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

INTEGRACE A AUTOMATIZACE NASAZENÍ AKTUALIZOVANÝCH MODULŮ PRO ZÁTĚŽOVÉ TESTOVÁNÍ

INTEGRATION AND AUTOMATION OF DEPLOYMENT OF UPDATED LOAD TESTING MODULES

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. Jakub Jedlička

AUTHOR

VEDOUCÍ PRÁCE — Ing. Mgr. Pavel Šeda, Ph.D.

SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Bc. Jakub Jedlička ***ID:*** 198597

Ročník: 2 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Integrace a automatizace nasazení aktualizovaných modulů pro zátěžové testování

POKYNY PRO VYPRACOVÁNÍ:

Cílem diplomové práce je migrace a aktualizace softwaru pro zátěžové testování. Student se seznámí se softwarem pro zátěžové testovaní Apache JMeter a již existujícími moduly rozšiřující možnosti testovaní. Operační systém zátěžového testeru bude aktualizován na systém Ubuntu ve verzi 20.04. Software pro zátěžové testovaní Apache JMeter bude aktualizován na verzi 5.5 se všemi rozšiřujícími moduly. Student zde provede rekonfiguraci buildovacího systému z Ant na Gradle či Maven a upraví moduly včetně jejich implementace, aby splňovaly požadavky či implementaci rozhraní nové verze Apache JMeteru. Student dále implementuje scripty usnadňující nasazení samotné aplikace včetně vybraných rozšiřujících modulů. Všem typům útoku v DDoS modulu bude přidána možnost nastavit parametry linkové vrstvy a vybraným typům útoků bude přidána podpora IPv6. Rozšiřující moduly budou patřičně otestovány na vhodném testovacím scénáři.

DOPORUČENÁ LITERATURA:

[1] RODRIGUES, Antonio Gomes, Bruno DEMION a Philippe MOUAWAD. Master Apache JMeter - From Load Testing to DevOps: Master performance testing with JMeter. Packt Publishing Ltd, 2019.

[2] ATAR, Afsana. Mastering JMeter 5.0. Packt Publishing, 2020.

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 19.5.2023

Vedoucí práce: Ing. Mgr. Pavel Šeda, Ph.D.

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

UPOZORNĚNÍ:

Autor diplomové práce nesmí při vytváření diplomové práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ABSTRAKT

Tato diplomová práce se věnuje zátěžovému testování pomocí nástroje JMeter a vývoji modulů pro útoky odepřením služeb (DoS). Práce se nejprve zabývá teorií, kde je popsáno zátěžové testování, využité nástroje pro tento typ testování a nakonec typy DoS útoků. Praktická část se v úvodu věnuje migraci operačního systému zátěžového testeru, na kterém probíhalo testování. Dále popisuje aktualizaci nástroje JMeter z verze 4.0 na verzi 5.5, aktualizaci přídavných modulů a opravu chyb, aby bylo možné moduly využívat bez chyb s nástrojem JMeter ve verzi 5.5. Aktualizovaný nástroj JMeter a přídavné moduly jsou nahrány na zátěžový tester. Pro budoucí aktualizace jsou vytvořeny automatizační skripty, které umožní aktualizovat systémové balíčky zátěžového testeru, aktualizovat jak přídavné moduly tak i nástroj JMeter. Následně jsou rozebrány provedená rozšíření modulů, které umožnily zvětšení rozsahu testů. Tyto rozšíření přidávají parametry linkové vrstvy a podporu pro IPv6. V závěrečné části jsou popsány nově vytvořené části modulu *DDoS*. Nové části modulu rozšiřují možnosti testování webových aplikací za použití více IP adres nebo testování pomocí pomalých útoků.

KLÍČOVÁ SLOVA

Config Element, DoS, DDoS, IPv6, JMeter, Sampler, SlowHTTPTest, Trafgen, Zátěžové testování

ABSTRACT

This thesis focuses on performance testing using JMeter and developing modules for denial of service (DoS) attacks. The thesis first deals with the theory, where performance testing is described, the tools used for this type of testing and finally the types of DoS attacks. The practical part starts with the migration of the operating system of the load tester on which the testing was performed. It also describes the upgrade of the JMeter tool from version 4.0 to version 5.5, updating the custom add-on modules and fixing bugs in modules so that the modules can be used without errors with the JMeter tool in version 5.5. The updated JMeter tool and custom add-on modules are uploaded to the load tester. For future updates, automation scripts are created to update the system packages of the load tester, updating both the custom add-on modules and the JMeter tool. The module extensions that have been made to increase the scope of the tests are then discussed. These extensions add mainly link layer parameters and support for IPv6. The final section describes the newly created parts of the *DDoS* module. The new parts of the module extend the capabilities of testing web applications using multiple IP addresses or testing using slow attacks.

KEYWORDS

Config Element, DoS, DDoS, IPv6, JMeter, Performance testing, Sampler, SlowHTTPTest, Trafgen

Vysázeno pomocí balíčku `thesis` verze 4.07; `http://latex.feec.vutbr.cz`

JEDLIČKA, Jakub. *Integrace a automatizace nasazení aktualizovaných modulů pro zátěžové testování*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2023, 77 s. Diplomová práce. Vedoucí práce: Ing. Mgr. Pavel Šeda, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Jakub Jedlička
VUT ID autora:	198597
Typ práce:	Diplomová práce
Akademický rok:	2022/23
Téma závěrečné práce:	Integrace a automatizace nasazení aktualizovaných modulů pro zátěžové testování

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 diplomové práce panu Ing. Mgr. Pavlu Šedovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Dále bych chtěl poděkovat rodině za trpělivost, podporu při studiu a vytváření této práce.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

V současné době se provádí zátěžové testování na většině webových aplikací a serverech. K zátěžovému testování lze použít nástroj Apache JMeter, který je zaměřen hlavně na testování webových aplikací, ale lze ho také využít i na testování některých protokolů na aplikační vrstvě. Jelikož každý den jsou prováděny DDoS (Distributed Denial of Service) útoky na servery, měly by se kromě pokusů o obranu proti těmto útokům servery také testovat na to, co se stane, když obrana selže.

Tato diplomová práce se zabývá vylepšením interoperability mezi nástrojem Apache JMeter ve verzi 5.5 a vyvíjenými moduly na testování DDoS útoků na fakultě elektrotechniky a komunikačních technologií vysokého učení technického v Brně. Protože tyto moduly byly vyvíjeny pro nástroj Apache JMeter ve verzi 4, je zapotřebí jejich uprava a otestování, aby je bylo možné využít s novou verzí nástroje Apache JMeter. Moduly bylo nutné také rozšířit, aby se výrazně zvětšily možnosti testování a tím také mít více testovacích scénářů.

V první kapitole je popsán pojem výkonnostní testování, k jakému účelu slouží a na jaké typy se dělí. V další části kapitoly jsou definovány nástroje pro zátěžové testování a jsou představeny nástroje Apache JMeter, Trafgen a SlowHTTPTest.

Druhá kapitola se věnuje programovacímu jazyku Java, ve kterém je nástroj Apache JMeter napsán. Dále jsou popsány nástroje využívané k automatizaci sestavování, spouštění a testování aplikací napsaných v programovacím jazyce Java.

Třetí kapitola přibližuje průběh DoS (Denial of Service) útok a jeho základní dělení. Dále jsou uvedeni vybraní zástupci DoS útoků, kteří jsou implementováni v modulu *DDoS* pro nástroj Apache JMeter.

Ve čtvrté kapitole je pojednáváno o migraci operačního systému zátěžového testeru. Je zde popsán původní a nový operační systém a poté průběh jeho aktualizace.

Kapitola pátá rozebírá aktualizaci nástroje Apache JMeter a rozšiřujících modulů. Konkrétně uvádí, co bylo provedeno ke zprovoznění jednotlivých modulů, jaká byla úskalí a k jakému účelu slouží.

Šestá kapitola se věnuje opravě chyb v jednotlivých modulech, které se nacházely už v původních modulech nebo vznikly až při aktualizaci modulů.

Sedmá kapitola popisuje skripty pro automatizaci vyžadovaných systémových balíčků na zátěžovém testeru. Dále se věnuje automatizaci sestavování a aktualizaci nástroje JMeter a vytvořených modulů.

Osmá kapitola se věnuje rozšířením přidaných do modulů *ServerEmulator* a *DDoS*. Rozšíření v této kapitole jsou rozebrány jak z programátorského hlediska tak i z hlediska uživatelského.

Poslední devátá kapitola se zabývá vývojem nových částí modulu *DDoS* sloužících k rozšíření možného testování.

1 Testování a nástroje pro testování

Tato kapitola má za cíl přiblížit výkonnostní testování a software používaný při testování. Podrobněji budou rozebrány softwarové nástroje Apache Jmeter a Trafgen.

1.1 Výkonnostní testování

Performance testing neboli výkonnostní testování se využívá k ověření nebo určení škálovatelnosti, rychlosti odezvy, stability a dalších vlastností pro aplikace nebo zařízení. Do výkonnostního testování patří zátěžové a kapacitní testování. Nejčastěji se výkonnostní testování provádí na webových aplikacích a serverech komunikujících s těmito aplikacemi pomocí API (Application Programming Interface) [1].

1.1.1 Zátěžové testování

Zátěžové testování lze rozdělit na takzvané load testing a stress testing.

Load testing slouží k testování a ověření, že aplikace nebo zařízení při normální a maximální zátěži zvládají fungovat dle požadovaných cílů. Testování umožňuje zjistit propustnost, dobu odezvy a jiné chování při testované zátěži [1][2].

Stress testing můžeme označit jako náporové testování a slouží k otestování limitů použitelnosti aplikace nebo zařízení nad dimenzované hodnoty. Dále lze testovat, co se děje po překročení této hranice. Aplikace nebo zařízení by nemělo při překročení kritické hranice způsobit jakoukoliv zranitelnost v systému nebo vedlejší efekty, kterými může být například poškození hardwaru [1][2].

1.1.2 Kapacitní testování

Capacity testing neboli kapacitní testování má za úkol zjistit, jestli jsou splněny specifikace aplikace nebo zařízení. Toto testování slouží k budoucímu plánování, jestli například při nárůstu uživatelů nebo zátěže nastane v určitém bodě problém [1][2].

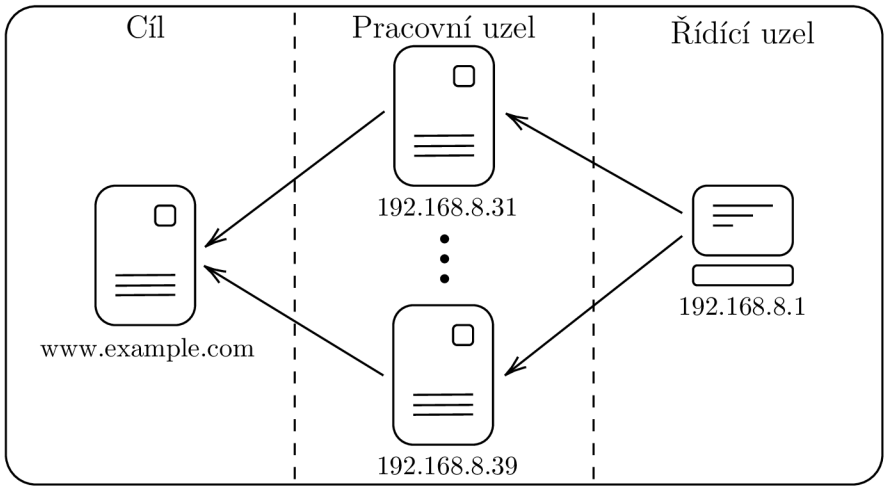
1.2 Software pro zátěžové testování

1.2.1 Apache JMeter

Apache JMeter je aplikace vyvíjená pod záštitou Apache Software Foundation. Aplikace je napsána v programovacím jazyce Java a má otevřený zdrojový kód (open source kód). Apache JMeter slouží k zátěžovému testování a měření výkonosti. První verze Apache JMeter byly určené k testování webových aplikací. Pomocí novějších

verzí lze testovat nejen webové aplikace, ale také například FTP (File Transfer Protocol), databáze pomocí JDBC (Java Database Connectivity), LDAP (Lightweight Directory Access Protocol), emailové protokoly jako POP3 (Post Office Protocol verze 3) nebo IMAP (Internet Message Access Protocol) a mnoho dalších služeb. Apache JMeter obsahuje GUI (Graphical User Interface), ve kterém se dají vytvářet a poté spouštět testy. Také lze spustit vytvořené testy přímo z terminálu (příkazové řádky). Za největší výhodu lze označit využití externích modulů pro testování, které lze přímo vyvinout nebo přidat od vývojářů třetí strany [3].

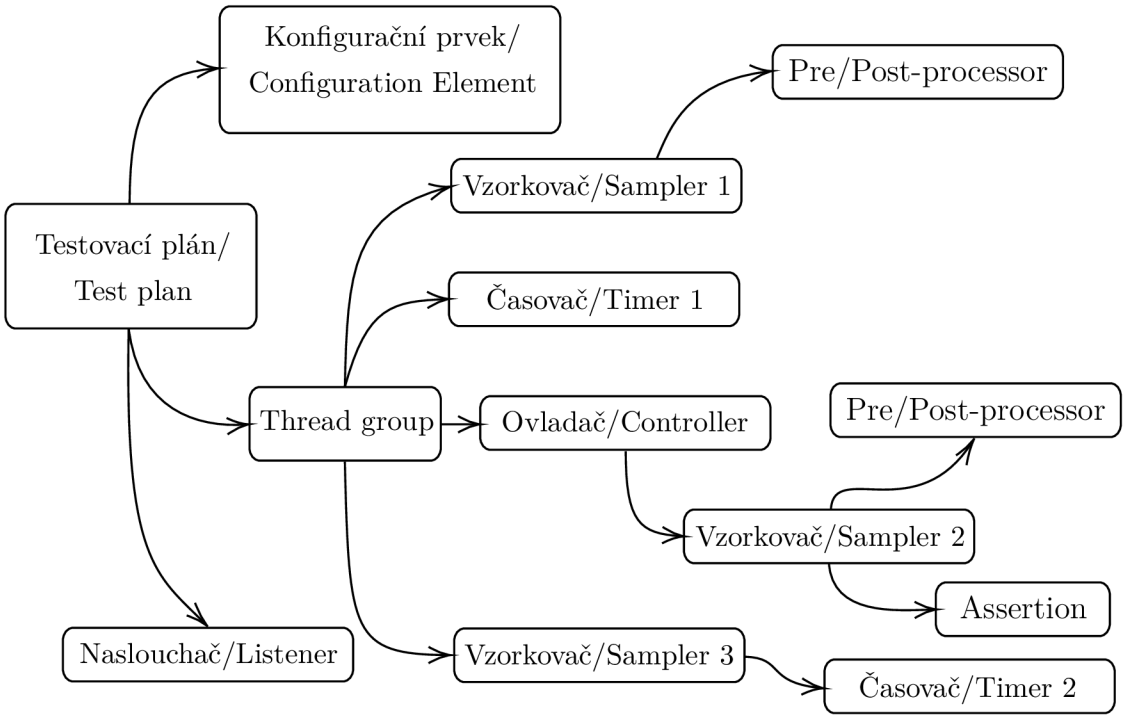
Apache JMeter lze využít pro testování z jedné stanice a také pro distribuované testování neboli testování z více zařízení současně. Při distribuovaném testování se v infrastruktuře nachází jeden řídící uzel a několik pracovních uzlů, které běží v módu *jmeter-server* a vykonávají dotazy na testovaný server nebo webovou službu. Řídící uzel má za úkol řízení a ovládání testů pro pracovní uzly. Na obrázku 1.1 lze vidět jednoduché znázornění infrastruktury pro distribuované testování [4].



Obr. 1.1: Ukázka infrastruktury při distribuovaném testování [4]

Základem každého testu v Apache JMeter je *test plan* neboli testovací plán. Testovací plán může obsahovat jeden nebo více komponentů, kterými jsou thread groups, listeners, timers, assertions, configuration elements, pre-processors a post-processors. Uvedené komponenty budou popsány dále v této podkapitole. Jak může vypadat propojení všech těchto komponentů lze vidět na obrázku 1.2. Testovací plán umožňuje konfigurovat spouštění *thread groups* jednotlivě nebo současně a poskytuje možnost definovat uživatelské proměnné, které mohou být dále použity ve skriptech [5][6].

Thread groups je vstupní bod pro jakýkoliv testovací plán a musí v něm být obsažen alespoň jedenkrát. *Thread groups* obsahuje ovladače (Controllers), vzorkovače (Samplers), které nelze umístit v testovacím plánu jinam a může obsahovat také



Obr. 1.2: Ukázkové propojení komponentů Apache JMeter [6]

posluchače (Listeners). Obsah *thread groups* se označuje jako test. *Thread groups* umožňuje nastavovat počet vláken neboli uživatelů, čas za jaký budou všechny tyto vlákna aktivní a počet opakování testů [5][6].

Controllers neboli ovladače se dělí na sampler controllers (vzorkovací ovladače) a logical controllers (logické ovladače). Vzorkovací ovladače posílají HTTP/S (Hypertext Transfer Protocol/Secure), FTP, JDBC a další dotazy na server. Logické ovladače se starají o logiku odesílání dotazů na server [5][6].

Samplers neboli vzorkovače odesílají dotazy na cílový server a zpracovává odpovědi. Apache JMeter obsahuje několik základních vzorkovačů. Příkladem může být JDBC dotaz, HTTP dotaz, FTP dotaz a další typy [5][6].

Listeners se dájí označit jako posluchače. Slouží k získávání výsledků testů, které lze poté zpětně analyzovat. Listener je možné přidat do testovacího plánu kdekoliv a získané výsledky lze ukládat také do souboru pro pozdější analýzu a zkoumání [5][6].

