíc základ pro hlubší pohled do problematiky penetračního testování.

2.1 Kybernetické hrozby a zranitelnosti

V dnešní době je kybernetická bezpečnost nezbytným prvkem v digitálním světě, kde organizace i jednotlivci čelí neustálým hrozbám a útokům. Sekce se zaměřuje na porozumění kybernetickým hrozbám a zranitelnostem, které mohou negativně ovlivnit integritu, dostupnost a důvěrnost webových stránek. Kybernetické hrozby jsou pestrým spektrem potenciálních nebezpečí, od běžných hrozeb, jako jsou malware a phishing, až po sofistikovanější techniky, jako jsou zero-day útoky, perzistentní hrozby a sociální inženýrství.

2.1.1 Zero-day

Zero-day útoky představují nebezpečí, kdy útočníci využívají zranitelnosti v softwaru nebo hardwaru, které nebyly dosud zjištěny nebo opraveny. Během "nultého dne" odhalení zranitelnosti útočníci útočí na systém, než je vyvinuto řešení.

Zero-day útoky jsou extrémně nebezpečné, protože se využívá chyba v bezpečnosti, která dosud není známa a tedy nemá žádnou dostupnou nápravu. Útočníci mohou vstoupit do systému, ovládnout ho a způsobit značné škody. Identifikace těchto hrozeb vyžaduje neustálou sledování bezpečnostních informací a rychlou reakci na nové potenciální hrozby.

(Kasperky, 2023)

2.1.2 Remote Code Execution (RCE)

Remote Code Execution (RCE) představuje závažnou hrozbu, kdy útočníci dokážou vzdáleně provádět kód na cílovém systému, což může vést k plnému ovládnutí zařízení. RCE může být využito k vložení škodlivého kódu, spouštění neoprávněných procesů nebo dokonce ke změně konfigurace systému. Zranitelnost je často spojena s chybami v kódu aplikací nebo neadekvátním zabezpečením síťových protokolů. Identifikace a oprava těchto chyb vyžaduje detailní analýzu kódu a pravidelné aktualizace.

Jeden známý příklad takového útoku je ransomwarový útok WannaCry, který se objevil v roce 2017. Tento škodlivý kód využíval exploit jménem EternalBlue, který unikl z Národní bezpečnostní agentury.

WannaCry se šíří automaticky bez účasti uživatele, šifruje uživatelské soubory a vyžaduje výkupné za jejich odemknutí. Útok se šíří sám po získání prvního přístupu do sítě. WannaCry využívá zranitelnost počítačů, které nebyly správně zabezpečeny aktualizacemi. V prvním čtvrtletí roku 2021 byl zaznamenán 53% nárůst firem postižených útoky WannaCry, jak uvádí studie Check Point Research.

Dalším významným příkladem RCE útoku je exploit nazývaný Log4J, který se dostal do povědomí v prosinci 2021. Tento útok využívá široce používanou open-source Java softwarovou knihovnu, která je nedílnou součástí mnoha aplikací napsaných v programovacím jazyce Java. Software Log4j slouží k zaznamenávání událostí a systémových zpráv a komunikuje o nich diagnostické zprávy správcům systému. Exploit zneužívá funkci Log4j, která umožňuje uživatelům vkládat vlastní kód do logovacích zpráv. Útočníci tak vzdáleně spouštějí příkazy na cílovém počítači prostřednictvím injekce zákeřného uživatelského vstupu do logovacích zpráv, které Log4j zpracovává jako instrukce. (Ben Lutkevich, 2023)

2.1.3 Malware

Malware, zkratka pro škodlivý software, zahrnuje různé typy programů navržených k poškození nebo neoprávněnému získání přístupu k počítačovým systémům. Mezi ně patří viry, trojské koně, ransomware a spyware.

Viry mohou infikovat soubory nebo programy a šířit se mezi uživateli. Trojské koně jsou skrývané v legitimním softwaru a provádějí nebezpečné akce bez vědomí uživatele. Ransomware šifruje data a požaduje výkupné za jejich obnovení. Spyware sleduje činnost uživatele a shromažďuje citlivé informace. Identifikace a odstranění malwaru vyžaduje efektivní antivirový software a pravidelné aktualizace. (Malwarebytes, 2023)

2.1.4 Phising

Phishing je technika sociálního inženýrství, při které útočníci klamou uživatele s cílem získat citlivé informace, jako jsou hesla nebo bankovní údaje. Často se provádí pomocí falešných e-mailů nebo webových stránek.

Falešné e-maily či webové stránky vypadají autenticky a přesvědčivě, což může způsobit, že uživatelé poskytnou citlivé informace. Vzdělávání uživatelů o nebezpečích phishingu a implementace efektivních filtrů na e-maily jsou klíčové k prevenci těchto útoků.

2.1.5 Sociální inženýrství

Sociální inženýrství zahrnuje manipulaci lidí s cílem získat důvěrné informace nebo přimět je k provedení akcí, které jsou pro útočníky výhodné. Technika sociálního inženýrstvý využívá lidskou důvěřivost nebo nedostatek obezřetnosti. Útočníci mohou vytvářet přesvědčivé příběhy nebo situace, aby získali citlivé informace od zaměstnanců nebo uživatelů. Vzdělávání o bezpečnostních postupech a důsledné ověřování identity jsou klíčové k prevenci útoků založen

2.2 Penetrační testování

Penetrační testování, často označované jako ethické hackování, představuje důležitou metodu v oblasti kybernetické bezpečnosti. Cílem techniky je simulovat skutečné kybernetické útoky na informační systémy s cílem identifikovat potenciální zranitelnosti. Tím, že napodobuje postupy skutečných útočníků, penetrační testování poskytuje organizacím jedinečný pohled na jejich bezpečnostní postavení. Testování zahrnuje systematický průzkum sítí, aplikací a infrastruktury s cílem odhalit slabá místa a potenciální rizika. Penetrační testování, ale nejen webových stránek se stalo nepostradatelným nástrojem pro organizace, které chtějí aktivně přistupovat k ochraně svých informačních aktivit. V jednotlivých sekcích se seznámíme se základy a projdeme veškeré druhy penetračního testování.

2.2.1 Testování webových stránek a aplikací

Nejdříve si zodpovězme co to je webová aplikace. Je to nějaká aplikace uložená někde na serveru a poskytnuta pomocí webových prohlížečů. Během testování webových aplikací je nezbytné pečlivě analyzovat každý prvek, od uživatelského rozhraní po serverovou infrastrukturu. To zahrnuje ověřování, zda jsou aplikace odolné vůči různým formám manipulace dat, ať už přicházejícím od uživatele nebo z jiných zdrojů. Důraz klade i na správnou konfiguraci a řízení přístupu k aplikacím, aby se zabránilo neoprávněnému přístupu a zneužití dat. Testování autentizačních a autorizačních mechanismů je klíčovým prvkem bezpečnostního hodnocení webových aplikací. Penetrační tester se při testovaní webových aplikací zaměřuje na detekci možných slabých míst v autentizačním procesu a zkoumání oprávnění uživatelů. Identifikace a odstranění případných nedostatků v těchto oblastech je klíčová pro zabránění neoprávněnému přístupu a zneužití účtů. Dále se webové aplikace často interagují s externími systémy a databázemi. Testování bezpečnosti sítě zahrnuje analýzu komunikace mezi webovým serverem a backendovými systémy. Penetrační tester hledá potenciální zranitelnosti v síťové vrstvě a sleduje možné útoky, které by mohly být iniciovány prostřednictvím síťových komunikací.

V rámci testování webových aplikací je důležité ověřit, zda se data přenášená mezi klientem a serverem šifrují pomocí bezpečných protokolů (např. TLS/SSL). Penetrační tester zkoumá konfiguraci šifrovacích algoritmů, délky klíčů a certifikátů s cílem identifikovat potenciální slabiny v šifrování a minimalizovat riziko odhalení citlivých informací během přenosu. Mnoho moderních webových aplikací využívá i rozhraní API (Application Programming Interface) pro komunikaci mezi komponentami. Testování bezpečnosti API zahrnuje analýzu autentizace, autorizace a manipulace s daty prostřednictvím API. Penetrační tester zkoumá možnosti neoprávněného přístupu k API, možné útoky typu "man-in-the-middle" a další bezpečnostní hrozby spojené s API.

Tímto způsobem penetrace v rámci testování webových stránek a aplikací zajistí komplexní přístup k odhalení a eliminaci potenciálních bezpečnostních rizik, která by mohla ohrozit integritu, dostupnost a důvěrnost webových systémů. Dva nejčastější útoky, které penetrační tester, použije v souvislosti s webovými aplikacemi jsou „Cross-Site Scripting“ útoky, které tvoří přibližně 40 % a „SQL Injection“ zranitelnosti, které zase tvoří přibližně 24 % všech útoků. (SecureOPS, 2021, str. 16)

2.2.2 Interní penetrační testování

Interní penetrační testování je zaměřeno na identifikaci zranitelností a odhalení potenciálních bezpečnostních hrozeb v rámci vnitřní sítě a systémů organizace. Interní fáze penetračního testování se věnuje hodnocení odolnosti firemní infrastruktury, která není přímo vystavena veřejnému internetu, ale může čelit různým hrozbám z vnitřních zdrojů.

Během interního penetračního testování se provádí důkladné zkoumání mechanismů vnitřní autentizace a oprávnění. Penetrační testeři analyzují například sílu uživatelských hesel, kontrolu přístupu k firemním zdrojům a způsoby, jakými jsou řešeny vnitřní autorizační procesy. Identifikace slabých míst v této oblasti může vést k neoprávněnému přístupu k důvěrným informacím a systémům.

Interní testování zahrnuje i skenování a analýzu interní sítě s cílem identifikovat potenciální zranitelnosti a slabiny. Tester sleduje možné přístupy k citlivým informacím, monitoruje síťový provoz a zkoumá, zda jsou vnitřní sítě odolné proti různým formám útoků, jako jsou odposlechy a odvrácené inženýrství. Dalším testování, které se v tomto segmentu provádí simulace ruzných interních útoků, kde využívají své znalosti o firemní infrastruktuře a procesech k identifikaci možných cílů a scénářů útoků. Cílem je následně odhalit, jak efektivně je organizace schopna detekovat a reagovat na interní hrozby a neautorizovanou činnost útočníka.

V částí interního testování se ale tester může setkat s testováním bezpečnosti interních aplikací. To zahrnuje analýzu interních webových aplikací, databází a dalších systémů, které mohou být potenciálním cílem interních útoků. Tester sleduje možné slabiny v kódu, šifrování dat a správě oprávnění.

Na konci po provedení celého testu a analýzy je vždy sepsán i detailní report všech identifikovaných rizik. Závěreční report i vždy obsahuje řešení zmíněných hrozeb, a většinou i návrhy a implementaci různých dalších opatření pro zlepšení celé bezpečnosti interní sítě.

2.2.3 Externí penetrační testování

Externí penetrační testování se zaměřuje na bezpečnostní hodnocení systémů a sítí organizace z pohledu potenciálního útočníka, který působí mimo firemní prostředí. Cílem je identifikovat a odstranit zranitelnosti, které by mohly být využity k neoprávněnému přístupu či útoku ze strany vnějších hrozeb.

Externí testování začíná skenováním přístupových bodů do firemní sítě. To zahrnuje jak webové aplikace, e-mailové servery, tak i firemní portály a další komponenty, které jsou přístupné z internetu. Cílem je identifikovat otevřené porty, zranitelnosti a potenciální bezpečnostní slabinu, které by mohly umožnit neautorizovaný přístup.

V dnešní době mnoho organizací využívá cloudové služby pro ukládání dat, provoz webových aplikací a další potřeby. Externí penetrační testování zahrnuje i hodnocení bezpečnosti těchto cloudových služeb. Tester se zaměřuje na konfiguraci, šifrování dat a správu přístupových práv, aby zajistil, že cloudové prostředí je odolné vůči vnějším útokům.

Externí penetrační testování zahrnuje simulaci různých typů útoků, které by mohly být spuštěny z vnějšího prostředí. Testeři využívají metody, jako jsou phishing, odposlechy, DNS útoky a skenování zranitelností, aby identifikovali možné scénáře útoků. Cílem je zjistit, jak efektivně

organizace reaguje na externí hrozby a zda jsou implementována odpovídající bezpečnostní opatření.

Externí penetrační testování představuje klíčový krok pro udržení robustní bezpečnosti organizace v dnešním prostředí, kde jsou vnější hrozby neustále přítomné. Pravidelné opakování těchto testů je nutné z důvodu rychle se pohybující kyberbezpečností, kde se může velice rychle objevit nový útočný vektor, pravidelné testování tedy napomáhá zabezpečit organizaci před těmi nejnovějšími kybernetickými hrozbami.

3 Metodologie Penetračního Testování

Metodologie penetračního testování hraje klíčovou roli v oblasti kybernetické bezpečnosti, poskytujíc strukturovaný a systematický přístup k identifikaci a hodnocení bezpečnostních zranitelností v informačních systémech. Metodologie slouží jako navigační nástroj pro odborníky v oboru, kteří provádějí penetrační testy, aby systematicky prozkoumali a vyhodnotili bezpečnostní postavení organizací. Metodologie poskytuje rámec, který pomáhá definovat postupy, techniky a cíle testování, což umožňuje konzistentní a komplexní přístup k identifikaci potenciálních hrozeb a zranitelností. Využití metodologií také zajišťuje, že testování je provedeno profesionálním a etickým způsobem, minimalizuje rizika a poskytuje relevantní výsledky.

