VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

LABORATORNÍ ÚLOHA SEZNAMUJÍCÍ STUDENTY SE SÍŤOVÝMI ÚTOKY
LABORATORY EXERCISE THAT PRESENTS NETWORK ATTACKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

Adam Dostál

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Zdeněk Martinásek, Ph.D.

BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Teleinformatika**
Ústav telekomunikací

Student: Adam Dostál ***ID:*** 164804
Ročník: 3 ***Akademický rok:*** 2015/16

NÁZEV TÉMATU:

Laboratorní úloha seznamující studenty se síťovými útoky

POKYNY PRO VYPRACOVÁNÍ:

Hlavním cílem bakalářské práce je navrhnout a vypracovat laboratorní úlohy seznamující studenty s penetračním testováním webových aplikací. Teoretická část bud obsahovat popis současného stavu problematiky (metodika, OWASP atd.). Dílčími úkoly je volba vhodných nástrojů k vytvoření laboratorní úlohy (Nessus, BackTrack/Kali) včetně realizace virtuální infrastruktury. Konkrétními výstupy práce budou komplexní laboratorní návody seznamující studenty se základními praktikami penetračního testování webových aplikací. Prezentujte nejméně tři útoky z OWASP top 10 (názorná prezentace využití nalezené zranitelnosti).

DOPORUČENÁ LITERATURA:

[1] STALLINGS, William. Cryptography and network security: principles and practice. Seventh edition. 731 pages. ISBN 01-333-5469-5.

[2] FADYUSHIN, Vyacheslav a Bruce HYSLOP. Instant penetration testing: Setting up a test lab how-to. 1. vyd. Birmingham: Packt Publishing, 2013, 74 s. ISBN 978-1-84969-412-4.

Termín zadání: 1.2.2016 ***Termín odevzdání:*** 1.6.2016

Vedoucí práce: Ing. Zdeněk Martinásek, Ph.D.
Konzultant bakalářské práce:

doc. Ing. Jiří Mišurec, CSc., *předseda oborové rady*

ABSTRAKT

Tato práce je zaměřena na penetrační testování webových aplikací. Teoretická část popisuje tuto problematiku a metodologii. V práci je zahrnuta bezpečnostní organizace *„The Open Web Application Security Project“* (OWASP), dokument OWASP Top 10 a prvních 5 zranitelností tohoto dokumentu. Poslední část uvádí linuxovou distribuci Kali Linux a několik nejpoužívanějších penetračních nástrojů.
Praktickou část tvoří testování prvních pěti zranitelností z dokumentu OWASP Top 10 2013. Obsahuje popis použitého SW pro realizaci útoků, virtuální infrastruktury a test jednotlivých zranitelností. Z praktické části je vytvořena laboratorní úloha „Penetrační testování webových aplikací“ a dodatečná úvodní úloha „Úvod do problematiky penetračního testování“.

KLÍČOVÁ SLOVA

Penetrační testování, zranitelnost, bezpečnost, OWASP, Kali Linux, webová aplikace, SQL.

ABSTRACT

This work is focused on penetration testing of web applications. The theoretical part describes this issue and methodology. The work includes security organization *"The Open Web Application Security Project"* (OWASP), document OWASP Top 10 and the first 5 vulnerabilities of this document. The last part introduces linux distribution Kali Linux and the several most used penetration tools.
The practical part consists of testing the first five vulnerabilities in the document OWASP Top 10 2013. It contains a description of the used SW for the realization of the attacks, virtual infrastructure and test of each vulnerabilities. From the practical part is created laboratory task "Penetration testing of web applications" and additional introductory task "Introduction into penetration testing".

KEYWORDS

Penetration testing, vulnerability, security, OWASP, Kali Linux, web application, SQL.

DOSTÁL, Adam *Laboratorní úloha seznamující studenty se síťovými útoky*: bakalářská práce. BRNO: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2016. 80 s. Vedoucí práce byl Ing. Zdeněk Martinásek, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Laboratorní úloha seznamující studenty se síťovými útoky“ jsem vypracoval(a) samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

BRNO ....

podpis autora(-ky)

PODĚKOVÁNÍ

Rád bych poděkoval panu Ing. Zdeňku Martináskovi, Ph.D. za odborné konzultace, připomínky a rady, které mi v průběhu psaní této práce poskytl.

BRNO . . . . . .

. . . . . .

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této bakalářské práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

BRNO ....

podpis autora(-ky)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OP Výzkum a vývoj
pro inovace

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Bezpečnost internetu, počítačů a síťové komunikace je celosvětově jedním z nejčastěji diskutovaných témat 21. století. Hlavním důvodem je, že při nedostatečném zabezpečení mohou být, ať už běžný uživatel nebo firemní společnost, vystaveni různým síťovým útokům, kde dochází k odcizení citlivých nebo soukromých dat, zneškodnění serveru z jeho běžné činnosti, napadení sociálních sítí, získání kontroly nad počítačem atd. Z těchto důvodů bylo zavedeno penetrační testování, které slouží k vyhodnocování zabezpečení síťových systémů a prostředků proti případnému napadení. Výsledky těchto testů jsou poté prozkoumány a případné nedostatky webových aplikací napraveny. Podle nedávné analýzy společnosti *„Veracode“*, 86% webových aplikací napsaných ve skriptovacím jazyce PHP obsahuje minimálně jednu cross-site scripting zranitelnost a 56% nejméně jednu SQL injection zranitelnost.

První kapitola teoretické části popisuje obecně penetrační testování, uvádí typy testů a jednotlivé kroky, podle kterých postupuje penetrační tým při testu webových aplikací. Druhá kapitola je věnována bezpečnostní organizaci „*Open Web Application Security Project*“ (OWASP) zabývající se bezpečností webových aplikací, financováním a vývojem desítek projektů, z nichž nejvýznamnější je OWASP Top 10, ve kterém je popsáno 10 nejzávažnějších zranitelností webových aplikací. V poslední kapitole je popsána linuxová distribuce Kali Linux, která je zaměřená na penetrační testy a obsahuje více než 300 nástrojů určených pro penetrační testování.

Cílem praktické části je realizace útoků na webovou aplikaci OWASP Mutillidae II a sepsání podrobného rozboru celého průběhu testování, ze kterého bude následně vytvořena laboratorní úloha seznamující studenty s penetračním testování webových aplikací. Jednotlivé útoky budou zaměřeny na prvních 5 zranitelností z dokumentu OWASP Top 10 2013.

Vzhledem ke správné korekci česko-anglického překladu bude práce v některých částech obsahovat anglické pojmy.

1 PENETRAČNÍ TESTOVÁNÍ

Jak již bylo uvedeno v úvodu, penetrační testy se provádí za účelem zjištění úrovně zabezpečení sítě nebo systému. Nejčastěji si tyto služby vyžadují firmy menších i větších rozsahů, aby nedocházelo k odcizení citlivých a neveřejných dokumentů, které by v případě krádeže mohly vést k finanční ztrátě. Osoba provádějící tyto testy se nazývá pen-tester nebo také etický hacker, jehož cílem je najít slabě zabezpečená místa, pomocí nich proniknout do určené infrastruktury a získat kontrolu nad celým systémem, či se pokusit získat citlivé informace. Rozsah testu je limitován přidělenými prostředky (čas, finance, personál), a z tohoto důvodu je potřeba se vždy zaměřit na nejkritičtější bezpečnostní místa, která představují pro firemní společnost největší hrozby.

Testování by mělo zahrnovat vše, kde hrozí riziko nežádoucího průniku do systému a odcizení dat nebo způsobení finanční škody.

Mezi tato rizika patří:

- E-mailové systémy
- Veřejné webové stránky
- Přístupová hesla
- DNS systémy
- Uložiště dat
- Informační systémy
- Síťové odposlouchávání
- CGI skripty
- LDAP systémy

Nejdůležitější webové aplikace k testování jsou právě takové, které využívají zejména zákazníci dané společnosti, např. internetové bankovnictví. Pokud by zde došlo k odcizení dat nebo peněz klienta, firma by ztrácela pověst mezi klienty, a tím i její hodnotu.

1.1 Typy testů

Penetrační testování kromě dalších subkategorií lze rozdělit na 2 základní (interní a externí) [1]:

Interní

Tento test se provádí z vnitřní strany sítě a představuje potencionálního útočníka, který se může vydávat za klienta nebo neloajálního zaměstnance a následně se pokusí o proniknutí do sítě společnosti.

Externí

Tento test se provádí z vnější strany testované oblasti a napodobuje vnější útoky (např. útočníka z internetu). Vnější penetrační testy se dále rozdělují na:

- VPN penetrační testy.
- Wi-fi penetrační testy.
- Penetrační testy webových aplikací.

Další rozdělení je podle způsobu provedení:

Manuální

Typ testu, který je vykonáván manuálně. Mezi výhody patří, že test lze zaměřit a přizpůsobit přesně pro určitou oblast se specifickými podmínkami. Další výhodou je, že člověk provádějící test musí mít velmi dobré znalosti problematiky penetračního testování, tzn. umí popsat a vysvětlit, co dělal, i lidem, kteří dané problematice nerozumějí. Nevýhodou je zároveň i již zmiňovaná velmi dobrá znalost penetračního testování a časová náročnost vzhledem k přípravě takového testu.

Automatizované

Test je vykonáván automaticky prostřednictvím nástroje vytvořeného profesionály, kteří se v této oblasti pohybují dlouhou dobu. Výhoda tohoto testu je, že testerovi stačí porozumět pouze tomu, jak se testovací nástroj ovládá, což ve výsledku vede k ušetření času oproti manuálnímu testu. Nevýhodou je nemožnost otestovat všechna zranitelná místa.

Semiautomatické

Kombinace manuálních a automatizovaných testů snažící se využít výhody obou testů a eliminovat nevýhody.

Podle úrovně znalostí o testovaném systému:

Black-box

Nejpoužívanější typ testu, kdy tester napodobuje útočníka z vnější strany, nemá o systému žádné informace a zná jenom vstupy a výstupy. Není potřeba žádná znalost programovacího jazyka ani zdrojového kódu. Nevýhodou je mít velmi dobré a široké znalosti problematiky penetračního testování.

White-box

Test, při kterém má tester plný přístup k veškerým informacím o systému či síti. Jinými slovy znalost vnitřní architektury, zdrojového kódu aplikace, typ a počet přítomných počítačových zařízení atd. Prakticky jde o kompletní analýzu zdrojového kódu, v němž se mohou vyskytovat chyby. S požadovanými znalostmi umožňuje na-

jít zranitelná místa v relativně krátké době a popřípadě kód optimalizovat pro vyšší bezpečnost aplikace. Nevýhodou je potřebná znalost programovacího jazyka, časová náročnost a úzké zaměření na kód a architekturu.

Grey-box

Kombinace obou typů testů black-box a white-box. Jedná se o snahu, použít co možná nejvíce výhod obou testů. Využívají se znalosti vnitřní logiky aplikace a test probíhá ze strany běžného uživatele nebo potencionálního útočníka.

1.2 Postup penetračního testování

Penetrační testování a celý jeho průběh se dělí do několika fází [2], [3], jejichž počet se pohybuje od čtyř do sedmi, viz obr. 1.1. Kolik fází bude při testování zahrnuto, záleží na zkušenostech penetračního týmu a požadavcích organizace. Jeho struktura je však pevně dána a do jisté míry je vždy stejná, proto zde bude popsáno pět fází.

Plánování

V této fázi dochází k přesnému a podrobnému stanovení, na jaké bezpečnostní části aplikace nebo systému bude test zaměřen. Jsou podepsány různé bezpečnostní smlouvy a penetrační tým určí rozsah testu, včetně finanční/časové náročnosti, případně typ nástrojů, které budou použity. Dále by tým měl zvážit případná rizika daného testu, tzn. musí se držet jasně daných pravidel ve smlouvě a volit takový postup, při kterém nedojde k odhalení neveřejných nebo tajných dat organizace.

Objevování/sběr dat

Fáze sbírání dat se považuje za začátek penetračního testování. Vybraný tým má za úkol zjistit co nejvíce užitečných informací o organizaci a jejich systémech (otevřené porty, verze operačních systémů, IP rozsahy, e-mailové zprávy, konfigurace zařízení, oprávněné osoby, nastavená pravidla firewallu atd.). Všechny tyto informace lze poměrně lehce získat pomocí běžně dostupných softwarových nástrojů (Nmap, Xprobe2, Queso apod.).

Odhalování zranitelností

Po úspěšném shromáždění všech užitečných informací o systému nastává další fáze odhalování zranitelností. Zde se používají automatizované nástroje s vlastní aktualizovanou databází (Nessus, Retina, ISS Scanner atd.), které obsahují velké množství zranitelností, jejich typy a detaily. Tyto nástroje srovnávají např. verze operačních systémů síťových prostředků, typ/datum aktualizací a po porovnání s databází vy-

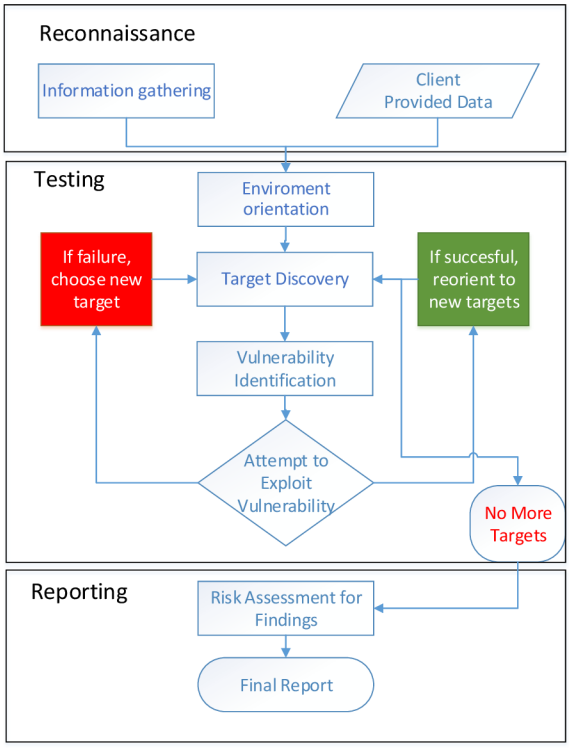
hodnotí případné zranitelnosti a nedostatky. V případě manuálního testu je potřeba, aby tester nespoléhal jen na nástroje, ale dokázal využít efektivně svoje zkušenosti a měl informace ohledně nejnovějších zranitelností a principu jejich zneužití.

Exploitace

Fáze, kdy nastává samotný proces prolamování bezpečnostních mechanizmů a zneužívání zjištěných zranitelností v předchozím kroku. Tester se snaží získat přístup do určitých zabezpečených oblastí, a pokud je úspěšný, může se mu naskytnout další možná cesta, která byla dříve nedostupná. V této chvíli by se měl tester vrátit na začátek a pokusit se shromáždit další informace na základě nově zjištěných.

Report

Poslední fází je zpráva, která shrnuje celkové výsledky penetračního testování, včetně podrobného postupu a obsahuje případné doporučení penetračního týmu na zlepšení bezpečnosti v kritických oblastech.



Obr. 1.1: Metodologie testování (překresleno)

2 BEZPEČNOSTNÍ ORGANIZACE

Nejznámější organizací věnující se bezpečnosti je „*The Open Web Application Security Project*“ (OWASP). Je to celosvětová nezisková charitativní nadace, zaměřená na zlepšování bezpečnosti softwaru. Nadace vznikla 1. prosince 2001 a o 3 roky později 21. dubna 2004 se z ní stala nezisková organizace ve Spojených státech amerických. Jejím cílem je, aby jednotlivci nebo organizace po celém světě měli neomezený přístup k informacím o bezpečnostních hrozbách a na jejich základě mohli provádět bezpečnostní opatření. To znamená, že veškeré OWASP nástroje, dokumenty a fóra jsou volně dostupné všem, kteří se chtějí buď dozvědět více o bezpečnosti webových aplikací nebo se podílet na bezpečnostním vývoji [4].

OWASP má v současné době přes 200 poboček po celém světě. Jedna z nich je i v České republice v Praze pod vedením Jana Kopeckého. Dále má OWASP více něž 142 aktivních projektů a každým dnem jim přichází žádosti o založení nových.

Nejvýznamnější z nich:

- OWASP CLASP
- OWASP WebGoat and WebScarab
- OWASP Developers Guide
- OWASP Testing Guide
- OWASP Top 10
- OWASP Codes of Conduct

Celý přehled projektů je k dispozici na [5].

2.1 OWASP Top 10

Nejvýznamnější dokument, na jehož vytvoření se podílí bezpečnostní experti z celého světa a který zveřejňuje 10 nejzávažnějších zranitelností webových aplikací [6].Je určen k bezplatnému používání a je licencován firmou „*Creative Commons*“ licencí Attribution-ShareAlike 3.0 [7]. Projekt založil v roce 2003 Dave Wichers a pod jeho vedením vychází každé 3 roky nová verze (od roku 2004 celkem 4) s aktuálními hrozbami.

Top 10 zranitelností webových aplikací v roce 2013:

- A1 Injection (Injektování).
- A2 Broken Authentication and Session Management (Chybná autentizace a správa relace).
- A3 Cross-Site Scripting (XSS).

- A4 Insecure Direct Object References (Nezabezpečené přímé odkazy na objekty).
- A5 Security Misconfiguration (Nezabezpečená konfigurace).
- A6 Sensitive Data Exposure (Expozice citlivých dat).
- A7 Missing Function Level Access Control (Chyby v řízení úrovní přístupů).
- A8 Cross-Site Request Forgery (CSRF).
- A9 Using Components with Known Vulnerabilities (Použití známých zranitelných komponent).
- A10 Unvalidated Redirects and Forwards (Neošetřené přesměrování a předávání).

V oblasti zabezpečení aplikací se tyto hrozby neustále mění, viz obr. 2.1. Každým rokem dochází ke vzniku nových a propracovanějších technologií, kde na jejich základě útočníci rozšiřují svoje znalosti a dovednosti, které použijí k důmyslnějším útokům.

OWASP Top 10 – 2010 (Previous)	OWASP Top 10 – 2013 (New)
A1 – Injection	A1 – Injection
A3 – Broken Authentication and Session Management	A2 – Broken Authentication and Session Management
A2 – Cross-Site Scripting (XSS)	A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References	A4 – Insecure Direct Object References
A6 – Security Misconfiguration	A5 – Security Misconfiguration
A7 – Insecure Cryptographic Storage – Merged with A9 →	A6 – Sensitive Data Exposure
A8 – Failure to Restrict URL Access – Broadened into →	A7 – Missing Function Level Access Control
A5 – Cross-Site Request Forgery (CSRF)	A8 – Cross-Site Request Forgery (CSRF)
<buried in A6: Security Misconfiguration>	A9 – Using Known Vulnerable Components
A10 – Unvalidated Redirects and Forwards	A10 – Unvalidated Redirects and Forwards
A9 – Insufficient Transport Layer Protection	Merged with 2010-A7 into new 2013-A6

Obr. 2.1: Porovnání Top 10 hrozeb 2010/2013

Z uvedeného porovnání je vidět, že první čtyři pozice kromě výměny A2 a A3 zůstaly stejné. Výskyt zranitelnosti A5 (CSRF) poklesl až na pozici A8, je to z důvodu snahy vývojářů a organizací tomuto riziku pokud možno co nejvíce zamezit. A7 společně s A9 bylo sloučeno a zahrnuje zranitelnost A6 (Sensitive Data Exposure). Poslední změnou je přidání nové zranitelnosti A9 (Using Components with Known Vulnerabilities).

2.2 Zranitelnosti A1-A5 (2013)

V této sekci bude popsáno prvních 5 nejčastěji se vyskytujících zranitelností webových aplikací, které budou následně v praktické části demonstrovány a použity jako podklad pro laboratorní úlohy.

A1 - Injection

Nejběžnější typ útoku injection je SQL. Princip útoku typu SQL injection [8], [9], [10], [11] je vložení vlastních příkazů do nezabezpečeného vstupu aplikace a prostřednictvím dotazu SELECT jsou tato vstupní data odeslána od klienta na webový server, kde dojde k narušení syntaxe kódu. Pokud je SQL injection úspěšné, může útočník číst data z databáze, měnit data databáze (vkládat, upravovat, mazat), provádět administrátorské operace s databází atd.

Útok

Obvykle k útoku dochází při přihlašování do webových aplikací. Uživatel zadá své jméno a heslo, které jsou poslány dotazem SELECT webovému serveru. Pokud zadané údaje souhlasí s údaji v databázi, je uživateli přístup povolen, v opačném případě zamítnut. Při nedostatečném zabezpečení vstupů může útočník vkládat své vlastní dotazy a dostat přístup do databáze. Na obrázku 2.2 se útočník přihlásí jako *‘ OR 1=1; **/– a heslo zakomentuje. Pokud nejsou řádně zabezpečeny vstupy, tak ho webová aplikace přihlásí jako prvního uživatele uvedeného v tabulce *Users*.

Jeden ze způsobů, jak zjistit, zda-li má webová aplikace SQL injection zranitelnost, je vložení neočekávaných znaků jako vstup, které by mohly změnit dotaz tak, že bude neplatný. Pokud zadáme znaky, které mají zvláštní význam v jazyce SQL a webová aplikace vygeneruje specifickou chybu, můžeme usuzovat, že zadaný vstupní znak je používán v dotazu SQL zranitelným způsobem.

Mezi tyto znaky patří:

- Apostrof (’) - označuje začátek nebo konec dat.
- Pomlčky (–) - označují SQL komentář.
- Středník (;) - označuje konec SQL příkazu.

Nejvíce se používá jednoduchá uvozovka (’). Pokud je cílová webová stránka zranitelná, tak se při jejím použití vygeneruje SQL chyba, ve které se budou vyskytovat klíčová slova „SQL“ nebo „MySQL“. Důvodem chyby je, že znak (’) je brán jako řetězcový oddělovač. Syntakticky je SQL dotaz spuštěn jako nesprávný (obsahuje příliš mnoho oddělovačů) a tudíž databázový systém vytvoří výjimku (chybu). Za nejčastější chyby lze označit takové, které se týkají buď databázového systému MySQL nebo Microsoft SQL Serveru.

Mějme webovou stránku:

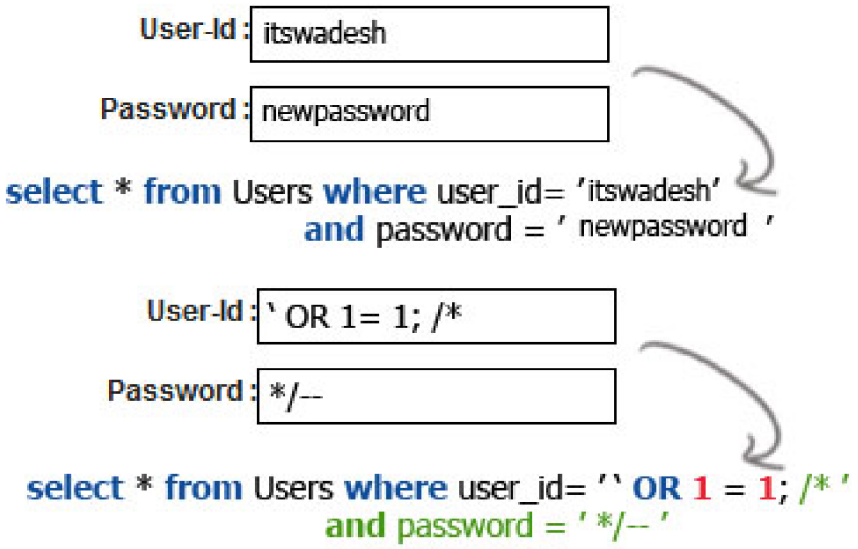
http://priklad.com/listproducts.php?cat=1

Část řetězce *.php* je indikátor, že cílový databázový systém by mohl být MySQL, který je ve většině případů zranitelný proti útoku SQL injection a dále je vidět, že *cat=1* odkazuje na konkrétní objekt, což značí metodu GET. Použitím neočekávaného znaku (') na konci URL, změní dotaz tak, že webová aplikace vygeneruje chybu indikující SQL zranitelnost:

Error: you have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' at line 1

V této chvíli útočník ví, že webová aplikace je zranitelná. Může zkoušet vkládat libovolné dotazy a pokusit se tak získat informace nebo kontrolu nad databází.

Dříve SQL injection nepředstavoval takovou hrozbu, protože bylo potřeba zadávat svůj vlastní SQL kód do webového formuláře manuálně. V současné době tomu je jinak, zejména kvůli automatizovaným nástrojům specializovaných pro SQL injection a proto představuje nejčastější zranitelnost webových aplikací posledních 6 let. Mnoho velkých firem nevěnovalo pozornost této chybě (i přes fakt, že proti SQL injection mohou být webové aplikace téměř zcela zabezpečeny), což vedlo v některých případech k velkým finančním ztrátám nebo odcizení neveřejných informací [12].



Obr. 2.2: Napadení pomocí SQL injection

Obrana

Způsobů ochrany je mnoho, ale mezi nejdůležitější patří používání API (Application Programming Interface), které nepoužívá interpret a poskytuje parametrizované rozhraní. Dalším způsobem je tzv. escapování (převod znaků na jiné odpovídající znaky) nebezpečných znaků jazyku SQL, mezi které patří zejména apostrof, pomlčky a středník.

A2 - Broken Authentication and Session Management

Broken Authentication and Session Management [13], [14] zahrnuje vše, co se týká ověřování uživatelů a řízení aktivní relace. Silnější metody autentizace existují a jsou k dispozici kombinující SW i HW na principu biometrie nebo kryptografických tokenů, ale vzhledem k vysokým finančním nákladům a náročnosti návrhu takového systému pro webové aplikace jsou velmi často vývojovými týmy opomíjeny. Tyto nedostatky mohou mít za následek ohrožení uživatelského soukromí nebo převzetí kontroly nad správou systému účtů.

Webové aplikace musejí mít zavedené relace pro sledování žádostí ze strany uživatelů. Protokol HTTP tuto funkci nepodporuje, a i přes skutečnost, že prostředí webových aplikací tuto možnost podporuje, vývojáři si preferují relace vytvářet sami, což vede mnohdy ke spoustě nevyžádaných chyb. Za časté chyby lze označit absenci šifrování při autentizaci údajů, odhalování ID relací v URL, neměnnost ID relace po přihlášení atd. Pokud nejsou tokeny relací dostatečně chráněny, není pro útočníka problém získat aktivní relace a vydávat se za uživatele.

Útok

Pro demonstraci útoku mějme webovou stránku aerolinek, jejichž systém vkládá do URL i ID relace:

```
http://priklad.com/sale/saleitems;jsessionid=6KELF6FERTOWPWKDSOS6SD
?dest=Island
```

Pokud právě přihlášený uživatel bude chtít někomu poslat tento odkaz o nabídce, aniž by věděl, že daný odkaz obsahuje ID o vlastní relaci, tak dotyčná osoba, které byl odkaz poslán při jeho použití, bude zároveň používat i relaci původního uživatele, a tedy i jeho např. kreditní kartu.