Timers neboli časovače lze využít pro vytvoření pauzy mezi jednotlivými dotazy, které jsou odesílány. Pokud časovač chybí, Apache Jmeter odesílá všechny dotazy bez prodlevy, což může vést k nerealistickému scénáři, protože reální uživatelé nemohou odesílat tolik dotazů v tak krátkém časovém úseku [5][6].

Assertions umožňuje ověřovat přijaté odpovědi, aby se zjistilo, jestli server vrací správné odpovědi a tím ověří, zda jsou stejné jako očekávané [5][6].

Configuration elemens neboli konfigurační prvky se využívají k upravení dotazu nebo přidání prvku dotazu, který je posílán vzorkovačem [5][6].

Pre-processor a **post-processor** slouží k vykonávání akcí před nebo po vykonání dotazu vzorkovačem. Pre-processor se využívá hlavně k úpravě proměnných v dotazu, které nebyly extrahovány z odpovědi. Post-processor je používán k získávání některých hodnot z odpovědí [5][6].

1.2.2 Trafgen

Pro testování nových nebo stávajících aplikací, protokolů nebo sítí slouží generátory provozu. Generátory jsou využívané pro vytváření pokusů ve zkušebních sítích tak, aby tyto sítě odpovídaly co nejvíce reálné infrastruktuře, která bude zavedena do provozu. Bez testování generace zátěže na pokusnou síť může dojít při nasazení do reálného prostředí k nedostatečnému výkonu nebo nepředvídatelnému chování [7].

Generátory se snaží tvářit jako velké množství zařízení. Proto v nich lze nastavovat parametry jako je zdrojová a cílová adresa, druh přenosového protokolu (UDP[1], TCP[2]), velikost uživatelských dat označovaných jako payload a další. Parametry, které se dají upravovat se liší v závislosti na generátoru [8].

Existuje mnoho generátorů provozu od různých společností. Za nejznámější lze označit Ostinato Packet Generator, TRex od společnosti Cisco, Iperf, Moongen Packet Generator a Trafgen. V navazujícím textu je přiblížen generátor provozu Trafgen, a to z důvodu jeho vyžití v praktické části této diplomové práce [8].

Trafgen je součástí balíčku *netsniff-ng*. Podpora balíčku *netsniff-ng* je pouze pro operační systém Linux. Z tohoto důvodu není co se týče nároků na paměť úložiště balíček příliš obsáhlý. Trafgen využívá paketové rozhraní, které umožňuje posílání paketů typu raw na druhé vrstvě OSI (Open Systems Interconnection) modelu. Generátor Trafgen umožňuje fuzz testování, jelikož konfiguraci vstupního paketu lze vytvářet s náhodnými hodnotami, kdy se každá hodnota vybere až při odesílání paketu. Tak je možné mít definovaný jeden vstupní paket, výstupních paketů pak bude více [9][10]. Trafgen podporuje preprocesorová makra jazyka C. Dále umožňuje načítání paketů z konfiguračního souboru, do kterého se dá vložit jeden nebo více paketů nebo lze tento konfigurační soubor vytvořit z PCAP[3] souboru. Na výpisu kódu 1.1 se nachází ukázka konfiguračního souboru, kde je zdrojová adresa generována až při odesílání ve zvoleném rozsahu zadaném decimálně [9].

[1] User Datagram Protocol
[2] Transmission Control Protocol
[3] Aplikační rozhraní pro zachytávání komunikace

Výpis 1.1: Příklad konfiguračního souboru s náhodnou zdrojovou adresou

```
#define ETH_P_IP  0x0800
{
  eth(daddr=72:65:84:83:85:78, saddr=69:77:73:75:85:39,
  proto=ETH_P_IP),
  ipv4(ttl=64, ver=4, flags=0b01000000, frag=0, df,
  da=192.168.9.3, sa=drnd(3931082007, 3939393939)),
  udp(sport=3108, dport=2007),
  fill('B',16),
}
```

1.2.3 SlowHTTPTest

Testování webových serverů pomocí HTTP/s protokolu může probíhat pomocí pomalých útoků. Pomalé útoky typu DoS nabízí nástroj SlowHTTPTest. Tento nástroj obsahuje 4 typy pomalých útoků, kterým lze zadat mnoho parametrů. Hlavními typy útoků jsou slow headers (Slowloris), slow Post (R.U.D.Y.[4]), slow Read a range attack (Apache killer). Nespornou výhodou je, že uvedený nástroj kromě více druhů útoků poskytuje také výstup útoku ve formátu `csv` a `html` [11].

[4] R U Dead Yet?

2 Programovací jazyk a nástroje

Tato kapitola se zaměří na programovací jazyk Java, ve kterém jsou napsané přídavné moduly a také celý Apache JMeter. Dále budou popsány základní nástroje pro řízení a automatizaci sestavování a testování aplikací napsaných v jazyce Java a Kotlin.

2.1 Programovací jazyk Java

Programovací jazyk Java je druhý nejpopulárnější jazyk dle PYPL (PopularitY of Programming Language Index[1]). Společnost Sun Microsystems, Inc. vydala v roce 1995 první dostupnou verzi. V roce 2009 byla Java odkoupena společností Oracle, která nabízí tento programovací jazyk jako proprietární, ale zároveň existuje i verze s otevřeným zdrojovým kódem pojmenovaná OpenJDK [12].

Java je objektově orientovaný multiplatformní jazyk, který je bezpečný, rychlý a dlouhodobě nabízí podporu některým verzím. Tyto verze se nazývají LST (Long Term Support) a jsou to verze 7, 8, 11 a 17 programovacího jazyka Java. Model dlouhodobé podpory zajišťuje vyvíjenému softwaru možnost setrvat na jedné verzi programovacího jazyka Java [13][14].

Při vydání Java 9 došlo asi k největší změně, kdy byl zaveden nový komponent s názvem *module*, který spojuje související balíčky. V ukázce níže je možné vidět rozdíl ve struktuře mezi Java 8 a Java 9 [15].

```
Balíček
└── Typy (třídy, rozhraní, ...)
    ├── Kód
    ├── Data
    └── Prostředky
        ├── xml
        ├── obrázky
        └── soubory properties
```

```
Modul
└── Balíček
    └── Typy (třídy, rozhraní, ...)
        ├── Kód
        ├── Data
        └── Prostředky
            ├── xml
            ├── obrázky
            └── soubory properties
```

Ke spuštění Java aplikace je potřeba JRE (Java Running Environment). JRE z největší části obsahuje JVM (Java Virtual Machine), které vytváří prostředí pro interakci mezi hardwarem a Java aplikací. Pro vytvoření aplikace je potřeba vytvořený kód zkompilovat. K tomu slouží JDK (Java Development Kit). JDK obsahuje základní knihovny pro vývojáře a JRE. Do verze 8 programovacího jazyka Java bylo JRE a JDK vydáváno zvlášť, od verze 9 je vydáváno pouze JDK [16][17].

[1] `https://pypl.github.io/PYPL.html`

2.2 Nástroje

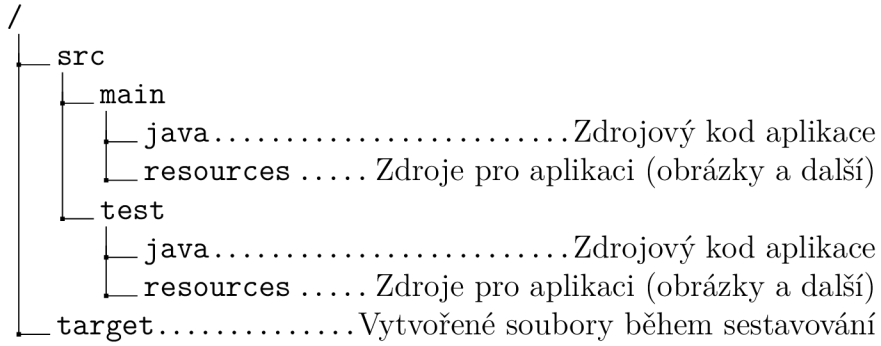
2.2.1 Apache Ant

Apache Ant (Another Neat Tool) je z nástrojů nejstarší a jeho hlavním cílem je provádět automatizaci pro standardní úlohy, které jsou využívané v projektech napsaných v programovacím jazyce Java. Do těchto úkolů lze zařadit kompilaci, zabalení jar souborů nebo spuštění unit testů. Apache Ant dále nabízí předem definované úlohy, které mohou být dále rozšířeny. Hlavní nevýhodou Apache Ant je chybějící manažer závislostí[2]. V nástrojích popsaných dále takové manažery umožňují stáhnout potřebnou knihovnu a tranzitivní závislosti této knihovny. Při absenci manažeru závislostí se musí všechny tyto knihovny importovat manuálně [18].

V nástrojích tento manažer umožňuje stáhnout potřebnou knihovnu, která může mít další stažené závislosti.

2.2.2 Apache Maven

Apache Maven je projektový manažer a nástroj pro sestavení softwaru napsaného v jazyce Java a Kotlin. Hlavním cílem Apache Maven je zjednodušit sestavování a správu projektu. Základem je konfigurační soubor POM (Project Object Model), který využívá struktury xml (Extensible Markup Language). Pomocí Apache Maven lze získat informace o projektu jako jsou výsledky unit testů a závislé knihovny. Apache Maven zavedl jednotné rozvržení adresářů pro projekt, které je zobrazeno na obrázku 2.1. Toto rozvržení usnadňuje přehlednost pro vývojáře, kteří pracují na více projektech [18][19].



Obr. 2.1: Standardní rozvržení Apache Maven projektu

[2] Anglicky dependency manager

Sestavování projektu probíhá v cyklech nazývaných fáze. Za nejčastější fáze se dají označit [18]:

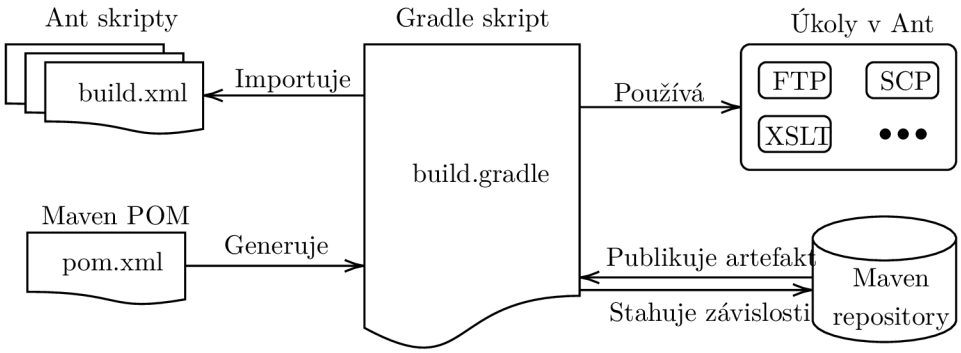
- Kompilace zdrojového kódu
- Spuštění jednotkových a integračních testů
- Sestavení artefaktu
- Vydání artefaktu[3] do lokálního nebo vzdáleného repozitáře

2.2.3 Gradle

Gradle je podobně jako Apache Maven nástroj pro sestavování softwaru. Gradle přebírá některé zásadní výhody, které přinesl Apache Maven, jako například jednotné rozvržení adresářů. Oproti Apache Maven je Gradle vysoce výkonný co se týká získávání závislostí[4] a také rychlosti při sestavování, jelikož ho paralelizuje. Dalším zásadním rozdílem je, že pro konfiguraci se využívá jazyka Groovy místo xml struktury. Pro dosáhnutí stejného výsledku v Gradle bude mít konfigurační soubor mnohem méně řádků než Apache Maven a je lépe čitelný, jelikož se spíše podobá skriptu než složité struktuře xml. Základním konfiguračním souborem je build.gradle [18][20].

Gradle má přesně definované tři fáze. První fází je inicializace, ve které se nastavuje prostředí pro sestavování. Druhou fází je konfigurace, která vytváří a konfiguruje graf úloh. Poslední fází je realizace, která postupně vykonává graf úloh v pořadí, ve kterém bylo určeno [18][21].

Gradle umožňuje integraci s dalšími nástroji pro sestavení jako je Apache Maven a Apache Ant. Na obrázku 2.2 je možná kombinace propojení těchto nástrojů pomocí Gradle [18].



Obr. 2.2: Propojení nástrojů pomocí Gradle [18]

[3]Artefakt je soubor, který projekt využívá nebo je projektem produkován. Nejčastěji je ve formátu jar.

[4]Anglicky dependency neboli knihovny pro jazyk Java

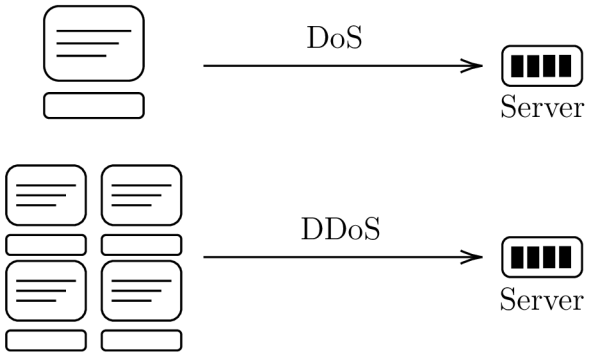
3 Útoky pomocí odepření služby

Kapitola se věnuje vysvětlení pojmů DDoS (Distributed Denial of Service) útok, DoS (Denial of Service) útok a možným typům těchto útoků. Dále jsou podrobněji popsány některé vybrané typy DDoS útoků.

3.1 Odepření služeb a distribuované odepření služeb

DoS neboli odepření služby je útok, jehož cílem je narušení služeb tím, že se nesnaží narušit službu samotnou, ale přístup k službě nebo serveru. Přístup se narušuje pomocí zasílání proudu paketů, které mají za úkol co nejvíce vytížit službu nebo server [22].

DDoS neboli distribuované odepření služby využívá základů DoS útoku, útok není veden z jednoho zařízení, ale z více zařízení. Rozdíl mezi DoS a DDoS útokem by se dal popsat pomocí vztahu mezi útočníkem a cílovou službou podobně jako v databázích. Pro DoS by to byl vztah 1:1 a pro DDoS N:1. Rozdíl mezi DoS a DDoS lze vidět na obrázku 3.1 [22].



Obr. 3.1: Rozdíl mezi DoS a DDoS útoky [23]

3.2 Základní dělení odepření služeb

DoS útoky lze dělit na dva základní typy:

- Útok na vyčerpání pásma
- Útok na vyčerpání zdrojů

Útok na vyčerpání pásma zaplavuje síť oběti nežádoucím provozem, který vytěžuje linku, takže běžní uživatelé budou vnímat zpomalení odezvy služby nebo přímo její výpadek. Tento typ útoku lze dělit na dva druhy. Prvním druhem je takzvaný záplavový útok, který má za cíl posílání velkého objemu paketů a tím co

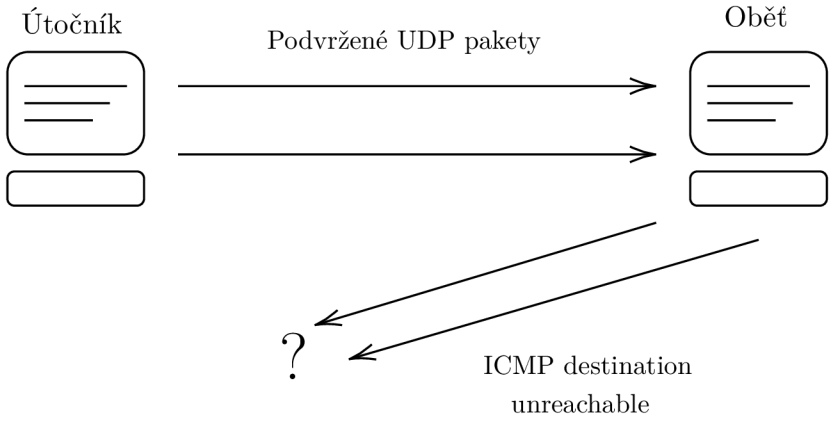
nejvíce zatížit propustnost linky nebo ji celou vyčerpat. Druhým je amplifikační útok, který posílá zprávu na takzvanou broadcast (všesměrovou) adresu a z tohoto důvodu všechny systémy v síti přeposílají zprávu na službu oběti. Broadcast adresa je adresa sloužící pro hromadné rozesílání zpráv [24].

Útok na vyčerpání zdrojů využívá zasílání poškozených paketů, který vyplýtvají zdroje služby nebo zdroje cílové sítě. Za zdroje lze označit například paměť zařízení nebo počet dostupných spojení [24].

3.3 Typy DoS útoků

3.3.1 UDP flood

UDP flood je jedním ze zástupců záplavového útoku. Útok využívá běžného chování serveru na UDP datagramy, kdy UDP protokol nevytváří spojení, ale funguje na bázi otázky a odpovědi. Při tomto typu útoku útočník odesílá UDP datagramy, které mají podvrženou zdrojovou IP (Internet Protocol) adresu a náhodný port. Oběť se následně snaží zjistit, jestli odesílatel naslouchá na portu. Pokud nenaslouchá, oběť posílá ICMP (Internet Control Message Protocol) pakety s obsahem „destination unreachable“ k oznámení, že nelze navázat spojení na daném portu. Jestliže útočník zasílá pakety oběti opakovaně, zaplní se fronta s přijatými datagramy a tím oběť přestane přijímat nová spojení. Tento útok je zobrazen na obrázku 3.2 [25][26].