3.1 Open Source Security Testing Methodology Manual

Open Source Security Testing Methodology Manual (OSSTMM) poskytuje metodologii pro důkladné bezpečnostní testování, známé jako OSSTMM audit. OSSTMM audit představuje přesné měření bezpečnosti na operační úrovni, které je prosté předpokladů a anekdotických důkazů.

Jelikož je OSSTMM otevřeným projektem, umožňuje každému bezpečnostnímu testerovi přispívat nápady pro provádění přesnějších, akčních a efektivních bezpečnostních testů. Otevřenost metodologie podporuje komunitní spolupráci a inovace v oblasti bezpečnostního testování.
Primárním účelem tohoto manuálu je poskytnout vědeckou metodologii pro přesné charakterizaci operační bezpečnosti (OpSec) prostřednictvím zkoumání a korelace výsledků testů způsobem konzistentním a spolehlivým.

Metodologie je přizpůsobitelná téměř jakémukoliv typu auditu, včetně penetračních testů, etického hackování, hodnocení bezpečnosti, hodnocení zranitelností, red-teaming, blue-teaming a dalších. Kdokoli, kdo používá tuto metodologii pro bezpečnostní testování a analýzu a dokončí platný Security Test Audit Report (STAR), provedl OSSTMM audit.

OSSTMM je založen na metodě "rozděl a panuj", což zahrnuje rekurzivní rozkládání problému na dva nebo více podproblémy stejného typu, dokud nejsou dostatečně jednoduché na přímé řešení. Tím je dosaženo systematického a strukturovaného přístupu k analýze bezpečnostních otázek.Pro zajištění bezpečnosti je nutné mít v místě kontroly, které minimalizují hrozbu samotnou nebo její účinky na akceptovatelnou úroveň ze strany vlastníka nebo manažera aktiva.

Kategorie omezení v rámci OSSTMM představují klíčový prvek pro strukturování a porozumění identifikovaným bezpečnostním omezením. Rozdělení omezení do pěti prvků – Exposure, Vulnerability, Weakness, Concern a Anomalies. Omezení umožňují získat jasný přehled o funkci každého omezení v operačním prostředí. Kategorizace dává organizacím přesnější nástroj pro nápravu bezpečnostních slabých míst a efektivní řízení rizik.

Tab. 1: Tabulka mapování omezení

Category		OpSec	Limitations
Operations		Visibility	Exposure
		Access	Vulnerability
		Trust	
Controls	Class A - Interactive	Authentication	Weakness
		Indemnification	
		Resilience	
		Subjugation	
		Continuity	
	Class B - Process	Non-Repudiation	Concern
		Confidentiality	
		Privacy	
		Integrity	
		Alarm	
			Anomalies

Zdroj: isecom.org (2010, s. 29)

Exposure poskytuje informace o interakci s cílem a přímo mapuje na Viditelnost a Přístup. Informace o exposure může pomoci útočníkovi obejít některé nebo všechny kontroly, a proto je Exposure také mapována na obě třídy kontrol. Exposure nemá význam sám o sobě, pokud neexistuje způsob, jak tuto informaci využít k exploataci aktiv nebo kontroly. Vulnerabilities, Weaknesses a Concerns hrají také roli při vážení hodnoty Exposure.

Vulnerability je chyba nebo nedostatek, který odepírá přístup k aktivům oprávněným osobám nebo procesům, nebo umožňuje privilegovaný přístup k aktivům neoprávněným osobám nebo procesům, a umožňuje neoprávněným osobám nebo procesům skrýt aktivity nebo sebe v rámci daného rozsahu. Vulnerability musí být mapována na všechny body interakce nebo OpSec a vzhledem k tomu, že Vulnerability může obejít nebo zneplatnit kontroly, musí být také zvážena při vážení Vulnerability.

Concern může být nalezena pouze v kontrolách třídy B, ale může ovlivnit OpSec, proto je mapována na všechny parametry OpSec a patří do této třídy procesních kontrol.

Weakness je chyba v kontrolách třídy A, která může ovlivnit OpSec, proto je mapována na všechny parametry OpSec, ačkoli patří do této interaktivní třídy kontrol.

Anomaly je jakýkoli neidentifikovatelný nebo neznámý prvek, který nebyl kontrolován a nelze ho zahrnout do běžných operací. Skutečnost, že nebyl kontrolován a nelze ho zahrnout, ukazuje přímý vztah k Důvěře. Anomaly také může způsobit anomálie ve fungování kontrol, a proto jsou zahrnuty při vážení.

3.2 OWASP Testing Guide

OWASP Testing Guide, který byl vyvinut organizací OWASP (Open Web Application Security Project) představuje klíčový nástroj v boji proti bezpečnostním hrozbám ve webových aplikacích. Jeho systematický přístup k testování bezpečnosti poskytuje odborníkům na bezpečnost, vývojářům a testerům strukturovaný rámec pro identifikaci a eliminaci potenciálních rizik. Proces testování, rozdělený do fází, nejenom umožňuje důkladné zkoumání aplikace, ale také pomáhá vytvářet jasný obraz o možných bezpečnostních slabých bodech a jejich potenciálních dopadech.

OWASP Testing Guide se rozděluje na dvě hlavní fáze: pasivní a aktivní. V pasivní fázi tester adoptuje roli pozorovatele a snaží se porozumět fungování celé webové stránky. Pro tento účel může využívat nástroje jako je Burp Suite, který umožňuje sledovat provoz mezi prohlížečem a serverem. Tester v Burp Suite často začíná v sekci "Proxy", kde zachytává veškerou komunikaci mezi prohlížečem a webovým serverem. Tím může analyzovat a manipulovat s HTTP požadavky a odpovědmi.

V rámci pasivní fáze může tester využít funkce Burp Suite pro odhalení informací o aplikaci, jako jsou otevřené porty, technologie použité v back-endu, a další. Zároveň může analyzovat správnost konfigurace a identifikovat potenciální bezpečnostní rizika.

Při přechodu do aktivní fáze tester využívá OWASP Testing Guide, který je rozdělen do jedenácti kategorií. V této fázi tester systematicky aplikuje jednotlivé testovací postupy na konkrétní aspekty webové aplikace. Například při testování autentizace (Authentication Testing) bude zkoumat, jak aplikace ověřuje identity uživatelů, zatímco v kategorii Input Validation Testing analyzuje, jak jsou zpracovávány vstupy od uživatelů.

Tímto strukturovaným přístupem kombinujícím pasivní a aktivní fázi, spolu s využitím nástrojů jako Burp Suite, získá tester komplexní pohled na bezpečnost webové aplikace a může identifikovat a adresovat potenciální rizika. V následujících odstavcích si predstavíme všech jedenáct kategorií a krátce si vysvětlíme jakou úlohu mají na starosti.

3.2.1 Information Gathering

Fáze Information Gathering hraje klíčovou roli při identifikaci potenciálních bezpečnostních hrozeb. Testovací tým se zaměřuje na sběr důležitých informací o webové aplikaci, včetně identifikace otevřených portů, detekce technologií a získávání detailů o síťové infrastruktuře. Analýza je klíčová pro pochopení prostředí aplikace a nasměrování následujících testovacích kroků.

3.2.2 Configuration and Deployment Management

Fáze Configuration and Deployment Management Testing se zaměřuje na důkladné zkoumání konfiguračních a nasazovacích procesů webové aplikace. Testování správnosti konfigurace je klíčové pro odhalení potenciálních bezpečnostních chyb, které by mohly vzniknout v důsledku nesprávně nastavených parametrů nebo chybných nasazovacích postupů.

3.2.3 Identity Management Testing

Fáze Identity Management Testing se specializuje na testování mechanismů řízení identity aplikace. Identifikace a ověření uživatelů jsou kritické pro zajištění bezpečnosti. Testování identity zahrnuje ověření, že systém adekvátně spravuje uživatelské účty, jejich změny a odstranění, a že přístup k citlivým datům je správně regulován.

3.2.4 Authentication Testing

V rámci Authentication Testing jsou zkoumány mechanismy ověřování uživatelů. Testovací tým identifikuje a prověřuje různé formy autentizace, včetně ověřování pomocí hesel, dvoufaktorové autentizace a dalších. Cílem je odhalit případné slabiny v autentizačním procesu, které by mohly umožnit neoprávněný přístup k aplikaci.

3.2.5 Authorization Testing

Fáze Authorization Testing se zaměřuje na testování oprávnění uživatelů k přístupu k různým částem webové aplikace. Testuje se, zda jsou oprávnění správně nastavena a zda uživatelé mají přístup pouze k tomu, co by měli. Cílem je odhalit potenciální chyby v regulaci oprávnění, které by mohly vést k neoprávněnému přístupu.

3.2.6 Session Management Testing

Testování správy relací se zaměřuje na bezpečnostní aspekty spojené s udržováním relací uživatele během jeho interakce s webovou aplikací. Identifikuje možné bezpečnostní hrozby, jako jsou session fixation a session hijacking, a zkoumá, zda jsou mechanismy správy relací dostatečně odolné vůči útokům.

3.2.7 Input Validation Testing

Fáze se zaměřuje na testování ověřování vstupních dat. Identifikuje a testuje se, jak aplikace zpracovává uživatelský vstup, aby se předešlo možným bezpečnostním rizikům, jako jsou SQL injection nebo cross-site scripting (XSS). Cílem je zajistit, že vstupy jsou validovány a filtrovány tak, aby minimalizovaly riziko útoků.

3.2.8 Error Handling

Fáze Error Handling se zabývá testováním reakce webové aplikace na chyby. Testuje se, zda jsou chybové zprávy vytvářeny bezpečným způsobem a zda nedávají neoprávněným jedincům příliš mnoho informací o vnitřní struktuře aplikace. Cílem je zajistit, že chybové zprávy jsou informativní pro uživatele, ale zároveň bezpečné.

3.2.9 Cryptography

Fáze Cryptography testuje šifrovací mechanismy používané pro zabezpečení citlivých dat. Zahrnuje testování síly šifrování, správné implementace kryptografických algoritmů a náležitého spravování klíčů. Cílem je zajistit, že šifrování je dostatečně odolné proti různým typům útoků.

3.2.10 Business Logic Testing

Fáze business logic testing se zaměřuje na testování obchodní logiky webové aplikace. Identifikuje a testuje, zda jsou obchodní procesy a pravidla řádně implementovány a zda nedochází k žádným neintuitivním či nebezpečným chováním aplikace vlivem nekorektních vstupů nebo manipulace s procesy.

3.2.11 Client Side Testing

Fáze Client Side Testing testuje bezpečnost na straně klienta webové aplikace. Zkoumá se, jak aplikace komunikuje s klientovým zařízením, jak jsou odesílána a zpracovává data na straně klienta a jak jsou implementovány bezpečnostní mechanismy v prohlížeči. Cílem je identifikovat potenciální bezpečnostní problémy související s klientem.

3.3 NIST SP 800-115

NIST SP 800-115 je standard zveřejněný Národním institutem pro standardy a technologii (National Institute of Standards and Technology - NIST) ve Spojených státech amerických. Tento standard se týká řízení bezpečnosti informací a je specificky zaměřen na postupy pro správu událostí a informací v oblasti bezpečnosti (Security Event Management, SEM).

Standard poskytuje doporučení a směrnice pro implementaci procesů a technik, které organizacím pomáhají monitorovat a reagovat na kyberbezpečnostní události v jejich informačních systémech. Jeho cílem je zlepšit schopnost organizací identifikovat, analyzovat a reagovat na bezpečnostní incidenty a události v reálném čase.

Během implementace jsou testerům poskytnuty nástroje a techniky pro identifikaci a využití bezpečnostních chyb. Využívání automatizovaných skenovacích nástrojů, analýza síťového provozu a manuální penetrace jsou klíčovými průmyslovými postupy, které NIST SP 800-115 podporuje. Kombinace umožňuje komplexní testování a identifikaci různorodých bezpečnostních rizik. Dále se i efektivně integruje s dalšími průmyslovými normami a rámcovými metodologiemi pro bezpečnost informačních systémů.

Po dokončení fáze provedení (execution phase) organizace provádějí důkladnou analýzu zjištěných zranitelností a nedostatků. Analýza je klíčová pro identifikaci kořenových příčin bezpečnostních nedostatků a umožňuje organizacím vypracovat konkrétní plán pro zlepšení celkové bezpečnosti.

NIST rovněž doporučuje, aby organizace vytvořily zprávu obsahující všechny nalezené zranitelnosti a doporučená opatření pro jejich odstranění. V této fázi je důležité stanovit priority pro odstraňování zranitelností na základě jejich vážnosti a potenciálního dopadu na bezpečnostní postavení organizace. (NIST SP 800-115, 2008)

NIST zdůrazňuje důležitost vytvoření opakovatelné a zdokumentované metodologie pro provedení bezpečnostních hodnocení. Metodologie NIST poskytuje konzistenci a strukturu hodnotícího procesu, usnadňuje začleňování nových členů týmu a adresuje omezení spojená s hodnocením. Organizace by měly rovněž stanovit cíle každého bezpečnostního hodnocení a přizpůsobit přístup podle těchto cílů. Používání kombinace technik umožňuje organizacím omezit rizika a optimalizovat využívání zdrojů. (NIST SP 800-115, 2008)

3.4 PTES

PTES je standard pro provádění penetračních testů, který poskytuje systematický přístup k celému procesu testování. Tento standard definuje postupy, fáze a cíle penetračního testování a slouží jako rozsáhlý rámec pro penetrační testery. PTES je navržen tak, aby byl komplexní a zároveň flexibilní, což umožňuje přizpůsobit testování konkrétním potřebám organizace.