Obrana

Ke snížení pravděpodobnosti tohoto typu útoku, by měli dodržovat určitá pravidla vývojáři webových aplikací i uživatelé:

- **Síla hesla** - každé heslo by mělo splňovat podmínky minimální délky a složitosti tzn. zadání velkého počátečního písmena s kombinací čísel, popřípadě

speciálních znaků. Uživatelé by také měli často měnit své hesla a nepoužívat žádná z předchozích.

- **Kontrola změny hesla** - mechanizmus změny hesla by měl být k dispozici všude, kde to vyžaduje situace. Uživatelé by měli být povinni při změně hesla zadat staré i nové heslo a systém by měl vyžadovat při těchto změnách nějakou formu autentizace (e-mail).
- **Používání hesla** - uživatelé by měli mít omezený počet pokusů při zadávání hesla během určité doby. Systém by neměl uvádět, zda-li bylo špatně zadáno heslo nebo jméno, měl by informovat uživatele o čase jejich úspěšného přihlášení nebo naopak uvádět čas a počet neúspěšných pokusů.
- **Uložení hesla** - všechna uložená hesla by měla být nějakým způsobem šifrovaná, aby nemohlo dojít k jejich expozici.
- **Ochrana ID relace** - pro ochranu celé relace se používá technologie SSL. Pokud systém tuto ochranu poskytuje, je nemožné zachytávání čísel relace mimo síť. Relace by se neměly vyskytovat v URL a měly by být dostatečně složité pro zabránění jejich uhodnutí.
- **Seznamy účtů** - systémy pro správu účtů by měly být navrženy tak, aby zabránily neoprávněnému nahlížení do seznamu účtů a dalších informací.

A3 - Cross-Site Scripting (XSS)

Cross-Site Scripting [15], [16] je zranitelnost, kdy se útočník snaží spustit škodlivý kód (skript) na straně klienta prostřednictvím jeho webového prohlížeče. Princip je takový, že útočník se snaží využít bezpečnostních chyb ve spouštěcích skriptech dané webové aplikace, pomocí nich měnit vzhled nebo obsah stránky a vytvořit tak odkazy, které vypadají jako důvěryhodné. Pokud nic netušící uživatel takový odkaz použije, tak dojde zároveň ke spuštění škodlivého skriptu, protože webové prohlížeče neposkytují žádný způsob, kterým by zjistily, že daný skript je nedůvěryhodný, což má za následek umožnění přístupu k probíhajícím relacím, soukromým informacím uložených v prohlížeči nebo přesměrování na jiné stránky apod. Tato metoda se nazývá reflected.

XSS útoky jsou rozděleny do třech kategorií:

- **Refleted** - útok typu *Reflected* je nejběžněji používaný. Princip spočívá v tom, že škodlivý kód je obsažen přímo v URL odkazu, který je odražen od webového serveru a dodán oběti jinou cestou např. prostřednictvím e-mailových adres. Pokud uživatel takový odkaz použije, prohlížeč vykoná i škodlivý skript, protože zdroj bude uveden jako důvěryhodný.
- **Stored** - útok typu *Stored* je nejnebezpečnější, protože skript, ve kterém je vložený škodlivý kód, je permanentně uložen na webovém serveru. Pokud uživatel zažádá server o nějakou uloženou informaci, tak spolu s touto informací

se spustí i nebezpečný skript. Nejčastěji se nebezpečný kód vkládá na webové stránky, které používá velké množství uživatelů z důvodu, že může postihnout všechny. Mezi tyto stránky lze zařadit např. různá fóra, rejstříky, sociální sítě atd.

- **DOM Based** - útok typu *DOM Based* je velice podobný útoku *reflected* s tím rozdílem, že škodlivý kód není odesílán na webový server, ale je zpracován až na straně klienta prostřednictvím JavaScriptu.

Útok

Pro demonstraci útoku typu reflected mějme webovou stránku používající skriptovací jazyk PHP s XSS zranitelnostmi. Původní tvar odkazu vypadá takto:

```
http://priklad.com/menu.php?user=xx
```

Někde ve zdrojovém kódu se vyskytuje následující řádek:

```
<?php echo $_GET['user']; ?>
```

Z výše uvedeného řádku je zřejmé, že pro úspěšný útok stačí upravit odkaz, aby vypadal takto:

```
http://priklad.com/menu.php?user=<script>alert('Útok XSS je úspěšný.');</sc
ript>
```

Pokud oběť takový odkaz použije, zobrazí se výstražné okno *alert* s nápisem „Útok XSS je úspěšný.“. V praxi se však používají nebezpečné skripty, které mohou způsobit určitou formu škody na straně uživatele.

Obrana

Nejlepším způsobem, jak se proti XSS útokům bránit, je escapování všech nedůvěryhodných dat a znaků v kontextu HTML. Dále se doporučuje používat validace vstupů a automatické sanitizační knihovny.

A4 - Insecure Direct Object References

Insecure Direct Object References [17] nastává, pokud vývojář zanechá nezabezpečené odkazy na interní objekty (klíč databáze, adresář, soubor atd.). Potenciální hrozbu poté představuje systémem autorizovaný uživatel, který může měnit parametry odkazující na určité objekty a dostat se tak k informacím, ke kterým nemá oprávněný přístup.

Útok

Mějme webovou stránku pro internetové bankovnictví, která používá přímé odkazy na účty. Uživatel je přihlášen ke svému účtu, jehož URL vypadá následovně:

```
http://priklad.com/accounts/details?accountnumber=235964658950
```

Účet autorizovaného uživatele má číslo *235964658950*. Pokud by však zaměnil v odkazu poslední číslici za kterékoliv jiné číslo:

```
http://priklad.com/accounts/details?accountnumber=235964658951
```

Mohlo by se stát, že dostane přístup k účtu a informacím, ke kterým nemá oprávnění.

Obrana

Principem ochrany proti této zranitelnosti je nutnost dodržovat určitá pravidla:

- Vyhnout se používání přímých odkazů na objekty a používat nepřímé odkazy pro každou relaci či uživatele zvlášť.
- Zamezit viditelnosti aktuálního ID objektu.
- Kontrola oprávnění při použití přímých odkazů, zda má při vstupu daný uživatel práva k danému objektu.

A5 - Security Misconfiguration

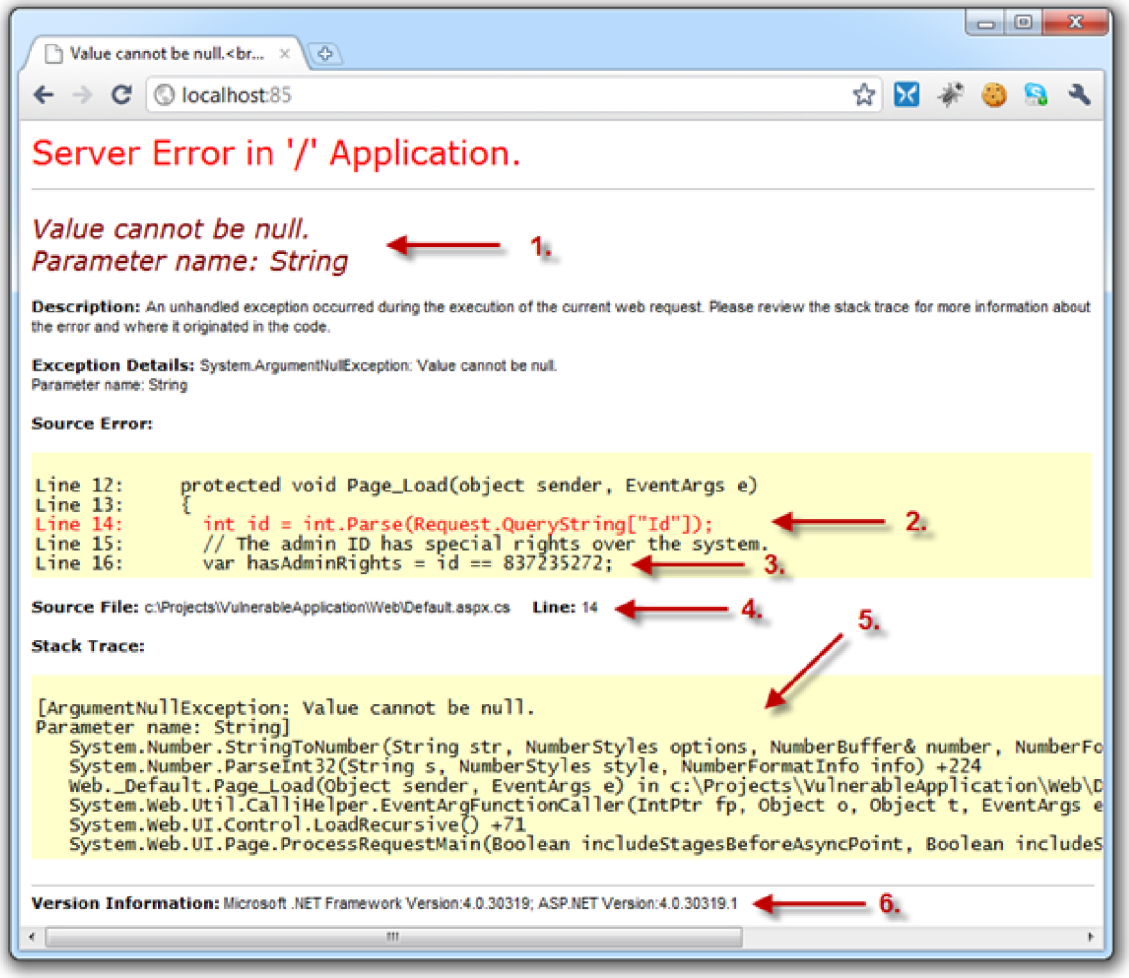
Security Misconfiguration [18] je zranitelnost, která souvisí s bezpečností webových serverů (ISS, Apache) a databázovými servery, mezi které patří zejména Microsoft SQL Server a MySQL. Správu a konfiguraci serverů mají na starost administrátoři, jejichž úkolem je udržet bezpečnost těchto služeb udržováním aktuálního softwaru, zakázáním nepotřebných služeb, měnit výchozí hesla, nastavení pravidel firewallu atd.

Útok

Pro demonstraci útoku mějme webovou aplikaci, která při výpisu chybového hlášení zobrazí následující informace, viz obr. 2.3 [19]

Z tohoto výpisu chybového hlášení může útočník zjistit níže uvedené informace a využít je pro svůj útok:

1. Informace o hodnotě a názvu parametru.
2. Informace o vnitřní implementaci parametru ID.
3. Citlivé detaily o struktuře kódu.
4. Fyzické umístění souboru na počítači vývojáře.



Obr. 2.3: Chybový výpis aplikace

5. Seznam metod volaných před chybou.
6. Verze .NET Framework používaný aplikací.

Obrana

Doporučená obrana proti tomuto typu útoku je včasné nasazení a udržování nových softwarových aktualizací a oprav, zajistit silnou architekturu aplikace, pravidelné skenování pro odhalení chybné konfigurace nebo zastaralých aktualizací, často měnit výchozí hesla, omezit výpis chybových hlášení a vypnutí možnosti výpisu adresářů.

3 KALI LINUX

Kali Linux [20] je linuxová open-source distribuce, založená na Debianu, sloužící pro penetrační testování a bezpečnostní průzkum webových aplikací. Obsahuje přes 300 penetračních nástrojů určených pro tyto účely. Kali Linux byl oficiálně spuštěn 13. dubna 2013 a vznikl jako kompletní přestavba linuxové distribuce BackTrack [21], který již nadále není vyvíjen. Projekt založil Mati Aharoni a Devon Kearns z organizace „*Offensive Security*“ [22], která má na starosti údržbu a financování.

Systém je volně ke stažení na oficiálních stránkách `https://www.kali.org/downloads` v 32/64 bitové verzi nebo obrazu (armel, armhf) založeném na ARM architektuře. Zprovoznění lze udělat několika způsoby: instalací na pevný disk, využití externího zařízení (flash disk, externí disk) bez nutnosti instalace nebo spuštěním pomocí softwaru VMware, popřípadě Oracle VM VirtualBox jako virtuální systém.

Nedávno (11. září 2015) byla spuštěna nová verze Kali Linux 2.0. Mezi hlavní změny oproti předchozí verzi 1.1.0a patří použití kernelu 4.0 založeného na distribuci Debian Jessie, rozšíření podpory hardware/síťových ovladačů a desktopových prostředí (gnome, kde, xfce, mate, e17, lxde, i3wm).

3.1 Nástroje Kali Linux

Vzhledem k velkému množství nástrojů pro penetrační testování webových aplikací, viz obr. 3.1, budou následovně popsány pouze některé z nejpoužívanějších. Jako zdroj informací je použit přehled nástrojů [23] a jejich oficiální stránky.

Metasploit Framework

Metasploit Framework je open-source nástroj, umožňující bezpečně simulovat útoky na síť k odhalení bezpečnostních chyb a prověření síly zabezpečení webových aplikací. Ovládá se přes příkazový řádek nebo přes grafické uživatelské prostředí (GUI) a vzhledem k jeho komplexnosti dokáže také pracovat s Armitage.[1]. Obsahuje přes 1300 exploitů, které každým dnem přibývají a více než 2000 modulů. Projekt má na starosti organizace „*Rapid7*“, poskytující kromě základní verze Metasploit Framework verzi zpoplatněnou (Metasploit Pro) a verzi komunitní (Metasploit Community).

[1] Armitage - nástroj, který zviditelňuje cíle a na jejich základě doporučuje exploity.

Obr. 3.1: Přehled nástrojů v prostředí Kali Linux

Wireshark

Wireshark je nejrozšířenější aplikace k zachytávání a analýze komunikace, procházející přes použitá síťová rozhraní. Poskytuje stovky analyzerů komunikačních protokolů, grafické uživatelské rozhraní (GUI), podporu různých operačních systémů (Windows, Linux, Mac OS X, BSD, Solaris), dešifraci protokolů (WPA, WEP), podrobnou filtraci zachycených informací atd. Dále umožňuje přepnutí síťové karty do promiskuitního režimu, který poté dokáže zachytit veškerý síťový provoz na daném médiu.

John the Ripper

John the Ripper je nástroj, který je zdarma a slouží k prolomení hesla. Původně byl vyvinut pouze pro operační systém Unix, nyní je možné ho spustit na patnácti různých platformách (DOS, Win32, BeOS, OpenVMS a zbylých 11 pro Unix systémy). Kombinuje řadu mechanismů k prolamování klíčů do jednoho společného, automaticky detekuje typ zašifrování daného hesla a obsahuje konfigurovatelný „cracker“ pro vlastní potřeby. Může být použit proti různým šifrovacím formátům hesel včetně typů, které se používají u Unix systémů (založených na DES, MD5, Blowfish), Kerberos AFS a Windows NT/XP/2002/2003 LM hash.

Nmap

Nmap (Network Mapper) je open-source aplikace pro objevování sítí a prozkoumání jejich bezpečnosti. Používá IP pakety k vyhledání dostupných cílových počítačů v síti, jaké služby host používá, typ spuštěného operačního systému, jaký typ filtru paketů je použit, pravidla firewallu atd. Podporuje velké množství platforem (Linux, Windows, Mac OS X, Solaris, IRIX, OpenBSD a další).

Standartní verze Nmap se ovládá přes příkazový řádek, ale k dispozici je i grafické prostředí (Zenmap) pro zjednodušení ovladatelnosti a vykreslení mapy sítě. Zadává se cíl a následně i profil, který určuje typ skenování (obvyklý sken, intenzívní sken, rychlý sken, sken s použitím TCP/UDP portů atd.). Výsledky jsou průběžně generovány do záložky Nmap Output, které lze podle potřeby různě filtrovat.

Burp Suite

Nástroj Burp Suite se používá pro bezpečnostní testování webových aplikací. Aplikace se skládá z několika dílčích modulů (proxy server, intruder, repeater, web spider atd.), které slouží k analýze a zmapování cíle, nalezení zranitelností a jejich následnému zneužití. Je napsán v programovacím jazyce Java, což umožňuje jeho použití na libovolném OS. K dispozici je verze neplacená (Free Edition) a verze placená (Professional Edition), rozšiřující verzi základní o určité doplňky pro efektivnější používání.

Sqlmap

Sqlmap je open-source nástroj určený k penetračnímu testování, který je specializován k automatickému odhalování a využívání SQL injection zranitelnosti. Kromě základních funkcí obsahuje mnoho specializovaných pro zkušené testery např. „fingerprinting“ databáze, načítání dat z databáze, přístup ke skrytým systémovým souborům a spouštění příkazů na operačním systému pomocí out-of-band připojení. Podporuje 6 SQL injection technik (boolean-based blind, time-based blind, error-based, UNION query, stacked queries, out-of-band).

OWASP ZAP

OWASP ZAP (Zed Attack Proxy) je open-source aplikace, určená ke skenování webových aplikací z hlediska bezpečnostních nedostatků. Jedná se o jeden z nejaktivnějších projektů organizace *„OWASP“*, díky čemuž je zajištěna neustálá podpora a aktuálnost. Stejně jako nástroj Burp Suite je napsán v programovacím jazyce Java, aby byla zajištěna funkčnost na libovolném OS a obsahuje i podobné moduly (web crawlers, passive scanner, automated scanner, proxy server atd.).

W3af

W3af (Web Application Attack and Audit Framework) je framework, jehož cílem je identifikovat a zneužívat veškeré nalezené zranitelnosti webových aplikací. Je kompletně napsaný v programovacím jazyce Python, obsahuje více než 130 pluginů, které používá pro identifikaci a využití nalezené zranitelnosti a nachází i využití v případě penetračního testování metodou black-box. Pro ovládání poskytuje grafické uživatelské rozhraní (GUI) nebo příkazový řádek.

XSSer

XSSer (Cross Site „Scripter”) je automatický framework, který slouží k detekci, exploitu a následného reportu dosažených výsledků XSS zranitelností webových aplikací. Obsahuje mnoho možností, jak se vyhnout určitým filtrům a používá různé speciální techniky k aplikaci škodlivého kódu.

WebScarab

WebScarab je projekt organizace *„OWASP“*, který slouží k analýze a zachytávání komunikace webového prohlížeče tzn. webové žádosti (HTTP i HTTPS) i webové odpovědi od serveru. Vzhledem k nepříliš přehlednému rozhraní je ve vývoji projekt WebScarab NG (Next Generation), který vznikl kompletním přepsáním WebScarab a zaměřuje se na přívětivější uživatelské rozhraní. Při vzniku nástroje OWASP ZAP se jeho vývoj stal pomalejším a větší pozornost je věnována právě zmíněnému nástroji, přesto se však jedná o velmi používaný a efektivní nástroj k penetračnímu testování webových aplikací.

Nessus Vulnerability Scanner

Nessus Vulnerability Scanner je celosvětově nejrozšířenější skener zranitelností vyvinutý společností *„Tenable Network Security“* (TNS) [24]. Testování funguje na principu kontroly konkretních chyb pomocí pluginů, které mají velmi rozsáhlé denně aktualizované databáze [25]. Nástroj pracuje na modelu klient-server, což znamená, že uživatel se pomocí webového prohlížeče může k serveru připojit z libovolné stanice

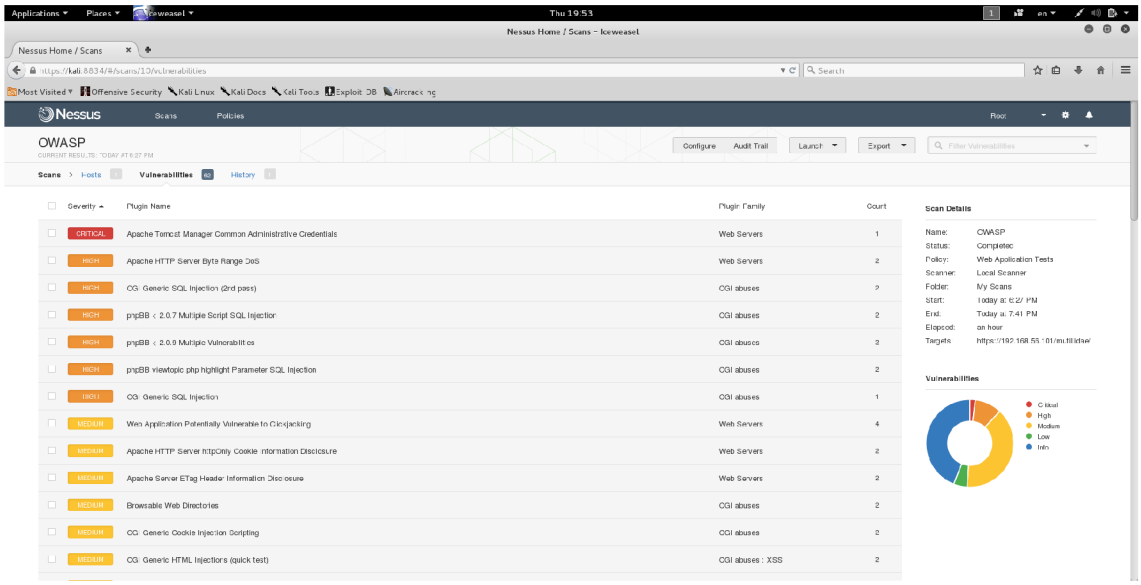
v síti. Umožňuje skenování pro následující typy zranitelnosti:

- Zranitelnosti, které umožní útočníkovi převzít kontrolu nad serverem nebo citlivými daty.
- Nesprávné konfigurace např. chybějící aktualizace.
- Výchozí hesla, běžné hesla, účty s chybějícími hesly.
- Odmítání služeb (Denial of service) proti TCP/IP.

Výsledky testu lze poté zaznamenat v různých formátech (XML, PDF, HTML, LaTeX). Nessus se také vyznačuje velmi přehledným a jednoduchým GUI 3.2, tudíž nástroj může efektivně používat i nový a nezkušený uživatel.

Nejnovější verze je Nessus 6.5 a byla vydána 15. října 2015. K dispozici jsou v současné době 4 různé verze z toho 3 placené (Nessus Manager, Nessus Cloud, Nessus Professional) pro profesionální použití penetračních týmů a jedna neplacená (Nessus Home). Nessus Home je určená pro nekomerční používání, avšak neposkytuje přístup k podpoře, je značně omezená a limitovaná maximálním počtem IP adres (16) v rámci jednoho testu. Nessus je navržen tak, aby mohl být dostupný a podporovaný pro různé OS a platformy.

Distribuce Kali Linux nástroj Nessus Vulnerability Scanner neobsahuje a je potřeba ho doinstalovat.



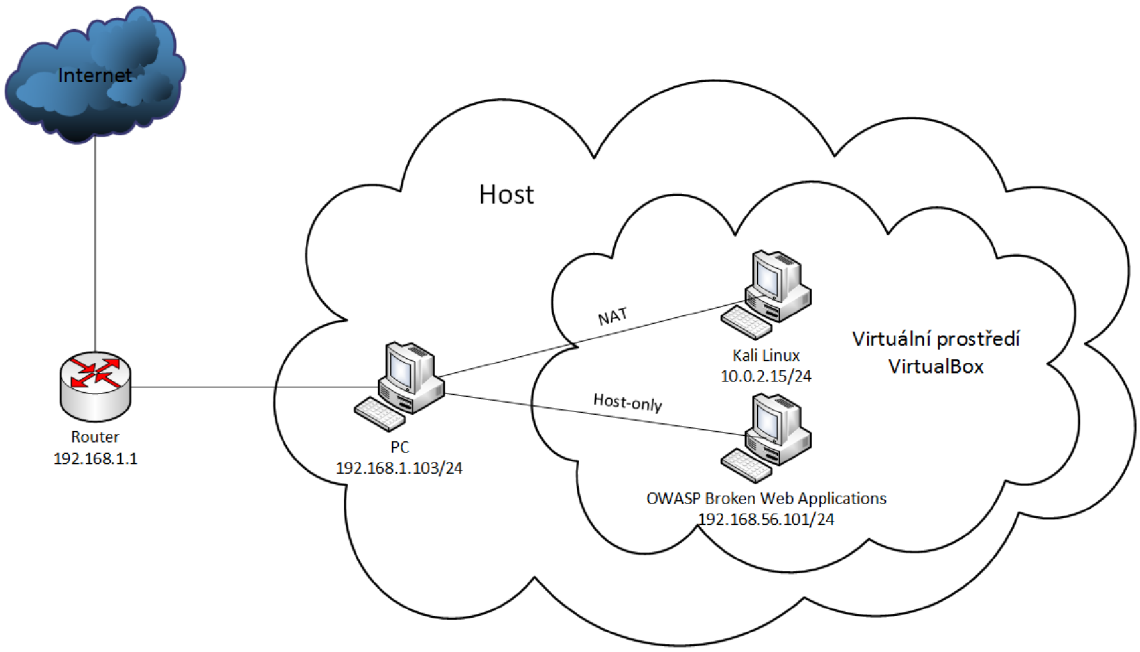
Obr. 3.2: Uživatelské prostředí nástroje Nessus

4 TESTOVÁNÍ ZRANITELNOSTÍ OWASP TOP 10

Cílem této kapitoly je provedení testu webové aplikace OWASP Mutillidae II na zranitelnosti A1-A5 z dokumentu *OWASP Top 10 2013* pomocí penetračních nástrojů v distribuci Kali Linux. Podrobný rozbor bude zároveň sloužit jako podklad pro vytvořenou laboratorní úlohu a návod pro vyučující.

4.1 Použitý software a vnitřní infrastruktura sítě

Vnitřní infrastruktura sítě je zakreslena na obr. 4.1. Prostřednictvím virtualizačního programu Oracle VM VirtualBox, umístěného na PC hosta, jsou spuštěny dvě virtuální stanice (Kali Linux a OWASP Broken Web Applications), které komunikují s hostitelským PC v režimu *NAT* a *Host only* z hlediska bezpečnosti sítě. Host všechny požadavky od stanic posílá routeru, který je zpracuje a přepošle dál v závislosti na požadavku. Infrastruktura na obrázku 4.1 je vytvořena podle mé vlastní.