Obr. 3.2: Ukázka průběhu UPD flood [27]

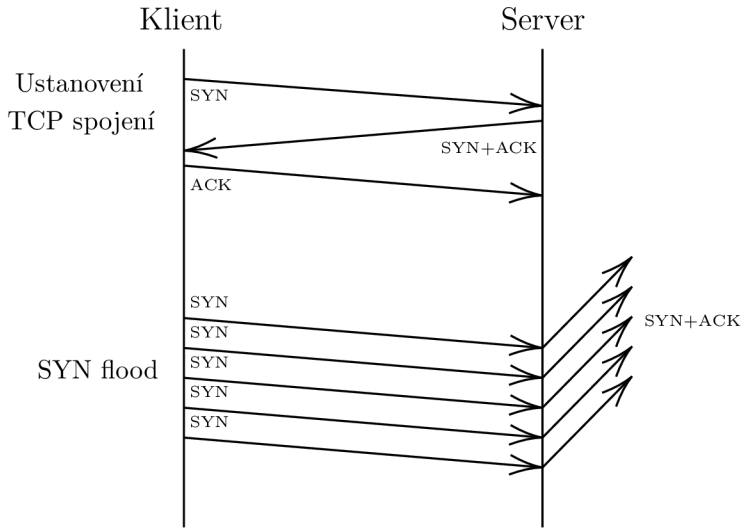
3.3.2 SYN flood

SYN flood je podobně jako UDP flood dalším známým zástupcem záplavových útoků. Tento útok využívá nutnosti TCP protokolu udržovat i polootevřené spojení po určitý časový úsek. TCP protokol potřebuje mít vytvořené spojení, aby strany

mohly komunikovat. Spojení se vytváří pomocí třícestného handshake (three-way handshake) a tento mechanizmus má tři fáze. V první fázi klient vysílá SYN požadavek. V druhé fázi server na tento požadavek odpoví SYN-ACK a ve třetí fázi klient odpoví ACK [26][28].

Útok probíhá tak, že útočník vysílá požadavky na nové spojení pomocí nastavení příznaku zprávy na SYN. Dále tento požadavek neobsahuje zdrojovou adresu zařízení útočníka, ale podvrženou adresu a server na ni odpovídá pomocí SYN-ACK. Server se tak nedočká ACK odpovědi a musí udržovat spojení polootevřené do doby, než vyprší určitý časový úsek [26][28].

Útok se snaží vyčerpat všechna dostupné spojení, které má server k dispozici. Po vyčerpání dostupných spojení nejsou legitimní uživatelé schopni vytvořit TCP spojení se serverem. Průběh navázání TCP spojení a útoku lze vidět na obrázku 3.3 [26][28].



Obr. 3.3: Ukázka průběhu SYN flood [28]

3.3.3 ICMP flood

Útok ICMP flood využívá k útoku protokol ICMP. ICMP protokol slouží k řešení problémů, které se vyskytují v síti. Protokol obsahuje několik druhů zpráv, které jsou posílány a nejvyužívanějším druhem je echo [26][29].

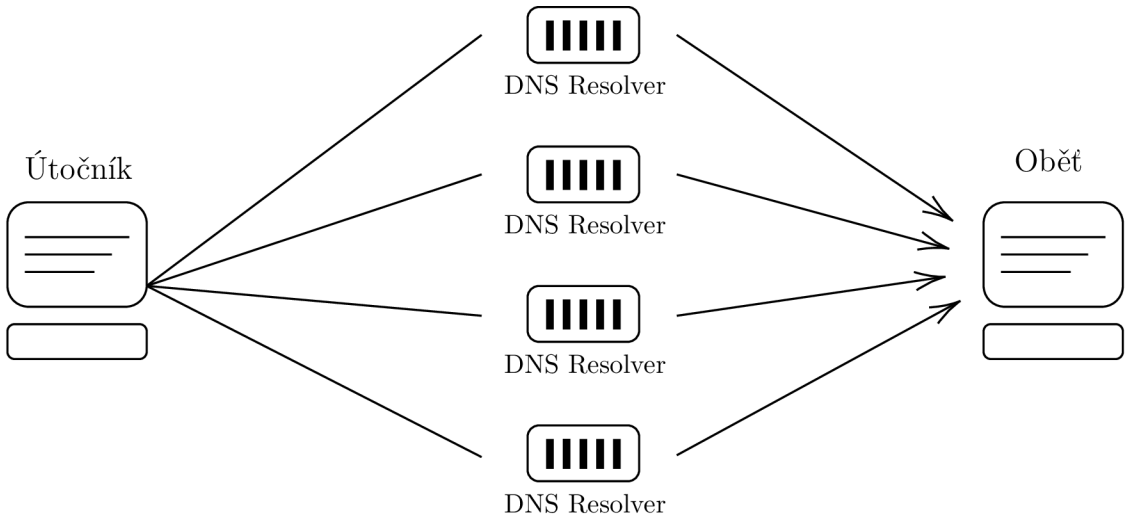
Útočník odesílá velké množství ICMP zpráv, na které se systém oběti snaží odpovídat. Při větším zatížení systému oběti nemusí zvládat odpovídat na všechny zprávy, jelikož útok zahltil síťové nebo systémové prostředky. K provedení DoS útoku je potřebné mít větší rychlost připojení než jakou má systém oběti, jelikož většinou se útočí na servery, které mají dostatečné systémové prostředky pro zpracování zpráv [26][29].

3.3.4 DNS flood

DNS flood používá stejný vektor útoku jako UDP flood. Jediným rozdílem je, že neposílá pakety s náhodným obsahem, ale žádosti o překlad doménového jména na IP adresu. Podobnost útoků je způsobena tím, že DNS (Domain Name System) využívá UDP na portu 53 [30].

3.3.5 DNS amplification

DNS amplification útok využívá dotazu o malé velikosti, na kterou DNS server odpovídá odpovědí o výrazně větší velikosti. Útočník zasílá požadavek o překlad doménového jména na IP adresu všem dostupným DNS serverům. Tento požadavek má podvrženou IP adresu, která je shodná s IP adresou oběti. DNS server poté neodpovídá útočníkovi, ale oběti. Dotazy jsou strukturovány tak, aby odpovědi od DNS serverů byly co nejobsáhlejší. Tímto způsobem je docíleno, že počáteční provoz útočníka je zesílen mnohonásobně. Na obrázku 3.4 lze vidět průběh útoku [31].



Obr. 3.4: Ukázka průběhu DNS amplification útoku [31]

3.3.6 Slowloris

Slowloris je nejznámějším zástupcem z kategorie pomalých útoků a byl vytvořen Robertem „RSnake“ Hansenem. Využívá zranitelnosti v HTTP (Hypertext Transfer Protocol) protokolu definovaného v RFC (Request For Comments) 2616. Zranitelnost spočívá v odeslání HTTP GET požadavku, který je záměrně nedokončený. HTTP služba při přijetí požadavku, který končí dvěma znaky pro nový řádek, pak nechá otevřené spojení. Výsledkem této zranitelnosti je vyčerpání možných spojení serveru [32].

Slowloris klient otevírá zároveň několik spojení pomocí nedokončených GET požadavků, dokud server nedosáhne limitu otevřených spojení. Před vypršením spojení Slowloris klient zašle další část požadavku, který je taktéž nedokončený. Tyto otevřená spojení poté znemožňují uživatelům přistoupit k webové službě, jelikož Slowloris klient udržuje svoje otevřená spojení po neurčitou dobu [32].

3.3.7 Slow POST

Slow POST častěji označován jako „R U Dead Yet?“ nebo „R.U.D.Y“ patří stejně jako útok Slowloris do kategorie pomalých útoků. Útok nezahlcuje server mnoha požadavky, ale vytváří několik zdlouhavých požadavků. Tyto požadavky mají za cíl udržet spojení po co nejdelší dobu, kdy odesílají data pomalým tempem. Útok je možné provést na každou webovou aplikaci, která využívá formuláře přijímané pomocí metody POST [33].

Průběh slow POST začíná nalezením formuláře ve webové aplikaci. Následně útočník odešle formulář, při kterém zachytí POST dotaz. V hlavičce POST dotazu bude upozornění, že obsah bude dlouhý. Tento dotaz se odesílá velice pomalu, protože jsou odesílána data po malých částech (až 1 byte) v náhodném intervalu. Náhodný interval se pohybuje kolem 10 sekund. Webový server ponechává toto spojení otevřené, jelikož se domnívá, že se jedná o uživatele s pomalým připojením. Útočník tyto pomalé dotazy vytváří zároveň, což vede k vyčerpání spojení [33].

3.3.8 Slow Read

Slow Read je dalším typem pomalého útoku. Tento útok se liší od Slowloris a slow POST útoku tím, že neodesílá pomalu HTTP dotazy, ale pomalu přijímá HTTP odpovědi. Slow Read zneužívá řízení toku protokolu TCP [34].

Útok začíná odesláním legitimního požadavku pomocí třícestného handshake (navázání TCP spojení). Následně útočník zpomalí HTTP odpovědi tak, že inzeruje malou velikost okna. Jestliže útočník bude inzerovat velikost okna rovnou nule, tak webový server přestane odesílat data. Přestože webový server neodesílá žádná data, udržuje spojení stále aktivní a tím útočník vyčerpá všechny spojení [34].

3.3.9 HTTP(S) flood

HTTP flood útok využívá základních dotazů HTTP protokolu a z tohoto důvodu je těžké rozeznat, jestli se jedná o útok nebo legitimní provoz. HTTP flood útok lze provádět pomocí GET nebo POST dotazů [35].

Útok pomocí **HTTP GET** dotazu posílá dotazy na webovou službu oběti a cílí hlavně na část webové služby, která obsahuje obrázky, soubory a další data v co

největší velikosti. Služba na tyto dotazy odpovídá a posílá odpovědi, které obsahují obrázky a soubory. Z tohoto důvodu dojde k zahlcení webové služby, která nezvládá odpovídat na všechny dotazy a tím nevyřizuje legitimní dotazy [35].

Útok pomocí **HTTP POST** dotazů je nejčastěji prováděn na webovou službu, která obsahuje nějaký formulář. Pokud je formulář odeslán, je od webové služby požadováno zpracování. Zpracování neprobíhá pomocí webové stránky, ale v pozadí na serveru. Server po přijetí formuláře obsah ve formě dat zpracovává, příkladem je autentizace uživatele v databázi. Tyto dotazy vyčerpají dostupné zdroje a omezí nebo kompletně zastaví webový server [35].

HTTPS flood je útok na stejném principu jako HTTP flood, jen komunikace mezi klientem a serverem je šifrována pomocí TLS (Transport Layer Security). HTTPS flood navíc od HTTP flood může způsobit vyčerpání výpočetních zdrojů serverů, protože při každém spojení provádí ustanovení nových klíčů pro asymetrickou kryptografii [36][37].

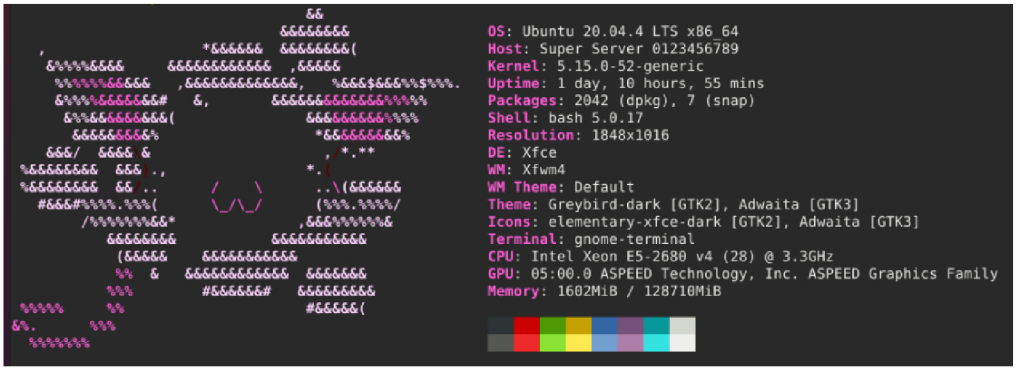
4 Migrace operačního systému

Kapitola stručně popisuje hardware zátěžového testeru a postup průběhu migrace na nový operační systém.

4.1 Hardware zátěžového testeru

Zátěžový tester zobrazený na obrázku 4.1 je osazen procesorem Intel Xeon E5-2680 v4, který disponuje 14 jádry a 28 vlákny. Základní frekvence procesoru je 2,40 GHz a při maximálním výkonu dosahuje až 3,30 GHz. Velikost operační paměti RAM (Random Access Memory), kterou má zátěžový tester k dispozici, je 128 710 MiB. Dále má šest síťových portů umožňujících maximální rychlost 1 Gbit/s a čtyři porty podporující rychlost až 10 Gbit/s. Všechny tyto parametry až na síťové porty lze vyčíst z obrázku 4.2, který zobrazuje výpis informací v příkazové řádce pomocí nástroje neofetch[1].

Obr. 4.1: Zátěžový tester



Obr. 4.2: Výstup programu neofetch

[1]Dostupný na `https://github.com/dylanaraps/neofetch`

4.2 Migrace na nový operační systém

Původní operační systém na zátěžovém testeru byla distribuce CentOS v majoritní verzi 7. Minoritní verzi nelze specifikovat, protože v průběhu migrace nebyla zaznamenána.

CentOS byla takzvaná downstream distribuce operačního systému Linux, která vychází z distribuce Red Hat Enterprise Linux (RHEL). Nové verze CentOS dostaly přídavek stream do jména distribuce a změnily se na takzvanou upstream distribuci, která nevychází z distribuce RHEL, ale slouží jako zdroj pro minoritní verze distribuce RHEL [38].

Distribuce operačního systému, na který byl zátěžový tester migrován, je Ubuntu ve verzi 20.04 LTS. Vývoj této distribuce je veden společností Canonical a ta je založena na distribuci Debian. Ubuntu má velké množství odnoží, které se liší hlavně vzhledem, jelikož používají různé DE (Desktop Enviroment) jako KDE, Xfce a další. Samotné Ubuntu používá v základu GNOME DE [39].

Na rozdíl od CentOS, kde má každá verze dlouhodobou podporu, Ubuntu vydává verze s dlouhodobou podporou každé dva roky. Verze Ubuntu jsou vydávány každých šest měsíců a ty, které nemají dlouhodobou podporu, jsou podporovány pouze devět měsíců. U verzí s dlouhodobou podporou je délka podpory stanovena na pět let.

Jak bylo zmíněno výše, migrace probíhala z CentOS 7 na Ubuntu 20.04 LTS. Přeinstalování bylo provedeno v následujících fázích:

První fází bylo stažení a otestování obrazu Ubuntu ve formátu iso. Na virtuálním stroji se otestovala správnost instalace obrazu Ubuntu.

Druhou fází bylo vytvoření zaváděcího[2] USB (Universal Serial Bus) flash disku pomocí nástroje Ventoy, který umožňuje zapsat více obrazů operačních systémů na jeden USB flash disk.

Ve **třetí fázi** se provedla instalace operačního systému Ubuntu na zátěžový tester. V této fázi se vyskytl problém, který se neprojevoval při testování na virtuálním stroji. Problém byl opraven po několika přeinstalováních systému a po přeinstalaci zaváděcího USB flash disku, který tento problém pravděpodobně způsoboval. Problém se projevoval tak, že instalace se provedla v pořádku, ale při restartování zátěžový tester nebyl schopný zavést operační systém. Operační systém Ubuntu byl instalován ve verzi 20.04, kdy se využívalo instalace pomocí GUI. Tento operační systém měl specifikaci pro osobní počítače a ne pro server, nebyla nainstalována minimalistická verze, ale základní, která obsahuje více balíčků.

[2] Často označováno jako bootovací

5 Aktualizace Apache JMeter a rozšiřujících modulů

Kapitola popisuje aktualizaci nástroje Apache JMeter na novější verzi a dále převod rozšiřujících modulů tak, aby byly podporovány v nové verzi Apache JMeter.

5.1 Aktualizace Apache JMeter

Před aktualizací byl nástroj Apache JMeter používán ve verzi 4. Tato verze využívala k sestavování a spouštění jednotkových testů nástroj Apache Ant. Do konfiguračního souboru pro nástroj Apache Ant bylo nastaveno i automatické sestavení a překopírování rozšiřujících modulů, které bude rozebráno v následující podkapitole.

Nástroj Apache JMeter byl aktualizován na nejnovější verzi, která v době této aktualizace byla 5.5. Oproti původní verzi se výrazně změnil, jelikož místo nástroje Apache Ant byl použit nástroj Gradle. Další důležitou změnou bylo, že nová verze je testována pouze s Java 11 a 17. Java ve verzi 8 byla podporována původní verzí nástroje Apache JMeter, ale v aktuální verzi už podporována není. Tento fakt se musel brát do úvahy, poněvadž to může ovlivnit chování rozšiřujících modulů.

Aktualizace verze nástroje Apache JMeter probíhala vytvořením privátního repozitáře na gitlabu. Do tohoto repozitáře byl nahrán kód z oficiálního repozitáře nástroje Apache JMeter[1] ve verzi 5.5. Následně bylo otestováno, zda lze sestavit nástroj Apache JMeter ze zdrojového kódu pomocí nástroje Gradle a poté nástroj Apache Jmeter spustit, jak lokálně na vývojářském počítači, tak na virtuálním počítači s operačním systémem Ubuntu ve verzi 20.04.