PTES hraje klíčovou roli v poskytování metodiky a struktury pro penetrační testování. Pomáhá zajistit, že penetrační testy jsou prováděny systematicky a že všechny relevantní aspekty jsou řádně pokryty. Organizace mohou využívat PTES k zajištění konzistentních a efektivních penetračních testů, což v konečném důsledku přispívá k zvýšení úrovně bezpečnosti informačního prostředí.

PTES se skládá ze sedmi klíčových fází, o kterých se v následujících odstavcích dozvíme více. těmi jsou pre-engagement, Intelligence Gathering, Threat Modeling, Vulnerability Analysis, Exploitation, Post Exploitation, Reporting. (The PTES Team, 2022)

3.4.1 Pre-engagement

Fáze pre-engagement představuje klíčový okamžik v definování rozsahu a cílů penetračního testování. Spolupráce s organizací v této fázi umožňuje stanovit nejdůležitější aspekty testování a vytvořit pravidla, která zajistí relevantnost a efektivitu celého procesu. Zkušení penetrační testři se snaží porozumět podnikovému prostředí, klíčovým informacím a nejcitlivějším částem systémů, aby identifikovali potenciální bezpečnostní slabiny. Důkladný průzkum prostředí a kvalitní komunikace s organizací v této fázi vytváří pevný základ pro úspěšné provedení testů.

3.4.2 Intelligence Gathering

Fáze sběru informací (Intelligence Gathering) hraje klíčovou roli při identifikaci a shromažďování relevantních údajů o cílovém prostředí. Při identifikaci fyzické polohy je důležité rozpoznat, zda se jedná o samostatnou budovu nebo o součást většího zařízení. Identifikace produktů a případných informací o jejich spuštění skrze firemní webové stránky, nové verze nebo prostřednictvím vyhledávače může poskytnout cenné náhledy do interních mechanismů cíle.

Často se stává, že podniky veřejně oznamují takové informace s cílem získat publicitu a informovat stávající nebo nové zákazníky o spuštění. Identifikace případných charitativních vazeb podniku skrze firemní webové stránky nebo vyhledávač může poskytnout cenný vhled do interních procesů a potenciálně i firemní kultury.

Důvodem pro sběr těchto informací je plné porozumění případné firemní agresivity. Autoři provádějí sběr otevřených informací s cílem určit různé body vstupu do organizace, které mohou být fyzické, elektronické a/nebo lidské. Mnoho společností opomíjí, jaké informace o sobě zveřejňují veřejně, a jak mohou být využity útočníkem s dostatečným odhodláním. Stejně tak mnoho zaměstnanců zanedbává uvědomění si informací, které o sobě veřejně zveřejňují, a jak mohou být informace použity proti nim či jejich zaměstnavateli.

Je důležité si uvědomit, co OSINT není. Otevřený zdrojový průzkum nemusí být přesný nebo aktuální. Informační zdroje mohou být záměrně/omylem manipulovány tak, aby odrážely chybná data, informace mohou zastarávat s časem nebo být neúplné. OSINT také nezahrnuje hledání v odpadcích nebo jakékoliv metody získávání firemních informací z fyzických předmětů nalezených na místě. (The PTES Team, 2022)

3.4.3 Threat Modeling

Hodnocení a modelování hrozeb a zranitelností jsou klíčové kroky v této fázi. Testeři zkoumají potenciální scénáře útoků a analyzují, jak by mohly být zneužity nalezené zranitelnosti. Úsilí poskytuje organizaci náhled na možná rizika a umožňuje zaměřit se na ty nejprioritnější oblasti. Využívání těchto informací zajišťuje, že penetrační testy budou zaměřeny na klíčové hrozby a umožňuje efektivní využití prostředků pro maximalizaci bezpečnosti.

V této fázi testovací tým provádí analýzu a diskuse o možných scénářích útoků, které by mohly ohrozit bezpečnostní postavení organizace. Identifikace, jak by mohli útočníci využít nalezené slabiny, a priorizace těchto rizik poskytuje směr pro následné kroky v procesu penetračního testování. Zjištěné informace jsou klíčové pro vytvoření účinné strategie, jak efektivně využít zdroje a zaměřit se na nejcitlivější aspekty bezpečnosti.

3.4.4 Vulnerability Analysis

Fáze analýzy zranitelností je klíčovým prvkem penetračního testování, který se zaměřuje na identifikaci a hodnocení bezpečnostních rizik spojených s nalezenými zranitelnostmi. Práce v rámci analýzy zranitelností je rozdělena do dvou hlavních oblastí: Identifikace a Validace. Během fáze Identifikace je klíčovým prvkem úsilí o objevení zranitelností. Zaměřuje se na identifikaci potenciálních bezpečnostních slabých míst, a to prostřednictvím rozsáhlého úsilí o odhalení různých typů zranitelností.

Následně je ve fázi Validace snaha o snížení počtu identifikovaných zranitelností na ty, které jsou skutečně platné a relevantní pro bezpečnostní posouzení. Tímto procesem je zajištěno, že výsledky penetračního testování odrážejí reálná bezpečnostní rizika, která by organizaci mohla ohrozit.

V rámci analýzy zranitelností se používá Vulnerability Testing, který zahrnuje aktivní i pasivní metody hodnocení. Pro aktivní testování je navržen automatizovaný skener, který slouží k posouzení sítí, hostitelů a přidružených aplikací. Před spuštěním jakéhokoli skenování pomocí nástroje jako je Nessus je důležité zajistit, že byl produkt aktualizován nejnovějšími podpisy. Produktem rozsáhlého výzkumu v oblasti kybernetické bezpečnosti je objevování zranitelností spolu s Proof of Concept nebo kódem pro zneužití. Výsledky z fáze identifikace zranitelností musí být individuálně ověřeny a tam, kde jsou dostupné konkrétní metody útoku, je třeba je ověřit.

Jedinou výjimkou jsou situace, kde zneužití vede k odepření služby (DoS), což je nutné zařadit do rozsahu pro účely ověření.

Při pokusu identifikovat, zda je zařízení, aplikace nebo operační systém zranitelný vůči útoku s využitím výchozích přihlašovacích údajů, postačí jednoduchý pokus o zadání známých výchozích hesel. Dobře definovaný seznam cílů by měl obsahovat mapování informací o verzi operačního systému, úrovni záplat a známých slabostech webových aplikací, blokovacích prahů a slabých portů pro útok. Identifikace slabých portů lze provádět pomocí technik jako je banner grabbing, nmap a spolehlivého úsudku. Mnoho portů a služeb může klamat nebo uvádět do omylu ohledně konkrétní verze. (The PTES Team, 2022)

3.4.5 Exploitation

Tady penetrační testři aktivně využívají nalezené zranitelnosti s cílem získat neoprávněný přístup. Jejich úkolem je simulovat reálný útok a otestovat, jak rychle a efektivně organizace dokáže reagovat a bránit se proti němu. Dále poskytuje příležitost testovacímu týmu nejenom ověřit existenci zranitelností, ale také získat hlubší porozumění reakční schopnosti organizace vůči skutečnému hrozbě. Využití různých technik a postupů při exploitaci pomáhá identifikovat možné slabiny v obranných mechanismech a zlepšit celkovou úroveň bezpečnosti.

3.4.6 Post Exploitation

Po úspěšném průniku a kompromitaci systému se spouštějí post-exploitation aktivity, které jsou klíčové pro porozumění reálnému dopadu úspěšného útoku. Aktivity se liší v závislosti na typu operačního systému a mohou zahrnovat od spuštění jednoduchého příkazu "whoami" až po enumeraci lokálních účtů. (The PTES Team, 2022)

Post-exploitation fáze je zaměřena na udržení dlouhodobého přístupu a získávání dalších informací o kompromitovaném systému. Testovací tým může provádět různé operace, včetně mapování sítě, sběru citlivých dat a zkoumání možností, jakými by mohl útočník nadále zneužívat kompromitovaný systém.

Aktivity v post-exploitation fázi jsou klíčové pro hodnocení celkové odolnosti organizace vůči prodlouženým kybernetickým hrozbám. Identifikované slabiny v této fázi umožňují organizaci adekvátně reagovat na potenciální rizika a posílit svou kybernetickou bezpečnostní strategii.

3.4.7 Reporting

V poslední fázi penetračního testování, tedy v reportingu, testovací tým poskytuje organizaci komplexní dokumentaci všech provedených kroků, identifikovaných zranitelností a doporučení pro zlepšení celkové bezpečnosti. Etapa reportingu je klíčovým prvkem, neboť získané informace mají posloužit jako nástroj pro organizaci k aktivnímu reagování na identifikované nedostatky a ke zvýšení úrovně kybernetické bezpečnosti.

V rámci reportingu jsou detailně popsány jednotlivé fáze penetračního testu, včetně pre-engagement, sběru informací, modelování hrozeb, analýzy zranitelností, exploatace a post-

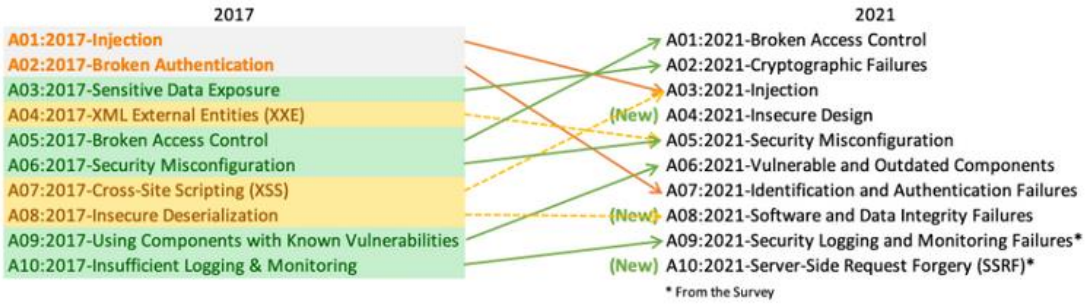
exploatace. Každá fáze je podrobně popsána včetně nalezených zranitelností, jejich míry rizika a možného dopadu na bezpečnostní postavení organizace.

Vzhledem k předešlým informacím může report obsahovat zajímavé aspekty týkající se současného stavu incidentní reakce organizace. Důraz může být kladen na schopnost organizace reagovat na simulovaný útok, jak efektivně byla identifikována a řešena bezpečnostní incidenty, a jak bylo otestováno napojení incidentního reakčního týmu na vedení organizace.

Kromě identifikovaných nedostatků může report obsahovat i pozitivní body, například uvedení bezpečnostních opatření, která prokázala svou efektivitu při odhalení a bránění v simulovaných útocích. Celkovým cílem reportingu je poskytnout organizaci kompletní pohled na její bezpečnostní postavení, nabídnout doporučení pro zdokonalení a podpořit celkovou kybernetickou rezilience.

4 OWASP TOP 10

Předem již zmíněná organizace OWASP (Open Web Application Security Project) vytváří mimo metodiku i seznam deseti nejčastějších bezpečnostních hrozeb, známý jako OWASP TOP 10. Těchto 10 hrozeb představuje klíčová rizika pro webové aplikace a poskytují důležitý rámec pro identifikaci a odstranění bezpečnostních nedostatků. OWAP aktualizuje TOP 10 seznam každé přibližně 4 roky, v době psaní bakalářské práce je poslední aktualizace z roku 2021, tudíž se následující seznam zranitelností může v následujících letech lišit, dále níže vypsané kapitoly jsou seřazené podle OWASP a to od těch nejčastějších.



Obr. 1: Vývoj OWASP TOP 10 2017-2022

Zdroj: *OWASP.org*. (2021)

Na výše zmiňovaném grafu, vytvořeném organizací OWASP, je patrný dynamický vývoj kyberbezpečnostního prostředí během čtyř let. Graf z roku 2017 do roku 2021 ukazuje výrazné změny v pořadí a podstatě hlavních bezpečnostních rizik webových aplikací. Zelené šipky znázorňují posuny výše, oranžové dolů, a čárkované šipky naznačují sloučení kategorií nebo jiné organizační úpravy. Zvláštní pozornost zasluhuje označení dvou hrozeb hvězdičkami, což znamená, že tyto hrozby byly vybrány komunitou a nemusí být podloženy konkrétními daty.

4.1 Broken Access Control

Problémy s řízením přístupu mohou umožnit neoprávněným uživatelům získat přístup k citlivým informacím nebo funkcím. Hrozba zahrnuje nesprávně nakonfigurované přístupové kontroly, nedostatečnou autentizaci a autorizaci. Při nesprávném nastavení může být uživatel schopen získat vyšší oprávnění, než mu bylo přiděleno, což může vést k úniku citlivých dat nebo neoprávněné manipulaci s funkcemi aplikace. Identifikace a náprava těchto nedostatků je klíčová pro udržení bezpečnosti aplikace.

Byl posunul nahoru z pátého místa na první pozici, což ho řadí mezi nejzávažnější rizika kybernetické bezpečnosti webových aplikací. Data naznačují, že průměrně 3,81 % testovaných aplikací obsahuje jednu nebo více zranitelností v této kategorii, s více než 318 tisíci případy těchto zranitelností. (OWASP, 2021)

4.2 Cryptographic Failures

Chyby v kryptografii mohou způsobit únik citlivých dat nebo umožnit útočníkům provádět útoky, jako jsou dešifrování šifrovaných zpráv nebo padělání digitálních podpisů. Nesprávné používání kryptografických algoritmů, slabé klíče nebo nedostatečná ochrana klíčů mohou znamenat zranitelnost. Správná implementace a údržba kryptografických postupů a pravidel je nezbytná pro odolnost proti těmto hrozbám.