Obr. 4.1: Vnitřní infrastruktura sítě

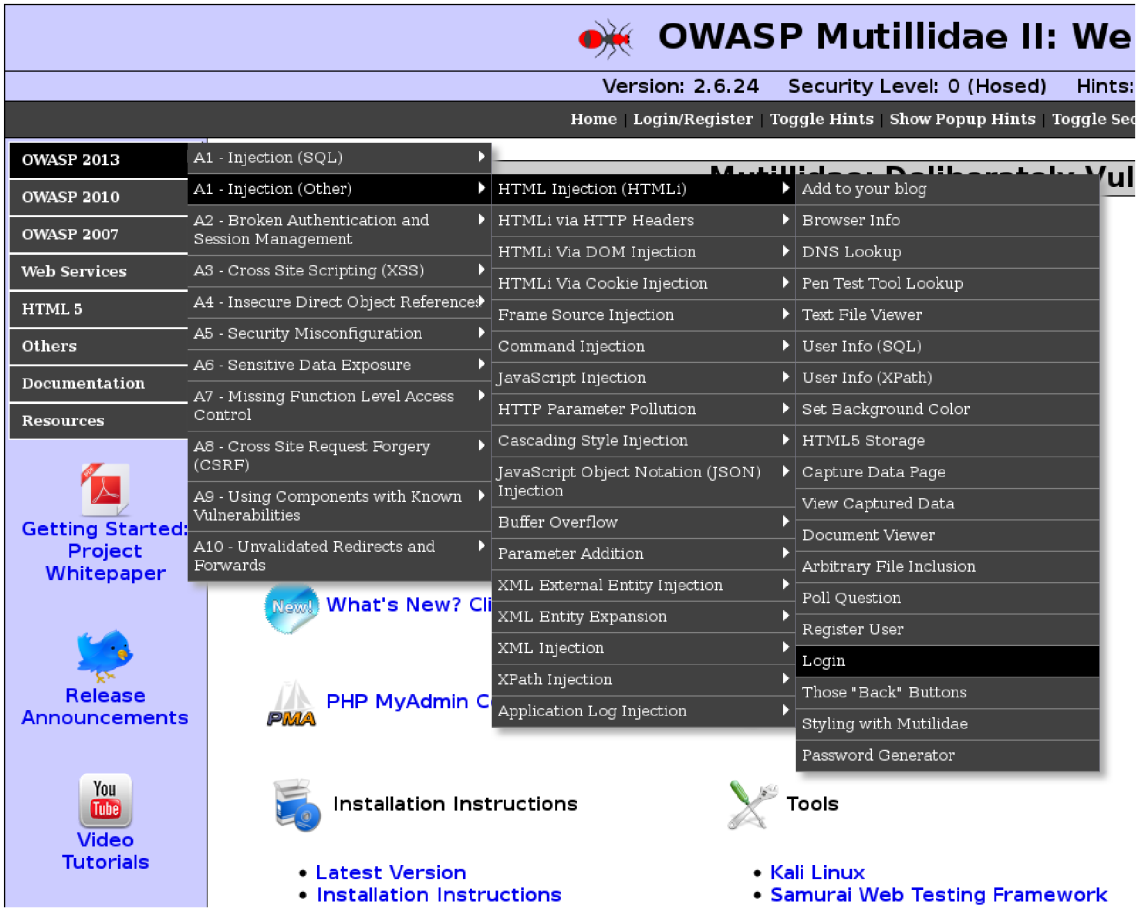
Oracle VM VirtualBox

Oracle VM VirtualBox [26] je virtualizační nástroj pro profesionální nebo osobní použití. Je vyvíjen společností *„Oracle“* ve dvou licenčních podobách (PUEL, GPL).

Obsahuje mnoho funkcí (podpora více jazyků, ukládání aktuálních snímků, sdílení složek, přenosnost, 3D virtualizaci, seamless mode atd.) a je navržen tak, aby mohl být spuštěn na operačních systémech Windows, Unix/Linux, Mac OS a Solaris.

OWASP Broken Web Applications

OWASP Broken Web Applications [27] je souhrn všech veřejně dostupných zranitelných webových aplikací určených pro penetrační testování např. Damn Vulnerable Web Application, bWAPP, OWASP WebGoat, OWASP Mutillidae II apod. Hlavním účelem je poskytnout možnost naučit se o bezpečnosti webových aplikací, vyzkoušet si manuální techniky testování, automatizovanými nástroji, analýzu zdrojového kódu a pozorování průběhu webových útoků. Projekt vede Chuck Willis a je sponzorovaný společností *„Mandiant“*.



Obr. 4.2: Zranitelnosti a podvolby zneužití

OWASP Mutillidae II

OWASP Mutillidae II [28] je open-source záměrně zranitelná webová aplikace, určená pro lidi zajímající se o webovou bezpečnost. Pro uživatele, kteří nechtějí mít na

starosti administraci serveru, může být nainstalována na operačních systémech Linux/Windows používající platformy LAMP (Linux, Apache HTTP Server, MySQL and PHP), WAMP (Windows Apache MySQL PHP) a XAMMP (Cross-platform Apache MySQL PHP). Projekt vede Jeremy Druin pod licencí GNU GPL v3 (open-source licence). Mutillidae poskytuje 10 zranitelností z každé nově vydané verze dokumentu OWASP Top 10, včetně desítek voleb u každé z nich, jakým způsobem zranitelnost zneužít, viz obr. 4.2. Dalšími funkcemi jsou rady pro pomoc uživateli, video tutoriály a přepínání mezi obtížnostmi zabezpečení jednotlivých zranitelností.

Kali Linux

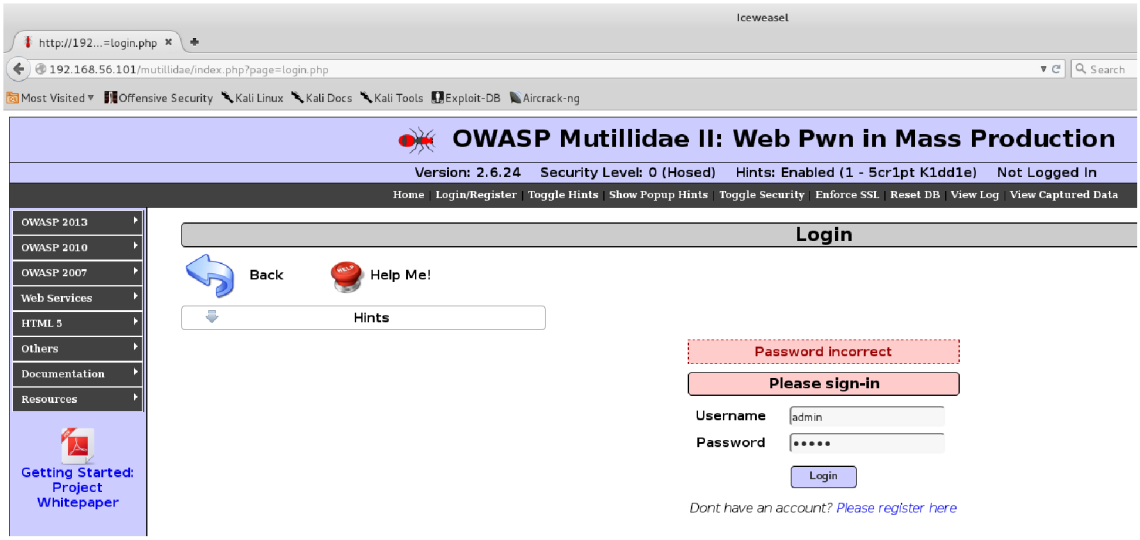
Pro prezentaci zranitelností bude využita distribuce Kali Linux a penetrační nástroje Sqlmap, Burp Suite a XSSer.

4.2 Zranitelnost A1 - SQL injection

Cílem této úlohy je získání uživatelských přihlašovacích údajů webové aplikace OWASP Mutillidae II pomocí SQL injection.

Prvním krokem je spuštění webového prohlížeče Iceweasel v prostředí Kali Linux, zadání IP adresy *192.168.56.101* (OWASP Broken Web Applications) a zvolení webové aplikace OWASP Mutillidae II.

Pro aplikaci SQL injection byl zvolen nástroj Sqlmap spolu s nástrojem Burp Suite a ve webové aplikaci OWASP Mutillidae II model zneužití prostřednictvím *Login* [1] (přihlášení), viz obr. 4.3 .

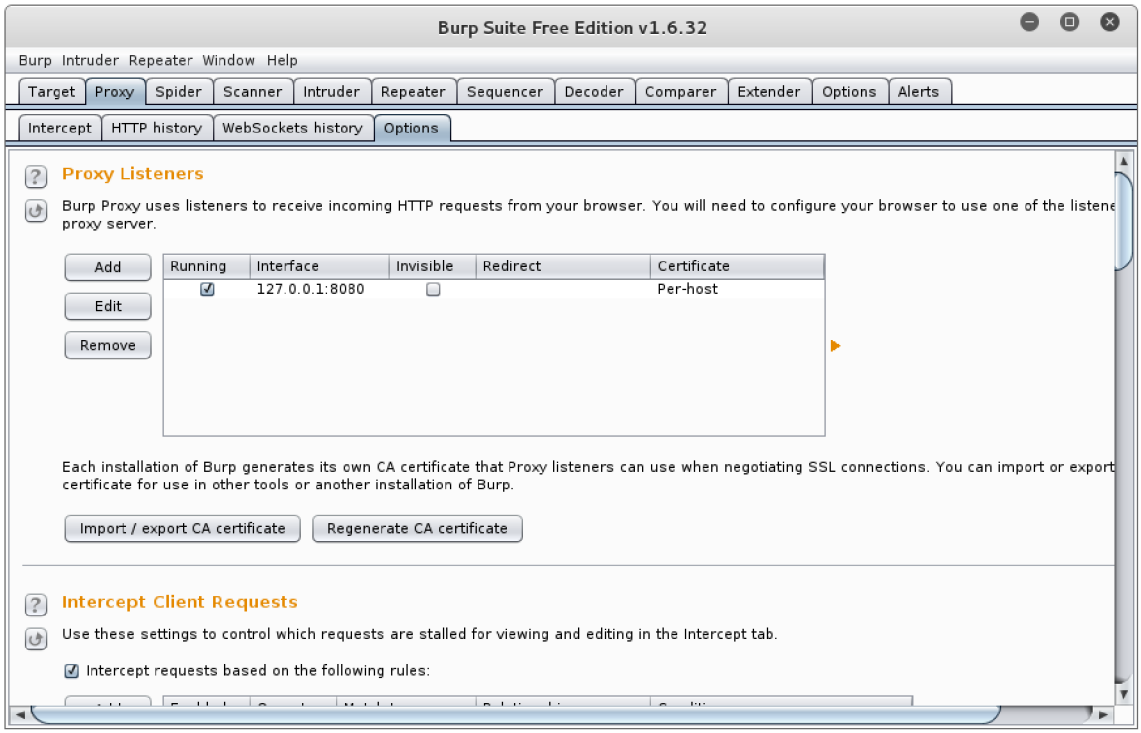


Obr. 4.3: Model zneužití zranitelnosti - *Login*

[1] „OWASP 2013“ -> „A1 - Injection (SQL)“ -> „SQLMAP Practice“ -> „Login“

Vzhledem k tomu, že je nemožné v tomto případě použít metodu GET (data jsou vidět v URL adrese), je potřeba zachytit HTTP komunikaci webového prohlížeče s webovým serverem a využít metodu POST (data nejsou vidět v URL adrese a přenášejí se skrytě prostřednictvím HTTP komunikace). Zachycení HTTP komunikace bude provedeno nástrojem Burp suite.

Po zapnutí nástroje Burp suite se přepneme do záložky ***Proxy/Options***, viz obr. 4.4 , kde se nastaví, na jakém portu bude nástroj naslouchat *8080*, IP adresu *127.0.0.1* a ujistíme se, že je aktivní (*Running*).



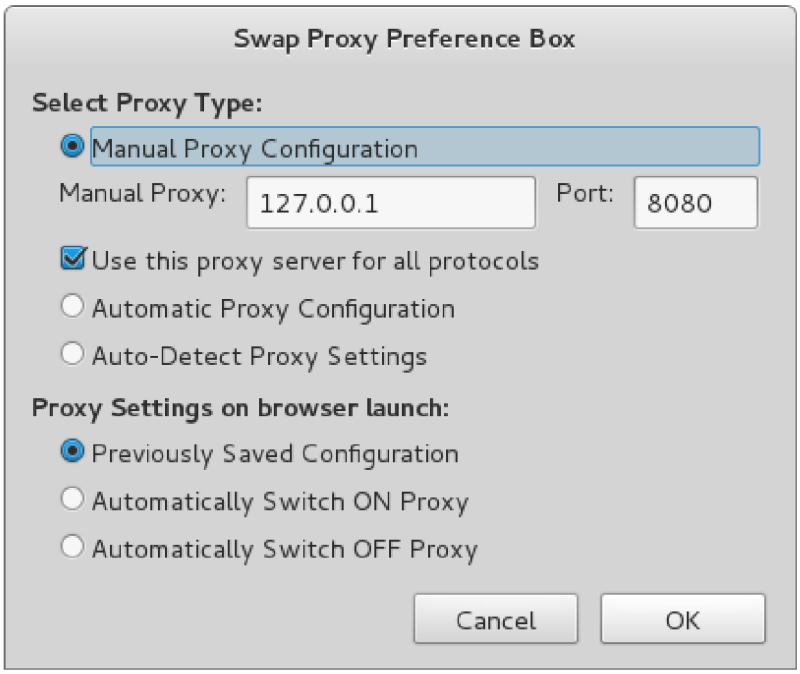
Obr. 4.4: Nastavení proxy portu a IP adresy

Dále je potřeba nastavit ve webovém prohlížeči stejné údaje, adresu HTTP proxy *127.0.0.1* a port *8080*. Po nastavení proxy serveru nebude možné používat prohlížeč k běžnému surfování po internetu, dokud nepřepneme zpět proxy na původní hodnoty. Z tohoto důvodu je v prohlížeči Iceweasel doinstalován doplněk Swap Proxy, který slouží k rychlému přepínaní mezi proxy servery, viz obr. 4.5.

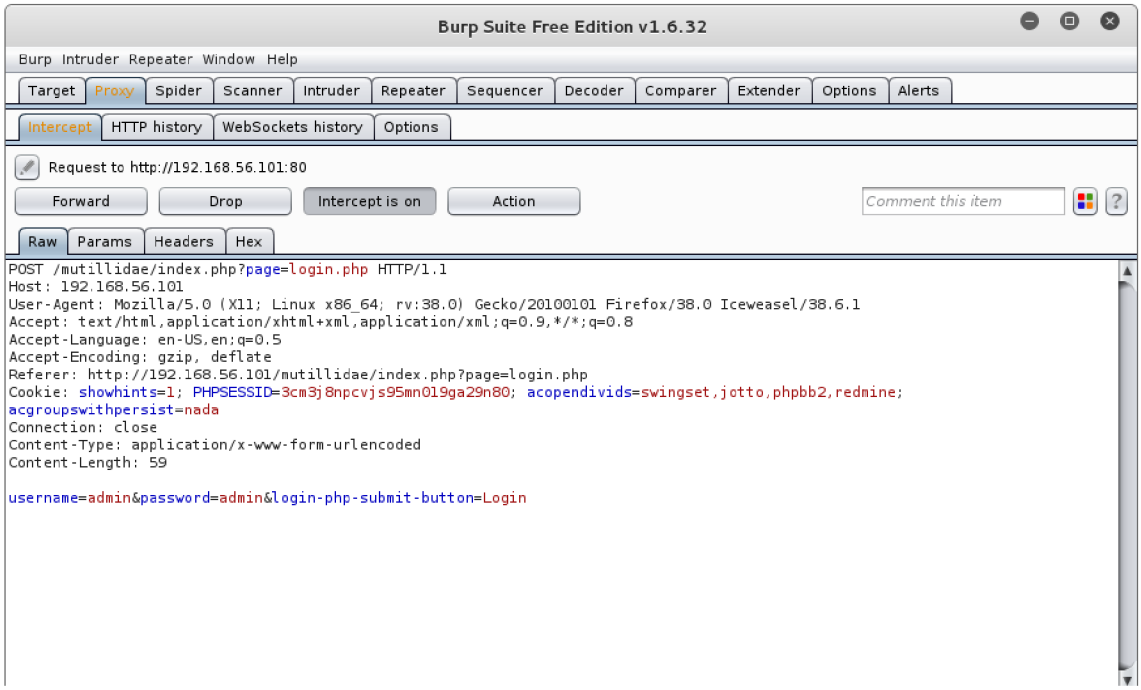
Po nastavení proxy serverů je již možné odchytit HTTP komunikaci. Vyplněním údajů username (uživatelské jméno), password (heslo) a potvrzením tlačítkem *Login* se zachytí HTTP komunikace, kterou lze vidět v záložce ***Proxy/Intercept*** v nástroji Burp Suite, viz obr. 4.6. Zde je vidět celý náš požadavek, který je možné si prohlédnout v záložkách:

- **Raw** - zobrazuje celý požadavek, který je odesílán webovému serveru.

- **Params** - zobrazuje názvy a hodnoty předávaných parametrů.
- **Headers** - zobrazuje HTTP hlavičky.
- **Hex** - zobrazuje cely požadavek v hexadecimálním zobrazení.



Obr. 4.5: Nastavení proxy ve webovém prohlížeči Iceweasel



Obr. 4.6: Zachycení HTTP komunikace

Kromě spousty užitečných informací v celém požadavku (metoda POST, kódování, user-agent, cookie), jsou nejdůležitější informací přímo zachycená data, tedy údaje username *admin* a password *admin*, které byly vloženy do požadavku tlačítkem *Login* a formát mají následující:

```
username=admin&password=admin&login-php-submit-button=Login
```

Zadané údaje sice nebyly správné pro úspěšné přihlášení do webové aplikace, ale nástroj Sqlmap bude potřebovat výše zmíněný formát dat k pokusu o SQL injection metodou POST. Získáním těchto dat bylo cílem nástroje Burp Suite a nyní lze použít nástroj Sqlmap.

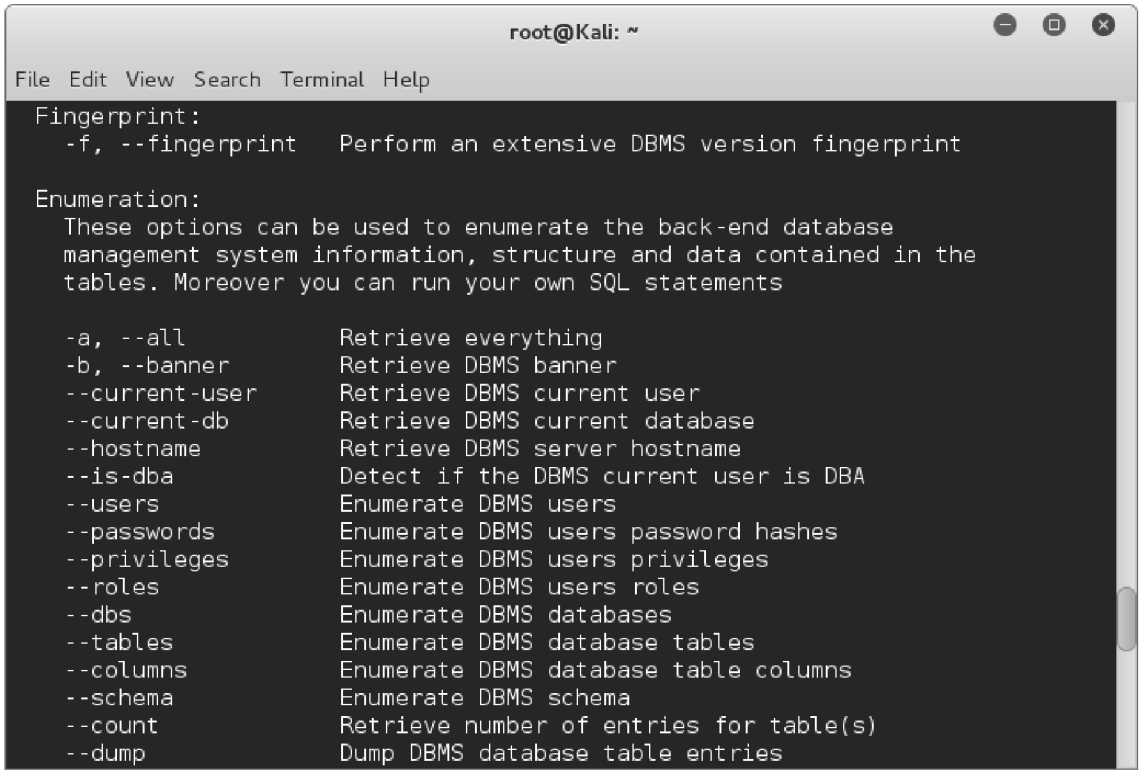
Sqlmap

Sqlmap se ovládá přes příkazový řádek a první věc, kterou by měl každý nezkušený uživatel udělat po spuštění terminálu, je zjistit, jaké možnosti nástroj poskytuje. Příkaz:

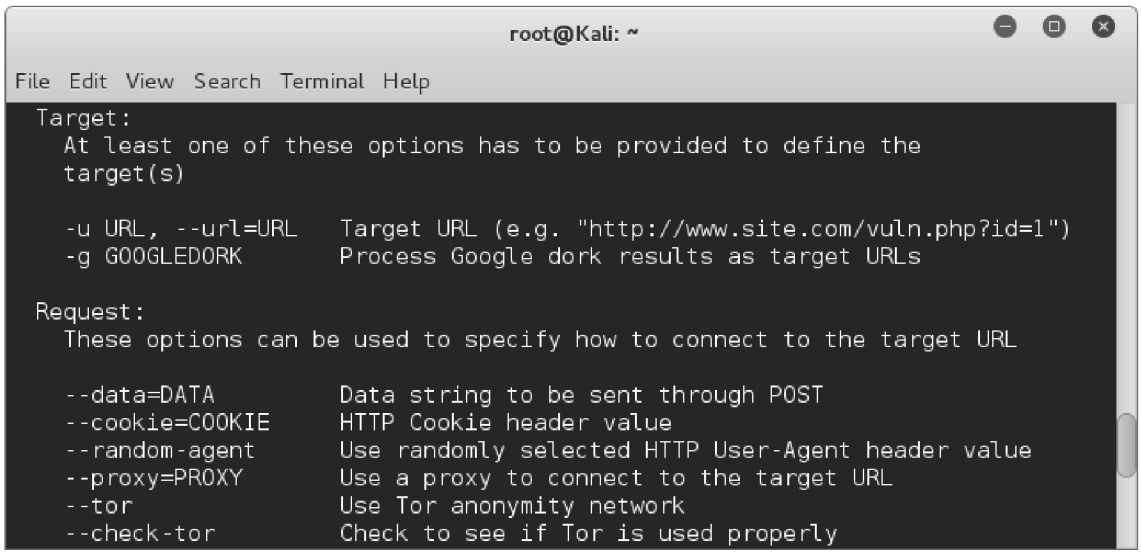
root@Kali:~# **sqlmap -help**

sqlmap - jméno binárního souboru

-help - zobrazení nápovědy



Obr. 4.7: Volby *Enumeration* a *Fingerprint*



Obr. 4.8: Volby *Target* a *Request*

Zobrazí se dlouhý výpis voleb a jejich příkazů, mezi které např. patří:

- **Target** - volba, která musí být vždy zadána a určuje cíl.
- **Request** - volby, které stanovují, jakým způsobem se připojíme k cílové URL.
- **Enumeration** - volby, které se používají k získání informací o systémové databázi, struktuře a údajů obsažených v tabulkách.
- **Fingerprint** - výpis konkrétní verze databáze.

Pro tuto úlohu jsou nejpodstatnější příkazy, obsažené pod výše zmíněnými volbami *Enumeration*, *Fingerprint*, viz obr. 4.7 a *Target*, *Request*, viz obr. 4.8.