Na vývojářském počítači se nevyskytl žádný problém se sestavením pomocí příkazu `./gradlew build` nebo `gradle build`. Při druhé verzi příkazu musí být gradle nainstalován na zařízení, kde se příkaz spouští. Na virtuálním počítači nastal problém při sestavování a to konkrétně při vytváření dokumentace. Komplikace je důsledkem chybějícího obsahu, starajícího se o dokumentaci v základním balíčku openJDK, který je stahován pomocí nástroje `apt-get`. Tuto komplikaci lze vyřešit dvěma způsoby. Prvním je doinstalování balíčku openJDK pro vývojáře, tento postup lze aplikovat pouze pokud jsou k dispozici na zařízení administrátorská práva, jinak není možné balíček nainstalovat. Druhým způsobem je přidání argumentu `--exclude-task :src:dist:javadocAggregate` pro nástroj Gradle. Takže příkaz pro sestavení bude mít podobu jedné z uvedených variant:

- `./gradlew build --exclude-task :src:dist:javadocAggregate`
- `gradle build --exclude-task :src:dist:javadocAggregate`

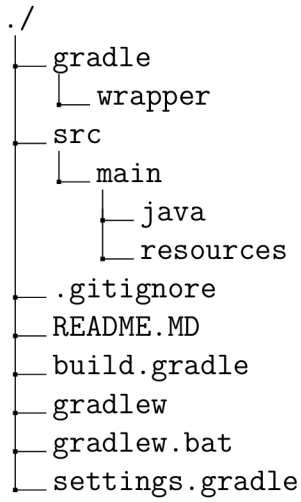
[1] `https://github.com/apache/jmeter`

5.2 Aktualizace rozšiřujících modulů

Rozšiřujících modulů pro aktualizaci bylo celkem sedm. Každý z těchto modulů byl v předchozí verzi přímo zahrnut v upravené verzi nástroje Apache JMeter. Protože moduly byly zahrnuty v repozitáři nástroje Apache JMeter, mohly by se v dlouhodobém časovém horizontu vyskytnout problémy s udržitelností a správou privátního repozitáře. Z tohoto důvodu byl pro každý modul vytvořen vlastní privátní repozitář. Rozdělení vede k jednodušší správě repozitáře a kódu i k možnosti sestavení každého modulu samostatně a nezávisle na ostatních modulech.

Jelikož nástroj Apache JMeter využívá nástroje Gradle, všechny rozšiřující moduly tento nástroj taktéž využívají. Rozšiřující moduly byly převáděny tak, aby podporovaly především programovací jazyk Java ve verzi 17. Toto umožní, že moduly bude možné používat po delší dobu. Každý modul byl vyzkoušen i s verzí 11 programovacího jazyka Java a žádný problém nebyl viditelný, ale v rámci této diplomové práce byla zaměřena pozornost na interoperabilitu mezi programovacím jazykem ve verzi 17 a nástrojem Apache JMeter s rozšiřujícími moduly.

Na obrázku 5.1 lze vidět základní strukturu rozšiřujících modulů, která je pro každý modul stejná.



Obr. 5.1: Základní struktura rozšiřujících modulů

Složka gradle s podsložkou wrapper obsahuje nástroj Gradle, aby nemusel být stahován zvlášť a bylo možné sestavit každý modul po naklonování repozitáře. Ke spuštění přiloženého nástroje Gradle se využívá souborů gradlew pro operační systém Linux nebo gradlew.bat pro operační systém Windows. Soubor .gitignore obsahuje definici složek a souborů, které při operaci `git add` nebudou přidány do repozitáře. Tento postup slouží k zamezení přidávání nepotřebných souborů do repozitáře.

Významná změna, ke které došlo v každém rozšiřujícím modulu, je odstranění souboru `messages.properties`, který byl využíván k získávání textových řetězců pro vkládání záměnou za proměnné. Toto umožňovalo měnit text na jednom místě, který se používal na více místech. Soubor byl odstraněn z důvodu nemožnosti načíst více těchto souborů ve stejný čas. Nástroj Apache JMeter využívá pouze svůj soubor, do kterého by obsah souboru .properties z každého modulu musel být nahrán. Tento způsob by vedl ke snížení modulárnosti modulů, což by bylo nežádoucí a také by mohlo dojít k neočekávaným chybám textů v GUI.

Problém byl vyřešen pomocí vytvoření souboru `PropertiesStrings.java`. V souboru je vytvořena vždy abstraktní třída obsahující konstanty datového typu String. Tento přístup využívá podobnou logiku, jen s rozdílem metody načítání textů pro jednotlivé části GUI. Ve výpisu kódu 5.1 lze vidět odlišnost mezi uvedenými přístupy. Rozdíl je zvýrazněn pomocí standardního označení nástroje git.

Výpis 5.1: Rozdíl přístupů při načítání textu

```
- private final String TOTAL_ROW_LABEL =
    JMeterUtils.getResString("web_generator_row_total");
+ private final String TOTAL_ROW_LABEL =
    JMeterUtils.getLocaleString(WEB_GENERATOR_ROW_TOTAL);
```

5.2.1 Modul WebGenerator

Modul WebGenerator slouží ke sběru statistik a následné analýze v průběhu testování. Z těchto analyzovaných statistik poté vygeneruje webovou stránku s výsledky testů.

K modulu WebGenerator byla přidána podpora nástroje Gradle, převedení řetězců textů do souboru `PropertiesStrings.java` a s tím spojenou úpravu kódu na místech, kde se tyto řetězce textů využívají. Nakonec bylo provedeno základní formátování kódu, aby lépe splňovalo standardy jazyka Java. Na výpisu kódu 5.2 z části souboru `build.gradle` lze vidět, jaké závislosti jsou potřebné pro uvedený modul, kdy `${apacheJMeterVersion}` je proměnná pro verzi Apache JMeter konkrétně 5.5. Závislosti pro potřeby daného výpisu byly rozděleny na více řádků, v souboru je každá závislost na jednom řádků. Tyto informace jsou platné i pro další podkapitoly migrace modulů.

Výpis 5.2: Závislosti modulu WebGenerator

```
dependencies {
    implementation group: 'org.apache.jmeter',
    name: 'ApacheJMeter_core',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.jmeter',
    name: 'ApacheJMeter_java',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.jmeter',
    name: 'ApacheJMeter_components',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.jmeter',
    name: 'jorphan',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.jmeter',
    name: 'ApacheJMeter',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.commons',
    name: 'commons-lang3', version: '3.12.0'

    implementation group: 'commons-io',
    name: 'commons-io', version: '2.11.0'
}
```

5.2.2 Modul NetEmulator

Modul NetEmulator má za úkol emulovat přenosové vlastnosti, mezi které patří například zpoždění, propustnost, ztrátovost a další.

Podobně jako u modulu WebGenerator byly převedeny řetězce textů do souboru `PropertiesStrings.java` a úpraven kód v místě používání těchto řetězců. Dále byl modul převeden na podporu nástroje Gradle a nakonec bylo aplikováno formátování kódu. Na výpisu kódu 5.3 z části souboru `build.gradle` jsou vypsány závislosti pro tento modul.

Výpis 5.3: Závislosti modulu NetEmulator

```
dependencies {
    implementation group: 'org.apache.jmeter',
    name: 'ApacheJMeter_core',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.jmeter',
    name: 'ApacheJMeter_java',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.jmeter',
    name: 'ApacheJMeter_components',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.jmeter',
    name: 'jorphan',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.jmeter',
    name: 'ApacheJMeter',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.httpcomponents',
    name: 'httpclient', version: '4.5.13'

    implementation group: 'org.apache.commons',
    name: 'commons-lang3', version: '3.12.0'

    implementation group: 'commons-io',
    name: 'commons-io', version: '2.11.0'
}
```

5.2.3 Modul AdvancedThreadGroup

Modul AdvancedThreadGroup přidává dva typy Thread Group pro nástroj Apache JMeter. První z nich je schodovitý a umožňuje nastavovat postupné zesilování útoku schodovitě. Druhý typ je komplexní a poskytuje detailnější nastavení Thread Group.

Podobně jako v předcházejících dvou modulech byla přidána podpora nástroje Gradle a naformátován kód tak, aby co nejvíce splňoval standardy jazyka Java. Dále byly taktéž převedeny řetězce textů do souboru `PropertiesStrings.java`. Závislosti tohoto modulu lze vidět na výpisu kódu 5.4.

Výpis 5.4: Závislosti modulu AdvancedThreadGroup

```
dependencies {
    implementation group: 'org.apache.jmeter',
    name: 'ApacheJMeter_core',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.jmeter',
    name: 'ApacheJMeter_java',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.jmeter',
    name: 'ApacheJMeter_components',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.jmeter',
    name: 'jorphan',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.jmeter',
    name: 'ApacheJMeter',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.commons',
    name: 'commons-lang3', version: '3.12.0'

    implementation group: 'commons-io',
    name: 'commons-io', version: '2.11.0'
}
```

5.2.4 Modul ReportGenerator

Modul ReportGenerator slouží k vytváření závěrečné zprávy neboli reportu. Tento report se generuje na základě souboru vytvořeného pomocí modulu NetAnalyzer. Výsledný report je generován v podobě webové stránky. Obsahuje čas začátku a konce testu, délku průběhu testu, velikost příchozího a odchozího provozu na vybraných síťových rozhraních a informace o využití procesoru a operační paměti RAM. Velikost provozu je zobrazena jak v grafu, tak i v samostatných polích pro jednotlivé hodnoty. Využití procesoru je vyobrazeno v grafu a tabulce, kde jsou hodnoty uvedeny jak pro jednotlivá jádra tak i jeho celková hodnota.

Do modulu byla přidána podpora nástroje Gradle, řetězce textů byly převedeny do souboru `PropertiesStrings.java` a kód byl naformátován tak, aby co nejvíce splňoval standardy jazyka Java. Dále byla implementována extrakce souborů, které se před aktualizací manuálně kopírovaly. Toto vylepšení umožňuje mít pouze jediný soubor, který se musí přidávat do nástroje Apache JMeter a zbytek bude proveden automaticky. Extrahované soubory v případě modulu ReportGenerator slouží jako zdrojové soubory pro webovou stránku závěrečné zprávy. Závislosti modulu lze vidět na výpisu kódu 5.5.

Při vytváření této rozšiřující funkce byla špatně zadána cesta pro extrakci souborů. Na chybu se přišlo až o 3 měsíce později, kdy u všech modulů byla prováděna validace očekávaného chování. Tato chyba byla opravena okamžitě po jejím objevení.

Výpis 5.5: Závislosti modulu ReportGenerator

```
dependencies {
    implementation group: 'org.apache.jmeter',
    name: 'ApacheJMeter_core',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.jmeter',
    name: 'ApacheJMeter_java',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.jmeter',
    name: 'ApacheJMeter_components',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.commons',
    name: 'commons-lang3', version: '3.12.0'

    implementation group: 'commons-io',
    name: 'commons-io', version: '2.11.0'
}
```

5.2.5 Modul NetAnalyzer

Modul NetAnalyzer slouží ke sběru dat, které jsou vizualizovány a zároveň ukládány do souboru, který se dá použít jako vstup pro modul ReportGenerator. Za data se dá označit zátěž procesoru a jeho jednotlivých jader a využití operační paměti RAM. Dále provoz na jednotlivých rozhraních, kdy lze vybrat jednotku provozu s veličinou bit/s nebo paket/s. Po startu modulu je ukazována aktuální rychlost provozu

na každém rozhraní pomocí měřiče rychlosti provozu. Měřič rychlosti provozu má kruhovitý tvar takzvaný „rate dial“. Aktualizace rychlosti provozu na měřiči probíhá v předem daných časových rozestupech volených uživatelem. Také lze ukládat provoz do souboru typu `.pcap` a zvolit maximální velikost tohoto souboru. Uložení do souboru umožňuje dále zkoumat provoz například pomocí nástroje Wireshark.

U modulu Netanalyzer byly provedeny standardní úkony jako u každého jiného modulu – přidání podpory nástroje Gradle, naformátování kódu a vytvoření souboru `PropertiesStrings.java` pro definované řetězce znaků. Závislosti tohoto modulu lze vidět na výpisu kódu 5.5. Některé závislosti je nutné společně s uvedeným modulem přidat do nástroje Apache JMeter, konkrétně jsou to jfreechart, pcap4j-core, jna a slf4j-api. Modul NetAnalyzer také potřebuje systémový balíček libcap-def, tento balíček může mít rozdílný název na jiných Linux distribucích než je Ubuntu.

Podobně jako u modulu ReportGenerator byla přidána implementace pro extrakci souborů z `jar` souboru z důvodu, že nelze spouštět skripty nacházející se uvnitř `jar` souborů.

Výpis 5.6: Závislosti modulu NetAnalyzer

```
dependencies {
    implementation group: 'org.apache.jmeter',
    name: 'ApacheJMeter_core',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.jmeter',
    name: 'ApacheJMeter_java',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.jmeter',
    name: 'ApacheJMeter_components',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.jmeter',
    name: 'jorphan',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.jmeter',
    name: 'ApacheJMeter',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.jfree',
    name: 'jfreechart', version: '1.5.3'
```

```
    implementation group: 'org.pcap4j',
    name: 'pcap4j-core', version: '1.8.2'

    implementation group: 'org.apache.commons',
    name: 'commons-lang3', version: '3.12.0'

    implementation group: 'commons-io',
    name: 'commons-io', version: '2.11.0'

    implementation group: 'org.apache.pdfbox',
    name: 'pdfbox', version: '2.0.26'

    implementation group: 'org.bouncycastle',
    name: 'bcpkix-jdk15on', version: '1.70'
}
```

5.2.6 Modul ServerEmulator

Modul ServerEmulator slouží k automatizovanému vytváření FTP a Apache HTTP serveru, které lze využít k testování. V případě FTP serveru lze nastavit IP adresu verze 4, na které bude naslouchat a bude připravený odpovídat na příchozí spojení. V případě Apache HTTP serveru je možné připravit server s určitou velikostí stránky a zvolit, jestli server bude využívat nezabezpečeného HTTP nebo zabezpečeného HTTP. Dále je možnost vybrat, na jakém portu bude server naslouchat a jakou IP adresu verze 4 bude využívat.

Modulu ServerEmulator byla přidána podpora pro extrakci potřebných souborů, podpora pro nástroj Gradle, vytvoření souboru `PropertiesStrings.java` pro definované řetězce znaků a byl naformátován kód. Závislosti modulu lze vidět na výpisu kódu 5.7. Modul taktéž vyžaduje systémové balíčky vsftpd pro FTP server a apache2 pro Apache HTTP server. Bez těchto systémových balíčků nelze jednotlivé servery spustit.

Z důvodu migrace na distribuci Ubuntu musel být upraven skript pro Apache HTTP server, jelikož na distribuci CentOS byl tento balíček pod názvem httpd. Byly upraveny cesty, kde se jednotlivé používané soubory nacházejí a aktualizován název služby, umožnující službu zapnout nebo vypnout. Také byly odebrány části kódu pro podporu rozšíření jádra SELinux (Security-Enhanced Linux), který je standardně pouze v distribucích založených na RHEL. Do ostatních Linux distribucí toto rozšíření jádra lze přidat, ale standardně se tento postup nepraktikuje. Nakonec byla do skriptu přidána proměnná, která obsahuje název služby pro jednoduší aktualizaci při změně názvu v případě přechodu na jiný typ webového serveru.

Výpis 5.7: Závislosti modulu ServerEmulator

```
dependencies {
    implementation group: 'org.apache.jmeter',
    name: 'ApacheJMeter_core',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.jmeter',
    name: 'ApacheJMeter_java',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.jmeter',
    name: 'ApacheJMeter_components',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.jmeter',
    name: 'jorphan',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.jmeter',
    name: 'ApacheJMeter',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.httpcomponents',
    name: 'httpclient', version: '4.5.13'

    implementation group: 'org.apache.commons',
    name: 'commons-lang3', version: '3.12.0'

    implementation group: 'commons-io',
    name: 'commons-io', version: '2.11.0'
}
```

5.2.7 Modul DDoS

Modul DDoS je nejrozsáhlejším a nejvíce komplikovaným modulem ze všech výše zmíněných modulů. Využívá se pro vytváření několika typů DDoS útoků. Provoz pro tyto útoky je generován generátorem provozu Trafgen. Ve všech typech DDoS útoků lze nastavovat parametry jako je cílová IP adresa ve verzi 4, síťové rozhraní ze kterého bude útok prováděn a podvržená IP adresa nebo její rozsah. Z rozsahu bude proveden výběr IP adresy. Další parametry jsou rozdílné v závislosti na typu útoku, jelikož v některých typech jsou nastaveny přímo v šabloně pro konfigurační soubor

generátoru provozu Trafgen. Těmito parametry mohou být zdrojová a cílová MAC (Media Access Control) adresa, velikost posílaných dat v paketu, cílový a zdrojový port a v případě ICMP flood typ ICMP zprávy.

V tomto modulu proběhlo při migraci více změn než u ostatních modulů. Uskutečnily se všechny standardní části migrace, jako je přidání podpory nástroje Gradle, naformátování kódu a vytvoření souboru `PropertiesStrings.java`. Dále byla implementována extrakce souborů sloužící pro extrahování konfiguračních šablon a skriptů pro spouštění a nastavení generátoru provozu Trafgen a skript v jazyce Python pro pomalý DoS útok SlowLoris. Závislosti modulu lze vidět na části výpisu kódu 5.8. Důležitým systémovým balíčkem, bez kterého nebude fungovat většina DDoS útoků, je `netsniff-ng`. Tento systémový balíček obsahuje generátor provozu Trafgen, který je fundamentálním pro většinu DDoS útoků.

Hlavní změna byla provedena u tříd vytvářejících Thread Group, které zobrazují pouze DoS útoky. Tato změna by se dala nazvat regresí, která vznikla při přechodu nástroje Apache JMeter na podporu jazyka Java ve verzích 9 a vyšších. Bylo zjištěno, že třída `TreeNode` nepředávala přímo objekty třídy `TestElement`, ale třídy `JMeterTreeNode`. Z tohoto důvodu musela být přidána proměnná, do které bude uložen objekt třídy `JMeterTreeNode` a ze kterého je načten objekt třídy `TestElement`.Tato regrese znemožňovala kompilaci modulu DDoS.