Kategorie se oproti roku 2017 posunula o jedno místo nahoru na druhou pozici. Kategorie se dříve nazývala A3:2017-Sensitive Data Exposure, ale nový název zdůrazňuje i selhání v oblasti šifrování. Oblast často vede k úniku citlivých dat nebo kompromitaci systému. (OWASP, 2021)

4.3 Injection

Hrozba injekce se týká situací, kdy jsou uživatelská data nebo jiný neověřený vstup vloženy přímo do programového kódu. SQL injection a XSS jsou příklady injekčních útoků. Útočník může vkládat škodlivý kód nebo příkazy, což může vést k neautorizovanému přístupu k databázím, manipulaci s obsahem stránek nebo provádění škodlivých akcí v kontextu uživatele. Kvalitní filtrování, validace a parametrizace vstupních dat jsou nezbytné pro zabránění injekčním útokům. 94 % testovaných aplikací bylo testováno na nějaký druh injection s 19% maximální mírou incidence. Kategorie zahrnuje 33 zranitelností a je nyní rozšířena o Cross-site Scripting. (OWASP, 2021)

4.4 Insecure Design

Nesprávný nebo nedostatečný návrh aplikace může vést k závažným bezpečnostním problémům. Insecure Design kategorie byla pro rok 2021 nově vytvořena a zahrnuje chyby v architektuře, které umožňují útočníkům obejít bezpečnostní opatření. Nedostatečná izolace komponent, nebo nesprávné rozhodnutí ohledně datových toků, mohou způsobit, že aplikace bude náchylná k útokům. Bezpečný a pečlivě promyšlený design aplikace je klíčový pro odolnost proti těmto rizikům.

4.5 Security Misconfiguration

Bezpečnostní konfigurace hraje klíčovou roli v odolnosti aplikace proti útokům. Chyby v konfiguraci mohou znamenat, že jsou nebo byly některé části aplikace nesprávně nastaveny, což může vytvářet zranitelné body. K tomu patří nejen serverové konfigurace, ale i konfigurace databází, úložišť a dalších komponent. Bezpečnostní konfigurace by měla být pečlivě navržena a pravidelně auditována, aby se minimalizovalo riziko nesprávné konfigurace, která může znamenat bezpečnostní hrozbu. Okolo 90% testovaných aplikací bylo zkoumáno kvůli nějaké formě nesprávné konfigurace s průměrnou mírou incidence 4,5% a více než 208 tisíci případy CWEs v této kategorii se tudíž řádí na páté místo seznamu z předchozího šestého. (OWASP, 2021)

4.6 Vulnerable and Outdated Components

Využívání zastaralých nebo zranitelných komponent může poskytnout útočníkovi snadný přístup k systému. Hrozba se týká všech komponent aplikace, včetně frameworků, knihoven a dalších softwarových součástí. Neudržované nebo nepřehledné třetí strany mohou obsahovat známé bezpečnostní chyby, které mohou být snadno zneužity. Pravidelná aktualizace a sledování záplat a aktualizací jsou nezbytné pro minimalizaci rizika zranitelností způsobených zastaralými komponentami.

4.7 Identification and Authentication Failures

Nesprávné provedení identifikace a autentizace může umožnit útočníkům neoprávněný přístup k účtům a datům uživatelů. Hrozba zahrnuje chyby v procesu ověřování totožnosti a autentizace, což může vést k neoprávněnému přístupu nebo dokonce k úplnému převzetí účtu. Silné a správně implementované postupy pro identifikaci a autentizaci jsou klíčové pro zajištění bezpečného prostředí.

4.8 Software and Data Integrity Failures

Integrita softwaru a dat je klíčová pro zajištění důvěryhodnosti a spolehlivosti aplikace. Nově v roce 2021 byla představena jako nová kategorie, která zaměřuje se na předpoklady týkající se aktualizací softwaru, kritických dat a CI/CD pipelinů bez ověření integrity. Chyby v integritě mohou umožnit útočníkům modifikovat data, šířit škodlivý kód nebo narušit integritu softwarových komponent. To může mít za následek ztrátu důvěryhodnosti uživatelů a spolehlivosti aplikace. Důkladná kontrola integrity softwaru a dat by měla být prováděna pravidelně, aby se minimalizovalo riziko integrity selhání. (OWASP, 2021)

4.9 Security Logging and Monitoring Failure

Neschopnost správně zaznamenávat a monitorovat bezpečnostní události může znamenat, že organizace bude mít omezenou schopnost rozpoznat nebo reagovat na bezpečnostní incidenty. Kategorie v roce 2017 známá jako „Insufficient Logging & Monitoring“ se zaměřuje na nedostatečné logování a monitorování, což v případě nezájmu ze strany kyberbezpečnostích specialistů a techniků může znamenat, že bezpečnostní hrozby nebudou odhaleny včas, což zvyšuje riziko nepovšimnutých útoků. Kvalitní systémy logování a monitorování jsou klíčové pro rychlé zjišťování a reakci na bezpečnostní incidenty.

4.10 Server-Side Request Forgery

Server-Side Request Forgery (SSRF) je technika, přidána na základě komunitního hlasu, při níž útočník přiměje server provádět neoprávněné operace na interní síti. To může zahrnovat čtení citlivých dat, zneužívání interních služeb nebo provádění dalších útoků. Identifikace a ochrana proti SSRF vyžadují pečlivou validaci uživatelských vstupů a řádnou konfiguraci serverových prostředí. Správná implementace těchto opatření je klíčová pro minimalizaci rizika ze strany SSRF.

5 Použité nástroje

Použití specializovaných nástrojů pro penetrační testování přináší několik výhod a je klíčové pro úspěšné identifikaci a řešení bezpečnostních nedostatků webových aplikací. První výhodou je automatizace procesu skenování a testování, která umožňuje rychlejší a efektivnější objevení potenciálních bezpečnostních rizik. Nástroje jako JoomScan, Burp Suite nebo ZAP jsou schopny provést systematickou analýzu a identifikaci zranitelností v rozsáhlých webových aplikacích během krátké doby, což značně šetří čas a zdroje oproti ručnímu testování.

Další výhodou je objektivita a konzistence výsledků. Automatizované nástroje provádějí testování podle předem definovaných pravidel a metodik, což minimalizuje riziko lidské chyby a zajistí konzistentní výsledky mezi opakovanými testy. To umožňuje lepší porovnání výsledků testů a identifikaci změn v bezpečnostním stavu aplikace v čase.

5.1 Burp Suite

Burp Suite je komplexní sada nástrojů navržená pro testování bezpečnosti webových aplikací. Jeho multifunkční povaha a široká škála funkcí umožňují penetračním testerům provádět rozsáhlou analýzu bezpečnostních aspektů webových aplikací. Jedním z hlavních prvků Burp Suite je Burp Proxy, který slouží k zachytávání a manipulaci s HTTP požadavky a odpovědi mezi klientem a serverem. Tento nástroj umožňuje penetračním testerům monitorovat provoz a analyzovat chování webové aplikace v reálném čase.

Další klíčovou součástí Burp Suite je Burp Scanner, který poskytuje automatizované skenování webových aplikací na zranitelnosti. Tento nástroj je schopen identifikovat různé typy bezpečnostních rizik, jako jsou SQL injection, cross-site scripting (XSS) nebo přístupová práva k souborům. Burp Scanner umožňuje penetračním testerům rychle identifikovat a prioritizovat zranitelnosti ve webových aplikacích a poskytuje detailní zprávy o nalezených bezpečnostních nedostatcích. Burp scanner je ale pouze dostupný v placené verzy softwaru, proto se vyplatí kombinovat s například ZAP, který je zcela zdarma.

Dále Burp Suite zahrnuje další užitečné nástroje, jako je Burp Repeater pro opakované testování konkrétních požadavků, Burp Intruder pro automatizované testování velkého množství vstupů, a Burp Spider pro procházení a mapování webových aplikací. Tato rozmanitost nástrojů umožňuje penetračním testerům provádět komplexní a důkladné testování bezpečnosti webových aplikací a identifikovat různé typy bezpečnostních rizik.

5.2 Joomscan

JoomScan je specializovaný nástroj určený pro testování bezpečnosti webových aplikací postavených na platformě Joomla. Jeho hlavním cílem je identifikovat a vyhodnotit bezpečnostní rizika a zranitelnosti v Joomla aplikacích. JoomScan provádí systematické skenování aplikace a analyzuje kód, soubory a konfigurace s cílem odhalit potenciální bezpečnostní slabiny.

Jedním z klíčových prvků JoomScan je jeho schopnost identifikovat známé zranitelnosti a slabiny v kódu Joomla aplikací. Nástroj prohledává soubory aplikace a porovnává je s databází

známých bezpečnostních chyb, což umožňuje rychle identifikovat potenciální rizika. Díky této funkci mohou penetrační testovací týmy prioritizovat opravy a zabezpečovací opatření podle závažnosti nalezených problémů.

Další funkcí JoomScanu je možnost provádět různé druhy testů, včetně SQL injection, cross-site scripting (XSS) a dalších typů útoků, které jsou často zneužívány v Joomla aplikacích. Nástroj umožňuje penetračním testerům detailně prozkoumat bezpečnostní postavení Joomla aplikace a identifikovat možné zranitelnosti, které by mohly být zneužity útočníky.

Celkově JoomScan přispívá ke zlepšení bezpečnosti Joomla aplikací tím, že umožňuje systematickou analýzu a identifikaci bezpečnostních rizik a zranitelností. Jeho použití umožňuje rychle a efektivně odhalit potenciální bezpečnostní nedostatky a poskytuje penetračním testovacím týmům informace potřebné k zabezpečení a ochraně Joomla aplikací.

5.3 ZAP (Zed Attack Proxy)

ZAP je open-source nástroj vyvinutý organizací OWASP (Open Web Application Security Project) pro testování bezpečnosti webových aplikací. Jedná se o multifunkční proxy server, který umožňuje penetračním testerům provádět pasivní i aktivní skenování webových aplikací s cílem identifikovat a odstranit bezpečnostní chyby a zranitelnosti.

Hlavní funkcí ZAP je jeho schopnost zachytávat a analyzovat veškerý provoz mezi webovým prohlížečem a cílovou webovou aplikací. Nástroj umožňuje penetračním testerům monitorovat a analyzovat každý HTTP požadavek a odpověď a identifikovat potenciální bezpečnostní rizika, jako jsou SQL injection, cross-site scripting (XSS) nebo nezabezpečené přenosy dat.

Kromě pasivního sledování ZAP také poskytuje funkce pro aktivní skenování webových aplikací na zranitelnosti. Tento nástroj umožňuje penetračním testerům provádět automatizované testy na různé typy bezpečnostních chyb a zranitelností a generovat podrobné zprávy o nalezených problémech. Díky tomu mohou penetrační testovací týmy rychle identifikovat a opravit bezpečnostní nedostatky ve webových aplikacích a zabezpečit je proti možným útokům.

ZAP, jakožto multifunkční nástroj pro testování bezpečnosti webových aplikací, poskytuje také funkci spideringu, která umožňuje systematické procházení webové aplikace a automatické identifikování odkazů na další stránky a adresáře. Funkce je obzvláště užitečná při mapování celkové struktury webové aplikace a identifikaci skrytých či nedostupných částí. Spidering v ZAP je navržen tak, aby byl robustní a efektivní, umožňuje nastavit různá pravidla pro procházení a filtrování odkazů, což umožňuje penetračním testerům provádět důkladné prozkoumání aplikace a identifikovat potenciální zranitelnosti nebo nezabezpečené oblasti. Integrace spideringu přímo do prostředí ZAP tak usnadňuje a zefektivňuje proces testování bezpečnosti webových aplikací a poskytuje penetračním testovacím týmům všechny potřebné nástroje pro provádění komplexních a důkladných bezpečnostních testů.

Celkově ZAP přispívá ke zlepšení bezpečnosti webových aplikací tím, že poskytuje penetračním testerům nástroje potřebné k identifikaci a odstranění bezpečnostních rizik a zranitelností. Jeho použití umožňuje penetračním testovacím týmům provádět komplexní a důkladné testování

bezpečnosti webových aplikací a poskytuje informace potřebné k zabezpečení a ochraně webových aplikací před možnými útoky.

5.4 Nikto

Nikto je další nástroj používaný k testování bezpečnosti webových aplikací, vyvinutý organizací Sullo. Jeho hlavním zaměřením je identifikace bezpečnostních rizik a zranitelností pomocí automatizovaného skenování webových aplikací. Nikto je navržen tak, aby poskytoval rychlé a efektivní skenování, které umožňuje identifikaci široké škály bezpečnostních problémů.

Hlavní funkcí Nikto je provádění pasivního i aktivního skenování webových serverů a aplikací s cílem odhalit možné bezpečnostní slabiny. Nástroj analyzuje webové stránky, skripty a konfigurační soubory a hledá známé zranitelnosti a chyby, jako jsou nezabezpečené adresáře, zastaralé verze softwaru nebo nezabezpečené konfigurace. Nikto také umožňuje testovat webové servery na různé typy útoků, včetně SQL injection, cross-site scripting (XSS) a dalších.

Díky své jednoduché konfiguraci a uživatelsky přívětivému rozhraní je Nikto oblíbeným nástrojem mezi penetračními testery a bezpečnostními profesionály. Poskytuje rychlý a efektivní způsob, jak identifikovat a odstranit bezpečnostní rizika ve webových aplikacích a zabezpečit je proti možným útokům. Nicméně, je důležité mít na paměti, že Nikto je pouze jedním z mnoha nástrojů v penetračním testerství a neměl by být používán jako jediný zdroj informací
o bezpečnosti aplikace. Kombinace různých nástrojů a metod testování je nejlepším přístupem k zajištění bezpečnosti webových aplikací.