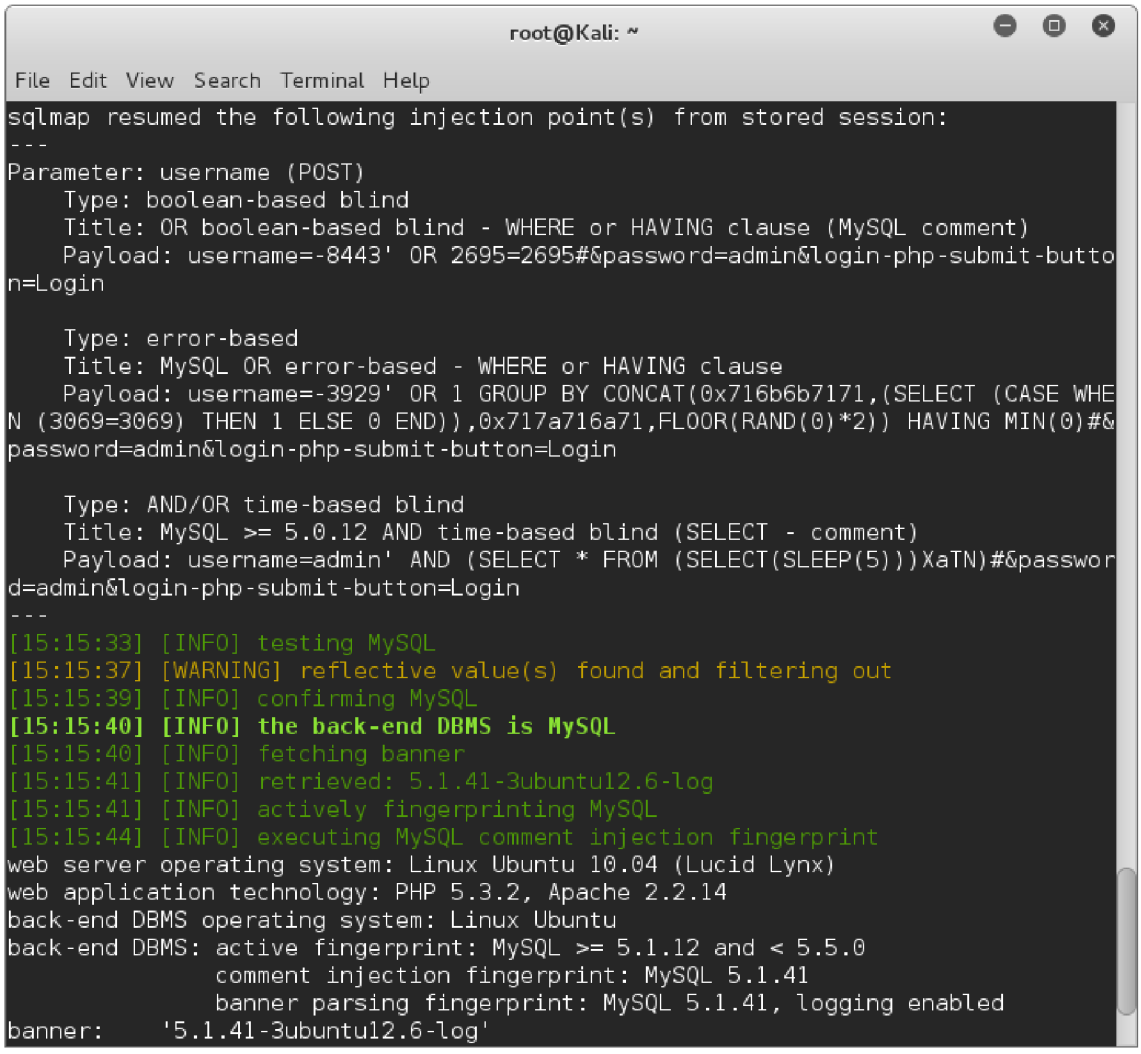
Dalším krokem je shromáždění základních informací o daném cíli, tzn. je potřeba vědět, co vlastně testujeme, protože spousty exploitů jsou velmi citlivé na konkrétní OS, služby a aplikace. V tomto případě je dobré vědět, o jaký DBMS (Database Management System - systém řízení báze dat) ve skutečnosti jde. Dále se přesvědčíme, zda-li je nástroj opravdu schopen použít SQL injection na cílovou webovou aplikaci. Příkaz:

root@Kali:~# **sqlmap -u** "http://192.168.56.101/mutillidae/index.php?page=login.php"–**data=**"username=admin&password=admin&login-php-submit-button=Login"**-f -b**

sqlmap - jméno binárního souboru
-u - označení cílové URL
–**data=** - datový řetězec poslán metodou POST
-f - výpis konkrétní verze DBMS
-b - výpis „banneru“ DBMS

Během testování se nás může program dotázat, jakým způsobem chceme pokračovat. Pro tuto úlohu to jsou následující dotazy a odpovědi:

- It looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y
- For the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] y
- Sqlmap got a 302 redirect to 'http://192.168.56.101:80/mutillidae/index.php?popUpNotificationCode=AU1'. Do you want to follow? [Y/n] y
- Redirect is a result of a POST request. Do you want to resend original POST data to a new location? [y/N] y
- POST parameter 'username' is vulnerable. Do you want to keep testing the others (if any)? [y/N] n



Obr. 4.9: Výpis úspěšné SQL injection, *fingerprint* a „banneru“

Po ukončení testu terminál vypíše informace, viz obr. 4.9, ze kterých kromě

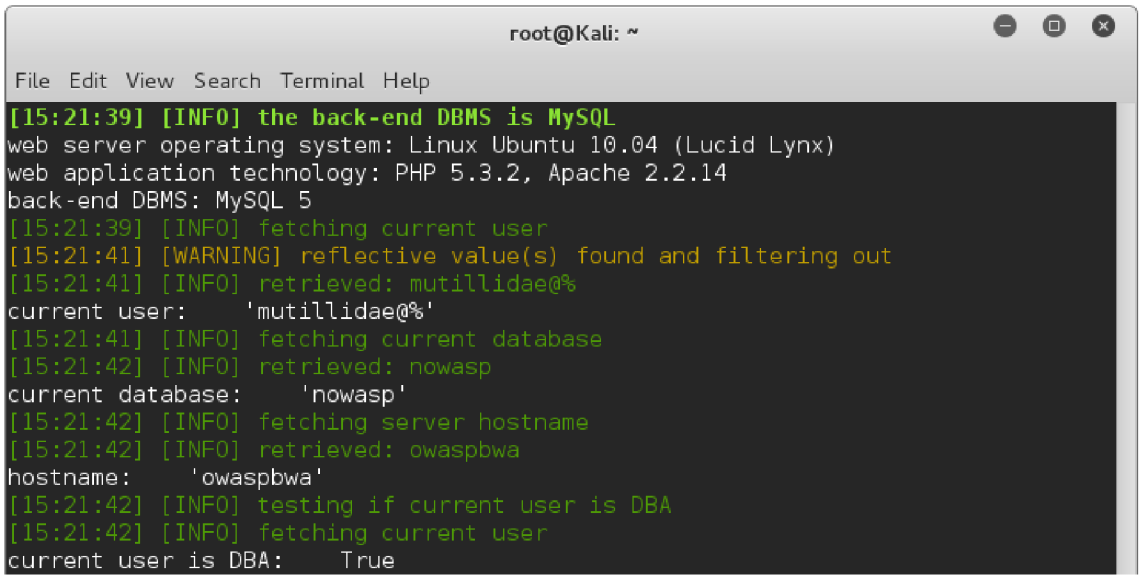
potvrzení o úspěšné aplikaci SQL injection lze zjistit následující:

- Cílový parametr je *username* a použitá metoda *POST*.
- Aplikovány byly 3 typy SQL injection (*boolean-based blind, error-based, AND/OR time-based blind*), kde nástroj uvádí jejich *Title* (záhlaví) a *Payload* (škodlivý kód).
- Technologie webové aplikace *PHP 5.3.2, Apache 2.2.14* a OS webového serveru *Linux Ubuntu 10.04*.
- Typ DBMS je *MySQL 5.1.41*, kde přesnou verzi zjistila metoda *fingerprint* rozborem „banneru“ a aplikací *comment injection*.

Dalším krokem bude zjištění jména aktuálního uživatele, aktuální databáze, názvu serveru, a zda-li je aktuální uživatel *DBA* (Database Administrator).
Příkaz:

root@Kali:~# **sqlmap -u** "http://192.168.56.101/mutillidae/index.php?page=login.php"–**data=**"username=admin&password=admin&login-php-submit-button=Login"–**hostname** –**is-dba** –**current-user** –**current-db**

sqlmap - jméno binárního souboru
-u - označení cílové URL
–**data=** - datový řetězec poslán metodou POST
–**hostname** - výpis názvu serveru DBMS
–**is-dba** - ověření, zda-li je aktuální uživatel *DBA* systému DBMS
–**current-user** - výpis aktuálního uživatele DBMS
–**current-db** - výpis aktuální databáze DBMS



Obr. 4.10: Výpis *current user*, *current database*, *hostname* a *DBA*

Z obrázku 4.10 je vidět, že nástroj úspěšně vypsal informace, které jsme požadovali. Aktuální uživatel je *mutillidae@%* a je *DBA*, aktuální databáze se jmenuje *nowasp* a název serveru je *owaspbwa*.

V tomto bodě jsme shromáždili dostatek informací o webové aplikaci a nyní je potřeba zjistit, jaké databáze jsou obsaženy v databázovém systému MySQL.
Příkaz:

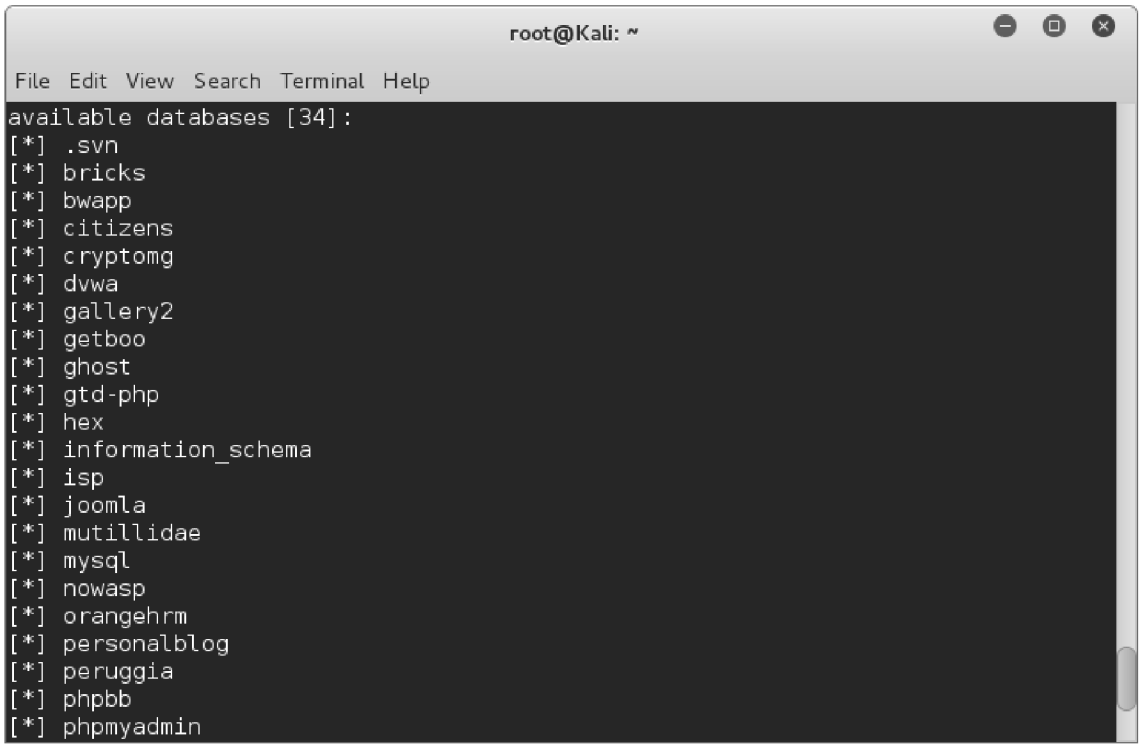
root@Kali:~# **sqlmap -u** "http://192.168.56.101/mutillidae/index.php?page=login.php"–**data=**"username=admin&password=admin&login-php-submit-button=Login"–**dbs**

sqlmap - jméno binárního souboru

-u - označení cílové URL

–**data=** - datový řetězec poslán metodou POST

–**dbs** - výpis databází DBMS



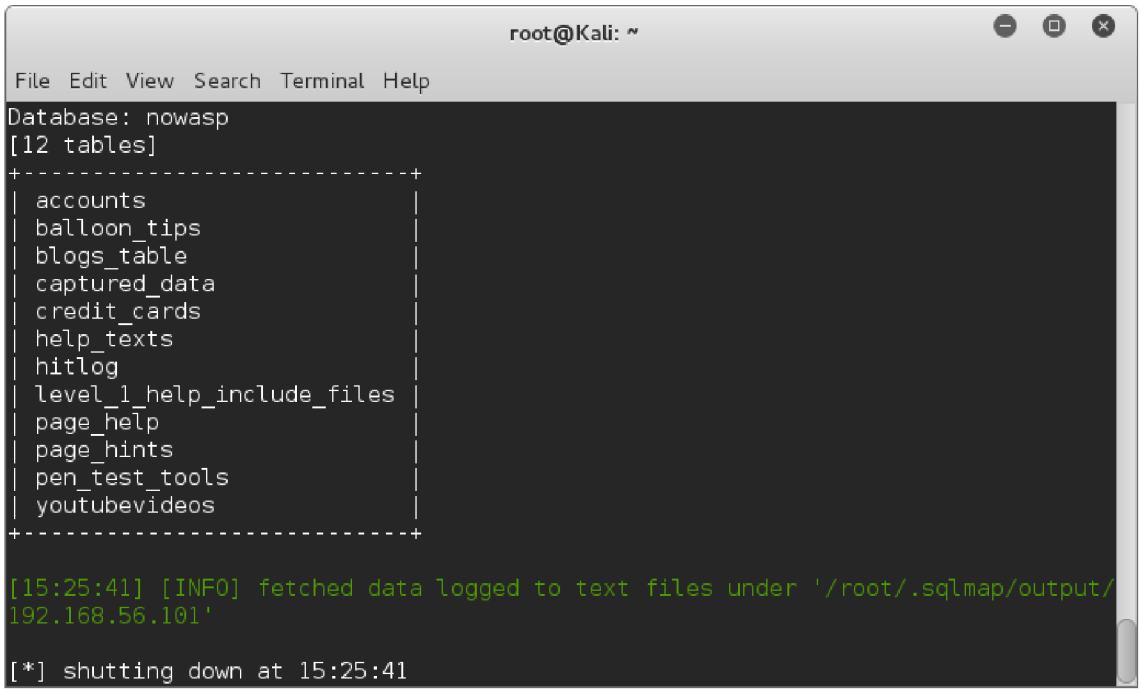
Obr. 4.11: Výpis *dbs*

Vzhledem k dlouhému seznamu databází je výpis zkrácen, viz obr. 4.11. *Information_schema* je součástí každé systémové databáze MySQL a obsahuje informace o všech objektech v instanci MySQL. V seznamu je uvedená i databáze *nowasp*, kterou jsme zjistili v předchozím kroku a právě na tu je potřeba se zaměřit. Každá

databáze se skládá z tabulek (*tables*) a ty obsahují nějaké sloupce (*columns*) s hodnotami. Nejdříve tedy odhalíme, jaké obsahuje tabulky.
Příkaz:

root@Kali:~# **sqlmap -u** "http://192.168.56.101/mutillidae/index.php?page=login.php"–**data=**"username=admin&password=admin&login-php-submit-button=Login"**-D** nowasp –**tables**

sqlmap - jméno binárního souboru
-u - označení cílové URL
–**data=** - datový řetězec poslán metodou POST
-D - určuje databázi DBMS, kterou má nástroj vypsat
–**tables** - výpis tabulek

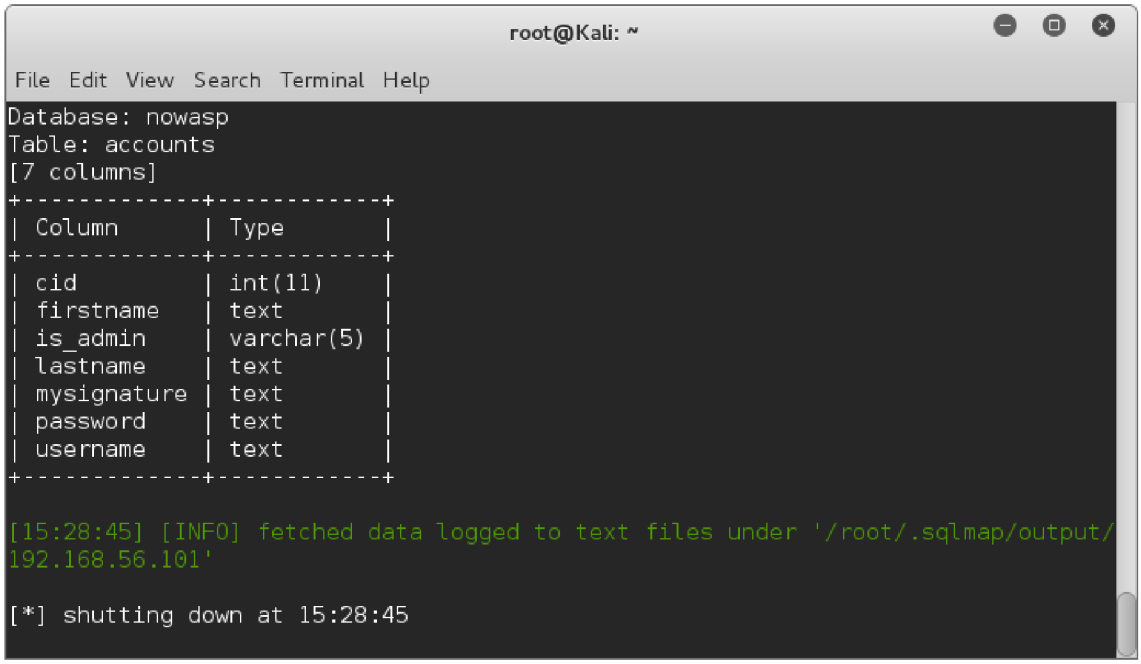


Obr. 4.12: Výpis *tables* databáze *nowasp*

Databáze *nowasp* obsahuje 12 tabulek, viz obr. 4.12. Tabulka, která představuje cíl této úlohy a disponuje potřebnými informacemi, je *accounts* (účty). Zjistíme tedy, jaké obsahuje sloupce.
Příkaz:

root@Kali:~# **sqlmap -u** "http://192.168.56.101/mutillidae/index.php?page=login.php"–**data=**"username=admin&password=admin&login-php-submit-button=Login"**-D** nowasp **-T** accounts –**columns**

sqlmap - jméno binárního souboru
-u - označení cílové URL
–**data=** - datový řetězec poslán metodou POST
-D - určuje databázi DBMS, kterou má nástroj vypsat
-T - určuje tabulku DBMS, kterou má nástroj vypsat
–**columns** - výpis sloupců



Obr. 4.13: Výpis *columns* tabulky *accounts*

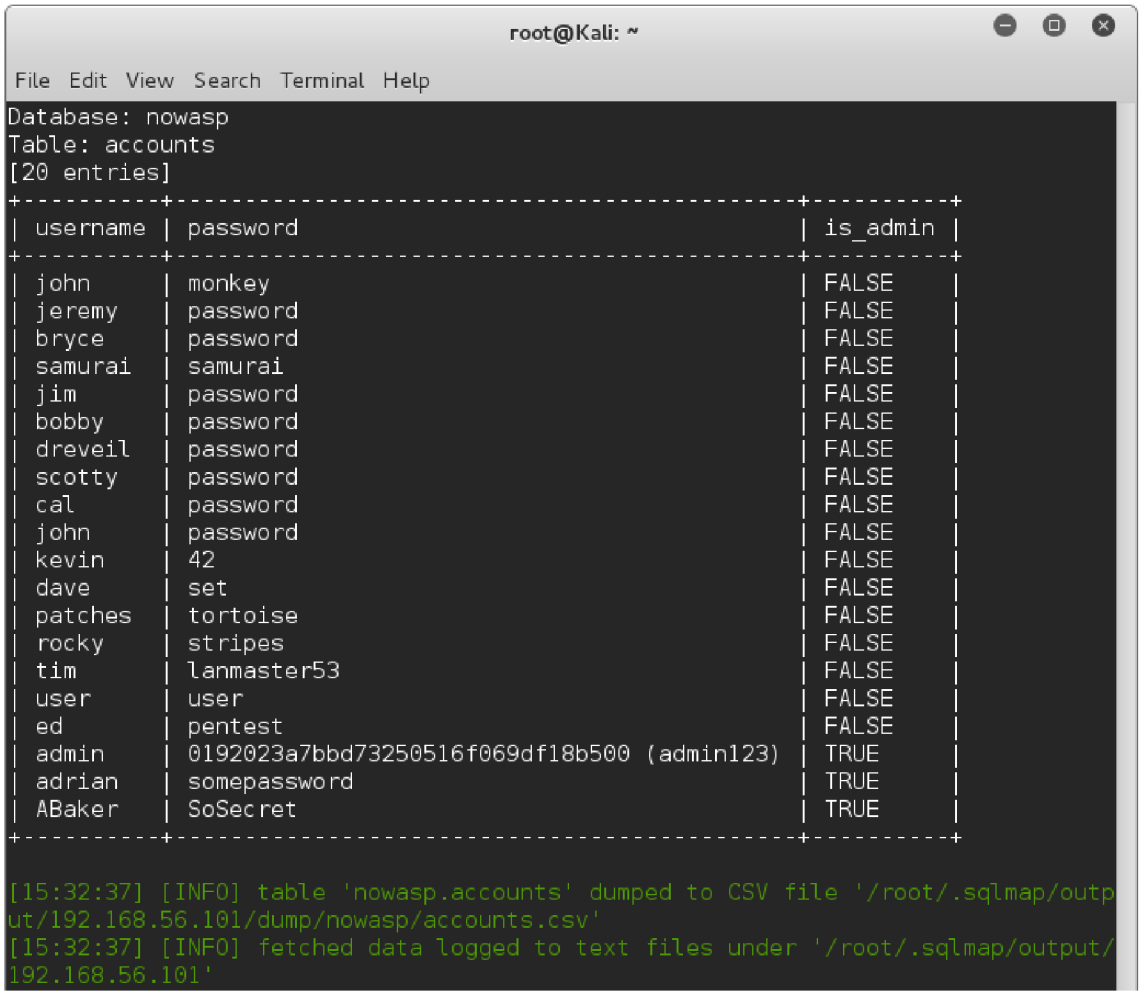
Z tabulky *accounts* jsme zjistili, že obsahuje 7 sloupců, viz obr. 4.13. Posledním krokem je vypsat záznamy sloupců *username, password, is_admin* a dozvědět se kromě uživatelských přihlašovacích údajů, zda-li mají administrátorské oprávnění. Příkaz:

root@Kali:~# **sqlmap -u** "http://192.168.56.101/mutillidae/index.php?page=login.php"–**data=**"username=admin&password=admin&login-php-submit-button=Login"**-D** nowasp **-T** accounts **-C** username,password,is_admin –**dump**
sqlmap - jméno binárního souboru
-u - označení cílové URL
–**data=** - datový řetězec poslán metodou POST
-D - určuje databázi DBMS, kterou má nástroj vypsat
-T - určuje tabulku DBMS, kterou má nástroj vypsat
-C - určuje sloupec DBMS, který má nástroj vypsat

–**dump** - výpis záznamů vybraných sloupců

Pokud nějaká hodnota (nejčastěji heslo) používá nějaký typ šifrování, pokusí se nástroj o dešifraci a dotáže se nás, jak chceme pokračovat. Odpovědi na dotazy byly zvoleny následovně:

- Do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] y
- Do you want to crack them via a dictionary-based attack? [Y/n/q] y
- What dictionary do you want to use?
  (1) default dictionary file '/usr/share/sqlmap/txt/wordlist.zip' (press Enter)
  (2) custom dictionary file
  (3) file with list of dictionary files
  > 1
- Do you want to use common password suffixes? (slow!) [y/N] n



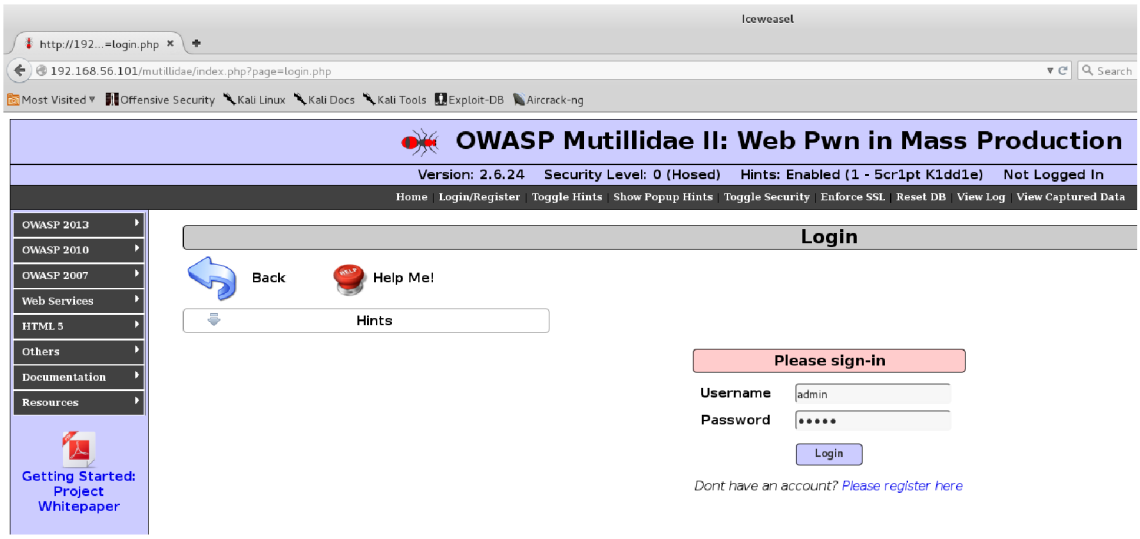
Obr. 4.14: Výpis záznamů sloupců *username*, *password* a *is_admin*

Vypsané záznamy, viz obr. 4.14, nám poskytly informace, které byly cílem této úlohy. Uživatelských účtů je celkově 20, z toho 3 mají administrátorská práva (*ABaker, adrian* a *admin*). Účet *admin* používá šifrování hesla typu MD5, které se podařilo úspěšně dešifrovat pomocí dictionary attack (slovníkový útok). Informace uvnitř tabulky *nowasp.accounts* nástroj zaznamenal do souboru *accounts.csv*, včetně cesty k němu. Také shromáždil veškerá data do adresáře *192.168.56.101*.

4.3 Zranitelnost A2 - Broken Authentication and Session Management

Cílem této úlohy je získání uživatelských přihlašovacích údajů webové aplikace OWASP Mutillidae II využitím metody dictionary attack (slovníkový útok).

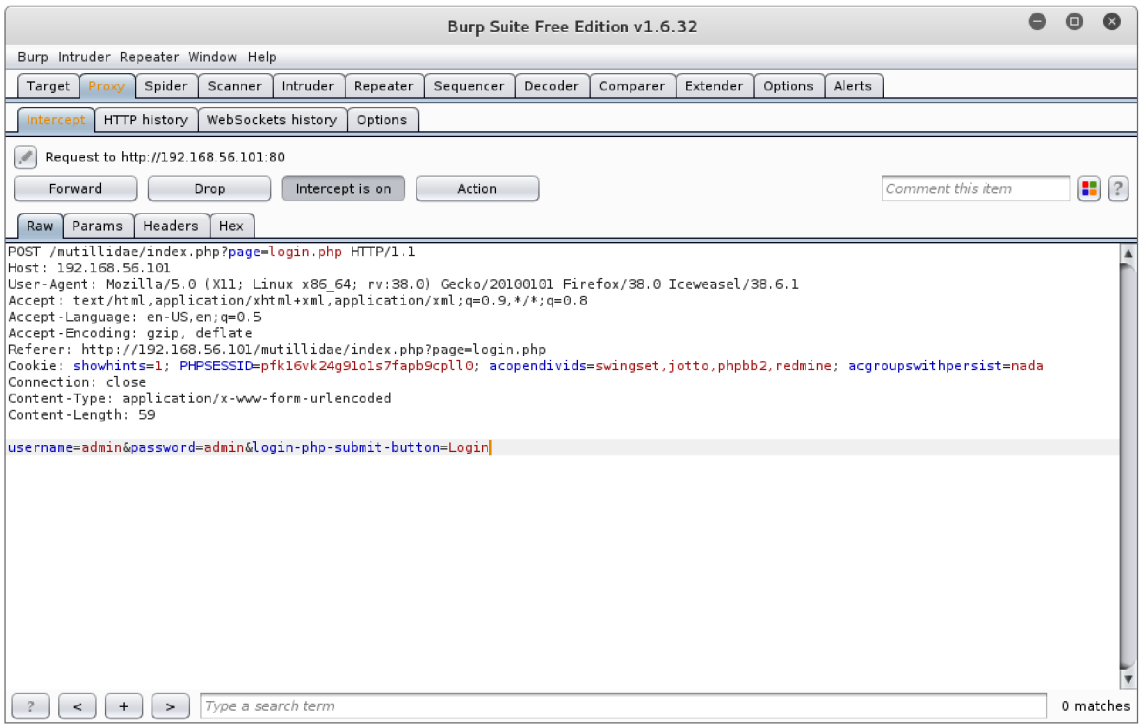
Pro demonstraci útoku byl zvolen nástroj Burp Suite a ve webové aplikaci OWASP Mutillidae II model pro zneužití opět prostřednictvím *Login* [2], viz obr. 4.15.