Výpis 5.8: Závislosti modulu DDoS

```
dependencies {
    implementation group: 'org.apache.jmeter',
    name: 'ApacheJMeter_core',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.jmeter',
    name: 'ApacheJMeter_java',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.jmeter',
    name: 'ApacheJMeter_components',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.jmeter',
    name: 'jorphan',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.jmeter',
    name: 'ApacheJMeter',
    version: "${apacheJMeterVersion}", withoutBom
```

```
    implementation group: 'org.apache.jmeter',
    name: 'ApacheJMeter_http',
    version: "${apacheJMeterVersion}", withoutBom

    implementation group: 'org.apache.httpcomponents',
    name: 'httpclient', version: '4.5.13'

    implementation group: 'org.apache.commons',
    name: 'commons-lang3', version: '3.12.0'

    implementation group: 'commons-io',
    name: 'commons-io', version: '2.11.0'

    implementation group: 'com.github.seancfoley',
    name: 'ipaddress', version: '5.3.4'
}
```

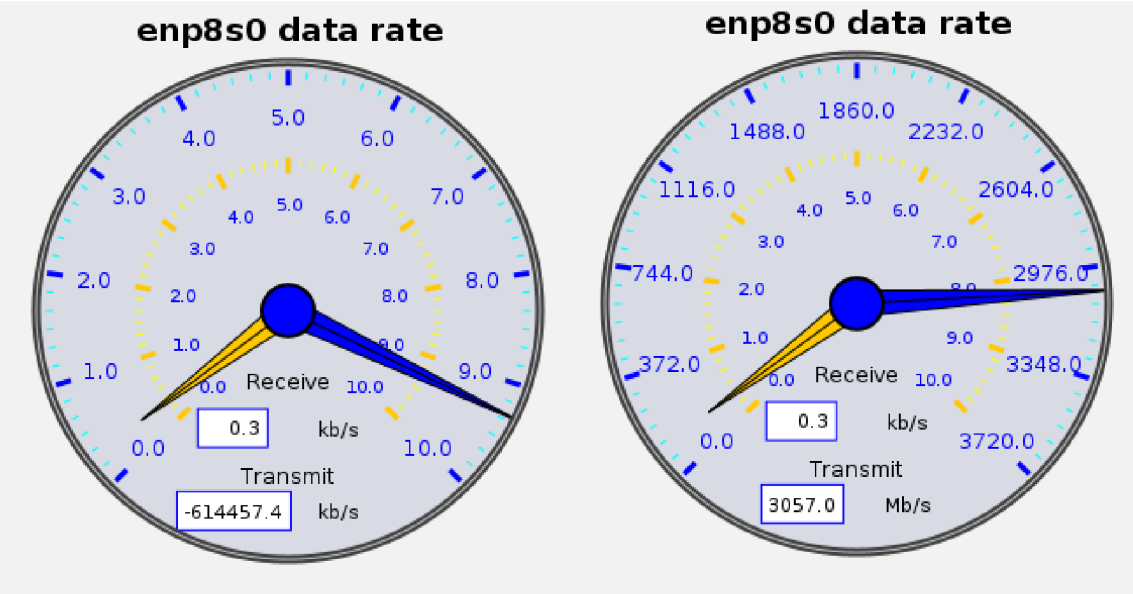
6 Oprava chyb

Kapitola popisuje nalezené a opravené chyby, které se nacházely také v původní verzi modulů a nebyly opraveny nebo vznikly při aktualizaci modulů, takže se jednalo o regresi. Každá podkapitola v této kapitole se zabývá chybami nalezenými ve specifickém modulu.

6.1 Modul NetAnalyzer

6.1.1 Oprava měřiče rychlosti provozu

Na obrázku 6.1 lze vidět porovnání před a po opravení chyby. Chyba se projevovala tak, že při zaznamenávání rychlosti provozu se na měřiči rychlosti provozu ukazovala záporná nebo nereálná hodnota v textových polích *Receive* a *Transmit*. Projevení této chyby lze vidět na levém měřiči rychlosti provozu. Výsledek po opravení chyby je zobrazen na pravém měřiči rychlosti provozu.



Obr. 6.1: Porovnání chyby a opravy

Chyba byla způsobena špatným návrhem při vytváření modulu. Z kódu modulu lze vyčíst, že bylo nejspíše předpokládáno nepřekročení rychlosti 1 Gbit/s. Tomuto předpokladu napovídají dvě věci. První je, že jednotkami na které se zaokrouhluje, jsou pouze kbit/s a Mbit/s, ale na jednotku Gbit/s se nezaokrouhluje, což může být také z důvodu zobrazení přesnějších výsledků. Druhou je zvolení datových typů `int` a `float` namísto datových typů `long` a `doule`, které mají velikost 64 bitů. Datové typy `int` a `float` mají velikost pouze 32 bitů.

Tato nedostatečná velikost způsobovala takzvaný „overflow“ neboli přetečení, což je chyba, kdy výsledek libovolné aritmetické operace je větší než hranice datového typu a tím dojde k neočekávanému chování. Jelikož maximální hodnota 32 bitového datového typu je dekadicky 2 147 483 647, v případě většího přenosu než přibližně 2 Gbit/s tak docházelo k přetečení.

Oprava této chyby vyžadovala záměnu datového typu `int` za datový typ `long` a záměnu datového typu `float` za datový typ `double`. Změna datových typů opravila nalezenou chybu a rychlost přenosu se zobrazuje bez problémů. Novou limitací je, že maximální rychlost přenosu nesmí přesáhnout hodnotu 9 223 372 036 854 775 807, což odpovídá přibližně 9 Ebit/s.

6.1.2 Oprava generování pcap souborů

Další chyba se projevovala při zaznamenávání provozu do `pcap` souboru. Při vytváření souboru docházelo k neočekávané chybě, která způsobovala nepředvídatelné chování aplikace. V průběhu opravy bylo zjištěno, že původní knihovna je zastaralá, nemá podporu a tak byla nahrazena novější knihovnou `pcap4j-core`.

Původní a nová knihovna při nastavení zaznamenávání provozu má pouze minimální odlišnosti. Kód byl upraven tak, aby podporoval tuto novou knihovnu. Dále byly jednotlivé fáze zachytávání ošetřeny `try catch` bloky, které umožní při chybě lépe zapsat chybu do souboru s logy a uvědomit uživatele, aniž by se vyskytlo nepředvídatelné chování aplikace.

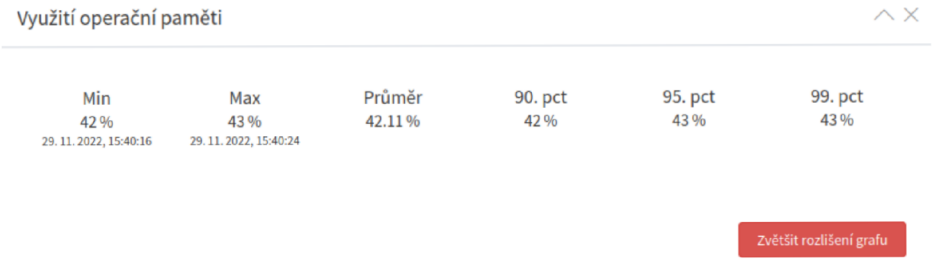
6.2 Modul ReportGenerator

U modulu ReportGenerator byla nalezena pouze jedna chyba. Chyba nastávala v generované webové stránce s výsledky modulu NetAnalyzer. Následek této chyby byl, že nebyly vykreslovány grafy a logo na webové stránce. Na obrázku 6.2 lze vidět webovou stránku bez vykreslených grafů.

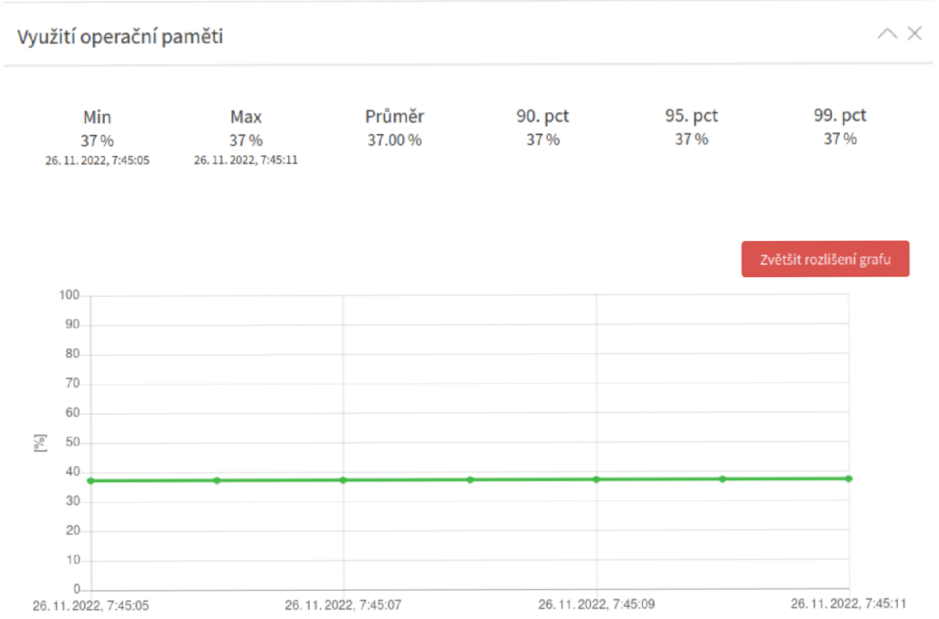
Chyba ve vykreslování loga byla způsobena nesprávným názvem souboru. Název souboru se lišil pouze tím, že správný soubor měl část názvu velkými písmeny. Logo se nevykreslovalo z důvodu, že operační systém Linux je citlivý na velikost znaků u názvů souborů a složek. Tento problém by neměl nastat na operačním systému Windows při zobrazení webové stránky s výsledky. Operační systém Windows není v základním nastavení závislý na velikosti znaků.

Chyba při vykreslování grafů byla způsobena chybějícím souborem obsahující kód v skriptovacím jazyce Javascript. Soubor byl součástí složky, která obsahovala zdrojové soubory pro webovou stránku, ale nebyla přidána do hlavičky hlavního

HTML souboru. Po přidání souboru `chart.js` do hlavičky byl tento problém vyřešen a výsledný graf lze vidět na obrázku 6.3.



Obr. 6.2: Webová stránka bez vykresleného grafu



Obr. 6.3: Webová stránka s vykresleným grafem

6.3 Modul ServerEmulator

6.3.1 Chybná syntaxe bash skriptu

Všechny chyby v modulu ServerEmulator byly spojené s vytvářením a spouštěním webového serveru Apache. První problém byl v samotných skriptech pro spuštění, kde funkce byly napsány jako `function nazev_funkce() { příkazy }`. Tato syntaxe není validní, jelikož spojuje dvě možnosti definice funkce v bash skriptech. Validní syntax pro definici může vypadat jako `function nazev_funkce { příkaz }` nebo `nazev_funkce() { příkazy }`. V případě opravy bylo využito druhé možnosti. Tento problém byl řešen společně s aktualizací názvů služeb ve skriptech pro modul ServerEmulator.

6.3.2 Chyba naslouchání na výchozím portu

Dalším problémem při spouštění webového serveru Apache bylo, že předdefinovaný soubor `ports.conf` obsahoval příkaz pro naslouchání na portu 80. To způsobovalo, že při nastavení portu 80 v GUI nástroje Apache JMeter nastala chyba při spuštění. Po nainstalování webového serveru Apache je ve výchozím nastavení v souboru `ports.conf` definován port 80 a proto byla přidána funkcionalita na přepis tohoto souboru tak, aby bylo možné naslouchat na jakémkoliv portu.

6.3.3 Chyba povolení SSL

Poslední chybou byla nemožnost vytvořit webový server Apache, který provozuje zabezpečené HTTP spojení. Chyba nejspíše vznikla v důsledku změny operačního systému. Pro opravení chyby musel být upraven bash skript pro spouštění webového serveru Apache, jelikož se před spuštěním musí zapnout SSL (Secure Sockets Layer) pomocí `a2enmod`. Ukázku zapnutí a vypnutí SSL lze vidět na výpisu kódu 6.1. Výpis dále obsahuje povolení konfiguračního souboru pro server, na kterém má být zapnuto SSL a zakázání konfiguračního souboru pro server bez zapnutého SSL.

Výpis 6.1: Zapnutí a vypnutí podpory SSL

```
# Zapnutí
sudo a2enmod ssl
sudo a2ensite default-ssl.conf
sudo a2dissite 000-default.conf
# Vypnutí
sudo a2dismod ssl
sudo a2ensite 000-default.conf
sudo a2dissite default-ssl.conf
```

Dále musel být přidán soubor `default-ssl.conf`, který obsahuje nastavení pro SSL na webovém serveru Apache. Zde je definována výchozí cesta k souborům nebo cesta k SSL certifikátu a klíči.

Nakonec musela být v souboru `ServerEmulatorCore.java` upravena logika spouštění webového serveru Apache. Toto obsahovalo přidání části kódu pro nahrazení portu v souboru `default-ssl.conf`, kdy se nahraje celý soubor do proměnné a pomocí funkce `replaceAll()` a RegEx (Regular Expression) se nastaví port, na kterém má webový server Apache naslouchat. V Java souboru pak bylo upraveno spouštění skriptu, do kterého byla přidána proměnná, obsahující jestli se má spouštět SSL.

6.4 Modul DDoS

6.4.1 Problém s DDoS útokem SlowLoris

Problémy při spouštění DoS útoku SlowLoris byly celkem dva. Prvním byla nemožnost spustit celý skript napsaný v jazyce Python. Toto bylo způsobeno tím, že v Linux systémech jsou často současně dvě verze jazyka Python, konkrétně verze 2 a verze 3. Důsledkem toho je, že Python ve verzi 3 se spouští příkazem `python3` a proto musel být příkaz na spouštění DoS útoku SlowLoris upraven tak, aby spouštěl Python ve verzi 3.

Druhý problém byl závažnější a způsoboval výstup několika tisíců chybových hlášení. Toto bylo zapříčiněno pokusy o spuštění už aktivního vlákna, které nebylo možné spustit. Problém byl vyřešen pomocí podmínky, která testuje, jestli je vlákno aktivní.

6.4.2 Chyba dělení nulou

U textových polí, kde je nastavená verifikace vstupu textového pole pomocí typu `naturalOrNothing` se nacházela chyba, že vstup byl prvně validován zda není prázdný a poté jestli se jedná o číslo větší než nula. Tato nekorektní validace způsobovala, že v pozadí byla do proměnné přiřazena hodnota 0, kterou se v průběhu provádění testu dělilo. Dělení nulou způsobovalo zamrznutí a pád celého nástroje Apache JMeter.

Z tohoto důvodu musela být logika kontroly vstupu upravena tak, aby bylo v jedné podmínce kontrolováno, jestli se ve vstupu jedná o číslo větší jak nula a zároveň jestli není prázdný.

6.4.3 Chyba při nastavení času ThreadGroup

Při zadání času potřebného k činnosti ThreadGroup nastávala chyba, která znemožňovala správné ukončení testu. Tuto chybu způsobovala špatná kontrola návratového kódu, protože návratový kód pro ukončené procesy pomocí signálů se vypočítává jako $128 + N$, kde N je hodnota signálu. Návratový kód se kontroloval pouze na správné vykonání a v tomto případě je návratový kód roven hodnotě 0. Při ukončení ThreadGroup po určitém čase se využívalo ukončení procesu pomocí signálu `SIGKILL`, který má hodnotu 9. V případě ukončení procesu pomocí signálu `SIGKILL` bude návratová hodnota 137. Proto se nemůže kontrolovat pouze návratová hodnota 0, ale také návratová hodnota 137. Pokud je návratová hodnota jiná než 0 nebo 137, je zaznamenána chybová hláška do logovacího souboru, ale k tomuto stavu by nemělo docházet.

6.4.4 Nefunkční DDoS StairGroup

StairGroup je rozšíření ThreadGroup, které umožňuje nastavování dynamických parametrů. V StairGroup se nacházela chyba, která způsobovala, že se zatížení neměnilo dle zadaných parametrů a po ukončení testu na pozadí stále fungovalo generování zátěže pomocí nástroje Trafgen. Chyba byla způsobena špatně nastavovaným atributem `dynamicRate`, který u StairGroup musí být nastaven jako `true`. Jelikož bylo nastaveno `false`, tak komponenty, které byly v StairGroup, nereagovaly na změny a vykonávaly pouze počáteční zátěž.

6.4.5 Chyba při spouštění vlastního konfiguračního souboru

DDoS - Trafgen Custom Config slouží ke spuštění vlastního konfiguračního souboru pro nástroj Trafgen. Při spuštění této konfigurace bylo testování zahájeno, ale výstupem byly pouze chybové hlášky, protože se konfigurační soubor nepovedlo načíst. Důvodem bylo vytváření konfiguračního souboru přímo v *TrafgenCustomSampler* a nevyužití třídy `TrafgenControl` pro spouštění ovládání nástroje Trafgen. Oprava této chyby byla provedena tak, že přímo v *TrafgenCustomSampler* je vytvořen samostatný konfigurační soubor. Tento soubor se poté předává třídě `TrafgenControl` pomocí „setteru“. Ve třídě byla upravena logika nastavování konfiguračního souboru, aby bylo možné rozlišit, jestli se jedná o vlastní konfigurační soubor a načíst ho místo předdefinovaných konfiguračních souborů. Do třídy proto bylo nutné přidat proměnnou `customConfig` datového typu boolean, která určuje, jestli má být využit abstraktní nebo vlastní konfigurační soubor.