5.5 Dirbuster

DirBuster je nástroj používaný k identifikaci skrytých adresářů a souborů na webových serverech. Jeho hlavním účelem je provádět systematické prohledávání adresářové struktury webového serveru s cílem odhalit potenciálně citlivé nebo nezabezpečené oblasti, které by mohly být zneužity útočníky.

Hlavní funkcí DirBuster je provádění slovníkových útoků na URL adresy webového serveru. Tento nástroj umožňuje uživatelům definovat vlastní slovníky nebo použít předdefinované seznamy slov k prohledávání adresářové struktury. DirBuster pak postupně testuje každou URL adresu na webovém serveru a hledá existenci adresářů a souborů na základě slov z vybraného slovníku.

Díky své schopnosti systematického prohledávání adresářové struktury webového serveru je DirBuster užitečným nástrojem pro penetrační testery při identifikaci potenciálně citlivých informací nebo skrytých funkcionalit ve webových aplikacích. Tento nástroj umožňuje rychle a efektivně identifikovat možná bezpečnostní rizika spojená se skrytými adresáři a soubory a poskytuje penetračním testovacím týmům důležité informace potřebné k zabezpečení a ochraně webových aplikací před možnými útoky.

Nicméně, je důležité zdůraznit, že v praxi byl také využit spideringový modul dostupný v nástroji ZAP (Zed Attack Proxy). Tato kombinace nástrojů umožňuje penetračním testerům provádět

důkladné prohledávání webových aplikací a identifikovat skryté adresáře a soubory, čímž přispívá k efektivnímu a komplexnímu testování bezpečnosti. Použití spideringu v ZAP poskytuje další možnost pro automatizované prohledávání aplikace a umožňuje získání uceleného obrazu o struktuře a možných zranitelnostech webové aplikace.

5.6 Wappalyzer

Wappalyzer je vysoce užitečný webový plugin pro kyberbezpečnost, který umožňuje identifikovat používané technologie na webových stránkách. Tento nástroj poskytuje cenné informace o různých technologiích, jako jsou správci obsahu (CMS), frameworky, databázové systémy, programovací jazyky, analytické nástroje a další. Je to nenahraditelný pomocník pro odborníky v oblasti bezpečnosti, vývojáře profesionály zabývající se kybernetickou bezpečností.

Jednou z hlavních funkcí Wappalyzer je automatické rozpoznávání technologií na webových stránkách pomocí analýzy zdrojového kódu. Plugin prohledává HTML, CSS, JavaScript a další zdrojové soubory a porovnává je s interní databází technologií. Jakmile identifikuje konkrétní technologie, zobrazí uživateli přehledné informace o nich přímo v prohlížeči.

Wappalyzer je nenahraditelným nástrojem nejen pro identifikaci používaných technologií na cílových webových stránkách, ale také pro sběr informací o bezpečnostních hrozbách a analýzu kybernetických rizik. Pomáhá například sledovat, které CMS jsou používány na konkurenčních webových stránkách, jaké frameworky dominují v daném odvětví nebo jaké bezpečnostní zranitelnosti mohou existovat na dané webové stránce.

5.7 SQLmap

SQLmap je open-source nástroj pro automatizovaný test SQL injection, který umožňuje penetračním testerům detekovat SQL injection zranitelnosti ve webových aplikacích. Tento nástroj je navržen tak, aby umožnil rychlé a efektivní testování bezpečnosti databází pomocí automatizovaných metod.

SQLmap dokáže automaticky detekovat SQL injection zranitelnosti v cílových webových aplikacích. Podporuje různé typy SQL injection útoků, včetně Union-based, Error-based, Blind, Time-based a dalších. SQLmap je kompatibilní s širokou škálou databázových systémů, včetně MySQL, PostgreSQL, Microsoft SQL Server, Oracle a dalších. Poskytuje detailní výstupy a informace o nalezených zranitelnostech, včetně SQL dotazů použitých k jejich detekci a exploataci.

SQLmap je nezbytným nástrojem pro penetrační testery a bezpečnostní odborníky, kteří provádějí testování bezpečnosti webových aplikací a hledají SQL injection zranitelnosti.

5.8 JSQLInjection

jSQLInjection je další open-source nástroj pro automatizované testování SQL injection zranitelností ve webových aplikacích. Tento nástroj je napsán v jazyce Java a poskytuje různé funkce pro detekci a exploataci SQL injection zranitelností. jSQLInjection umožňuje

automatizované testování SQL injection zranitelností v cílových webových aplikacích. Stejně jako sqlmap podporuje širokou škálu databázových systémů. jSQLInjection je navržen tak, aby byl snadno použitelný a poskytoval uživatelsky přívětivé rozhraní pro provádění testů SQL injection.

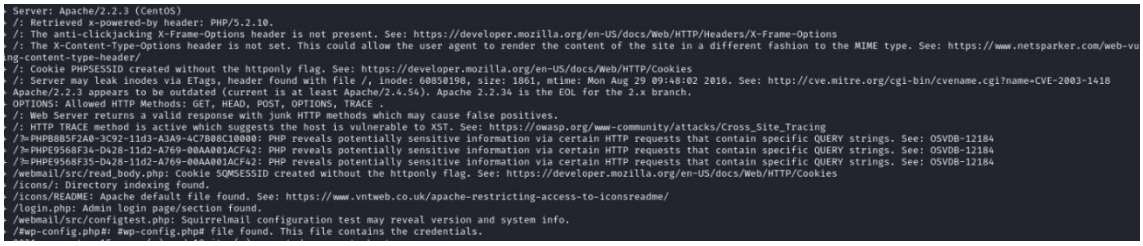
jSQLInjection je užitečným nástrojem pro penetrační testery a bezpečnostní odborníky, kteří provádějí testování bezpečnosti webových aplikací a hledají SQL injection zranitelnosti. Díky svému jednoduchému použití a široké podpoře databázových systémů je ideální volbou pro testování bezpečnosti webových aplikací postavených na různých technologiích.

6 První web

Jedná se o starší bakalářskou práci s registrem oznámení podle zákona o střetu zájmů. Přesto, že se jedná o malou stránku pouze s přihlašovacím oknem, tak zde byli nalezeny chyby. V souladu s uchováním informací nebudou domény jmenovány konkrétně namísto bude vždy doména nahrazena za domena.cz.

6.1 Nikto scan

Nástroj Nikto je využíván pro skenování a odhalování chyb v bezpečnosti webových aplikací. Pro spuštění skenu jsme použili příkaz nikto -h domena.cz, kde -h specifikuje cílovou stránku. Z tohoto skenu jsme získali několik důležitých informací.



Obr. 2: Nikto scan prvního webu

Zdroj: *OWASP.org*. (2021)

Z obrázku 2 je vidět hned několik informací, mezi ty důležité patří:

Server: Apache/2.2.3 (CentOS): Tato informace poskytuje identifikaci webového serveru a jeho verze. Zastaralá verze Apache 2.2.3 může být náchylná k různým zranitelnostem a bezpečnostním hrozbám, což zvyšuje riziko úspěšného útoku.

X-Powered-By header: PHP/5.2.10: Tato hlavička odhaluje, že webová aplikace využívá PHP verze 5.2.10. Starší verze PHP mohou obsahovat známé bezpečnostní chyby, které mohou být zneužity útočníky k provádění různých typů útoků.

Chybí anti-clickjacking hlavička X-Frame-Options: Absence této ochranné hlavičky může umožnit útočníkům vložit obsah do webové stránky pomocí clickjacking útoků, což může vést k podvodným akcím nebo krádeži citlivých informací od uživatelů.

Chybí hlavička X-Content-Type-Options: Nedostatek této hlavičky může umožnit útočníkům manipulovat s typem obsahu webové stránky a potenciálně vyvolat bezpečnostní problémy nebo útoky, jako je MIME sniffing.

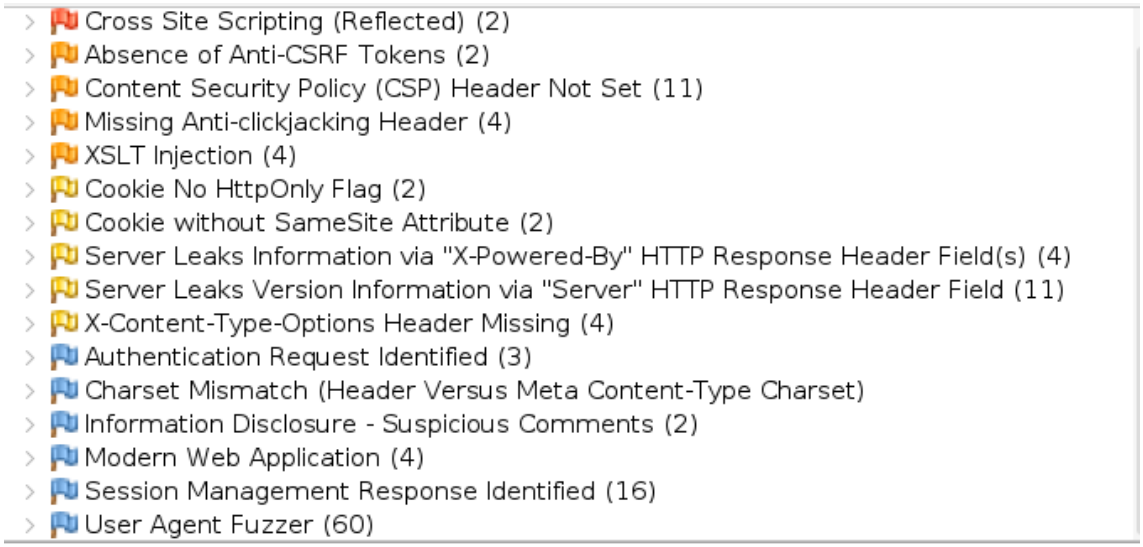
Cookie PHPSESSID byla vytvořena bez flagu httponly: Absence httponly flagu u této cookie může umožnit útočníkům přístup k obsahu cookies pomocí JavaScriptu, což zvyšuje riziko útoků a možnost neoprávněného přihlášení k účtům uživatelů především pokud použito s kombinací sociálního inženýrství.

6.2 ZAP (Zed Attack Proxy)

Nejprve jsme spustili automatický sken pomocí ZAP, který procházel webové stránky a hledal potenciální zranitelnosti. Tento proces zahrnoval identifikaci typických bezpečnostních chyb, jako jsou SQL injection, Cross-Site Scripting (XSS), chybějící zabezpečení cookies a další. Automatické skenování poskytlo rychlý přehled o možných zranitelnostech a umožnilo nám identifikovat potenciálně problematické oblasti webové aplikace.

Po dokončení automatického skenování jsme přešli k manuálnímu prozkoumání zjištěných oblastí. Prozkoumali jsme každý alert individuálně, abychom porozuměli jeho kontextu a závažnosti. Během tohoto procesu jsme se zaměřili na ověření a potvrzení nalezených zranitelností a posouzení jejich reálného rizika.

Díky kombinaci automatického skenování a manuálního prozkoumání pomocí ZAP jsme byli schopni identifikovat a vyhodnotit řadu zranitelností ve webové aplikaci. Tento přístup nám umožnil získat komplexní pohled na bezpečnostní stav aplikace a identifikovat klíčové oblasti, které vyžadují pozornost a opravy.



Obr. 3: ZAP Alerty pro první web

Zdroj: *vlastní práce*

Na obrázku 3 jsou vidět všechny nálezy od ZAP. U některých nálezech, především těch označených modře jako “informational”, se dá říct, že nejsou nějak důležité, ale přesto je dobré se na ně vždy podívat, jestli neobsahují nějaké zajímavé informace o testovaném prostředí. Z těchto nálezů mezi důležité a potvrzené nálezy patří:

Cross Site scripting (reflected): Tento alert naznačuje možnost Cross-Site Scripting (XSS) útoku přes reflexi. Konkrétně bylo zjištěno, že stránka a parametr http://domena.cz/login.php?accesscode= je náchylný k XSS útoku, kdy útočník může vložit škodlivý kód do stránky a tím ohrozit uživatele.

Použitý útok je: http://domena.cz/login.php?accesscode="><script>alert(1);</script> kde "><script>alert(1);</script> je samotný XSS útok.

Obr. 4: Ukázka XSS Zranitelnosti

Zdroj: *vlastní práce*

Cross-Site Scripting (XSS) je typ útoku na webové aplikace, který umožňuje útočníkovi vložit a spustit škodlivý kód (nejčastěji JavaScript) do webové stránky zobrazené uživatelům. XSS útok je často využíván k získání citlivých informací, přesměrování uživatelů na podvodné webové stránky nebo převzetí kontroly nad uživatelským účtem. XSS může být kategorizován jako "reflected" (očekávaný vstup, který je zobrazen zpět na stránce) nebo "stored" (útočníkův kód je uložen na serveru a je zobrazen každému, kdo navštíví nezabezpečenou stránku).CSP, která definuje, které zdroje mohou být na stránce načítány, což může minimalizovat riziko útoku XSS. (Portswinger)

Pro ochranu před XSS útoky je vhodné provádět úpravu vstupních dat například filtrováním, validací a vhodným zpracováním dat. V následujích úryvku PHP kódu je návrh jak by se XSS zranitelnosti dalo předejít.

```
<?php
$accesscode = isset($_GET['accesscode']) ? htmlspecialchars($_GET['accesscode']) : ''; ?>
```

V příkladu je proměnná $accesscode ošetřena funkcí htmlspecialchars, což zajistí, že jakákoli potenciálně škodlivá data budou převedena na jejich bezpečnou HTML reprezentaci, čímž se minimalizuje riziko XSS útoku. Toto ošetření je důležité pro veškerá data získaná z uživatelského vstupu, která jsou následně zobrazena na stránce.