Obr. 4.15: Model zneužití zranitelnosti - *Login*

Již známým postupem je potřeba nastavit proxy servery webového prohlížeče Iceweasel a nástroje Burp Suite pro zachytávaní HTTP komunikace. Po nastavení proxy serveru vyplněním přihlašovacích údajů username *admin* a password *admin* a potvrzení tlačítkem *Login* dojde k zachycení HTTP komunikace nástrojem Burp suite, viz obr. 4.16.

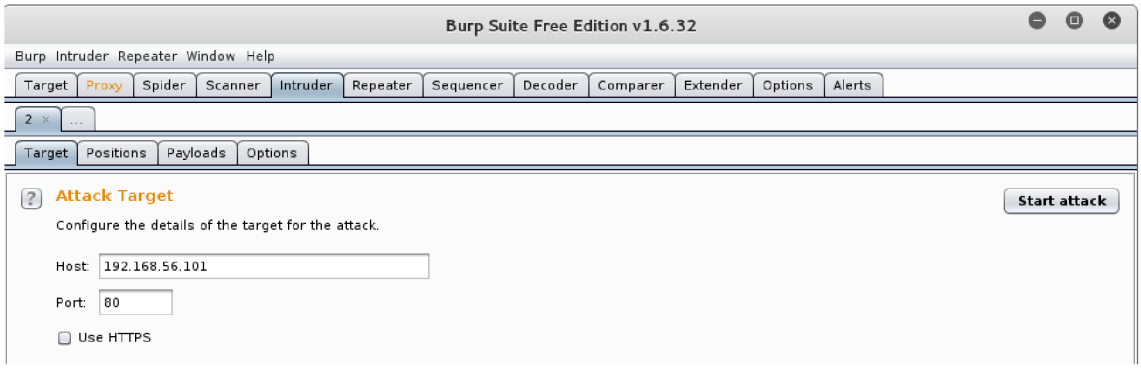
[2] „OWASP 2013“ -> „A2 - Broken Authentication and Session Management“ -> „Authentication Bypass“ -> „Via Brute Force“ -> „Login“



Obr. 4.16: Zachycení HTTP komunikace

Nyní kliknutím pravého tlačítka myši kdekoliv v okně s požadavkem se ukáže menu, ze kterého zvolíme volbu *Send to Intruder*. Intruder je modul, prostřednictvím kterého lze vykonávat útoky typu dictionary attack a brute force, manipulovat s parametry, detekovat zranitelnosti SQL injection a cross-site scripting.

Přepnutím se do záložky ***Intruder/Target*** zkontrolujeme adresu našeho cíle *192.168.56.101* a port *80*, viz obr. 4.17.

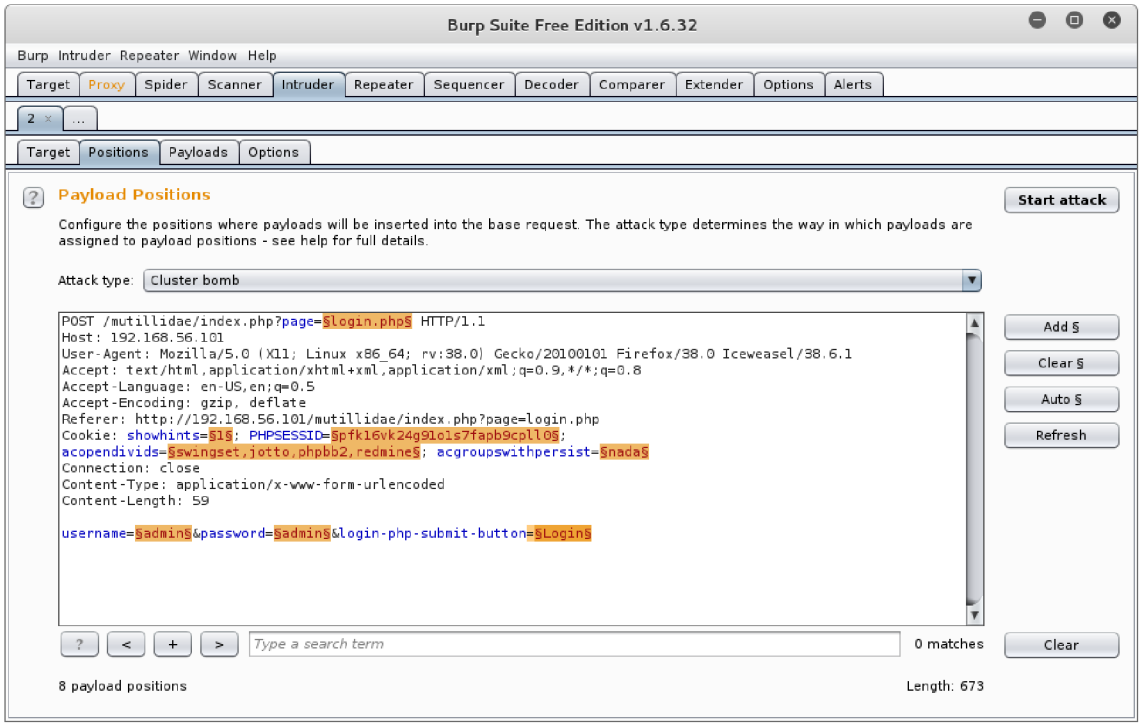


Obr. 4.17: Volba adresy cíle a portu

V záložce ***Intruder/Positions*** je potřeba určit typ útoku a zvolit místa v požadavku, na která bude Intruder vkládat námi zadané hodnoty. V našem požadavku Intruder automaticky vyznačil místa, vhodná pro vkládání hodnot. Tato místa jsou

označena paragrafem § a barevně zvýrazněna, viz obr. 4.18. V tomto případě si sami chceme zvolit proměnné hodnoty k testování. Kliknutím na tlačítko *Clear §* se všechny automaticky zvolené proměnné hodnoty vymažou. Nyní zvolíme pouze hodnotu proměnných username a password (*„admin“*) a kliknutím na tlačítko *Add §* zadefinujeme místa, na která bude Intruder vkládat uživatelská jména a hesla ze slovníku a porovnávat je, čímž bude zjišťovat, zda-li daná kombinace je ve webové aplikaci registrována či nikoliv.

Attack type (typ útoku) zvolíme *Cluster bomb*. Ten umožňuje v požadavku označit až 8 míst k porovnávání a je nejvhodnější pro hledání správné kombinace uživatelského jména a hesla.



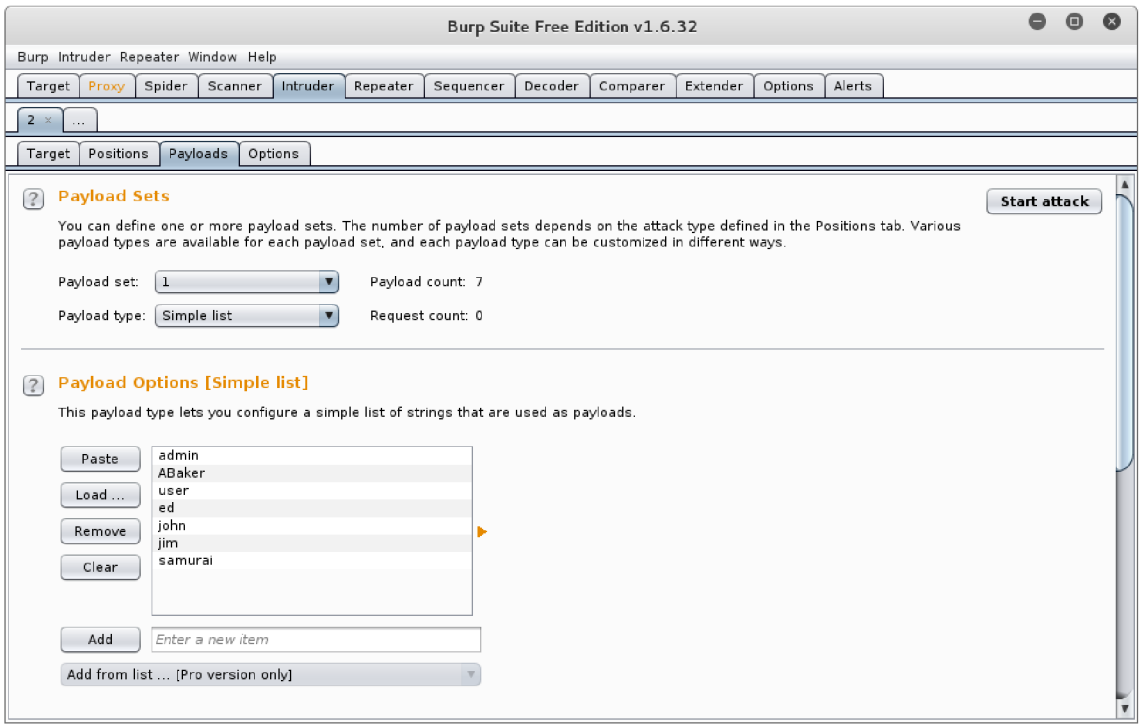
Obr. 4.18: Volba typu útoku a vstupních hodnot

Záložka ***Intruder/Payloads*** je posledním bodem nutným před zahájením útoku. Zde musíme určit, jaké hodnoty se budou na zvolená místa vkládat.

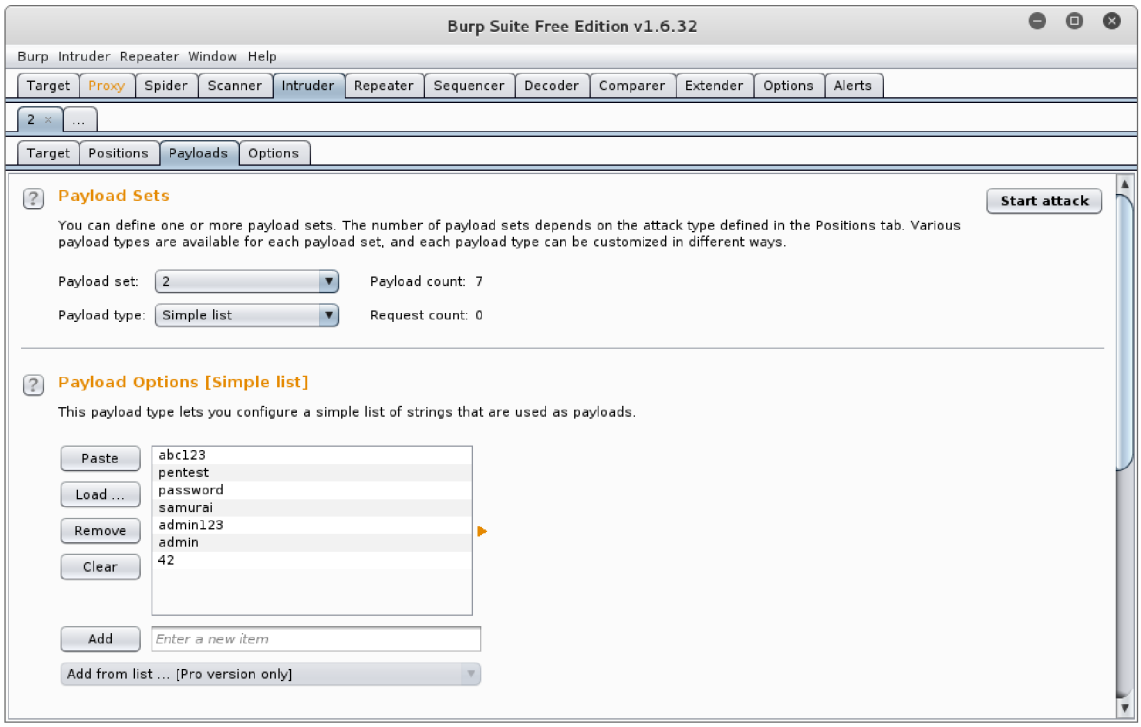
V Payload set bude po rozbalení seznamu na výběr z tolika možností, kolik jsme jich v požadavku označili, tedy v našem případě 2, kde vybereme nejprve *1*.

Payload type určuje požadovaný typ „payloadu“ z rozbalovacího seznamu. Tedy v tomto případě stačí zvolit *Simple list*, který je určen pro jednoduché srovnávání několika hodnot ze slovníku.

Payloads Options upřesňuje, jaké hodnoty chceme srovnávat. Kliknutím na tlačítko *Load ...* vybereme soubor *userlist.txt*, který obsahuje 7 uživatelských jmen z databáze pro demonstraci útoku, viz obr. 4.19.



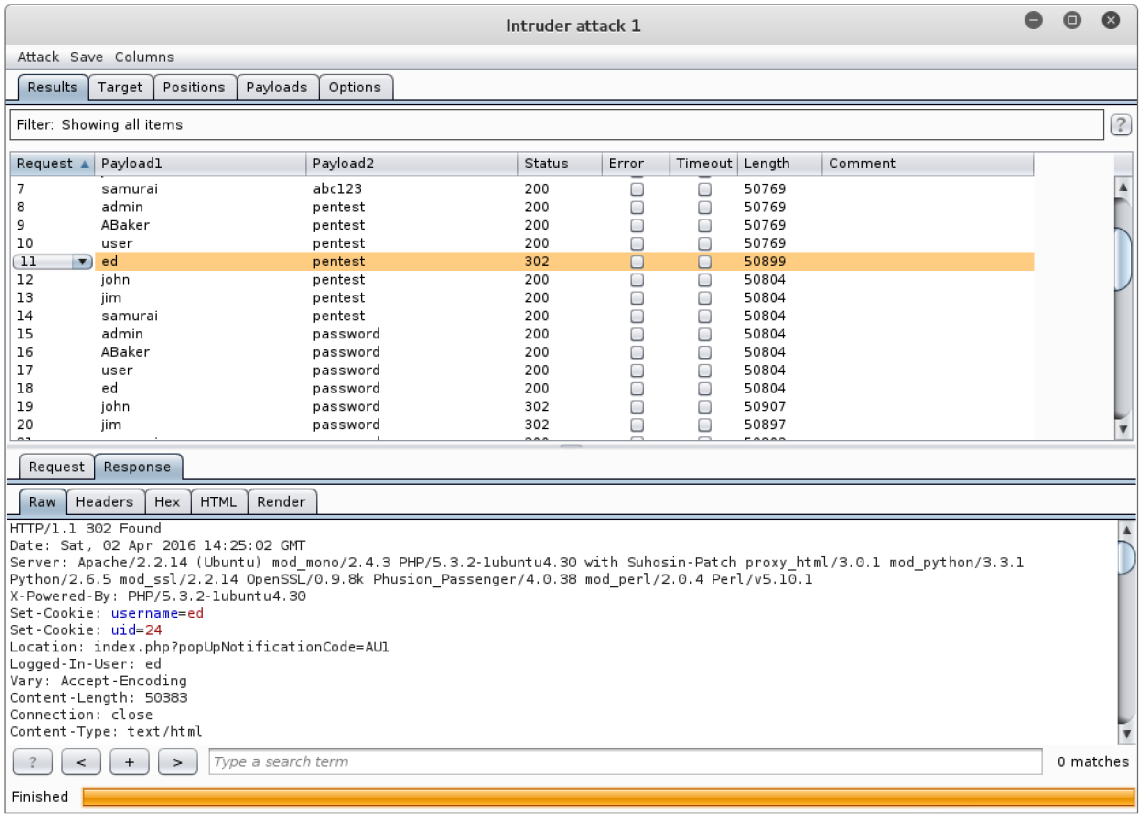
Obr. 4.19: Volba prvních vstupních hodnot (uživatelská jména)



Obr. 4.20: Volba druhých vstupních hodnot (uživatelská hesla)

Přepnutím Payload set na číslo *2* a kliknutím na tlačítko *Load ...* vybereme

soubor *passlist.txt*, který obsahuje 7 uživatelských hesel, viz obr. 4.20. V této chvíli stačí kliknout na tlačítko *Start attack*, a zahájit útok.



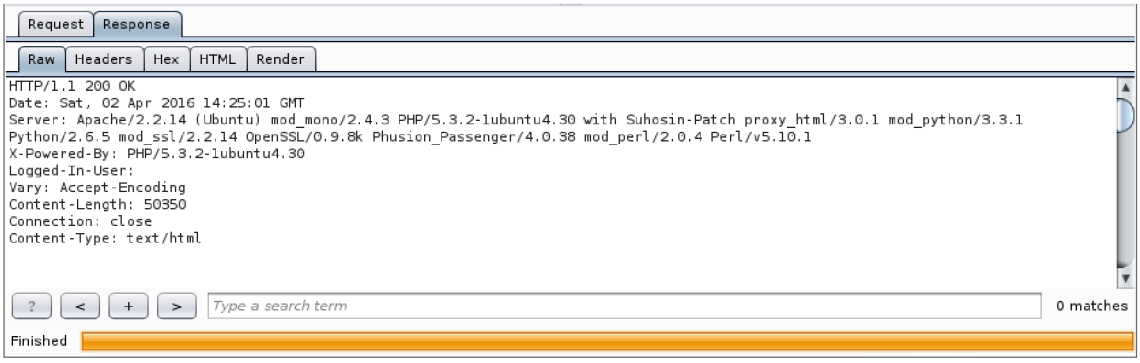
Obr. 4.21: Výsledky slovníkového útoku

Po dokončení útoku nástroj zobrazí výsledky, viz obr. 4.21. Celkově provedl 49 požadavků (Request). Z obrázku je vidět pouze část jednotlivých požadavků (kombinací) uživatelských jmen (Payload1), hesel (Payload2), statusu (Status) a délky požadavku (Length).

Pokud chceme vědět, jaké kombinace jsou správné, tak nejjednodušším způsobem je zkontrolovat číslo statusu. Status s číslem *302* značí shodu uživatelského jména a hesla ve webové aplikaci. Tento případ platí pro požadavky *11, 19* a *20*.

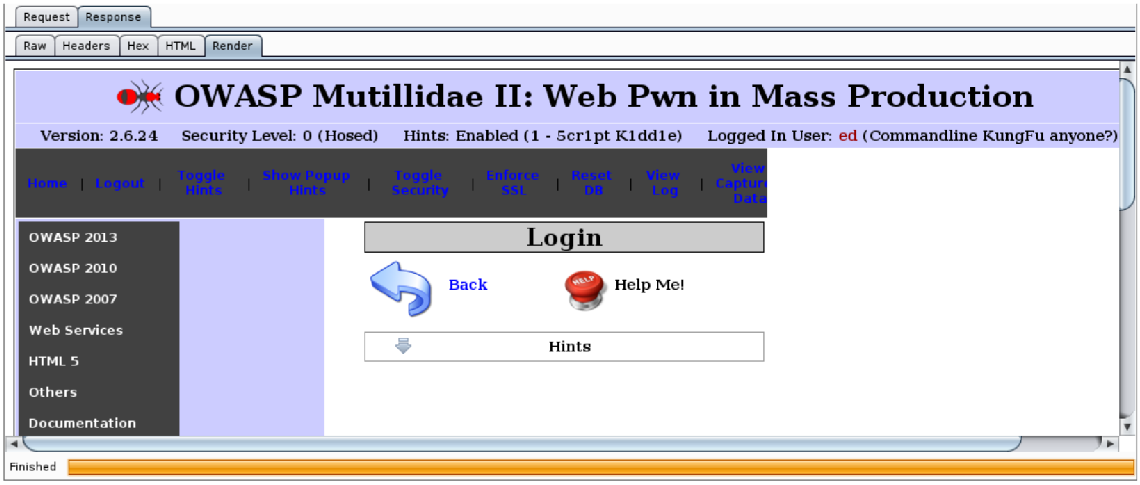
Dalším způsobem je podívat se na odpověď (Response) od serveru. Požadavek číslo *11* srovnával uživatelské jméno *ed* s heslem *pentest*. Při přepnutí do záložky ***Response/Raw*** je vidět odpověď od webového serveru. Je-li kombinace správná, budou zde položky *Set-Cookie, Location* a u *Logged-In-User* uvedené jméno uživatele.

Kombinace uživatelského jména *user* a hesla *pentest* je nesprávná a odpověď od webového serveru vypadá jako na obr. 4.22 s označením statusu *200*.



Obr. 4.22: Odpověď webového serveru při nesprávné kombinaci

Nástroj také umí vykreslit do jisté míry podobu webové stránky z požadavku a odpovědi webového serveru v záložce ***Response/Render***, viz obr. 4.23. Je to také další možnost, jak si ověřit, zda-li je kombinace uživatelského jména *ed* a hesla *pentest* správná. Zde tomu tak je a a jsme přihlášeni jako uživatel *ed*.



Obr. 4.23: Přibližná podoba webová stránky

Celkem ze 49 kombinací byly správné 4:

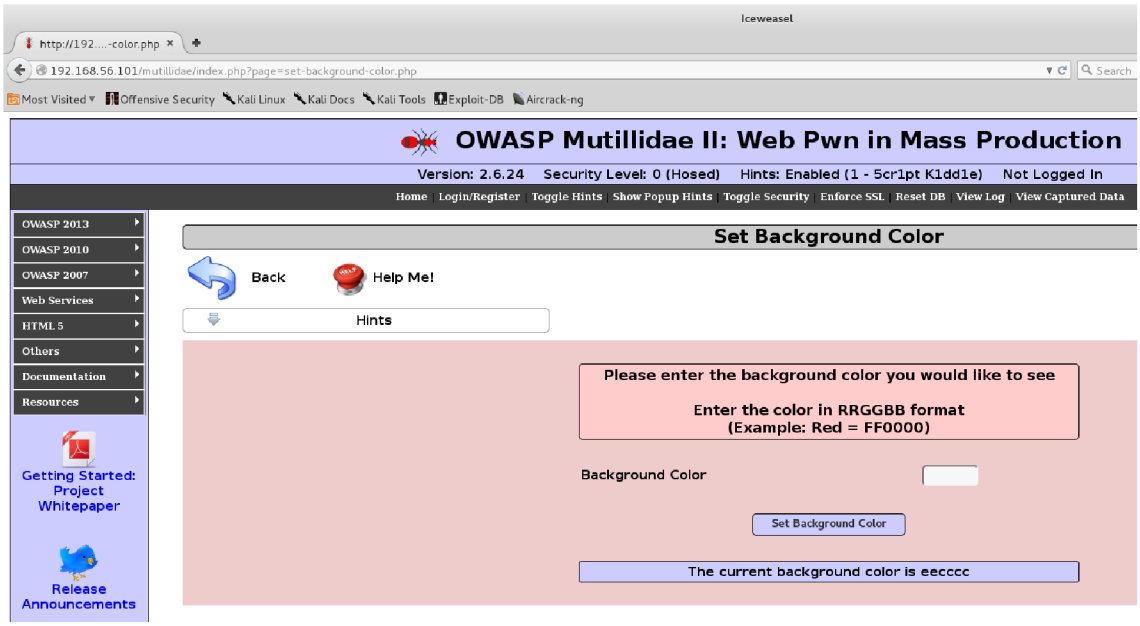
- *ed/pentest*
- *john/password*
- *jim/password*
- *samurai/samurai*

4.4 Zranitelnost A3 - Cross-Site Scripting

Cílem této úlohy je vyhledání potenciálně nebezpečných skriptů nástrojem XSSer, které mohou mít vliv na webovou aplikaci OWASP Mutillidae II, vybrat nějaký ze

skriptů a demonstrovat metodu **Reflected** i **Stored** (viz kapitola 2.2).

Ve webové aplikaci OWASP Mutillidae II byl zvolen model pro zneužití *Set Background Color* [3], viz obr. 4.24. Tento model byl zvolen především z toho důvodu, že nástroj XSSer vyžaduje určitou formu adresy URL pro aplikaci škodlivého kódu.



Obr. 4.24: Model zneužití zranitelnosti - *Set Background Color*

Otevřením terminálu a zadáním následujícího příkazu:

```
root@Kali:~# xsser –gtk
```

dojde ke spuštění graficky uživatelského rozhraní, viz obr. 4.25. K zahájení útoku je nutné přepnout Fly mode na *Intruder*, což zajistí testování cílové URL adresy přímo námi vložené. Vložená URL adresa:

```
http://192.168.56.101/mutillidae/index.php?page=set-background-color.php
```

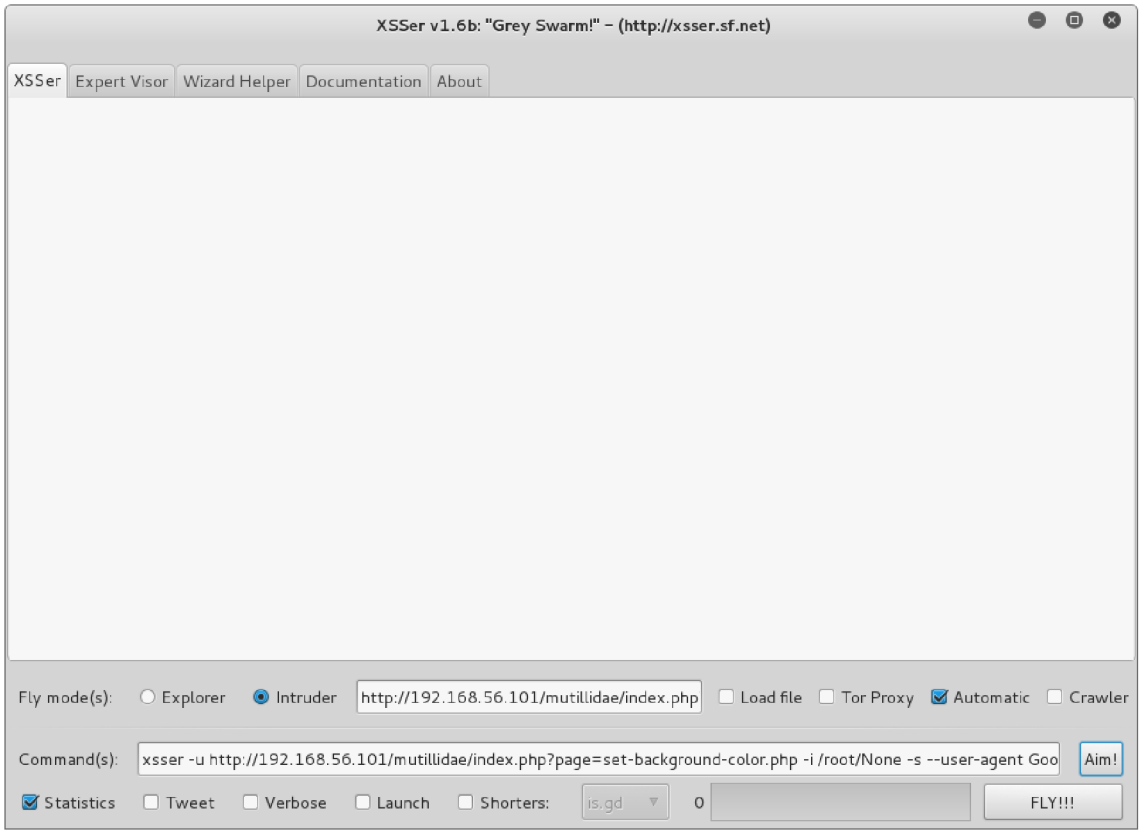
Dále je nutné zaškrtnout režim testování na *Automatic* a kliknout na tlačítko *Aim!*. V poli Command je nyní vidět celý příkaz, který nástroj vykoná:

```
xsser -u http://192.168.56.101/mutillidae/index.php?page=set-background-color
.php -i /root/None -s –user-agent Googlebot/2.1 (+http://www.google.com/bot.
```

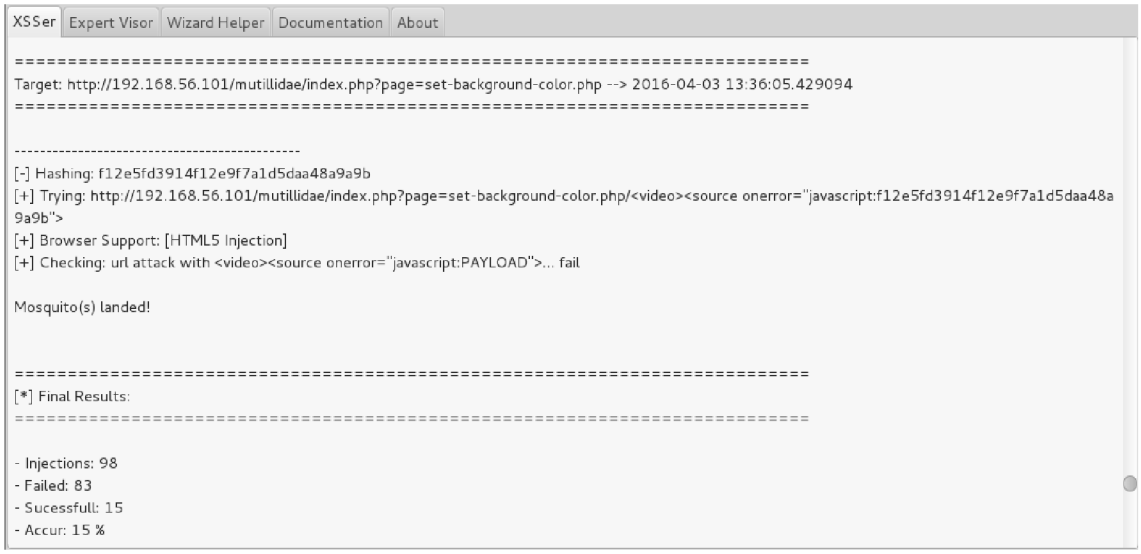
[3] „OWASP 2013“ -> „A3 - Cross Site Scripting (XSS)“ -> „Reflected (First Order)“ -> „Set Background Color“

html) –threads 5 –timeout 30 –retries 1 –delay 0 –auto

K útoku je vše potřebné nastaveno a stačí kliknout na tlačítko *FLY!!!* k zahájení.