6.5 Domovský adresář nástroje JMeter

Chyba s domovským adresářem byla nalezena v každém z modulů a znemožňovala spustit nástroj JMeter s jakoukoliv cestou ke spouštěcímu skriptu `jmeter`. Nástroj JMeter bylo možné spustit pouze ze složky `bin`, která se nachází ve složce `jmeter`. Tato chyba by byla problematická hlavně pro neznalé uživatele. Nutno říci, že při použití vytvořených automatických skriptů, popsaných v následující kapitole, chyba nemohla nastat.

Chyba byla opravena nahrazením původního výběru domovského adresáře nástroje JMeter za statickou metodu `getJMeterHome` ze třídy `JMeterUtils`, která je součástí zdrojového kódu nástroje JMeter.

7 Automatizace pro nasazení a aktualizaci nástroje JMeter

Kapitola rozebírá automatizaci nasazení a aktualizaci testeru a nástroje JMeter pomocí skriptů sloužících k automatizaci.

7.1 Příprava testeru

Jak bylo zmíněno v kapitole 4.2 Migrace na nový operační systém, na testeru je instalován operační systém Ubuntu 20.04 LTS. Skript, který je napsán v jazyce Bash pro přípravu testeru, by měl být kompatibilní se všemi operačními systémy založenými na distribuci Debian, které mají nainstalován nástroj `apt-get`. Testování proběhlo pouze na podporované verzi operačního systému Ubuntu, takže na jiných verzích nebo distribucích nemusí být balíčky dostupné nebo jsou dostupné pod jiným názvem.

Výpis 7.1: Skript pro instalaci balíčků

```
#!/bin/bash

JAVA_PACKAGES="openjdk-17-jdk openjdk-17-jre openjdk-17-doc"

# Apache2, vsftpd and links are for server emulator module
# netsniff-ng incude trafgen for ddos module
# libcap-def used for net analyzer module
JMETER_MODUL_PACKAGES="apache2 vsftpd netsniff-ng
libpcap-dev links"

# Usefull packages for developers
DEV_PACKAGES="nload"

# Packages used by build and start script
SCRIPT_PACKAGES="tar zip git gradle"

PACKAGES_TO_INSTALL="${JAVA_PACKAGES} ${JMETER_MODUL_PACKAGES}
${DEV_PACKAGES} ${SCRIPT_PACKAGES}"

for i~in $PACKAGES_TO_INSTALL; do
  sudo apt-get install -y $i
done
```

Jednotlivé balíčky a postup jejich instalace je zobrazen na výpisu kódu 7.1. Každý balíček je instalován samostatně z důvodu, že při chybě se nenainstaluje pouze specifický balíček. Kdyby se instalovaly všechny balíčky zároveň, dojde k tomu, že v případě chyby se nenainstaluje žádný balíček. Jednotlivé balíčky jsou rozděleny do částí, kdy jedna část slouží k přidání balíčků pro vývojáře a ostatní jsou povinné pro správné fungování nástroje Jmeter. V balíčcích pro vývojáře chybí balíček pro nástroj Wireshark, sloužící pro zachytávání komunikace, a to z důvodu jeho velikosti, která je mnohonásobně větší než u ostatních balíčků. Při vývoji na virtuálních strojích by mohlo nastat vyčerpání paměti pro ukládání dat. Tento balíček si každý vývojář může doinstalovat samostatně.

Aktualizace balíčků probíhá pomocí spuštění stejného skriptu nebo ji lze provést pomocí `sudo apt-get update && sudo apt-get upgrade`, který neaktualizuje pouze vybrané balíčky, ale všechny nainstalované balíčky.

Po každém spuštění skriptu je na konci vytvořen archív ve formátu `zip` obsahující nástroj JMeter se všemi moduly a zdrojovými soubory pro JDK. Tento archív slouží k možnému přenosu na jiná zařízení, která nemusí mít například přístup k internetu. Archív lze samostatně rozbalit nebo použít jako vstupní soubor pro skript sloužící ke spuštění nástroje JMeter.

7.2 Příprava nástroje JMeter

Skript pro přípravu a aktualizaci nástroje JMeter obsahuje 5 volitelných argumentů. Argumenty se dají zadat jak v krátkém tak i v dlouhém formátu. V tabulce 7.1 lze vidět jednotlivé argumenty a jejich účel.

Tab. 7.1: Popis argumentů skriptu build and deploy

`-h/-help`	Výpis nápovědy.
`-p/-build_path`	Zadání cesty k nástroji JMeter a modulům nebo k zadání cesty, kde se mají stáhnout a nainstalovat.
`-j/-java_path`	Zadání cesty k JDK, v případě využití specifického JDK.
`-d/-download_jdk`	Argument bez parametru. Stážení openJDK ve verzi definované skriptem.
`-b/-build_jmeter`	Argument bez parametru. Sestavení nástroje JMeter.

Skript pro instalaci a aktualizaci využívá stejný základ, ve kterém jsou definovány všechny rozšiřující moduly, které mají být součástí nástroje JMeter. Dále jsou v něm definovány potřebné knihovny pro jednotlivé moduly, které jsou stahovány z centrálního repozitáře maven, jako soubory ve formátu `jar`.

Jelikož jsou nástroj JMeter a přídavné moduly uložené v privátních repozitářích, nemá k nim kdokoliv přístup. Z důvodu bezpečnosti a pro možnost kontroly přístupu byl vytvořen klíč pro nasazení (deploy key) ve službě GitLab. Tento klíč umožní každému získat zdrojové kódy a má nastavené, že může pouze číst z repozitářů, ale nemá povolen zápis. Uvedený přístup má z pohledu údržby testeru výhodu, že nemusí být vytvořen nový uživatel ve službě GitLab a přidán mu SSH (Secure Shell) klíč.

Průběh přípravy je zobrazen v podobě vývojového diagramu na obrázku 7.1

7.2.1 Prvotní instalace nástroje JMeter

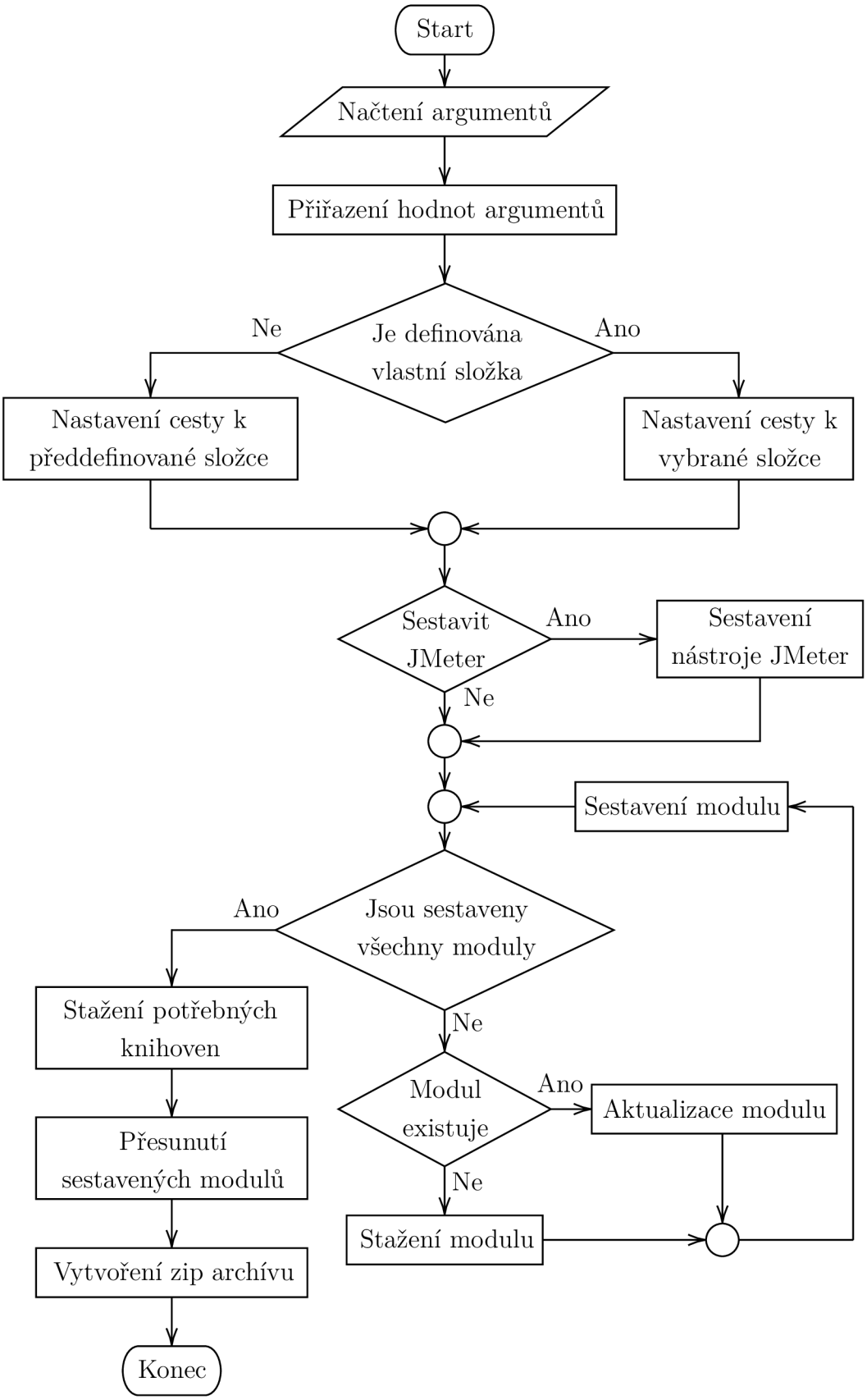
Pro prvotní instalaci je nutné provést více kroků než které jsou prováděny v případě aktualizace. Přidaným krokem oproti aktualizaci je nutnost sestavit celý nástroj JMeter. Aby tento krok nemusel být opakován při každém spuštění, byl přidán skriptu argument `-b`, kterým se spustí sestavení nástroje JMeter.

7.2.2 Aktualizace nástroje JMeter

Pokud se vyskytne změna ve vzdáleném repozitáři, musí uživatel manuálně spustit skript. Skript automaticky rozlišuje, jestli už je daný přídavný modul nainstalován a je nutné provést aktualizace. Průběh této automatické kontroly lze vidět na výpisu kódu 7.2. Lze aktualizovat všechny přídavné moduly naráz nebo nástroj JMeter a všechny přídavné moduly. Aktualizace pouze určitých přídavných modulů není implementována a to z důvodu, že by vyžadovala velké množství uživatelské interakce. Výsledkem by bylo ušetření pouze několika sekund. Stejně jako v případě instalace se pro aktualizaci nástroje JMeter využije argument `-b`.

Výpis 7.2: Instalace a aktualizace přídavných modulů

```
#Function have one argument and it's module name
clone_repository() {
    eval cd $BUILD_PATH
    if [ ! -d $1 ]; then
        git clone "https://${DEPLOY_USER}:${TOKEN}@gitlab.com/
        ${GITLAB_MODULE_PATH}/$1.git"
    else
        cd $1
        git fetch
        git rebase origin/main
    fi
}
```



Obr. 7.1: Vývojový diagram pro build and deploy skript

7.3 Spuštění nástroje JMeter

Skript pro automatické spuštění nástroje JMeter slouží k rozbalení a následnému spuštění nástroje JMeter nebo pouze ke spuštění ze specifikovaného adresáře, kde se nástroj JMeter nachází. Obsahuje 4 argumenty, kdy 3 jsou volitelné a jeden je povinný. Všechny parametry jsou v tabulce 7.2. Jak je z tabulky patrné, povinným argumentem je cesta, která slouží k lokaci nástroje JMeter nebo ji lze použít i jako cestu pro rozbalení archívu obsahujícího nástroj JMeter. Skript spouští nástroj JMeter jako superuživatel pomocí `sudo` s argumentem `-E`, který umožní zachovat proměnné. Příkaz na spuštění nástroje JMeter obsažený ve vytvořeném skriptu vypadá následovně `sudo -E ./jmeter`.

Tab. 7.2: Popis argumentů skriptu start jmeter

`-h/-help`	Výpis nápovědy.
`-p/-path`	Zadání cesty k nástroji JMeter. Nelze použít v kombinaci s argumentem `-z`.
`-j/-java_path`	Zadání cesty k JDK, v případě využití specifického JDK.
`-z/-zip_path`	Zadání cesty k zip archívu v kterém je obsažen nástroj JMeter. Nelze kombinovat s argumentem `-p`.

8 Rozšíření jednotlivých modulů

Kapitola popisuje vyvinuté rozšíření pro vybrané moduly. Každé rozšíření je jednotlivě rozebráno jak po stránce vývojové, tak po stránce uživatelské.

8.1 Modul ServerEmulator

8.1.1 Změna velikosti generované stránky

V modulu ServerEmulator při spouštění HTTP/S serveru bylo potřeba upravit velikost stránky tak, aby uživateli byla umožněna větší kontrola při volbě testovacích parametrů. Před touto úpravou bylo možné pouze zvolit velikost stránky v celých číslech[1], kdy nejmenší možná velikost stránky byla 1 kiB.

Umožnění generace stránky podporující velikost menší než 1 kiB bylo dosaženo záměnou datového typu `Integer` za datový typ `Double`. Záměnou byla dosažena desetinná přesnost umožnující zvolit velikost generované stránky menší než 1 kiB. Změny musely být provedeny jak v logice kódu, tak i v nastavení objektu třídy `SpinnerModel` z balíčku `javax.swing`[2].

V aktuální verzi při psaní této diplomové práce je nejmenší velikost stránky kterou lze vygenerovat přibližně 0,3 kiB (300B). Menší velikost nelze zvolit, jelikož stránka obsahuje text upozorňující na to, že stránka byla vygenerována automaticky. V případě, že uživatel vybere stránku o větší velikosti jak 0,3 kiB, je na stránku přidána výplň v podobě řetězce skládajícího se ze znaku `*`. Vytvoření této výplně probíhá ve dvou fázích, které lze vidět na výpisu kódu 8.1.

Výpis 8.1: Doplnění řetězce na zvolenou velikost

```
for (int i~= 1; i~<= (htmlSize / 100) - 3; i++) {
    htmlWriter.write("**********************************
    **************************************************
    **************");
}
double htmlSizeRounded =
    Math.round(htmlSize * Math.pow(10, 3)) / Math.pow(10, 3);
for (int i~= 0;
    i~< (((htmlSizeRounded / 100) -
    (int) (htmlSizeRounded / 100)) * 100); i~+= 3) {
        htmlWriter.write("**");
}
```

[1] V rámci zaokrouhlení jednotek na kiB.

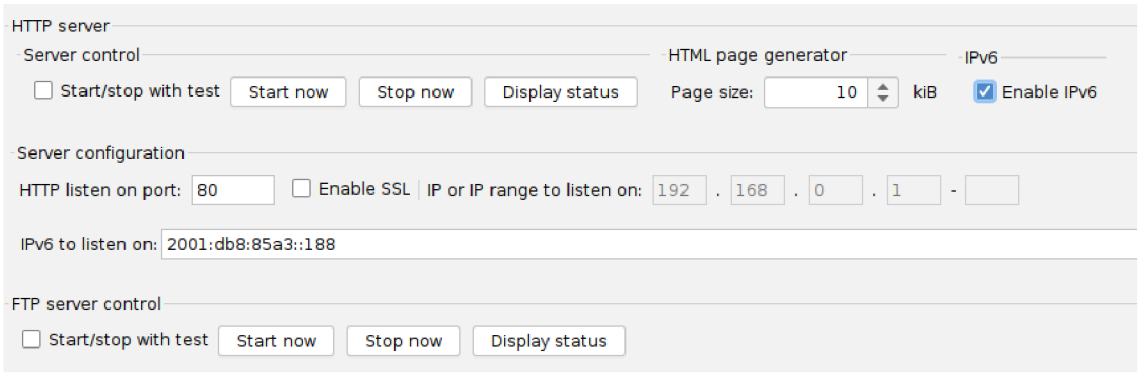
[2] Obsahuje komponenty pro vytváření uživatelského rozhraní.

8.1.2 Rozšíření o podporu IPv6 pro HTTP server

Do modulu ServerEmulator byla implementována podpora IPv6 pro HTTP/S server. Implementace podpory IPv6 umožňuje rozšíření testovacích scénářů pro lokální testování. Přidání tohoto vylepšení souviselo s vytvořením komponenty *Sampler*, sloužící pro generaci HTTP/S požadavků na webový server. Tato komponenta bude popsána v jedné z navazujících podkapitol.

Z programátorského hlediska se pouze upravila logika tak, aby bylo možné přidávat i IPv6 adresu a ne pouze IPv4 adresu. S úpravou logiky bylo spojeno také přidání rozlišení, zda uživatel zapíná webový server pro IPv4 nebo IPv6. Poslední částí, která byla implementována, je uživatelské rozhraní obsahující zaškrtávací políčko pro povolení nebo zakázaní IPv6 a dále textové pole pro zadání IPv6 adresy.

Ovládání z pohledu uživatele se oproti předchozí verzi, která neobsahovala podporu IPv6 nezměnilo. Na obrázku 8.1 v pravém horním rohu lze vidět zaškrtávací tlačítko, které umožňuje, jaký typ IP adresy se má použít. V případě, je-li políčko „zaškrtnuté“, bude webový server využívat zvolenou IPv6 adresu. Tuto adresu lze určit v textovém poli s označením „IPv6 to listen on“. Poté může uživatel spustit webový server stejným způsobem jako tomu bylo v předešlé verzi.