Absence of Anti-CSRF Tokens: Tento alert upozorňuje na nepřítomnost Anti-CSRF tokenů. Formulář na stránce obsahuje akci "login.php?accesscode=", která není chráněna CSRF tokenem. Absence CSRF tokenu je zranitelnost, která může umožnit útočníkovi provádět Cross-CSRF útoky se zaměřují na vykonání neautorizovaných akcí uživatelem, který je přihlášen do cílového účtu. K tomu dochází, když útočník vytvoří požadavek, který vypadá jako legitimní, a přesvědčí uživatele, aby ho spustil (například kliknutím na odkaz nebo odesláním formuláře). V případě, že webová aplikace neprovádí dostatečnou kontrolu pravosti požadavků pomocí Anti-CSRF tokenů, může být zranitelnost využita k provedení škodlivých akcí, jako je změna nastavení účtu, odeslání falešných platby nebo smazání dat. (Invicti)

Proti CSRF útokům je vhodné používat náhodně generované a unikátní tokeny, které jsou součástí každého formuláře nebo požadavku. V následujícím příkladu je ukázka jak by se CSRF útokům dalo předejít.

```
<form action="login.php" method="post">
  <input type="hidden" name="csrf_token" value="<?php echo generate_csrf_token(); ?>">
  <!-- Další potřebné okna formáře -->
  <button type="submit">Submit</button>
</form>
```

Kód generuje náhodně generovaný token pomocí funkce generate_csrf_token() a vloží ho do skrytého pole formuláře. Při zpracování formuláře na serveru se tento token ověří s tokenem uloženým v relaci nebo jiném úložišti, což zabrání CSRF útokům.

Funkce generate_csrf_token není základní funkcí php a proto si jí musí vývojáři nebo administrátoři vytvořit sami. Zde je jednoduchý návrh jak by funkcem mohla vypadat.

```
<?php
function generate_csrf_token() {
  return bin2hex(random_bytes(32));
}
```

Zmíněný kód vytvoří náhodný token o délce 64 znaků (každý znak pak reprezentuje 4 bity).

CSP Header not set: Tento alert indikuje, že na stránce není nastavena Content Security Policy (CSP) hlavička. CSP je bezpečnostní mechanismus, který pomáhá chránit webové aplikace proti XSS útokům a dalším bezpečnostním hrozbám omezováním, které zdroje mohou být načteny na stránce. Nastavení CSP umožňuje omezit, které zdroje jsou povoleny pro konkrétní webovou stránku, což může snížit riziko úspěšných útoků. (Scanrepeat)

```
Content-Security-Policy: default-src 'self'
```

Úryvek nastavuje Content Security Policy (CSP) pomocí HTTP hlavičky nebo přímo v HTML stránce. CSP definuje, odkud mohou být načítány zdroje (např. skripty, styly) a jakým způsobem mohou interagovat s obsahem stránky, což pomáhá chránit před různými druhy útoků, jako je třeba XSS.

Missing Anti-clickjacking header: Tento alert upozorňuje na nepřítomnost anti-clickjacking hlavičky. Absence této ochrany může umožnit útočníkům provádět clickjacking útoky, kde se obsah jiné stránky zobrazuje v iframe na cílové stránce, čímž může vést k uživatelskému klamání nebo zneužití. Použitím X-Frame-Options header s hodnotou "DENY" nebo "SAMEORIGIN" může zabránit útokům clickjacking tím, že omezí, jak může být stránka vložena v rámci jiného webu. (Iothread)

```
X-Frame-Options: DENY
```

Tímto přidáme X-Frame-Options header s hodnotou "DENY", což zakazuje vložení stránky do rámce <frame>, <iframe> nebo její zobrazení v iframe jiného webu.

Cookie no httpOnly flag: Tento alert indikuje, že některé cookies na stránce neobsahují httpOnly flag. To znamená, že tyto cookies mohou být zranitelné vůči útokům na prostředí, protože mohou být přístupné skriptům JavaScript na stránce. Přidání httponly flagu k cookies může zabránit přístupu k obsahu cookies pomocí skriptů JavaScript a snížit tak riziko útoků na prostředí. (BeagleSecurity, 2018)

```
setcookie("session_id", $session_id, [
   'expires' => time() + 86400,
   'path' => '/',
   'domain' => našedoména.cz',
   'secure' => true,
   'httponly' => true,
   'samesite' => 'Strict'
]);
```

PHP Kód nastavuje httponly flag k cookies vytvořených na straně serveru, což brání přístupu k obsahu cookies pomocí JavaScriptu, což snižuje riziko útoků na sezení.

Cookie Without SameSite attribute: Tento alert upozorňuje na nepřítomnost SameSite atributu u některých cookies. SameSite atribut pomáhá chránit proti CSRF a některým dalším útokům tím, že omezuje, kdy jsou cookies odesílány při cross-site požadavcích. Nastavení atributu SameSite na "Strict" nebo "Lax" může snížit riziko útoků tím, že omezí, jak jsou cookies odesílány v případě cross-site požadavků. (StackHawk)

```
setcookie("session_id", $session_id, [
  ...
  'samesite' => 'Strict'
]);
```

Server leaks Info via X-Powered-By and via Server HTTP Response header field: Tento alert označuje únik informací ze serveru skrze X-Powered-By a Server HTTP Response header field. Tato informace by mohla být využita útočníky k identifikaci použitého softwaru a jeho verzí, což může zvýšit riziko úspěšného útoku, pokud jsou známy známé zranitelnosti tohoto softwaru. Ideálním řešením je vypnutí zobrazování serverových informací v HTTP hlavičkách což může snížit riziko útoků tím, že omezí dostupnost citlivých informací. (StackHawk)

```
ServerTokens Prod // Pro Apache
ServerSignature Off // Pro Nginx
```

X content type options Header missing: alert nás upozorňuje na nepřítomnost X-Content-Type-Options hlavičky. Tato hlavička pomáhá chránit proti MIME type sniffing útokům tím, že určuje, jaký typ obsahu může být vykreslen v prohlížeči, a tím snižuje riziko úspěšného útoku pomocí manipulace s typem obsahu. Použití X-Content-Type-Options header s hodnotou "nosniff" může zabránit prohlížeči ve snaze měnit MIME typy a snížit tak riziko útoků. (StackHawk)

```
Header set X-Content-Type-Options: nosniff
```

MIME type sniffing, také známý jako content-type sniffing nebo media type sniffing, je proces, který webový prohlížeč používá k identifikaci typu obsahu souboru na základě jeho obsahu, i když je mime type explicitně určen v HTTP hlavičce. Webové prohlížeče toto občas používají, když server neodešle správnou MIME type hlavičku nebo pokud je hlavička chybějící.

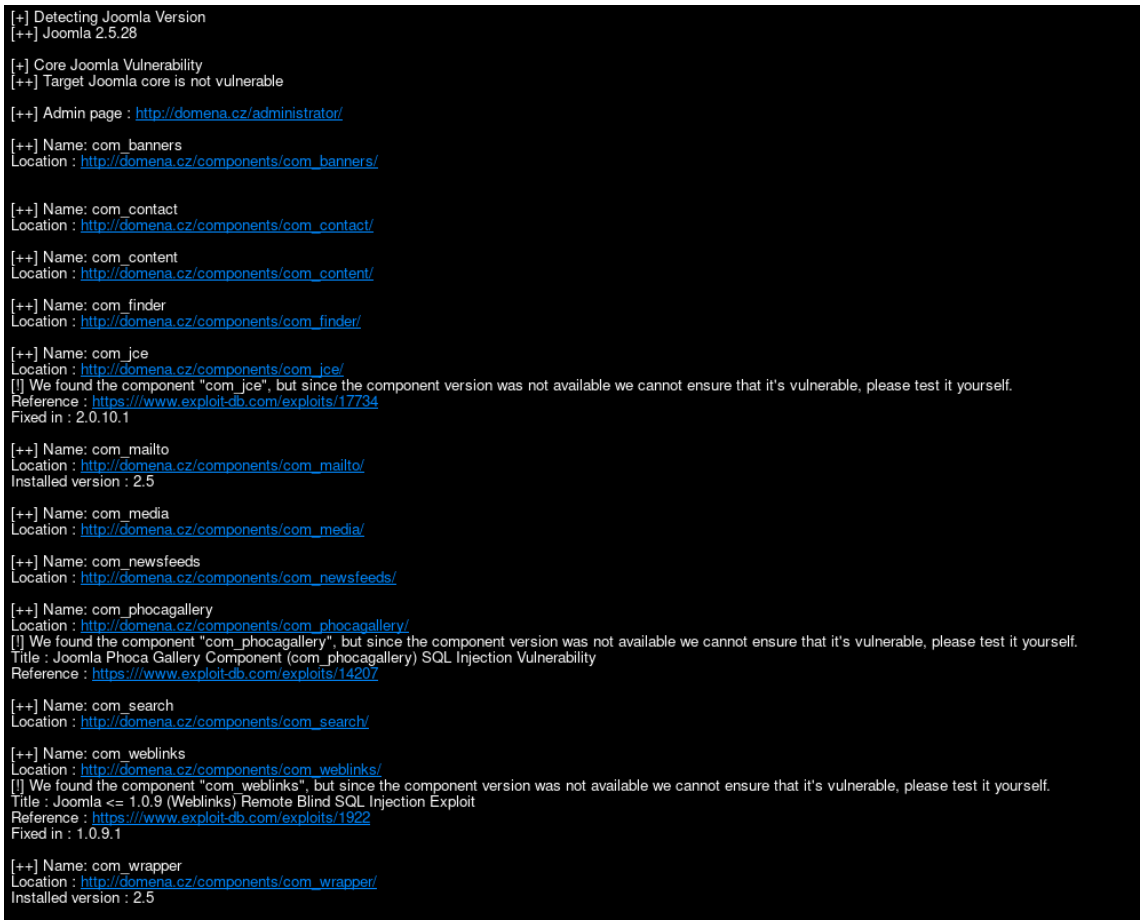
MIME type sniffing může být užitečné, když je soubor poskytnutý serverem špatně označen, což může vést k lepšímu zobrazení nebo interpretaci obsahu souboru pro uživatele. Nicméně, toto chování může také otevřít dveře k bezpečnostním rizikům, protože webové prohlížeče mohou zkoumat obsah souboru a interpretovat jej jinak, než bylo zamýšleno. (Keycdn, 2023)

7 Druhý web

Nyní provedeme se podíváme na test druhé stránky, pomocí webového pluginu wappalyzer jsme zjistili, že se jedná o stránku postavenou na Joomla CMS, dále jsme zjistili, že používá MooTools JavaScript framework a knihovnu jQuery. Dále jsme zjistili, že jako u předchozí stránky, je web hostovaný na CentOS, používá PHP 5.2.10 a Apache 2.2.3. Ve zdrojovém kodu stránky jsme následně nalezli komentář "<!-- Created by Artisteer v4.0.0.58475 -->", který zobrazuje informaci o softwaru a jeho verzi, který byl použit při vytváření šablony webu. Jelikož se tentokrát jedná o Joomla stránku, tak využijeme i další nástroj joomscan o kterém se piše výše v sekci použité nástroje.

7.1 Joomscan

Jelikož výstup z joomscan je v tomto případě dlouhý tak bude vybráno pouze to důležité.



Obr. 5: Joomscan druhého webu

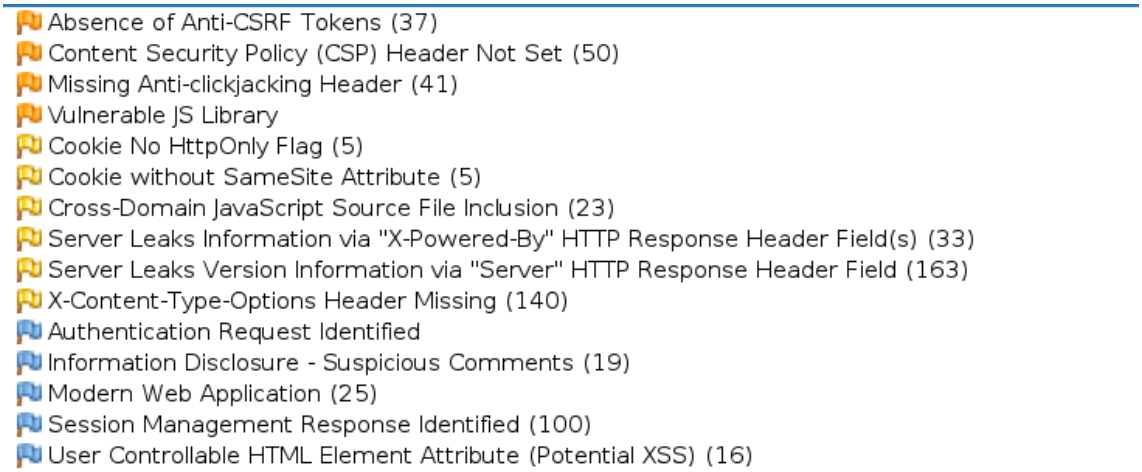
Zdroj: *vlastní práce*

Na obrázku je vidět hned několik důležitých informací, zjistili jsme verzy joomly, že není zranitelná, administrátorskou stránku, která nepodléhá injection útokům a podařilo se nám

zjistit jaké pluginy stránka používá. U tří nainstalovaných pluginů se našli zranitelnosti, ale jelikož neznáme jejich verze, musíme manuálně otestovat zda tuto zranitelnost najdeme. Začali jsme prvním ohroženým pluginem a to "com_jce", ten se nedokázalo potvrdit, jelikož funguje pouze v Image, Media, Template a File manager sekcích, které se nepodařilo najít. Pak jsme se přesunuli na druhý ohrožený plugin a to "com_phocagallery". Verze 2.7.3 tohoto pluginu by měla mít SQL Injection zranitelnost pomocí: "index.php?option=com_phocagallery&view=categories&Itemid=injection. Tuto zranitelnost jsme otestovali třemi způsoby, a to manuálně a pomocí nástrojů sqlmap a jSqlInjection, ani jeden způsob neodhalil SQL Injection zranitelnost a proto se posuneme dál na plugin "com_weblinks". Tento plugin by měl být zranitelný na blind SQL injection útok z roku 2006 a to pomocí php scriptu na stránce exploid-db.com, tento script se boužel nepodařilo zprovoznit a tudiž nemůžeme tuto instalaci považovat za zranitelnou. Obecně se doporučuje vždy aktualizovat na nejnovější verze aby se předešlo vzniku případným zranitelnostem.