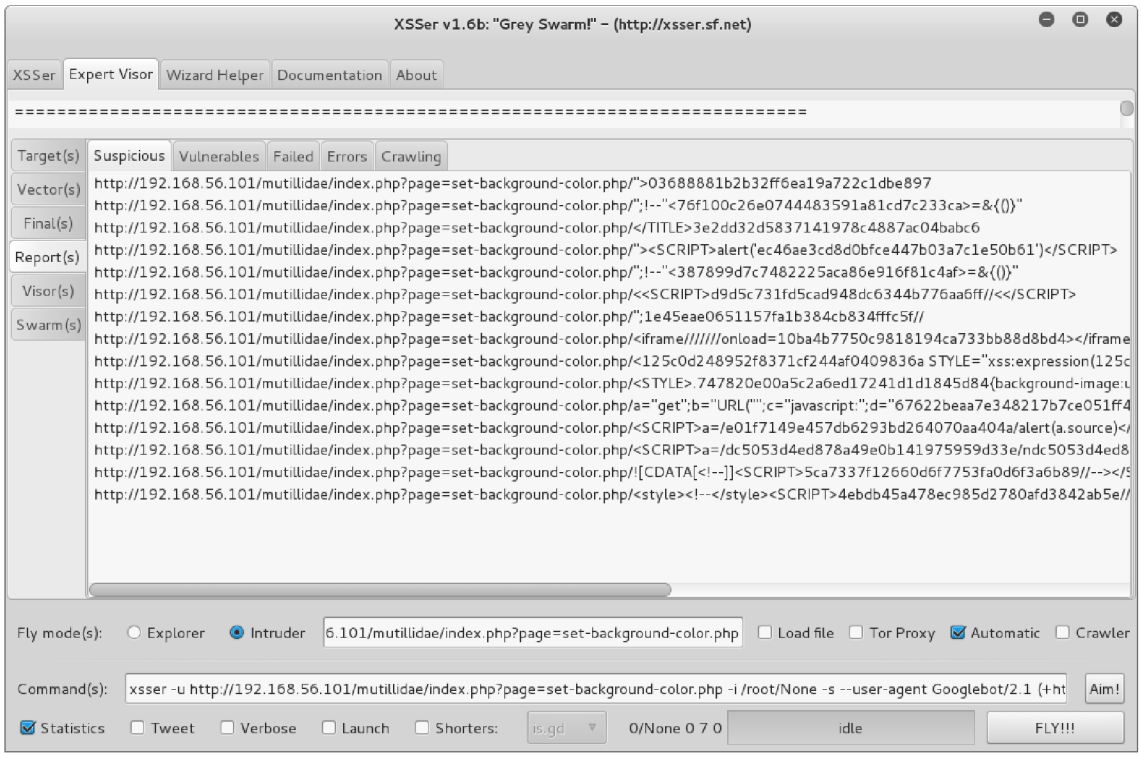
Obr. 4.25: Graficky uživatelské rozhraní a nastavení útoku



Obr. 4.26: Výsledky a celkový průběh útoku

Po vykonání útoku nástroj vypíše do hlavního okna celý postup testování, viz obr. 4.26. Výpis je velmi dlouhý, proto byl zvolen obrázek, ze kterého je vidět pokus o aplikaci škodlivého kódu a výsledky. Nástroj provedl celkem 98 Injections (injekcí), z toho 83 s označením Failed (neúspěšný), 15 s označením Succesful (úspěšný) a Accuracy (přesnost) byla 15%.

V záložce ***Expert Visor/Report/Suspicious*** je vidět všech 15 injections s označením Suspicious (podezřelé), viz obr. 4.27, tedy takové, které mohou mít vliv na webovou aplikaci.

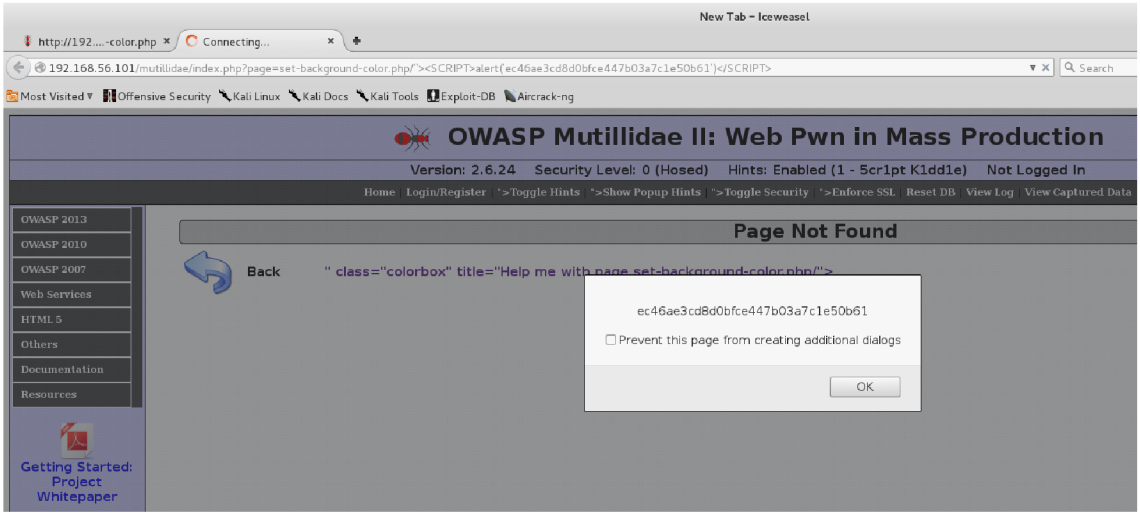


Obr. 4.27: Report suspicious injections

Pro příklad zkusíme použít nástrojem vytvořenou URL se skriptem *alert* (výstraha):

```
http://192.168.56.101/mutillidae/index.php?page=set-background-color.php/“>
<SCRIPT>alert(’ec46ae3cd8d0bfce447b03a7c1e50b61’)</SCRIPT>
```

Po vložení odkazu do webového prohlížeče Iceweasel je vidět, že způsobil zobrazení výstražného okna *alert* s nápisem *ec46ae3cd8d0bfce447b03a7c1e50b61* a ovlivnil i horní menu webové aplikace. Fakt, že jsme přinutili webovou aplikaci k zobrazení výstražného okna, značí zranitelnost proti útokům cross-site scripting.

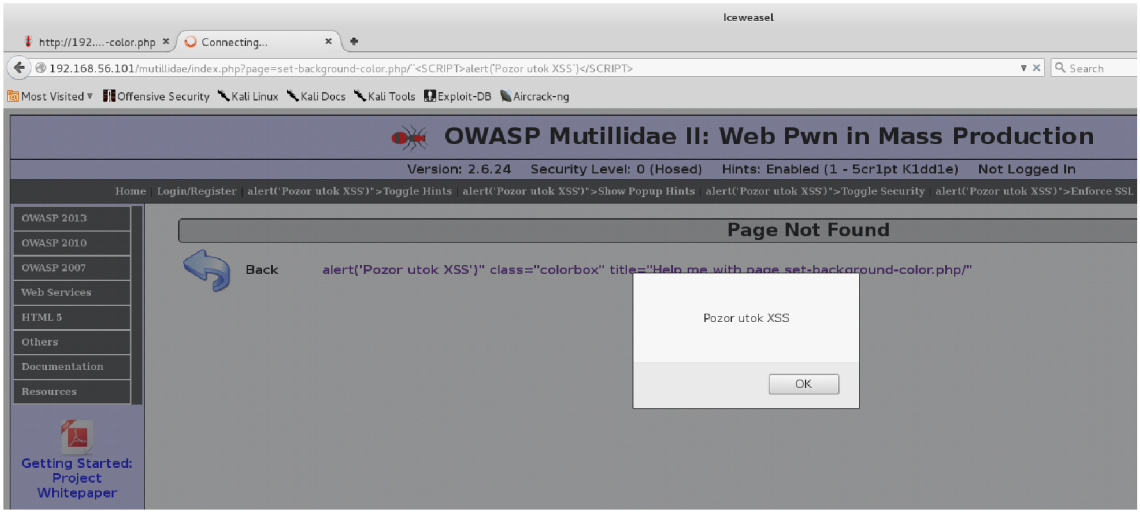


Obr. 4.28: Použití skriptu *alert*

Nepatrně upravíme předchozí skript:

```
http://192.168.56.101/mutillidae/index.php?page=set-background-color.php/“
<SCRIPT>alert(’Pozor utok XSS’)</SCRIPT>
```

Po použití takové URL se výstražné okno *alert* změní na nápis *Pozor utok XSS* a dochází i k výrazné změně horního menu ve formě *alert(’Pozor utok XSS’)“>*, viz obr. 4.29.



Obr. 4.29: Úprava skriptu *alert*

V praxi útočníci vytváří mnohem nebezpečnější skripty, které se navíc kódují podle ASCII tabulky, aby odkazy URL vypadaly věrohodně při metodě **Reflected**.

Zde šlo především o demonstraci, jak jednoduše si může nezkušený uživatel, který nerozumí skriptovacímu jazyku, ověřit pomocí automatické aplikace škodlivého kódu nástrojem XSSer, zda-li je cílová webová aplikace zranitelná proti útoku cross-site scripting metodou **Reflected**.

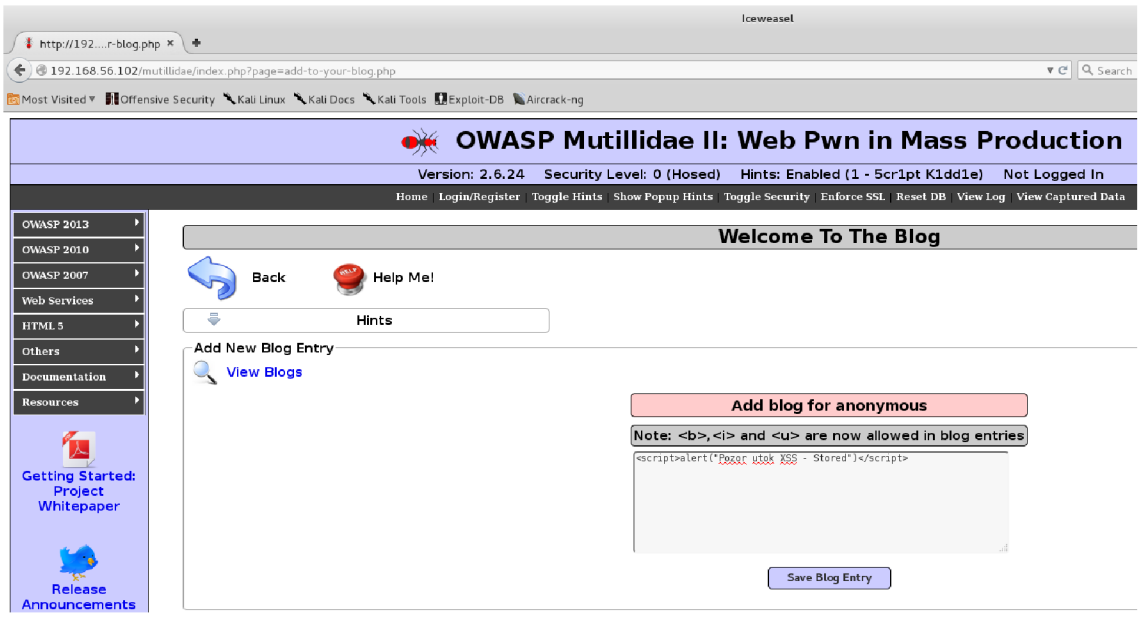
Pro prezentaci metody **Stored** byl zvolen model pro zneužití *Add to your blog* [4], viz obr. 4.30.

Jako příklad bude použit skript *alert* z předchozího úkolu. Abychom ho mohli uložit do databáze, je nutná úprava na tvar:

```
<script>alert("Pozor utok XSS - Stored")</script>
```

Potvrzením tlačítka *Save Blog Entry* se skript uloží do databáze a způsobí zobrazení výstražného okna *alert* s nápisem *Pozor utok XSS - Stored*. V tomto případě to však na rozdíl od metody reflected bude mít dopad na více věcí.

V případě, že by nyní někdo chtěl použít model *Add to your blog*, se nejprve zobrazí výstražné okno *alert* a až poté bude moci blog využít.

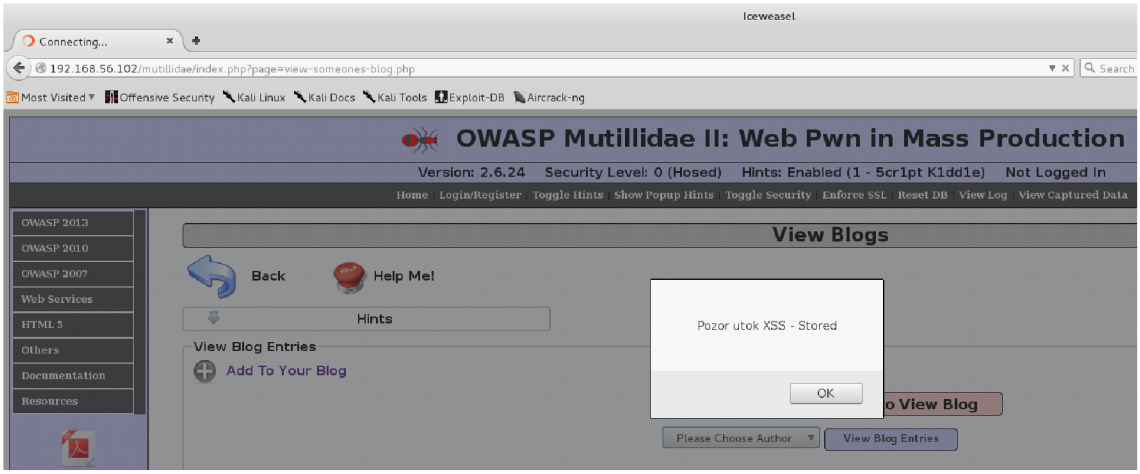


Obr. 4.30: Model zneužití zranitelnosti - *Add to your blog* a skript *alert*

Pokud zvolíme tlačítko *View Blogs*, vybereme z příspěvků od uživatelů volbu *Show All* a potvrdíme *View Blog Entries*, opět dojde k zobrazení výstražného okna *alert*, viz obr. 4.31. Je to z důvodu, že jsme chtěli vypsat všechny uložené blogy

[4] „OWASP 2013“ -> „A3 - Cross Site Scripting (XSS)“ -> „Persistent (Second Order)“ -> „Add to your blog“

v databázi od všech uživatelů včetně toho, který skript do databáze vložil, což zavinilo spuštění i skriptu *alert*. V praxi je metoda **Stored** mnohem nebezpečnější než metoda **Reflected**, protože může postihnout více uživatelů najednou a je pravděpodobnější, že uživatel nechtěně klikne např. na nějakou formu reklamy na webových stránkách (dojde ke spuštění skriptu), než na ojedinělý odkaz v e-mailu (metoda **Reflected**).



Obr. 4.31: Zobrazení výstražného okna *alert* při volbě *Show All*

4.5 Zranitelnost A4 - Insecure Direct Object References

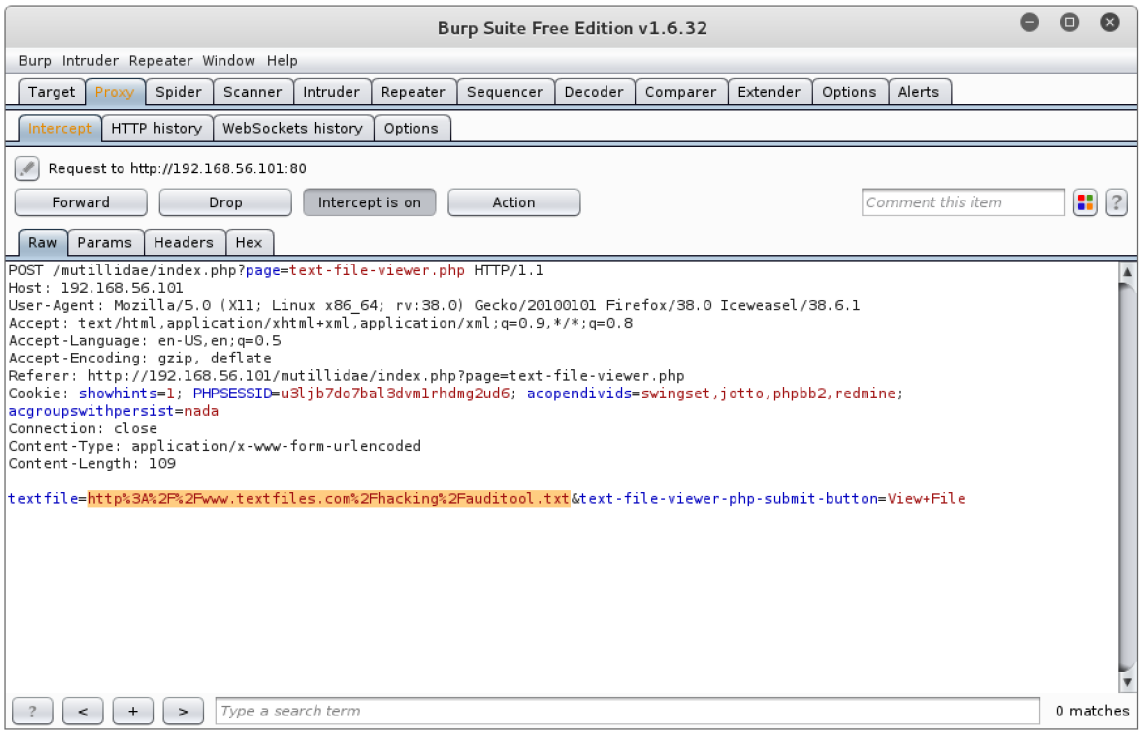
Cílem této úlohy je využít přímých odkazů na objekty a získat informace o webové aplikaci OWASP Mutillidae II.

Obr. 4.32: Model zneužití zranitelnosti - *Text File Viewer*

Pro demonstraci útoku byl zvolen nástroj Burp Suite a ve webové aplikaci OWASP Mutillidae II model pro zneužití *Text File Viewer* [5], viz obr. 4.32.

Již známým postupem z předchozích úloh musíme nejprve nastavit proxy servery webového prohlížeče Iceweasel a nástroje Burp Suite k zachycení HTTP komunikace. Z nabídky textových souborů byl vybrán soubor *Intrusion Detection in Computers by Victor H. Marshall (January 29, 1991)*. Po kliknutí na tlačítko *View File* dojde k zachycení komunikace, viz obr. 4.33. Vyznačený textový řetězec můžeme nyní smazat a místo něj napsat cestu k souboru *passwd*:

textfile=***/etc/passwd***&text-file-viewer-php-submit-button=View+File



Obr. 4.33: Zachycení HTTP komunikace

Kliknutím na tlačítko *Forward* odešleme upravený požadavek webovému serveru ke zpracování a v prohlížeči Iceweasel dostaneme odpověď, viz obr. 4.34. Jedná se o textovou databázi, obsahující informace o uživatelích, kteří se mohou přihlásit do systému a uživatelských identitách OS vlastnící běžící procesy. Útočník by se mohl ze souboru dozvědět uživatelská jména, šifrovaná hesla, čísla UID (User ID number), čísla GID (Group ID number), celá jména uživatelů, domovské adresáře a příkazový terminál (shell).

[5] „OWASP 2013“ -> „A4 - Insecure Direct Object References“ -> „Text File Viewer“

File: /etc/passwd

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
syslog:x:101:102::/home/syslog:/bin/false
klog:x:102:103::/home/klog:/bin/false
mysql:x:103:105:MySQL Server,,,:/var/lib/mysql:/bin/false
landscape:x:104:122::/var/lib/landscape:/bin/false
sshd:x:105:65534::/var/run/sshd:/usr/sbin/nologin
postgres:x:106:109:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
messagebus:x:107:114::/var/run/dbus:/bin/false
tomcat6:x:108:115::/usr/share/tomcat6:/bin/false
user:x:1000:1000:user,,,:/home/user:/bin/bash
```

Obr. 4.34: Výpis souboru *passwd*

Kromě souboru *passwd* může útočník získat informace o běžících procesech na serveru, verzi serveru atd. prostřednictvím souborového systému *proc*. Stejným postupem jako u souboru *passwd* (musíme znovu zachytit HTTP komunikaci) zjistíme verzi kernelu:

textfile=***/proc/version***&text-file-viewer-php-submit-button=View+File

Z obrázku 4.35 je vidět konkretní verze používaného OS a datum poslední aktualizace.

File: /proc/version

```
Linux version 2.6.32-25-generic-pae (buildd@rothera) (gcc version 4.4.3 (Ubuntu 4.4.3-4ubuntu5) ) #44-Ubuntu SMP Fri Sep 17 21:57:48 UTC 2010
```

Obr. 4.35: Výpis souboru *version*

Směrovací tabulku:

textfile=***/proc/net/route***&text-file-viewer-php-submit-button=View+File

File: /proc/net/route

Iface	Destination	Gateway	Flags	RefCnt	Use	Metric	Mask	MTU	Window	IRTT
eth0	0038A8C0	00000000	0001	0	0	0	00FFFFFF	0	0	0

Obr. 4.36: Výpis souboru *route*

ARP tabulku:

textfile=***/proc/net/arp***&text-file-viewer-php-submit-button=View+File

Poskytuje informace pro vyhledání IP adres pro jiné interní servery.

File: /proc/net/arp

IP address	HW type	Flags	HW address	Mask	Device
192.168.56.1	0x1	0x2	0a:00:27:00:00:00	*	eth0

Obr. 4.37: Výpis souboru *arp*

Konfigurační soubor Apache serveru:

textfile=***/etc/apache2/apache2.conf***&text-file-viewer-php-submit-button=View+File

V tomto souboru je uvedená celá konfigurace webového serveru Apache. Obrázek 4.38 reprezentuje vzhledem k dlouhému výpisu pouhý zlomek.

File: /etc/apache2/apache2.conf