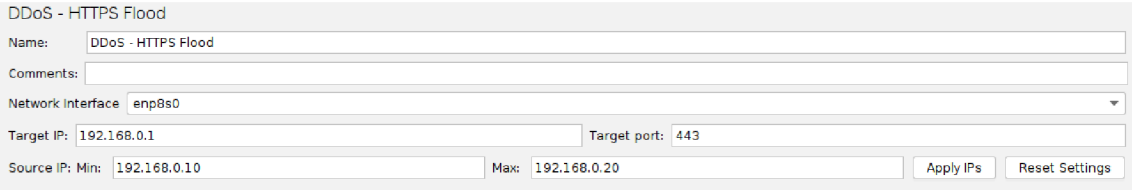
Obr. 8.1: Modul ServerEmulator s přidanou podporou IPv6

8.2 Modul DDoS

8.2.1 Přidání portu do HttpsFlood

V *DDoS - HTTPS Flood* chyběla možnost zvolení portu pro testování HTTPS. Zde byl nastaven výchozí port na 443, pro většinu testovacích scénářů je toto nastavení dostatečné. Neposkytuje ale možnost testování v případě, že by webový server využíval jiný port než 443. Z tohoto důvodu byla přidána možnost zvolit si cílový port. Změna probíhala v přidání proměnné, která přebírá zvolený vstup od uživatele

a v přidání testového pole v uživatelském rozhraní. Přidané textové pole v uživatelském rozhraní lze vidět na obrázku 8.2, kde je výchozí hodnotou 443.



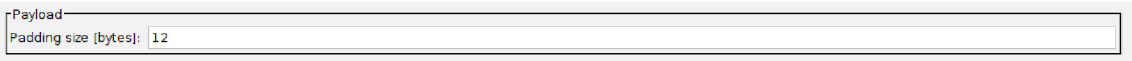
Obr. 8.2: Upravené uživatelské rozhraní pro *DDoS - HTTPS Flood*

8.2.2 Přidání „payload“ pro UDP a ICMP flood

Datový obsah neboli „payload“ bylo možné zvolit pouze v testu pro SYN flood. U zbylých testů, které měly možnost nastavení datového obsahu, byl nastaven jenom staticky v konfiguračním souboru nástroje Trafgen. Z tohoto důvodu pokud uživatel chtěl zvětšit nebo zmenšit „payload“, musel ho přepsat v šabloně pro konfigurační soubor. Tato šablona není perzistentní, takže se při každém spuštění nástroje JMeter přepíše do výchozího stavu.

Proto bylo statické nastavení konfigurační souboru upraveno tak, aby bylo možné tuto hodnotu zadávat dle vstupu uživatele. Tato změna se dotkla testů pro UDP a ICMP flood. V obou případech se přidávalo pouze textové pole pro zadání vlastní velikosti hodnoty datového obsahu. Z programátorského hlediska se jednalo o přidání textového pole pro uživatele a proměnné pro udržení hodnoty datového obsahu a s tím potřebné logiky. Dále změnou `fill('B',22)` v konfiguračním souboru za `fill('B',##IcmpFloodSampler.payload)` v případě ICMP flood. Pro UDP flood změna byla `fill('B',##UdpFloodSampler.payload)`.

Přidaný vstup v uživatelském rozhraní lze vidět na obrázku 8.3, kdy je pro SYN, UDP a ICMP flood stejné.



Obr. 8.3: Upravené uživatelské rozhraní pro *DDoS - HTTPS Flood*

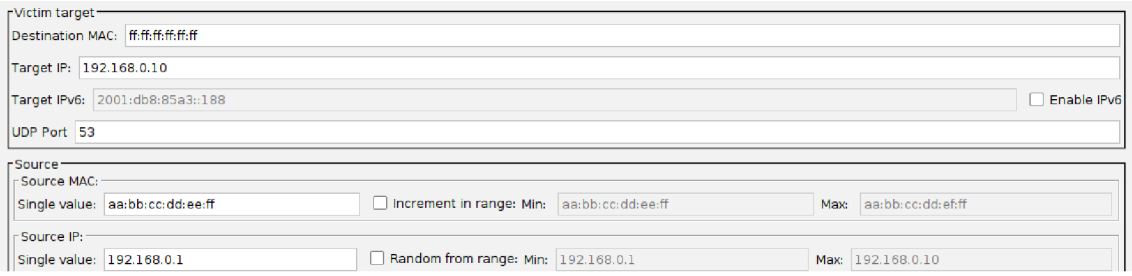
8.2.3 Rozšíření o volbu zdrojové a cílové MAC adresy

V testech pro UDP flood, DNS flood a DNS amplification chyběla možnost zadání zdrojové a cílové MAC adresy. Tato možnost nechyběla v žádném zbývajícím testu

pro DDoS (SYN, NTP a ICMP flood) založeném na nástroji Trafgen. Z programátorského hlediska byly upraveny abstraktní konfigurace[3], ve kterých byla staticky zadána zdrojová a cílová MAC adresa. V případě UDP flood byly statické hodnoty nahrazeny proměnnými `##UdpFloodSampler.dmac` a `##UdpFloodSampler.smacTG`. Pro DNS flood a DNS apmlification vypadá záměna podobně, pouze část proměnné `UdpFloodSampler` má podobu `DnsFloodSampler` pro DNS flood a `DnsAmpSampler` pro DNS amplification. Dále bylo doplněno uživatelské rozhraní pro zdrojovou a cílovou MAC adresu a s tím spojená logika pro uchování a předání zadaných hodnot.

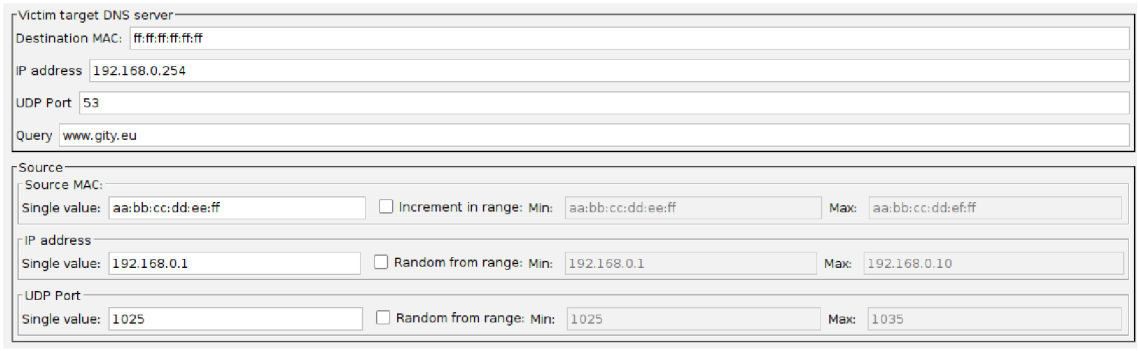
Uživatelské rozhraní se liší od již implementovaných testů, které mají volbu zdrojové a cílové MAC adresy společně viz obrázek 8.4. Z obrázku je viditelné, že rozložení prvků uživatelského rozhraní je po vrstvách ISO/OSI. V případě UDP flood, DNS flood a DNS amplification je zdrojová a cílová MAC adresa umístěná tak, aby byla vhodně zařazena do již existujícího rozložení uživatelského rozhraní. Rozložení tedy není dle vrstev ISO/OSI, ale dělí se na nastavení hodnot oběti a zdrojových hodnot, které využije útočník. Implementaci uživatelského rozhraní lze vidět na obrázku 8.5 pro UDP flood, na obrázku 8.6 pro DNS flood a na obrázku 8.7 pro DNS amplification. Na obrázcích je zdrojová adresa, označená jako *Source MAC*, která má možnost volby mezi jednou hodnotou adresy a více adresami ze zvoleného rozsahu. Cílová adresa označená jako *Destination MAC* může obsahovat pouze jedinou hodnotu. Ovládání se neliší od již implementovaných možností, například jako u SYN flood.

Obr. 8.4: Uživatelské rozhraní pro nastavení linkové vrstvy *DDoS - SYN Flood*

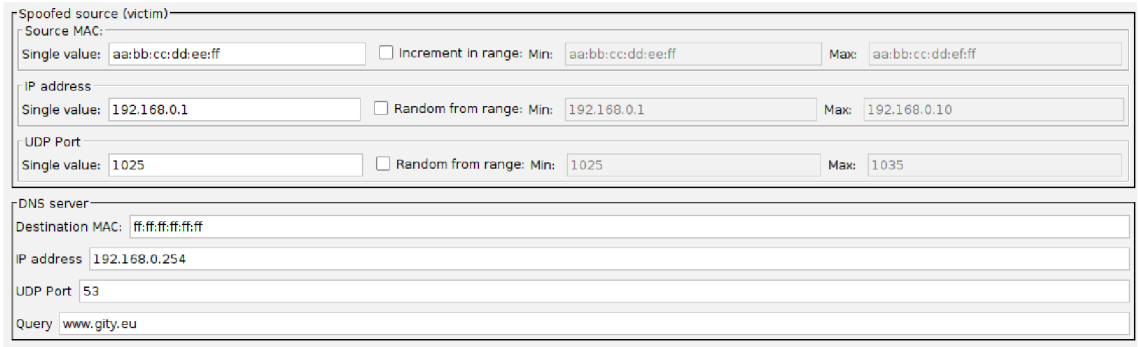


Obr. 8.5: Upravené uživatelské rozhraní pro *DDoS - UDP Flood*

[3]V podobě šablony



Obr. 8.6: Upravené uživatelské rozhraní pro *DDoS - DNS Flood*



Obr. 8.7: Upravené uživatelské rozhraní pro *DDoS - DNS Amplification*

8.2.4 Přidání IPv6 adres pro SYN a UDP flood

Přidání IPv6 pro testy SYN a UDP flood umožnilo zvětšit počet možných testovacích scénářů na dvojnásobné množství. Rozšíření o IPv6 probíhalo ve dvou fázích. Tato podkapitola se bude zabývat první fázi, což bylo přidání zdrojové a cílové IPv6, druhá fáze bude rozebrána a popsána v následující podkapitole.

V první fázi bylo nejdůležitější přidání konfiguračního souboru pro nástroj Trafgen. Konfigurační soubor se liší od konfiguračního souboru pro IPv4, proto musely být přidány dva abstraktní konfigurační soubory pro SYN a UDP flood. Ukázku konfiguračního souboru lze vidět na výpisu kódu 8.2. Jelikož po přidání konfiguračního souboru pro IPv6 mají SYN a UDP flood dva konfigurační soubory, musela být upravena logika, aby bylo možné načítat správný konfigurační soubor. Logika spočívala v přidání kontroly o jakou verzi IP adresy se jedná a uložení do proměnné `configFileSuffix` třídy `TrafgenControl`. Ukládání do této proměnné probíhá pomocí „setteru“. Ve výpisu kódu 8.3 lze vidět výběr abstraktního konfiguračního souboru a jeho uložení do proměnné.

Výpis 8.2: Ukázka konfiguračního souboru pro SYN fload s využitím IPv6

```
#define ETH_P_IP  0x86DD
{
    eth(daddr=##SynFloodSampler.dmac,
        saddr=##SynFloodSampler.smacTG, proto=ETH_P_IP),
    ipv6(ttl=##SynFloodSampler.ttl, ver=6,
        da=##SynFloodSampler.targetIPv6,
        sa=##SynFloodSampler.sourceIPv6),
    tcp(sport=##SynFloodSampler.sourcePortTG,
        dport=##SynFloodSampler.dPort, seq=drnd(), aseq=0,
        hlen=40, syn, win=##SynFloodSampler.winSize),
    fill('B',##SynFloodSampler.payload),
}
```

Výpis 8.3: Generace konfiguračního souboru v závislosti na typu IP adresy

```
if(this.getPropertyAsBoolean(IPV6ENABLED)) {
    trafgenControl.setConfigFileSuffix(configFileSuffixIpv6);
} else {
    trafgenControl.setConfigFileSuffix(configFileSuffix);
}
```

Uživatelské rozhraní byly rozšířeno o textové pole pro zadání zdrojové a cílové IPv6 adresy. Dále zde bylo přidáno „zaškrtávací“ políčko pro výběr, zda uživatel chce využívat IPv6.

8.2.5 Rozšíření IPv6 o možnost zvolit adresy v rozsahu

Tato podkapitola popisuje druhou fázi přidání podpory IPv6 pro testování SYN a UDP flood. Druhá fáze spočívala v rozšíření volby zdrojové IPv6 adresy tak, aby bylo možné zvolit rozsah a počet adres, které se mají využít ze zvoleného rozsahu.

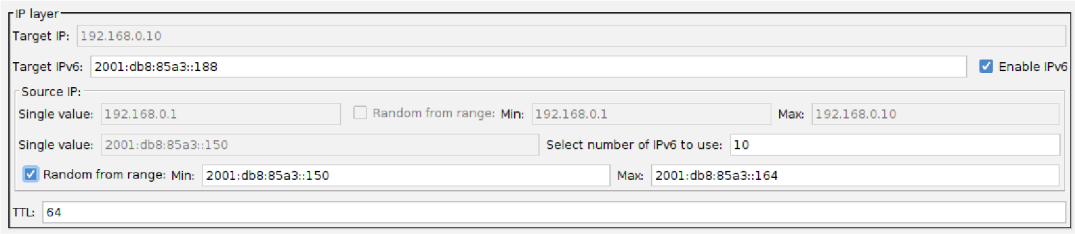
Z programátorského hlediska se jednalo o změnu, která byla rozsáhlá. Limitace konfiguračního souboru pro nástroj Trafgen neumožňovala náhodné zvolení přímo v souboru, jak je tomu v případě IPv4. Limitace byla způsobena z důvodu, že interní funkce `drnd` nástroje Trafgen podporuje pouze celá čísla do velikosti 2^{32}. Hodnota 2^{32} odpovídá IPv4 adrese 255.255.255.255. Pro použití pro IPv6 by tato funkce musela podporovat čísla do velikosti 2^{128}. Tento problém byl vyřešen tak, že konfigurační soubor může obsahovat více definovaných paketů mezi kterými lze náhodně vybírat. Proto musela být přidána nová metoda, která umožňovala vytvářet konfigurační soubor obsahující více paketů. S tím byla spojena také implementace logiky pro rozlišení, jestli použít metodu pro generování konfiguračního souboru obsahujícího

jeden nebo více paketů. Příkaz pro spouštění nástroje Trafgen musel být upraven tak, aby podporoval náhodný výběr z více paketů v jednom konfiguračním souboru. V případě, obsahuje-li konfigurační soubor více paketů, je k příkazu pro spouštění přidáno `-rand`.

Dále bylo nutné přidat metody pro práci s IPv6 adresami, jelikož muselo být ze zadaného rozsahu vybráno N adres, kde N značí počet adres z rozsahu, které chce uživatel vybrat. Výběr adres by měl být v náhodném pořadí a zároveň kontrolován, jestli není rozsah menší, než uživatelem vybraný počet adres. Pro práci s IPv6 adresami byla přidána knihovna `com.github.seancfoley:IPAddress`[4].

Z uživatelského pohledu při „zaškrtnutí“ políčka pro výběr s názvem *Enable IPv6* bude testování probíhat s IPv6 adresami. Uživatel do textového pole *Target IPv6* zadává IPv6 adresu cílového zařízení (oběti). Počet zdrojových adres závisí na „zaškrtnutí“ políčka pro výběr označeného jako *Random from range*. V případě, že není políčko „zaškrtnuté“, je použita pouze jedna IPv6 adresa. V opačné situaci je použito N adres, kdy N odpovídá hodnotě zadané uživatelem do textového pole označeného *Select number of IPv6 to use*. Dále uživatel musí uvést nejmenší a největší adresu z rozsahu, ze kterého chce generovat IPv6 adresy. V případě spuštění testu, kdy je rozsah zvolených IPv6 adres menší, než počet adres, které uživatel chce použít, je zobrazeno upozornění, které umožňuje rozšířit rozsah IPv6 adres. Toto upozornění je znázorněno na obrázku 8.8. Uživatelské rozhraní pro SYN flood je zobrazeno na obrázku 8.9, kdy pro UDP flood je stejné, jen se liší v rozložení jednotlivých prvků[5].

Obr. 8.8: Upozornění na nedostatečný rozsah IPv6 adres



Obr. 8.9: Upravené uživatelské rozhraní pro *DDoS - SYN Flood s IPv6 adresami*

[4]Knihovnu lze nalézt na `https://github.com/seancfoley/IPAddress`
[5]Stejně jako bylo zmíněno v kapitole o přidání MAC adres

9 Vývoj nových částí modulu DDoS

Kapitola se zabývá vývojem `Sampler` *DDoS – Sampler with interface* sloužícímu k testování HTTP/S dotazů a podpůrnému `Config element` *CSV Dataset for set and randomize IP*. Dále se věnuje interoperabilitě mezi těmito dvěma prvky a implementaci třech typů pomalých DoS útoků.

9.1 Vývoj DDoS – Sampler with interface

Sampler DDoS – Sampler with interface je rozšíření již existujícího *Sampler HTTP Request*, který je jedním ze základních *Sampler* nástroje JMeter. Použití základního *Sampler* bylo nedostatečné a pro uživatele bylo složité nastavovat, jaké síťové rozhraní se má využít. Dále bylo implementováno zjednodušení výběru typu IP adresy a implementace interoperability s konfiguračním prvkem *CSV Dataset for set and randomize IP*, který je popsán v následují podkapitole.

DDoS – Sampler with interface z programátorského hlediska neupravuje žádným způsobem „backend“ část, ale upravuje pouze uživatelské rozhraní, aby bylo jednoduší na úpravu. Uživatelské rozhraní bylo značně zjednodušeno, když byla odstraněna celá „Advanced“ část. Prvek pro nastavení zdrojové adresy z odstraněné části byl výrazně upraven a přesunut do hlavní části. Zbylé prvky nebyly potřebné, jelikož obsahovaly nepodstatné nastavení pro účely testování HTTP/S dotazů v rámci testovacích scénářů pro tento *Sampler*.