7.2 ZAP (Zed Attack Proxy)

Podíváme se opět do nástroje ZAP, kde jsme spustili automatické skenování a později i spider pro zjistění skrytých souborů a adresář díky kterému jsme zjistili pár uživateli nepřístupných částí stránky. A to https://domena.cz/67, kde jsme našli stránku s informacemi o joomle 1.5, jelikož víme, že tato joomla má verzi 2.5.28 tak se jedná pouze o informativní záležitost nikoliv zranitelnost. Dále http://domena.cz/73fe na této adrese se nechází kopie stránky nabízíme, opět pouze informativní. Na stránce /75 potom kopie vozový park. Na 76 potom stránku o nabídce práce, která není uživateli jinak dostupná pomocí klasických odkazů. Na /77 potom stránka s blogem lorem ipsum, také uživateli nedostupná normálními způsoby. /78 je stránka se souborem ke stažení a poslední /79 je potom kopie stránky kontakty. Zatímco stránky neobsahují zranitelnosti, nemusí být vhodné mít desítky uživateli nepřístupných stránek případně kopií stránek. V případě kdyby se v některé kopii nacházely nějaké vstupní prvky, které v této instanci nejsou zabezpečeny, vystavuje se stránka zvýšenému nebezpečí.



Obr. 6: ZAP alerty pro druhý web

Zdroj: *vlastní práce*

Nyní se přesuneme zpět k alertům, kde jich z obrázku 9 vidíme 15. Většina z nich již byla vypsána a vyřešena v předchozím webu a tudiž je přeskočíme. Upozornění, která jsme neřešili jsou: **Vulnerable JS library - jquery, version 1.7.1.** Je to alert, který upozorňuje na zranitelnou verzi knihovny jQuery, konkrétně verze 1.7.1, která je používána na webové stránce. Knihovna jQuery je často používaná a zajišťuje různé funkce pro manipulaci s obsahem stránky, animace, události a AJAX požadavky. Obsahuje všechny tyto CVE: CVE (Common Vulnerabilities and Exposures) jsou unikátní identifikátory pro známé bezpečnostní chyby. V tomto případě je knihovna jQuery verze 1.7.1 zranitelná a obsahuje několik CVE identifikátorů:

CVE-2020-11023, CVE-2020-11022, CVE-2015-9251, CVE-2019-11358, CVE-2020-7656 a CVE-2012-6708

Každý z těchto identifikátorů označuje konkrétní bezpečnostní chybu, která byla nalezena ve verzi jQuery 1.7.1. Některé z těchto chyb mohou být závažné a umožnit útočníkovi provést různé druhy útoků, jako je například Cross-Site Scripting (XSS), únik informací nebo vzdálené vykonání kódu. Je důležité aktualizovat tuto knihovnu na nejnovější a bezpečnější verzi, která neobsahuje tyto zranitelnosti, aby byla zajištěna bezpečnost webové aplikace.

Cross-Domain JavaScript Source File Inclusion: Tento alert upozorňuje na možnost vložení externího JavaScriptového souboru z jiné domény do HTML kódu stránky. Tato chyba je také známá jako Cross-Domain Script Inclusion a je způsobena nebezpečným použitím funkce <script src="">. (StackHawk)

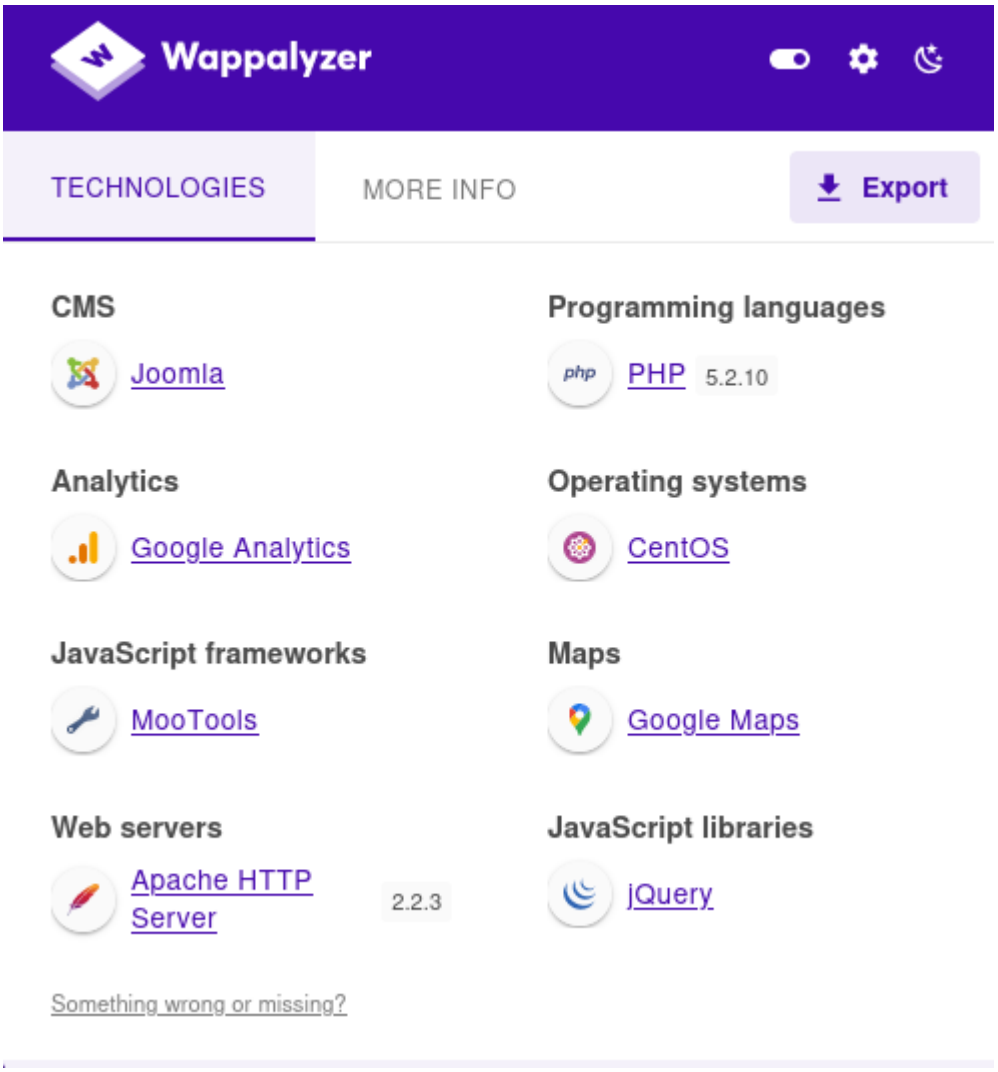
Je také klasifikován pod OWASP kódem A08.
<script src=" http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>

Tento kód načítá externí JavaScriptový soubor z domény html5shiv.googlecode.com. Pokud útočník dokáže ovládat obsah tohoto souboru, může vložit škodlivý kód do webové stránky, což může vést k různým druhům útoků, včetně Cross-Site Scripting (XSS) nebo zneužití uživatelských relací.

Je důležité používat pouze důvěryhodné zdroje JavaScriptových souborů a vyhnout se načítání souborů z neověřených domén, aby se minimalizovala rizika bezpečnostních hrozeb.

8 Třetí web

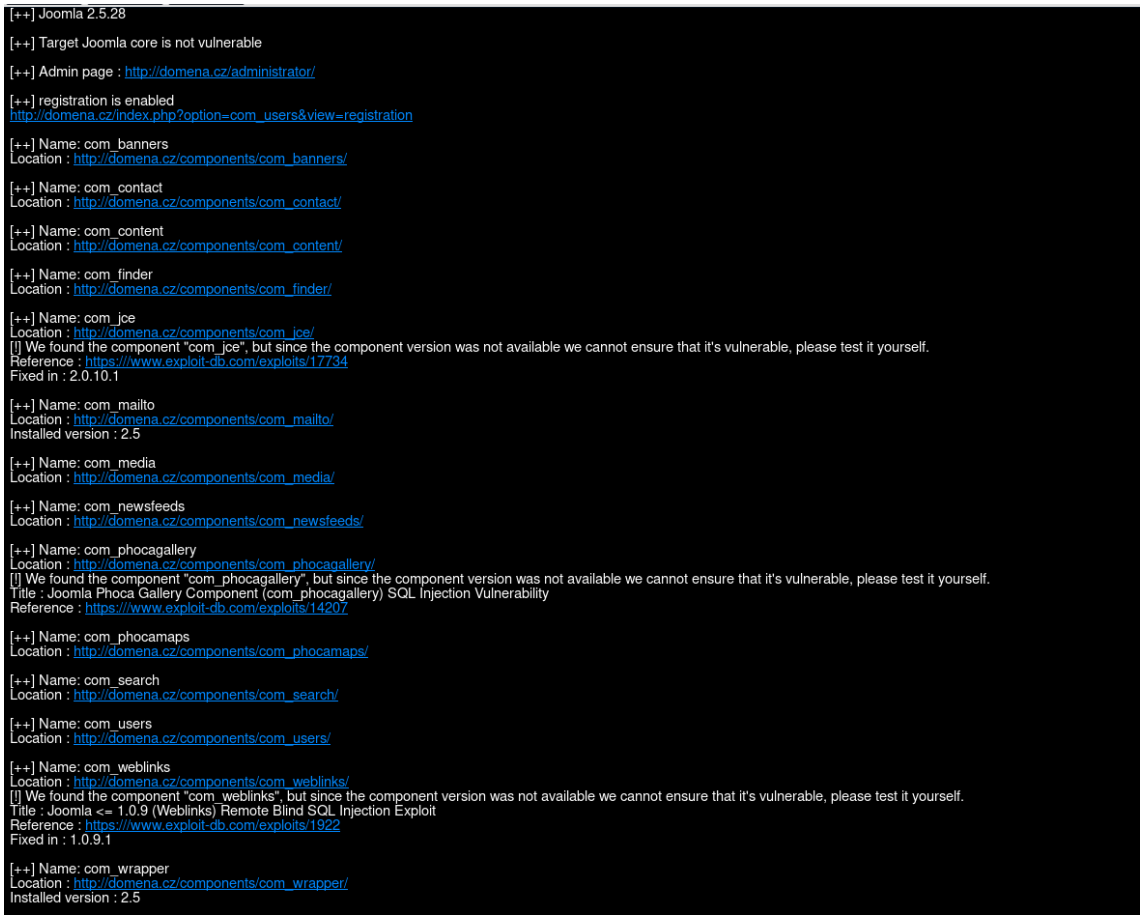
Tento web je také postaven na Joomla CMS, opět jsme jako první provedli rychlou kontrolu přes prohlížečový plugin Wappalyzer, kde jsme zjistili, že web využívá Joomlu, Gogle Analytics a Maps, a další informace, jelikož jsme se nedozvěděli moc dalších informací tak použijeme další nástroje, opět zase začneme nástrojem joomscan a poté se podíváme zpět do ZAPu a projdeme si další upozornění.