```
#
# Basec upon the NCSA server configuration files originally by Rob McCool.
#
# This is the main Apache server configuration file.  It contains the
# configuration directives that give the server its instructions.
# See http://httpd.apache.org/docs/2.2/ for detailed information about
# the directives.
#
# Do NOT simply read the instructions in here without understanding
# what they do.  They're here only as hints or reminders.  If you are unsure
# consult the online docs. You have been warned.
```

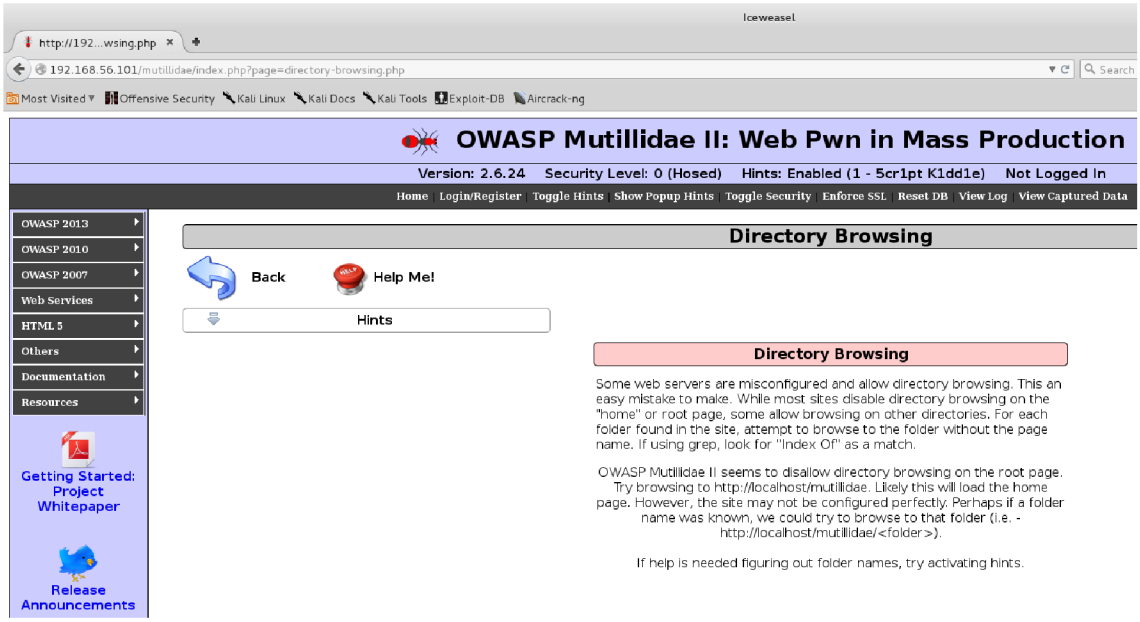
Obr. 4.38: Výpis části souboru *apache2.conf*

Pokud je webová aplikace nezabezpečená a používá přímé odkazy na objekty, může se útočník dozvědět velké množství užitečných informací o systému a uživatelích. Záleží pouze, jakou má celkovou znalost o cílovém systému a jeho struktuře, aby mohl přistupovat k jednotlivým souborům a číst jejich obsah.

4.6 Zranitelnost A5 - Security Misconfiguration

Cílem této úlohy je využít nezabezpečenou konfiguraci k prozkoumání adresářů a souborů, které by neměly být neoprávněným osobám přístupné a získat tak informace o webové aplikaci OWASP Mutillidae II.

Pro demonstraci útoku byl zvolen nástroj Burp Suite a ve webové aplikaci OWASP Mutillidae II model pro zneužití *Directory Browsing* [6], viz obr. 4.39.



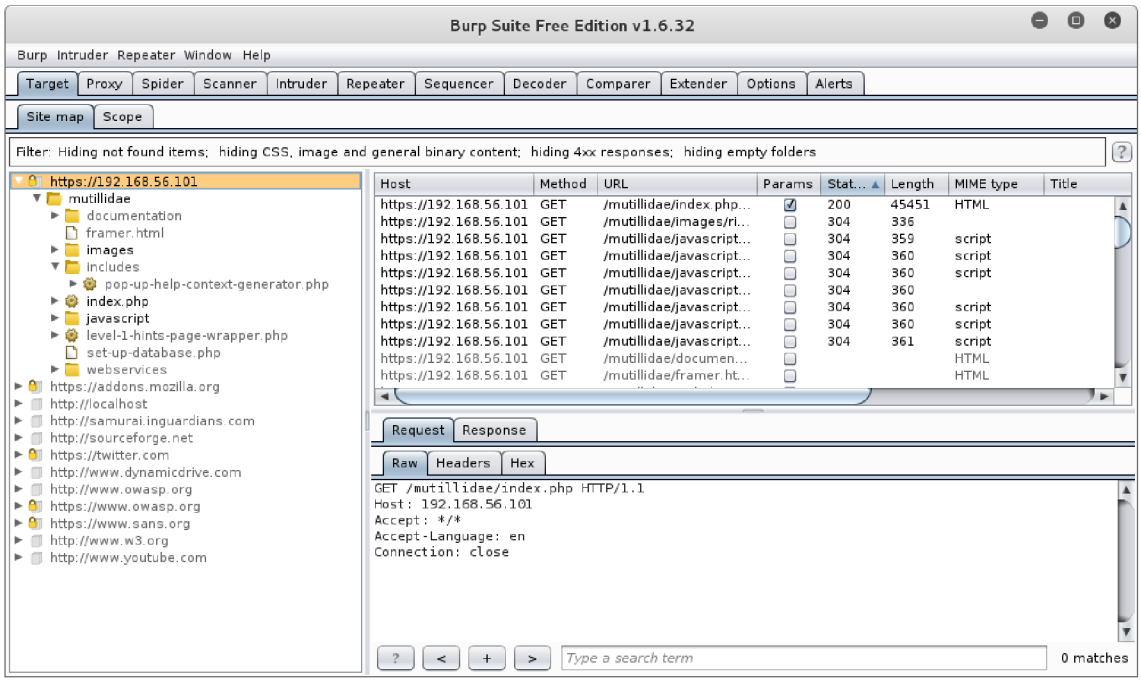
Obr. 4.39: Model zneužití zranitelnosti - *Directory Browsing*

Nejprve je nutné opět zachytit HTTP komunikaci nástrojem Burp Suite. V tomto případě dojde k zachycení stisknutím tlačítka *F5* sloužícího k obnovení webové stránky. Po zachycení HTTP komunikace vypneme zachytávání kliknutím na tlačítko *Intercept is on* v záložce ***Proxy/Intercept*** a přepneme se do záložky ***Target/Site map***.

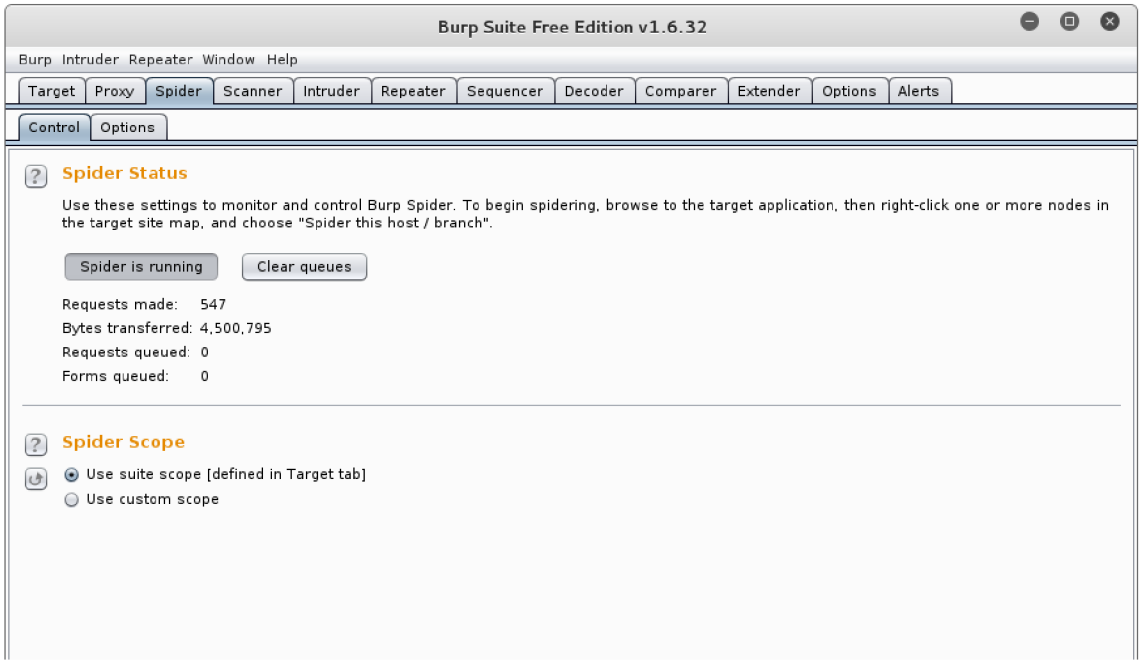
Site map (mapa stránky) shromažďuje veškeré informace, které nástroj shromáždil v průběhu testování. V této úloze zde nástroj shromáždil všechny informace ze zachyceného požadavku a pomocí passive spidering (pasivní „spidering“ - metoda,

[6] „OWASP 2013“ -> „A5 - Security Misconfiguration“ -> „Directory Browsing“

která zpracuje všechny HTTP požadavky/odpovědi, z nichž vybuduje podrobný obsah o aplikaci) další obsah, který lze vyvodit z odkazů aplikace, formulářů, vazeb atd., viz obr. 4.40.



Obr. 4.40: Shromážděný obsah o webové aplikaci ze zachycené HTTP komunikace



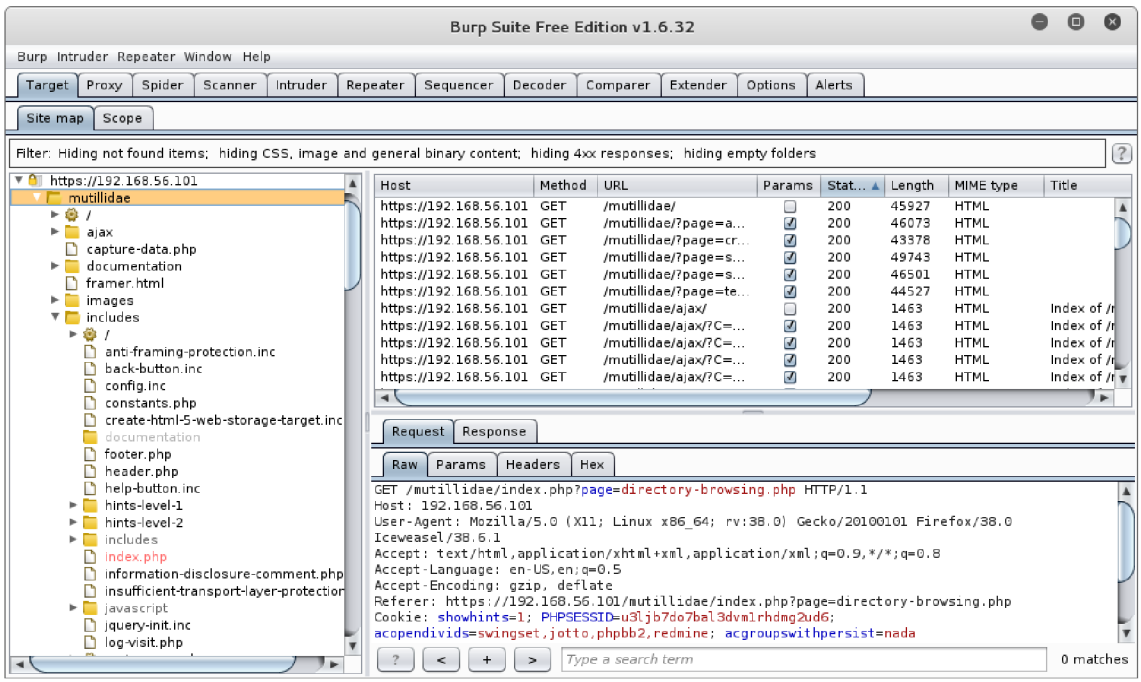
Obr. 4.41: Zkoumání složky *mutillidae* a jejího obsahu

Abychom mohli prohlížet veškerý obsah webové aplikace, je nutné použít modul Spider, který automaticky prozkoumává cookies, inicializuje spojení s webovou aplikací, vypisuje a mapuje různé stránky a parametry cílové webové stránky.

Kliknutím pravým tlačítkem na podsložku *mutillidae* hlavní větve *https://192.168.56.101* v záložce ***Target/Site map*** a zvolením *Spider this branch*, odešleme příkaz modulu Spider k průzkumu této podsložky. Nástroj nás upozorní, že zvolená položka není ve vymezeném rozsahu testování, a zda-li chceme rozsah automaticky upravit. Kliknutím na *Yes* Burp Suite vymezí pole průzkumu pouze na zvolenou složku *mutillidae* a všechny její podsložky.

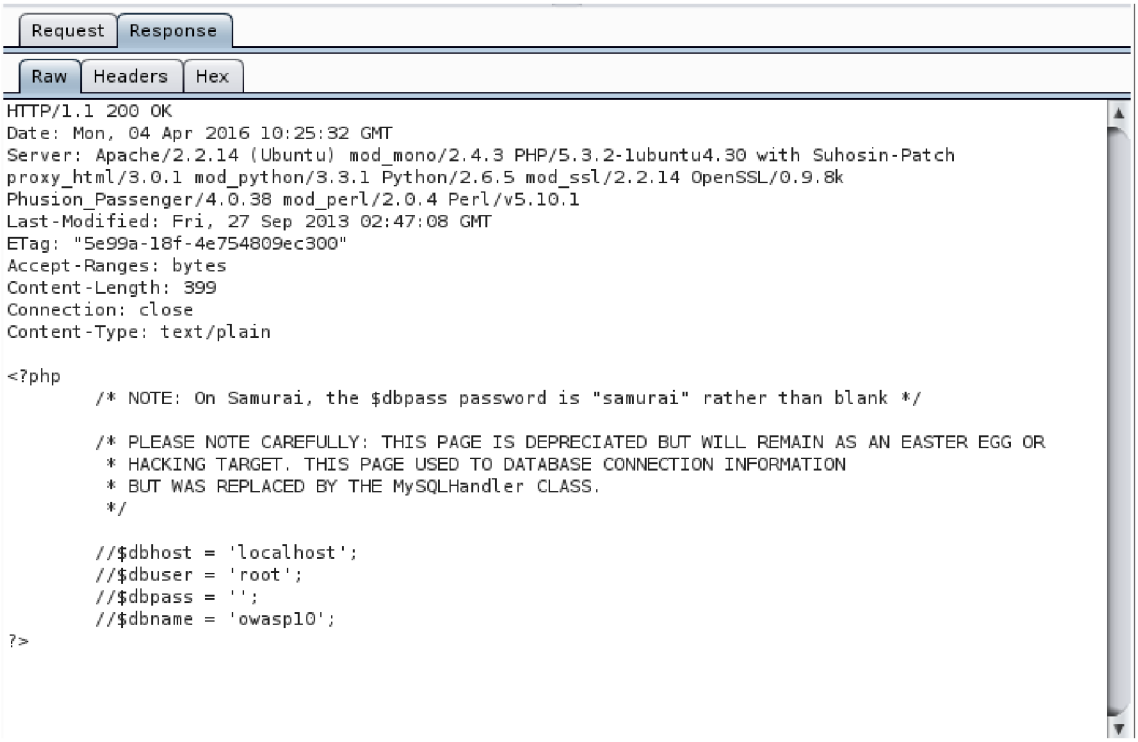
Poté nástroj provede průzkum, viz obr. 4.41. Po dokončení je z obrázku vidět celkový počet provedených požadavků (Requests made) a přenesených bytů (Bytes transferred).

V této chvíli bude shromážděný obsah v záložce ***Target/Site map*** rozšířen, viz obr. 4.42. Například adresář *includes* obsahuje mnohem více souborů než na obrázku 4.40, což bylo účelem, protože nyní je lze možné prohlížet i soubory, které byly v předchozím kroku skryté a mohou nám poskytnou cenné informace o webové aplikaci.

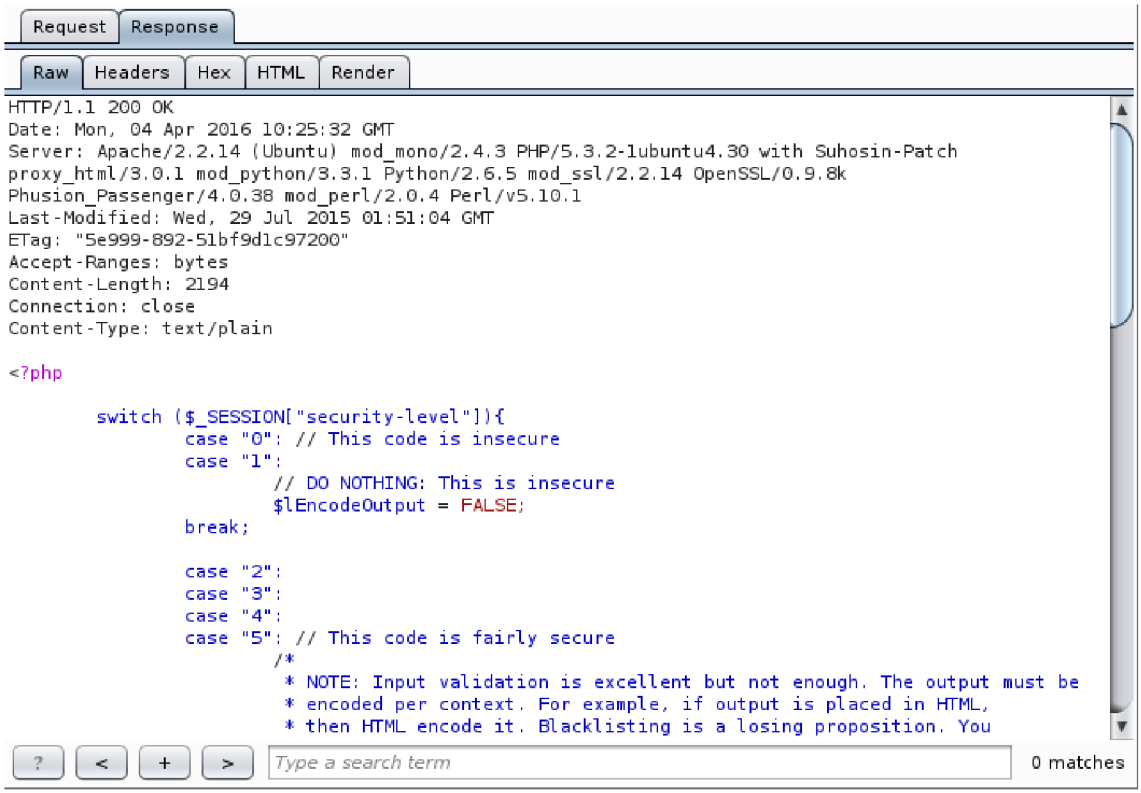


Obr. 4.42: Shromážděný obsah modulem Spider

Pokud bychom zvolili adresář *includes* a v něm soubor *config.ini*, tak v záložce ***Response/Raw*** je zobrazen obsah tohoto souboru, viz obr. 4.43. Zde se útočník dozví název hosta databáze (*localhost*), uživatelské jméno (*root*), heslo (-) a název databáze (*owasp10*).



Obr. 4.43: Obsah souboru *config.inc*



Obr. 4.44: Obsah souboru *back-button.inc*

Při volbě souboru *back-button.inc* útočník dostane přístup ke zdrojovému kódu. V něm jsou nevyužité funkce, např. *case "2":* a *case "3":*, které jsou považovány za nebezpečné, viz obr. 4.44.

Tato úloha demonstruje nejběžnější nezabezpečenou konfiguraci, kde většina vývojářů spoléhá na skryté adresáře a soubory. Jedinou ochranou je zde předpoklad, že útočník nezjistí jejich názvy, protože k nim neexistují žádné vazby. Nicméně, tyto názvy lze uhodnout nebo prostřednictvím automatizovaných nástrojů odhalit.

5 ZÁVĚR

V práci je nejprve obecně popsáno, co je penetrační testování, co vše by mělo zahrnovat, na jaké typy testů se dělí a metodologie testování.

Dále je uvedena bezpečnostní organizace *„OWASP“*, věnující se bezpečnosti webových aplikací. Jsou zde zmíněny její nejvýznamnější projekty a popsán dokument OWASP Top 10, který zveřejňuje každé tři roky 10 nejzávažnějších zranitelností webových aplikací. Další část tvoří základní popis prvních pěti zranitelností (Injection, Broken Authentication and Session Management, Cross-Site Scripting, Insecure Direct Object References a Security Misconfiguration) z verze dokumentu OWASP Top 10 2013. U každé z nich je uveden příklad útoku a způsob obrany.

Poslední teoretická část je zaměřena na linuxovou distribuci Kali Linux, určenou k penetračnímu testování a průzkumu bezpečnosti. Kromě základních informací o distribuci, jsou zmíněny způsoby jejího zprovoznění a nejnovější verze Kali Linux 2.0. Další část tvoří popis několika nejpoužívanějších nástrojů pro penetrační testování, včetně skeneru zranitelností Nessus.

Praktickou část tvoří testování prvních pěti zranitelností z dokumentu OWASP Top 10 2013, kde je nejprve uveden použitý SW pro realizaci útoků a vnitřní infrastruktura sítě. Celý rozbor této části je určen k porozumění vytvořené laboratorní úlohy a zároveň slouží jako podrobný návod pro vyučující.

Prvním úkolem bylo získání uživatelských přihlašovacích údajů webové aplikace OWASP Mutillidae II prostřednictvím zranitelnosti SQL injection. Nejprve dojde k zachycení HTTP komunikace (požadavku) nástrojem Burp Suite. Následně je použit nástroj Sqlmap k aplikaci SQL injection. Výsledkem bylo získání základních informací o webové aplikaci a dvaceti uživatelských přihlašovacích údajů.

Druhý úkol je zaměřený na zranitelnost Broken Authentication and Session Management a získání uživatelských přihlašovacích údajů webové aplikace OWASP Mutillidae II metodou dictionary attack. Zachycený požadavek nástrojem Burp Suite je odeslán modulu Intruder, který metodou dictionary attack porovná kombinace uživatelských jmén a hesel. Výsledkem bylo 45 nesprávných kombinací a 4 správné.

Třetí úkol je věnovaný zranitelnosti Cross-Site Scripting. Cílem je vyhledání potenciálně nebezpečných skriptů, které mohou mít vliv na webovou aplikaci OWASP Mutillidae II a znázornit metodu reflected i stored. Nejprve nástrojem XSSer došlo k vyhledání nebezpečných skriptů a následně byla provedena ukázka se skriptem *alert* (metoda reflected). Poté bylo prezentováno použití skriptu *alert* i pro metodu stored.

Čtvrtý úkol je zaměřen na zranitelnost Insecure Direct Object References, kde je cílem, zneužitím této zranitelnosti, získat informace o webové aplikaci OWASP Mutillidae II. Nástrojem Burp Suite je zachycen požadavek, který je následně pozmě-

něn za přímý odkaz na objekt (soubor). Výsledkem bylo prozkoumání pěti souborů webové aplikace.

Poslední úkol se zabývá zranitelností Security Misconfiguration. Cílem je provedení průzkumu adresářů webové aplikace OWASP Mutillidae II, které by neměly být přístupné neoprávněným osobám. Nejprve je odchycena HTTP komunikace a poté následné odeslání podsložky *mutillidae* modulu Spider k průzkumu. Výsledkem je nahlížení do adresářů a souborů, které jsou za normálních okolností skryty.

Z praktické části je v příloze vypracována laboratorní úloha „Penetrační testování webových aplikací“ a „Úvod do problematiky penetračního testování“, která popisuje základní princip zranitelností SQL injection a Broken Authentication and Session Management a slouží jako úvod před hlavní laboratorní úlohou.

LITERATURA

[1] FADYUSHIN, Vyacheslav a Bruce HYSLOP *Instant Penetration Testing: Setting Up a Test Lab How-to.* Birmingham: Packt Publishing, 2013. 88 s. ISBN 978-1849694124.

[2] SELECKÝ, M. *Penetrační testy a exploitace.* 1. vyd. Brno: Computer Press, 2012. 304 s. ISBN 978-80-251-3752-9.

[3] SAINDANE, M. *Penetration testing - a systematic approach* [online]. 11. 8. 2015. [cit. 3. 11. 2015]. Dostupné z: <`http://www.infosecwriters.com/Papers/MSaindane_Pentest.pdf`>.

[4] TESAURO, M. *The Open Web Application Security Project* [online]. 2001, 30. 10. 2015. [cit. 5. 11. 2015]. Dostupné z: <`https://www.owasp.org/index.php/Main_Page`>.

[5] TESAURO, M. *OWASP Projects* [online]. 2001, 27. 10. 2015. [cit. 5. 11. 2015]. Dostupné z: <`https://www.owasp.org/index.php/Category:OWASP_Project#tab=Project_Inventory`>.

[6] WICHERS, D. *The OWASP Foundation* [online]. 4. 10. 2015. [cit. 6. 11. 2015]. Dostupné z: <`https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project#tab=OWASP_Top_10_for_2013`>.

[7] *Creative Commons* [online]. 2001, 5. 11. 2015. [cit. 5. 11. 2015]. Dostupné z: <`http://creativecommons.org/licenses/by-sa/3.0/`>.

[8] TESAURO, M. *SQL Injection* [online]. 14. 8. 2014. [cit. 8. 11. 2015]. Dostupné z: <`https://www.owasp.org/index.php/SQL_Injection`>.

[9] BOYD, S. *Applied Cryptography and Network Security.* Yellow Mountain: Springer Berlin Heidelberg, 2004. 292 s. ISBN 978-3-540-22217-0.

[10] *SQL Injection Are Your Web Applications Vulnerable?* [online]. 25. 3. 2002. [cit. 8. 11. 2015]. Dostupné z: <`https://www.defcon.org/images/defcon-10/dc-10-presentations/dc10-spett-sqlinjection/dc10-spett-sqlinjection.pdf`>.

[11] ANDREU, A. *Professional Pen Testing for Web Applications.* Indianapolis: Wiley Publishing, 2006. 546 s. ISBN 978-0-471-78966-6.

[12] *SQL injection Hall-of-Shame.* [online]. [cit. 8. 11. 2015]. Dostupné z: <`http://codecurmudgeon.com/wp/sql-injection-hall-of-shame/`>.

[13] TESAURO, M. *Broken Authentication and Session Management* [online]. 22. 4. 2010. [cit. 8. 11. 2015]. Dostupné z: <https://www.owasp.org/index.php/Broken_Authentication_and_Session_Management>.

[14] STUTTARD, D. *The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws*.2. vyd. Indianapolis: John Wiley & Sons, 2011. 914 s. ISBN 978-1-118-02647-2.

[15] TESAURO, M. *Cross-Site Scripting (XSS)* [online]. 2. 2. 2016. [cit. 14. 3. 2016]. Dostupné z: <https://www.owasp.org/index.php/Cross-Site_Scripting>.

[16] KLEIN, A. *Cross Site Scripting Explained* [online]. [cit. 14. 3. 2016]. Dostupné z: <https://crypto.stanford.edu/cs155/papers/CSS.pdf>.

[17] TESAURO, M. *Insecure Direct Object References* [online]. 14. 6. 2013. [cit. 16. 3. 2016]. Dostupné z: <https://www.owasp.org/index.php/Top_10_2013-A4-Insecure_Direct_Object_References>.

[18] TESAURO, M. *Security Misconfiguration* [online]. 23. 6. 2013. [cit. 16. 3. 2016]. Dostupné z: <https://www.owasp.org/index.php/Top_10_2013-A5-Security_Misconfiguration>.

[19] HUNT, T. *OWASP Top 10 for .NET developers part 6: Security Misconfiguration* [online]. 10. 12. 2010. [cit. 16. 3. 2016]. Dostupné z: <http://www.troyhunt.com/2010/12/owasp-top-10-for-net-developers-part-6.html>.

[20] *Kali Linux The Ultimate Penetration Testing Platform* [online]. 2014. [cit. 9. 11. 2015]. Dostupné z: <http://docs.kali.org/introduction/what-is-kali-linux>.

[21] *BackTrack Linux – Penetration Testing Distribution* [online]. 2014. [cit. 9. 11. 2015]. Dostupné z: <http://www.backtrack-linux.org>.

[22] *Offensive Security Training, Certifications and Services* [online]. 2007. [cit. 9. 11. 2015]. Dostupné z: <https://www.offensive-security.com>.

[23] *Kali Linux tools listing* [online]. 2014. [cit. 10. 11. 2015]. Dostupné z: <http://tools.kali.org/tools-listing>.

[24] *Tenable Network Security* [online]. 2016. [cit. 10. 3. 2016]. Dostupné z: <https://www.tenable.com/about-tenable/about-us>.

[25] *Tenable Network Security documentation* [online]. 14. 12. 2015. [cit. 10. 3. 2016]. Dostupné z: <https://docs.tenable.com/nessus/>.

[26] *Oracle VM VirtualBox* [online]. 26. 3. 2016. [cit. 26. 3. 2016]. Dostupné z: <https://www.virtualbox.org/>.

[27] *OWASP Broken Web Applications Project* [online]. 6. 3. 2016. [cit. 26. 3. 2016]. Dostupné z: <https://www.owasp.org/index.php/OWASP_Broken_Web_Applications_Project>.

[28] *OWASP Mutillidae 2 Project* [online]. 2. 3. 2015. [cit. 26. 3. 2016]. Dostupné z: <https://www.owasp.org/index.php/OWASP_Mutillidae_2_Project>.

[29] GREGR, F. *Penetrační testy a odhalování zranitelností síťových prvků* Vysoké učení technické v Brně, 2015. 74 s.

[30] NOVOTNY, J. *Elektronická bezpečnost veřejných služeb* Vysoká škola polytechnická Jihlava, 2015. 53 s.

[31] STALLINGS, W. *Cryptography and network security: principles and practice.* 7. vyd. Prentice Hall, 2006. 900 s. ISBN 978-0-13-609704-4.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

SQL	Structured Query Language
DNS	Domain Name System
CGI	Common Gateway Interface
LDAP	The Lightweight Directory Access Protocol
VPN	Virtual Private Network
ISS	Internet Security System
OWASP	The Open Web Application Security Project
SW	Software
HW	Hardware
HTTP	The Hypertext Transfer Protocol
ID	Identifier
SSL	Secure Sockets Layer
DOM	Document Object Model
HTML	HyperText Markup Language
HTTPS	Hypertext Transfer Protocol Secure
URL	Uniform Resource Locator
ARM	Acorn RISC Machine
GUI	Graphical User Interface
BSD	Berkeley Software Distribution
OS	Operating system
WPA	Wi-Fi Protected Access
WEP	Wired Equivalent Privacy
DOS	Disk Operating System
DES	The Data Encryption Standard

LM	LAN Manager
TCP	The Transmission Control Protocol
UDP	The User Datagram Protocol
VM	Virtual Machine
DBMS	Database Management System
PHP	PHP: Hypertext Preprocessor
DBA	Database Administrator

SEZNAM PŘÍLOH

A LABORATORNÍ ÚLOHA - PENETRAČNÍ TESTOVÁNÍ WEBOVÝCH APLIKACÍ

Cílem úlohy je seznámit se s penetračním testováním webových aplikací automatizovanými nástroji v prostředí Kali Linux a prvními pěti zranitelnostmi webových aplikací z dokumentu OWASP Top 10 2013.

Zadání

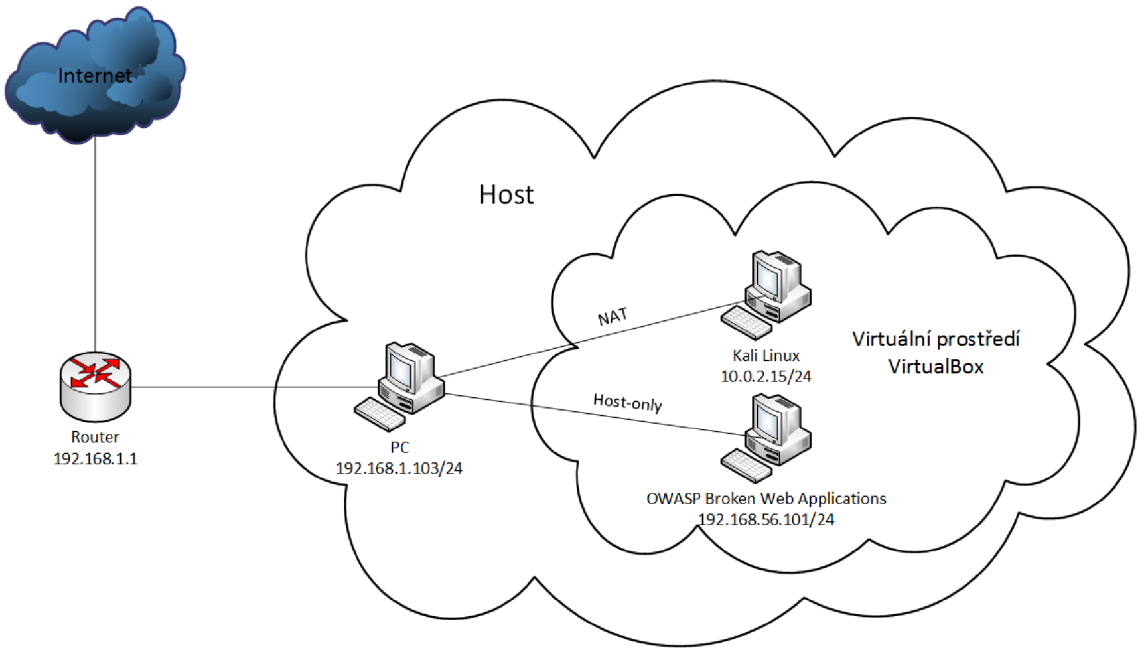
1. Seznámit se s prostředím webové aplikace OWASP Mutillidae II.
2. Získat uživatelské přihlašovací údaje webové aplikace OWASP Mutillidae II pomocí SQL injection.
3. Získat uživatelské přihlašovací údaje webové aplikace OWASP Mutillidae II využitím metody dictonary attack.
4. Vyhledat potenciálně nebezpečné skripty nástrojem XSSer a vyzkoušet vliv skriptu *alert* na webovou aplikaci OWASP Mutillidae II metodou reflected i stored.
5. Využít přímých odkazů na objekty a získat informace o webové aplikaci OWASP Mutillidae II.
6. Využít nezabezpečenou konfiguraci k prozkoumání adresářů a souborů webové aplikace OWASP Mutillidae II.

Postup pro vypracování

Vnitřní infrastruktura sítě je zakreslena na obr. A.1. Prostřednictvím virtualizačního programu Oracle VM VirtualBox, umístěného na PC hosta, jsou spuštěny dvě virtuální stanice (Kali Linux, OWASP Broken Web Applications), které komunikují s hostitelským PC v režimu *NAT* a *Host only* z hlediska bezpečnosti sítě. Host všechny požadavky od stanic posílá routeru, který je zpracuje a přepošle dál v závislosti na požadavku.

1)

- Spusťte Oracle VM VirtualBox a v něm stanice Kali Linux a OWASP Broken Web Applications.
- Stanici OWASP Broken Web Applications ponechte spuštěnou na pozadí a přihlaste se do systému Kali Linux jako uživatel *root* s heslem *toor*.
- Otevřete webový prohlížeč Iceweasel, zadejte IP adresu *192.168.56.101* (OWASP Broken Web Applications), zvolte webovou aplikaci OWASP Mutillidae II a zorientujte se v prostředí.



Obr. A.1: Vnitřní infrastruktura sítě

2)

- Model zneužití zvolte **„OWASP 2013“ -> „A1 - Injection (SQL)“ -> „SQLMAP Practice“ -> „Login“**.
- V URL adrese můžete vidět, že webová aplikace nepoužívá metodu GET, ale POST. Je tedy potřeba zachytit HTTP komunikaci webového prohlížeče s webovým serverem.
- Spusťte nástroj Burp Suite a přepněte do záložky ***Proxy/Options***. Nastavte IP adresu *127.0.0.1*, port *8080* a ujistěte se, že naslouchání je aktivní (zatržené tlačítko *Running*).
- Pro úspěšné zachycení HTTP komunikace je potřeba nastavit stejnou IP adresu i port ve webovém prohlížeči Iceweasel. V pravém horním rohu otevřete možnosti a pravým tlačítkem klikněte na doplněk *Swap Proxy*. Otevře se nastavení, kde zadejte stejné údaje jako v nástroji Burp Suite, zavřete nastavení a klikněte levým tlačítkem na položku *Swap Proxy*, která se zvýrazní, což značí, že přepnutí proxy serveru je aktivní. V této fázi nepůjde používat webový prohlížeč k běžnému surfování. V případě potřeby musíme vypnout proxy server opětovným kliknutím na *Swap Proxy*.
- Nyní vyplňte ve webové aplikaci libovolné jméno a heslo a potvrďte tlačítkem *Login*. V nástroji Burp Suite se přepněte do záložky ***Proxy/Intercept*** a prohlédněte si zachycený požadavek. Ve spodní části jsou vámi vyplněné údaje, které budete potřebovat k použití nástroje Sqlmap.

- Spusťte terminál, zadejte následující příkaz a podívejte se na možnosti nástroje Sqlmap. Věnujte pozornost zejména volbám *Enumeration*, *Fingerprint*, *Target* a *Request*.
  Příkaz:

```
root@Kali:~# sqlmap -help
```

- Dalším krokem je shromáždění základních informací o webové aplikaci a přesvědčení se, zda-li je nástroj schopen úspěšně provést SQL injection.
  Příkaz:

```
root@Kali:~# sqlmap -u "http://192.168.56.101/mutillidae/index.php?page=login.php"–data="username=„vámi zadané jméno“&password=„vámi zadané heslo“&login-php-submit-button=Login"-f -b
```

- Pokud se vás nástroj dotáže na tyto otázky, zvolte následující odpovědi:
  - It looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y
  - For the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] y
  - Sqlmap got a 302 redirect to 'http://192.168.56.101:80/mutillidae/index.php?popUpNotificationCode=AU1'. Do you want to follow? [Y/n] y
  - Redirect is a result of a POST request. Do you want to resend original POST data to a new location? [y/N] y
  - POST parameter 'username' is vulnerable. Do you want to keep testing the others (if any)? [y/N] n
- Po dokončení testování nástroj zobrazí výsledky a informace o webové aplikaci, které si prohlédněte. Smažte parametry **-f -b** na konci předchozího příkazu a místo nich zadejte parametry k zjištění názvu serveru DBMS, aktuálního uživatele DBMS, aktuální databáze DBMS a zda-li je aktuální uživatel DBA systému DBMS. Jednotlivé parametry je nutné oddělovat mezerou.
- Dále je nutné zjistit, jaké databáze webová aplikace obsahuje. Opět smažte parametry pro výpis předchozích informací a nahraďte je parametrem **–dbs**.
- Zvolte cílovou databázi, kterou víte z předchozích kroků a vypište její tabulky.
  Příkaz:

```
root@Kali:~# sqlmap -u "http://192.168.56.101/mutillidae/index.php?page=login.php"–data="username=„vámi zadané jméno“&password=„vámi
```

zadané heslo“&login-php-submit-button=Login"**-D** „cílová databáze“ –**tables**

- Vyberte tabulku, obsahující uživatelské účty a vypište její sloupce. Postup je stejný jako v příkazu výše. Parametr s cílovou databází zůstává, jen je potřeba použít parametry pro volbu tabulky a výpis sloupců, které obsahuje.
- Nyní podobným způsobem zvolte sloupce s uživatelskými jmény a hesly ve formátu **-C** *uživatelská jména,hesla* (nepoužívejte mezery za čárkou) a na konec příkazu přidejte parametr –**dump**, který slouží pro vypsání hodnot těchto sloupců.
- Pokud nějaká hodnota (nejčastěji heslo) používá nějaký typ šifrování, nástroj se pokusí o dešifraci a dotáže se, jak chceme postupovat. Zvolte následující odpovědi:
  - Do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] y
  - Do you want to crack them via a dictionary-based attack? [Y/n/q] y
  - What dictionary do you want to use?
    (1) default dictionary file '/usr/share/sqlmap/txt/wordlist.zip'
    (2) custom dictionary file
    (3) file with list of dictionary files
    > 1
  - Do you want to use common password suffixes? (slow!) [y/N] n
- Prohlédněte si vypsané uživatelské údaje a zkuste některý z nich použít pro přihlášení do webové aplikace OWASP Mutillidae II pro ověření správnosti. Zkuste si také vypsat sloupec *is_admin*.

3)

- Model zneužití zvolte **„OWASP 2013“ -> „A2 - Broken Authentication and Session Management“ -> „Authentication Bypass“ -> „Via Brute Force“ -> „Login“**.
- Stejným postupem jako v předchozím úkolu zachyťte HTTP komunikaci. Po úspěšném zachycení klikněte pravým tlačítkem kamkoliv v okně s požadavkem. Ukáže se menu, ze kterého vyberte volbu *Send to Intruder*.
- Přepněte se do záložky ***Intruder/Target***, nastavte IP adresu našeho cíle *192.168.56.101* a port *80*.
- V záložce ***Intruder/Positions*** je potřeba určit typ útoku a zvolit místa v požadavku, na která bude Intruder vkládat námi zadané hodnoty. Kliknutím na tlačítko *Clear §* se všechny automaticky označené proměnné hodnoty vymažou. Nyní označte pouze vámi vyplněné uživatelské jméno a heslo (pokud jste zvolili jako uživatelské jméno např. *„student“* a heslo *„student1“*, tak

označíte jen tyto hodnoty) a kliknutím na tlačítko *Add §* zadefinujeme místa, na která bude Intruder vkládat uživatelská jména a hesla ze slovníku. Attack type zvolte *Cluster bomb*.

- Přepněte se do záložky ***Intruder/Payloads***. Zde je nutné určit, jaké hodnoty se budou na zvolená místa vkládat. V Payload set bude po rozbalení seznamu na výběr ze dvou možností, kde vyberte nejprve *1*. Payload type zvolte *Simple list*. V Payloads Options klikněte na tlačítko *Load ...* a vyberte z domovského adresáře soubor *userlist.txt*.
- Nyní přepněte Payload set na číslo *2* a kliknutím na tlačítko *Load ...* vyberte z domovského adresáře soubor *passlist.txt*. Klikněte na tlačítko *Start attack* k zahájení útoku.
- Po dokončení útoku nástroj zobrazí výsledky. Prohlédněte si jednotlivé kombinace uživatelských jmen a hesel, záložky ***Response/Raw*** a ***Response/Render*** a zkuste zjistit, podle čeho můžete poznat, že daná kombinace je správná. Prostřednictvím těchto kombinací se poté zkuste přihlásit do webové aplikace OWASP Mutillidae II.

4)

- Model zneužití zvolte **„OWASP 2013“ -> „A3 - Cross Site Scripting (XSS)“ -> „Reflected (First Order)“ -> „Set Background Color“**.
- Otevřete terminál a zadáním následujícího příkazu spusťte graficky uživatelské rozhraní nástroje XSSer.
  Příkaz:

```
root@Kali:~# xsser –gtk
```

- Ve spodní části přepněte Fly Mode na *Intruder* a do prázdného pole vložte URL adresu. Zatrhněte režim testování na *Automatic* a kliknete na tlačítko *Aim!*. Pro zahájení útoku klikněte na tlačítko *FLY!!!*.
- Po vykonání útoku nástroj vypíše do hlavního okna celý postup testování, který si prohlédněte a zjistěte, kolik úspěšných a neúspěšných injections nástroj vykonal. V záložce ***Expert Visor/Report/Suspicious*** jsou vidět všechny injections s označením suspicious, tedy takové, které mohou mít vliv na webovou aplikaci.
- Otestujte nástrojem vytvořenou URL se skriptem *alert*. Mělo by se zobrazit výstražné okno, což je hlavní indikátor, že webová aplikace je zranitelná proti útokům cross-site scripting.
- Upravte předchozí tvar skriptu na následující a podívejte se, jaký vliv bude mít skript na webovou aplikaci:

```
http://192.168.56.101/mutillidae/index.php?page=set-background-color.
php/“<SCRIPT>alert('Pozor utok XSS')</SCRIPT>
```

- Nyní zvolte model pro zneužití **„OWASP 2013“ -> „A3 - Cross Site Scripting (XSS)“ -> „Stored (Second Order)“ -> „Add to your blog“**.
- Upravte skript z předchozího úkolu na následující tvar:

```
<script>alert("Pozor utok XSS")</script>
```

- Upravený skript napište do prázdného textového pole a klikněte na *Save Blog Entry*. Mělo by dojít k zobrazení výstražného okna *alert*. V tomto případě to však, na rozdíl od metody reflected, bude mít dopad na více věcí.
- Zvolte tlačítko *View Blogs*, vyberte z příspěvků od uživatelů volbu *Show All* a potvrďte tlačítkem *View Blog Entries*. Zobrazí se výstražné okno *alert*. Přepněte se na jakýkoliv jiný model zneužití a poté se zkuste zpět přepnout na model této úlohy. Opět se zobrazí výstražné okno *alert*. V obou případech se zamyslete proč tomu tak je.

5)

- Model zneužití zvolte **„OWASP 2013“ -> „A4 - Insecure Direct Object References“ -> „Text File Viewer“**.
- Již známým postupem zachyťte HTTP komunikaci. Jediný rozdíl bude, že po nastavení proxy serveru zvolte ve webové aplikaci libovolný soubor a klikněte na tlačítko *View File*.
- V zachyceném požadavku smažte zvýrazněnou část:

```
textfile=http%3A%2F%2Fwww.textfiles.com%2Fhacking%2Faudi
tool.txt&text-file-viewer-php-submit-button=View+File
```

- Nahraďte jí cestou k souboru *passwd*:

```
textfile=/etc/passwd&text-file-viewer-php-submit-button=View+File
```

- Kliknutím na tlačítko *Forward* v nástroji Burp Suite odešlete upravený požadavek webovému serveru ke zpracování a v prohlížeči Iceweasel dostaneme odpověď. Prohlédněte si obsah souboru a vysvětlete, proč je důležitý.

- Nyní stejným postupem zachyťte HTTP komunikaci, zobrazte si obsah následujících souborů, zjistěte, k čemu jsou určeny a proč jsou důležité:

```
/proc/version
/proc/net/route
/proc/net/arp
/etc/apache2/apache2.conf
```

6)

- Model zneužití zvolte **„OWASP 2013“ -> „A5 - Security Misconfiguration“ -> „Directory Browsing“**.
- Zachyťte HTTP komunikaci nástrojem Burp Suite. Po nastavení proxy serveru stiskněte tlačítko *F5,* sloužící k obnovení webové stránky, čímž se zachytí i HTTP komunikace. Poté vypněte zachytávání kliknutím na tlačítko *Intercept is on* v záložce ***Proxy/Intercept*** a přepněte se do záložky ***Target/Site map***.
- Prohlédněte si podsložku *mutillidae* hlavní větve *https://192.168.56.101*. Kliknutím pravým tlačítkem na podsložku *mutillidae* se zobrazí menu, ze kterého zvolte volbu *Spider this branch*. Nástroj se vás dotáže, zda-li chcete automaticky upravit rozsah průzkumu. Zvolte *Yes*, čímž se nástroj zaměří pouze na zvolenou složku *mutillidae* a všechny její podsložky.
- Přepněte se do záložky ***Spider/Control*** a vyčkejte, dokud počet požadavků ke zpracování (Requests queued) nebude na hodnotě 0.
- Přepněte se zpět do záložky ***Target/Site map***, prozkoumejte shromážděný obsah podsložky *mutillidae* modulem Spider a srovnejte s předchozím obsahem. Věnujte pozornost zejména složce *includes*.
- Vyberte soubor *back-button.inc*, jehož obsah uvidíte v záložce ***Response/Raw***. Zobrazí se vám zdrojový kód, který obsahuje spousty nevyužitých funkcí např. *case "2":* a tím pádem představuje riziko pro webovou aplikaci.
- Najděte i jiné soubory s potenciálně nebezpečným zdrojovým kódem a analyzujte je. Dále vyhledejte soubor, ze kterého zjistíte název hosta databáze, název databáze, uživatelské jméno a heslo.

B LABORATORNÍ ÚLOHA - ÚVOD DO PROBLEMATIKY PENETRAČNÍHO TESTOVÁNÍ

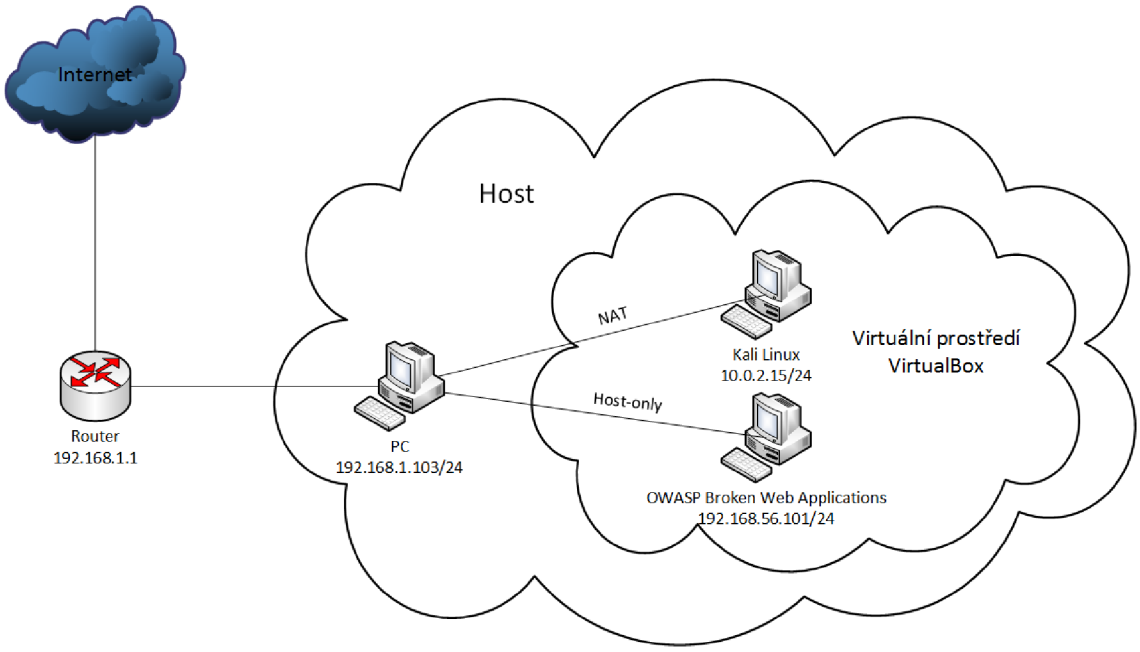
Cílem úlohy je studenty seznámit se základním principem zranitelností SQL injection a Broken Authentication and Session Management z dokumentu OWASP Top 10 2013.

Zadání

1. Seznámit se s prostředím webové aplikace OWASP Mutillidae II.
2. Vyzkoušet si základní metody SQL injection pro úspěšné přihlášení do webové aplikace OWASP Mutillidae II.
3. Porozumět problematice zranitelnosti Broken Authentication and Session Management prostřednictvím změn hodnot parametru uid v zachyceném cookie.

Postup pro vypracování

Vnitřní infrastruktura sítě je zakreslena na obr. A.1. Prostřednictvím virtualizačního programu Oracle VM VirtualBox, umístěného na PC hosta, jsou spuštěny dvě virtuální stanice (Kali Linux, OWASP Broken Web Applications), které komunikují s hostitelským PC v režimu *NAT* a *Host only* z hlediska bezpečnosti sítě. Host všechny požadavky od stanic posílá routeru, který je zpracuje a přepošle dál v závislosti na požadavku.



Obr. B.1: Vnitřní infrastruktura sítě

1)

- Spusťte Oracle VM VirtualBox a v něm stanice Kali Linux a OWASP Broken Web Applications.
- Stanici OWASP Broken Web Applications ponechte spuštěnou na pozadí a přihlaste se do systému Kali Linux jako uživatel *root* s heslem *toor*.
- Otevřete webový prohlížeč Iceweasel, zadejte IP adresu *192.168.56.101* (OWASP Broken Web Applications), zvolte webovou aplikaci OWASP Mutillidae II a zorientujte se v prostředí.

2)

- Model zneužití zvolte **„OWASP 2013“ -> „A1 - Injection (SQL)“ -> „SQLi - Bypass Authentication“ -> „Login“**.
- Nejprve napište do textového pole uživatelského jména jednoduchou uvozovku ***’*** a potvrďte tlačítkem *Login*. Uvozovka by měla donutit webovou aplikaci k zobrazení výjimky, což indikuje její zranitelnost proti útoku SQL injection. Prohlédněte si z chybového hlášení vámi vyplněný a poslaný typ dotazu (query) webovému serveru.
- Mezi nejjednodušší způsob úspěšného přihlášení pomocí SQL injection je splnění podmínky. Předpokládejme, že nevíme uživatelské jméno ani heslo. Zadejte do textového pole uživatelského jména ***’ or ’1’=’1***, heslo vyplňte stejně a zkuste se přihlásit. Budeme přihlášeni jako první uživatel, který je uveden v databázi, což je ve většině případech *admin*. Autentizace proběhla úspěšně, protože podmínka 1=1 je vždy pravdivá.
- Dalším ze způsobů je použití znaků, označující v jazyce SQL komentář. Zadejte do textového pole uživatelského jména ***’ or ’1’=’1’ #*** a zkuste se přihlásit. Autentizace proběhne opět úspěšně, protože vše za znakem *#* je zakomentováno, tedy i přihlašovací heslo.
- V případě, že bychom věděli uživatelská jména, ale nikoliv patřičná hesla, postačí zadat požadované uživatelské jméno a použít komentář pro zbytek dotazu. Předpokládejme, že víme o registrovaném uživateli *user*. Zadejte do textového pole uživatelského jména ***user’ #*** a zkuste se přihlásit. Budeme přihlášeni jako uživatel *user*.

3)

- Klikněte na položku v horním menu ***Login/Register*** a proveďte registraci s libovolnými údaji. Nyní opět klikněte na položku ***Login/Register***, stiskněte *Alt* pro zobrazení menu webového prohlížeče a zvolte volbu ***Nástroje/Cookies Manager+***. Označte všechny zaznamenané cookies a vymažte je.

- Přihlaste se do webové aplikace pod registrovanými údaji a podívejte se, jaké cookies probíhající relace zachytil doplněk Cookies Manager+. Věnujte pozornost cookie *uid*, který označuje identifikační číslo a pořadí uživatele v databázi.
- Stejným postupem zaregistrujte nového uživatele a prohlédněte si i jeho cookies po přihlášení. U cookie *uid* klikněte na *Edit* a změňte hodnotu *Obsah* o 1 menší a potvrďte tlačítkem *Save*. Nyní, stisknutím tlačítka *F5*, byste měli být v pravém horním rohu webové aplikace přihlášeni jako předchozí uživatel, kterého jste zaregistrovali. Při nedostatečné ochraně autentizace můžete být tímto způsobem přihlášeni jako kterýkoliv uživatel registrovaný v databázi. Postupným zmenšováním parametru *uid*, zkuste zjistit všechny uživatele webové aplikace OWASP Mutillidae II.

C OBSAH PŘILOŽENÉHO CD

- Elektronická verze práce