Obr. 7: Wappalyzer třetího webu

Zdroj: *vlastní práce*

8.1 Joomscan



Obr. 8: Joomscan třetího webu

Zdroj: *vlastní práce*

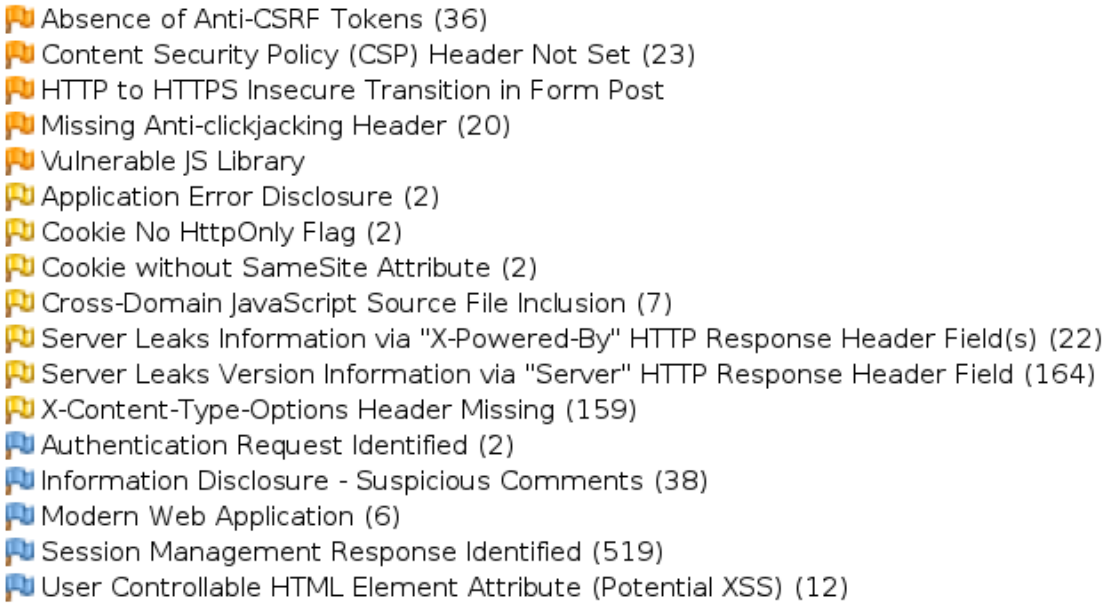
Na obrázku můžeme vidět, že její verze 2.5.28, Core neobsahuje žadné zranitelnosti, nalezli jsme administrátorskou stránku, která je nepřístupná uživateli, a vrátí nás na domovskou stránku. Dále jsme ale narazili na stránku s možnou registrací, jelikož stránka neobsahuje žádný blog, ale je to jen stránka o možném ubytování, tudíž by stránka s registrací neměla být veřejně dostupná. Stránka opět využívá několika pluginů a 3 z nich měli ve specifikovaných verzích určitou zranitelnost.

com_jce – na tuto komponentu jsme již narazili v předchozím webu a proto víme, že potřebujeme Image, Media, Template nebo File Manager sekce, tyto sekce se nepodařilo nalézt proto v tuto chvilí nemužeme říci, že je komponenta zranitelná.

com_phocagallery – s tímto jsme se také setkali v předchozí ukázce, mělo by se jednat o SQL Injection zranitelnost, použijeme tedy opět sqlmap a jSQLInjection nástroje, díky kterým test můžeme automatizovat. Sqlmap nám prozdrail pouze, že web je chráněn nějakým WAF (Web Application Firewall" a dle jSQLinjection, který prošel stovkami pokusů došel k závěru, že SQL Injection možná není.

com_weblinks – je opět známá komponenta, boužel jelikož se jedná o stejný exploit se stejným PHP scriptem, který nejde spustit tak nemůžeme otestovat.

8.2 ZAP



Obr. 9: ZAP alerty třetího webu

Zdroj: *vlastní práce*

Z obrázku výpisu upozornění můžeme vidět 18 upozornění, s většinou jsme se už seznámili a tudiž je nebudeme zase rozebírat. Ale podívame se na alert Vulnerable JS Library, jak již víme jedná se o jQuery knihovnu tentokrát ve verzi v1.4.2, tato verze je ještě starší než jsme se setkali v předchozí ukázce. Tato stará verze má hned několik CVEs a to CVE-2011-4969, CVE-2020-11023, CVE-2020-11022, CVE-2015-9251, CVE-2019-11358, CVE-2020-7656 a CVE-2012-6708.

HTTP to HTTPS Insecure Transition in Form Post: Označuje situaci, kdy je formulář odesílán z HTTPS stránky na HTTP adresu. To znamená, že citlivá data z formuláře jsou přenášena po nezabezpečené cestě, což vytváří riziko zachycení nebo úpravy těchto dat útočníkem. Chyba Spadá do kategorie OWASP_2021_A02. Nejefektivnějším řešením je zajistit, aby veškeré formuláře byly odesílány pomocí HTTPS. To znamená, že adresa v atributu action formuláře by měla začínat https:// namísto http://. (Iothread)

Další možností je aktivace HSTS: Implementace HTTP Strict Transport Security (HSTS) politiky na serveru pomůže zabezpečit, že prohlížeč automaticky použije HTTPS při komunikaci se serverem, aniž by se nejprve pokusil o HTTP spojení.

Strict Transport Security Header Not Set: Upozorňuje nás na skutečnost, že stránka neposkytuje HTTP Strict Transport Security (HSTS) hlavičku. HSTS zajišťuje, že prohlížeč při komunikaci s webovou stránkou vždy použije protokol HTTPS, což pomáhá chránit uživatele před útoky typu

man-in-the-middle a proti přesměrování na nezabezpečené verze stránek. Chyba HSTS spadá v tabulace OWASP TOP 10 na páté místo.

Přidáním HSTS Hlavičky by webový server měl poskytovat HTTP Strict Transport Security (HSTS) hlavičku v odpovědi na požadavky klientů a měla by indikovat, že pro danou doménu a její poddomény by měl prohlížeč vždy používat HTTPS. Je důležité nastavit vhodnou délku platnosti HSTS hlavičky, aby byla účinná a zároveň minimalizovala možné problémy s nefunkčním přístupem k serveru.

(StackHawk)
Zde je návrh na vyřešení problému:

Header always set Strict-Transport-Security: max-age=31536000; includeSubDomains

max-age je klíčový parametr, který určuje, jak dlouho má prohlížeč uložit informaci o tom, že daná doména podléhá HSTS politice. Hodnota je udávána v sekundách. Například hodnota 31536000 odpovídá jednomu roku.

IncludeSubDomains je volitelný parametr a indikuje, zda by měla být HSTS politika aplikována i na všechny poddomény dané domény. Pokud je tento parametr přítomen a má hodnotu true, pak platí HSTS politika nejen pro aktuální doménu, ale i pro všechny její poddomény.

9 Závěr

V závěrečné části bakalářské práce jsme prozkoumali a analyzovali různé aspekty kybernetické bezpečnosti s důrazem na současné hrozby a zranitelnosti. Teoretická část nám poskytla hlubší pohled na některá konkrétní kybernetická rizika, jako jsou zero-day útoky, Remote Code Execution (RCE), malware, phishing a sociální inženýrství.

V rámci metodologie penetračního testování jsme studovali několik standardů a přístupů, jako je Open Source Security Testing Methodology Manual (OSSTMM), OWASP Testing Guide a Penetration Testing Execution Standard (PTES). Tyto metodologie nám poskytly strukturovaný rámec pro provedení penetračních testů, ať už interně nebo externě.

Ve čtvrté kapitole jsme se podrobněji zaměřili na OWASP TOP 10, což jsou deset nejvýznamnějších bezpečnostních rizik pro webové aplikace. Každá z těchto kategorií, jako Broken Access Control, Cryptographic Failures, Injection, Insecure Design, Security Misconfiguration, Vulnerable and Outdated Components, Identification and Authentication Failures, Software and Data Integrity Failures, Security Logging and Monitoring Failure a Server-Side Request Forgery, byla detailně popsána včetně praktických dopadů na webové prostředí.

V praktické části naší práce jsme se zaměřili na konkrétní webové prostředí, kde jsme využili nástroje pro skenování a zjišťování informací. Prováděli jsme penetrační testy a identifikovali zranitelnosti. Získané poznatky nám umožnily navrhnout konkrétní bezpečnostní opatření a řešení, která povedou k eliminaci identifikovaných rizik a zvýšení odolnosti testovaného webového prostředí vůči kybernetickým hrozbám.

Celkově lze konstatovat, že cíle práce byly úspěšně naplněny. Přínos spočívá nejen v identifikaci konkrétních bezpečnostních rizik pro testované webové prostředí, ale také v návrhu opatření, která posílí kybernetickou bezpečnost jako celek. Práce nabízí přínosné poznatky a doporučení nejen pro konkrétní webové prostředí, ale i pro obecný kontext kybernetické bezpečnosti.

Seznam použité literatury

Albanian Hacking Crew. (Červenec, 2010). Joomla! Component Phoca Gallery 2.7.3 - SQL Injection. In: Albanian Hacking Crew. Exploit Database: A comprehensive archive of computer security vulnerabilities. [online]. Dostupné z: https://www.exploit-db.com/exploits/14207

AmnPardaz Security Research & Penetration testing Group. (2011). Joomla! Component joomlacontenteditor 2.0.10 - Multiple Vulnerabilities. In: AmnPardaz Security Research & Penetration testing Group. Exploit Database: A comprehensive archive of computer security vulnerabilities. [online]. Dostupné z: https://www.exploit-db.com/exploits/17734

Iothread. HTTPS to HTTP Insecure Transition in Form Post. In: Iothread. Iothreat: Cybersecurity Blog [online]. Dostupné z: https://www.iothreat.com/blog/https-to-http-insecure-transition-in-form-post

Iothread. Missing Anti clickjacking header. In: Iothread. Iothreat: Cybersecurity Blog [online]. Dostupné z: https://www.iothreat.com/blog/missing-anti-clickjacking-header

Invicti. (Květen, 2023). How to protect your websites and web apps with anti-CSRF tokens. In: Invicti. Invicti Security Blog [online]. Dostupné z: https://www.invicti.com/blog/web-security/protecting-website-using-anti-csrf-token/

Jijith Rajan, BeagleSecurity. (červen, 2018). Cookie session without 'HttpOnly' flag. In: BeagleSecurity. BeagleSecurity Blog [online]. Dostupné z: https://beaglesecurity.com/blog/vulnerability/cookie-session-without-httponly-flag.html

Keycdn. (2023, duben) What is mime sniffing. In: Keycdn. Keycdn Support: Knowledge Base [online]. Dostupné z: https://www.keycdn.com/support/what-is-mime-sniffing

Kaspersky Lab. (Datum publikování není uvedeno). Zero-day exploit. In: Kaspersky Lab. Kaspersky Resource Center [online]. [cit. 2023-11-25]. Dostupné z: https://www.kaspersky.com/resource-center/definitions/zero-day-exploit

Lutkevich, B. (2023, únor). Remote Code Execution. In: TechTarget. SearchWindowsServer: Remote Code Execution (RCE) [online]. (cit. 2023-11-25). Dostupné z: https://www.techtarget.com/searchwindowsserver/definition/remote-code-execution-RCE

Malwarebytes. (2023, září). Malware. In: Malwarebytes. Malwarebytes Blog [online]. (cit. 2023-11-25). Dostupné z: https://www.malwarebytes.com/malware

Nist.org. GUIDE TO INFORMATION SECURITY TESTING AND ASSESSMENT. In: Nist.org. National Institute of Standards and Technology [online]. [cit. 2023-12-30]. Dostupné z: https://csrc.nist.rip/library/NIST%20SB%202008-12%20Guide%20To%20Information%20Security%20Testing%20And%20Assessment.pdf

OWASP. (Datum publikování není uvedeno). OWASP Top Ten. In: OWASP. Open Web Application Security Project [online]. [cit. 2023-10-22]. Dostupné z: https://owasp.org/www-project-top-ten/

OWASP Organization. OWASP Testing Guide. In: OWASP. Open Web Application Security Project [online]. [cit. 2023-12-30]. Dostupné z: https://owasp.org/www-project-web-security-testing-guide/assets/archive/OWASP_Testing_Guide_v4.pdf

Portswigger. Reflected XSS. In: Portswigger. PortSwigger Web Security: Online Academy [online]. Dostupné z: https://portswigger.net/web-security/cross-site-scripting/reflected

Herzog, P. & ISECOM. (Datum publikování není uvedeno). The Open Source Security Testing Methodology Manual. In: ISECOM. The Institute for Security and Open Methodologies. [online]. [cit. 2023-12-30]. Dostupné z: https://www.isecom.org/OSSTMM.3.pdf

PTES Team. Pentest-standard PTES. In: PTES Team. Pentest-standard PTES [online]. [cit. 2024-04-01]. Dostupné z: https://buildmedia.readthedocs.org/media/pdf/pentest-standard/latest/pentest-standard.pdf

Scanrepeat. Content Security Policy (CSP) Header Not Set. In: Scanrepeat. Scanrepeat: Web Security Knowledge Base [online]. [cit. 2024-04-01]. Dostupné z: https://scanrepeat.com/web-security-knowledge-base/content-security-policy-csp-header-not-set

SecureOPS. (2021, červen). Everything You Need to Know About Penetration Testing. In: SecureOPS. SecureOPS Blog [online]. [cit. 2024-04-01]. Dostupné z: https://www.secureops.com/wp-content/uploads/2021/06/SecureOps-Types-of-Penetration-Testing-v6.pdf

StackHawk. Strict-Transport-Security Header Not Set. In: StackHawk. StackHawk Documentation [online]. [cit. 2024-04-01]. Dostupné z: https://docs.stackhawk.com/vulnerabilities/10035/

StackHawk. Cross-Domain JavaScript Source File Inclusion. In: StackHawk. StackHawk Documentation [online]. [cit. 2024-04-01]. Dostupné z: https://docs.stackhawk.com/vulnerabilities/10017/

StackHawk. x-content-type-options-header-missing. In: StackHawk. StackHawk Documentation [online]. [cit. 2024-04-01]. Dostupné z: https://docs.stackhawk.com/vulnerabilities/10021/

StackHawk. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s). In: StackHawk. StackHawk Documentation [online]. [cit. 2024-04-01]. Dostupné z: https://docs.stackhawk.com/vulnerabilities/10037/

StackHawk. Cookie Without SameSite Attribute. In: StackHawk. StackHawk Documentation [online]. [cit. 2024-04-01]. Dostupné z: https://docs.stackhawk.com/vulnerabilities/10054/