Jak již bylo zmíněno, probíhala pouze úprava uživatelského rozhraní a proto byla vytvořena jenom třída `InterfaceHttpSamplerGui`. Tato třída využívala již vytvořené rozložení třídy `UrlConfigGui` stejně jako třída `HttpTestSampleGui`, ze které vytvořená třída vychází. Dále zde byl přidán výběr síťového rozhraní, výběr typu IP adresy a náhodné IP adresy. Pro správnou funkčnost výběru náhodné IP adresy je potřeba zmíněný konfigurační prvek. Na výpisu kódu 9.1 lze vidět ukázku toho, jak probíhá nastavení IP adresy dle síťového rozhraní, typu IP adresy a náhodné IP adresy na základě volby uživatele. Tento výběr je poté nastaven v podobě IP adresy do parametru nástroje JMeter označeného jako `IP_SOURCE`. Pro útok z více IP adres se do parametru nepředává specifická IP adresa, ale proměnná `VARIABLE_FOR_RANDOM_IP`.

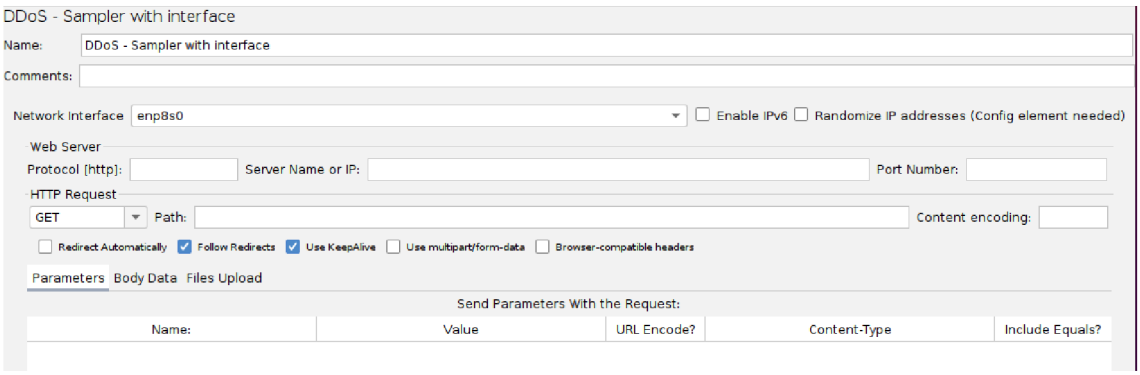
Z pohledu uživatele je nastavování parametrů jednoduché, jak lze vidět na obrázku 9.1. Uživatel musí pro testování nastavit parametry webového serveru, na který má být prováděn útok. V případě, že má být útok veden z jedné IP adresy, uživatel vybírá síťové rozhraní z kombinovaného pole[1] označeného jako „Network in-

[1] Z anglického výrazu Combo box

terface“ a vybírá možnost využití IPv6 adresy pomocí „zaškrtnutí“ políčka „Enable IPv6“. Jestliže políčko „Enable IPv6“ není „zaškrtnuté“, je využito IPv4 adresy. Pokud má být útok veden z více IP adres, uživatel „zaškrtne“ políčko označené jako „Random IP addresses (Config element needed)“, ke kterému je potřebné mít přidaný zmíněný konfigurační prvek. V případě více adres není potřebné vybírat síťové rozhraní a typ IP adresy, jelikož tyto parametry se nastavují přímo v konfiguračním prvku.

Výpis 9.1: Výběr zdrojové IP adresy v závislosti na nastavení

```
if (!IPV6_ENABLED_VALUE && !ENABLE_MULTIPLE_IP_ADDRESS) {
    ipAddres = DDoSUtils
        .getIpFromInterface(selectedDev.toString(),
        DDoSUtils.VersionOfIP.IPV4);
} else if (IPV6_ENABLED_VALUE &&
            !ENABLE_MULTIPLE_IP_ADDRESS) {
    ipAddres = DDoSUtils
        .getIpFromInterface(selectedDev.toString(),
        DDoSUtils.VersionOfIP.IPV6);
} else {
    ipAddres = "${" + VARIABLE_FOR_RANDOM_IP + "}";
}
sampler.setProperty(HTTPSamplerBase.IP_SOURCE, ipAddres);
sampler.setProperty(HTTPSamplerBase.IP_SOURCE_TYPE,
    HTTPSamplerBase.SourceType.HOSTNAME.ordinal());
```

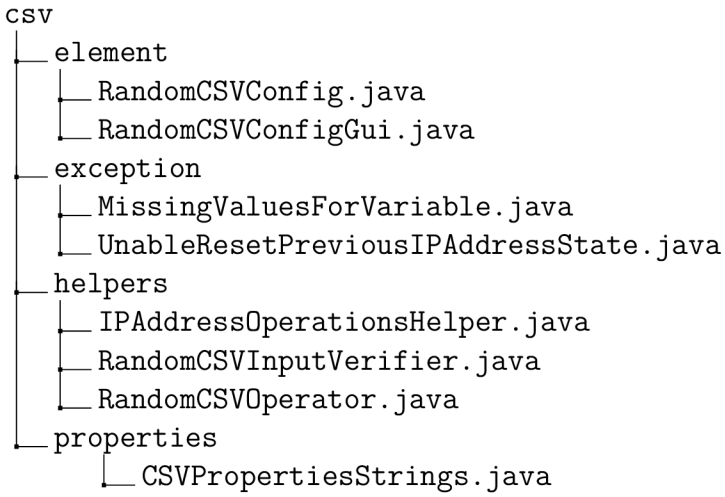


Obr. 9.1: Uživatelské rozhraní *DDoS – Sampler with interface*

9.2 Vývoj CSV Dataset for set and randomize IP

CSV Dataset for set and randomize IP je konfigurační prvek a byl vyvinut s primárním cílem usnadnění práce s IP adresami. Sekundárně se dá využít pro nastavování proměnných nástroje JMeter. Konfigurační prvek umožňuje přidat a odstranit IP adresy zadané do textového pole na vybrané síťové rozhraní. Dále umožňuje tyto IP adresy nastavit do specifické proměnné nástroje JMeter a tím je využít v jiných prvcích nástroje JMeter. Konfigurační prvek je navržen tak, aby bylo z něho možné v případě potřeby vytvořit samostatný modul, jelikož přímo neslouží k testování DDoS útoků.

Z programátorského hlediska vzhledem k návrhu se liší od ostatních částí modulu DDoS. Strukturu balíčků lze vidět na obrázku 9.2. Oproti ostatním částem DDoS modulu obsahuje vlastní výjimky, vlastní soubor pro řetězce textu a třídy sloužící pro obecné operace při práci s konfiguračním prvkem.



Obr. 9.2: Struktura konfiguračního prvku *CSV Dataset for set and randomize IP*

Uživatelské rozhraní je navrženo jednoduše, stejně jako třída `RandomCSVConfig`. Nejsložitější logika probíhá v třídě `IPAddressOperationsHelper`, která slouží pro přidání a odebrání IP adres na specifické síťové rozhraní a operace jak s IPv4 tak i IPv6 adresami. Třída `RandomCSVInputVerifier` má za úkol kontrolu vstupů v uživatelském rozhraní. Třída `RandomCSVOperator` je určena k načtení vstupu z textového pole a v případě, že má být konfigurační prvek využit pro nastavení specifické proměnné pro IP adresu, roztřídí načtený text do dvou listů dle typu IP adresy.

Z pohledu uživatele umožňuje konfigurační prvek rozsáhlé nastavení. Uživatelské rozhraní lze vidět na obrázku 9.3. Do textového pole lze vložit IP adresy třemi způsoby. Prvním je načtení ze souboru pomocí kliknutí na tlačítko „Browse...“, pomocí kterého lze vybrat soubor. Druhým způsobem je ruční zadání IP adres. Tyto

dva způsoby lze kombinovat, takže je možné načíst soubor a poté manuálně přidat nebo odebrat jednu nebo více adres. Druhý způsob slouží jako doplňující, takže první způsob musí být vždy použit jako první a poté lze přidat manuálně IP adresy. Třetím způsobem je generace N adres z určitého rozsahu, kde N je počet adres, který bude vybrán z rozsahu a je zadáván uživatelem. Pro využití této možnosti musí uživatel zaškrtnout políčko „Use random range“. Třetí způsob lze také kombinovat s druhým, ale stejně jako v předchozím případě je druhý způsob pouze komplementární.

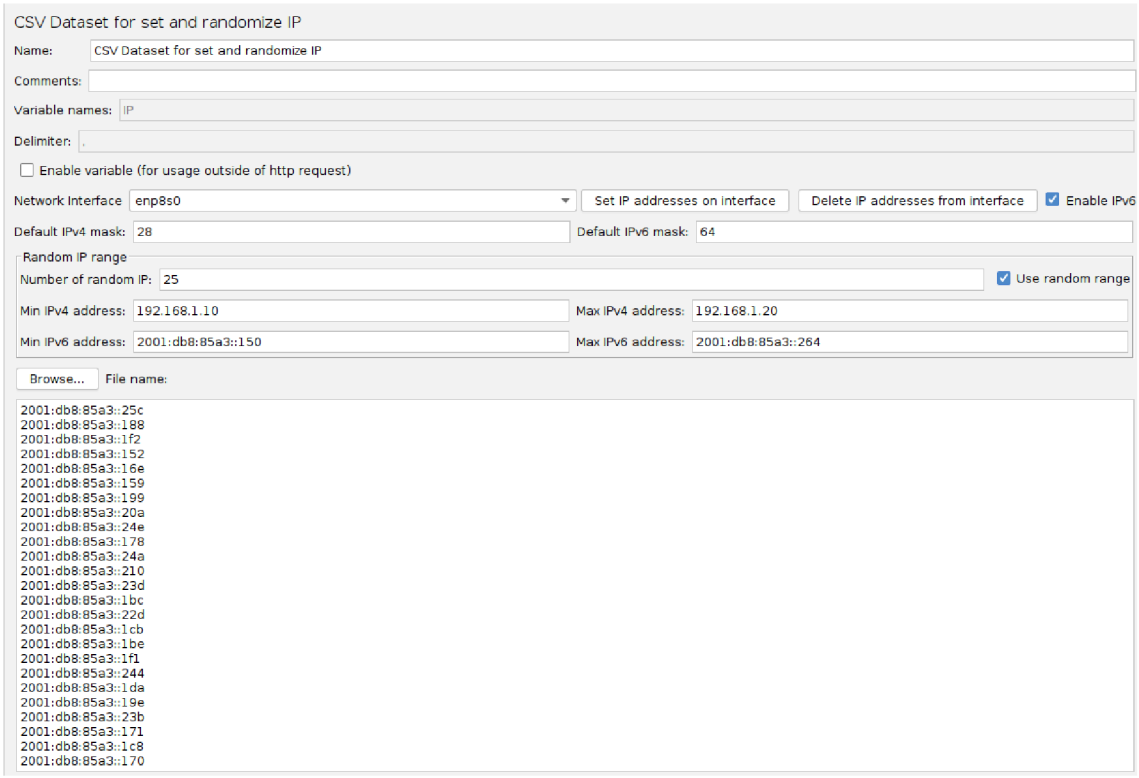
Pro načtení IP adres na síťové rozhraní slouží tlačítko s názvem „Set IP addresses on interface“. Tlačítko má také úlohu vygenerování IP adres v určitém rozsahu, kdy tyto vygenerované adresy budou zároveň nastaveny. Pro odstranění IP adres ze síťového rozhraní se využívá tlačítko „Delete IP addresses from interface“, které odstraní všechny uvedené adresy v textovém poli ve spodní části uživatelského rozhraní. V případě přidání již existující zadané IP adresy na síťové rozhraní bude IP adresa považována za přidanou. Podobně tomu je při odstranění IP adres, kdy se pokládá neexistující adresa za odstraněnou. Vybrat na jaké síťové rozhraní se má přidání nebo odstranění IP adres aplikovat, lze zvolit z kombinovaného pole s názvem „Network interface“. Dále lze volit masku pomocí CIDR (Classless Inter-Domain Routing) notace pro oba dva typy IP adress.

Je také nutné podotknout, že přidávání IP adres není propojené s nastavením specifické proměnné pro nástroj JMeter, jelikož tato proměnná se nastavuje až po začátku testu na rozdíl od přidání IP adres na síťové rozhraní, které ovládá pouze uživatel. Jediným společným prvkem jsou zapsané adresy v textovém poli ve spodní části uživatelského rozhraní.

9.3 Vzájemná interoperabilita

Interoperabilita mezi *DDoS - Sampler with interface* a *CSV Dataset for set and randomize IP* slouží k možnosti testovat HTTP/S dotazy z více IP adres. K tomu se využívá nastavení proměnné nástroje JMeter a tato proměnná je pojmenována jako `VARIABLE_FOR_RANDOM_IP`. Pokud je na prvku *Sampler* zvolena možnost pro výběr z více IP adres a zároveň je konfigurační prvek součástí testovacího plánu v nástroji JMeter, je při každém vzorkování[2] vybrána náhodná IP adresa ze zadaných IP adres v konfiguračním prvku. Pro správné fungování musí mít jak *Sampler* tak i konfigurační prvek nastavenou stejnou verzi IP adres. Pokud IP adresy nebudou nastaveny na stejnou verzi, může dojít k nefunkčnosti testů nebo k nežádoucím výsledkům při testování.

[2] *Sampler* při každém dotazu využívá metody Sample a z ní získává výsledky.



Obr. 9.3: Uživatelské rozhraní *CSV Dataset for set and randomize IP*

Hlavní nevýhodou tohoto přístupu je, že v případě oddělení konfiguračního prvku od DDoS modulu musí být kód synchronizován tak, aby byla používána proměnná se stejným názvem. Tento způsob nejde žádným způsobem obejít, jelikož by mělo být dbáno na to, aby každý přídavný modul byl nezávislý na dalším přídavném modulu.

9.4 Implementace pomalých DoS útoků

Z důvodu rozšíření možností testování musely být přidány pomalé útoky slow headers (Slowloris), slow POST (R.U.D.Y.) a slow Read. Tyto útoky využívají nástroj SlowHTTPTest[3]. V modulu DDoS již existovala vlastní implementace útoku Slowloris, která nebyla optimalizována a nesplňovala všechny požadavky a proto byla označena za zastaralou a nepodporovanou. Všechny 3 typy pomalých útoků využívají stejného základu s využitím rozdílného *Sampler* a rozdílného uživatelského rozhraní. V další části podkapitoly bude popsán společný základ z programátorského hlediska a poté bude rozebráno uživatelské rozhraní pro jednotlivé útoky.

Společný základ lze popsat pouze z programátorského hlediska, jelikož má na starosti jenom správný průběh výkonu testů. Společný základ je navržen podobným

[3]Lze najít na `https://github.com/shekyan/SlowHTTPTest`

způsobem jako pro testy, které využívají nástroj Trafgen, jen je v tomto případě využit nástroj SlowHTTPTest.

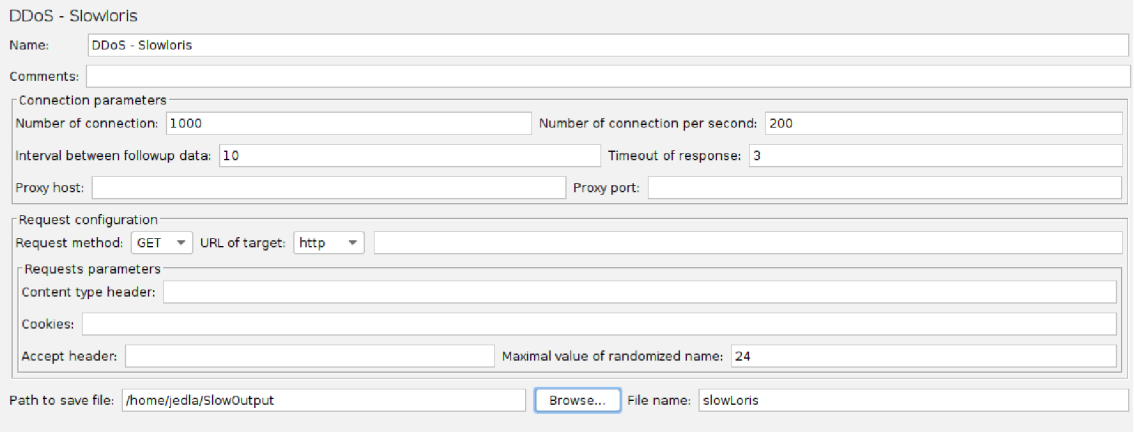
Byla vytvořena třída `SlowAttacksControl`, která slouží ke spouštění a ukončení nástroje SlowHTTPTest. Třída využívá vícevláknovosti (multi-threding), kdy spouští nástroj SlowHTTPTest v samostatném vlákně. Při vykonávání tohoto vlákna je prováděna kontrola *standard error* výstupu (stderr), která byla určena k zastavení v případě, kdy nástroj SlowHTTPTest je ukončen chybou. Zároveň probíhá kontrola *standard output* výstupu (stdout), jestli se na výstupu objeví text *Connection refused*. V případě, že se objeví, znamená to, že spojení nebylo navázáno. To může značit, že nebylo možné vůbec navázat spojení nebo cílový server byl zahlcen po určitou dobu a přestal přijímat nová spojení, takže byl i nástroj SlowHTTPTest ukončen. O jaké ukončení se jedná, se kontroluje v *Sampler* jednotlivých útoků, kdy se vypočítává čas začátku a konce testu. Pokud je rozdíl těchto hodnot menší jak 6 sekund, pravděpodobně se jedná o chybně zadané parametry. V případě rozdílu těchto hodnot větších jak 6 sekund se jedná o přetížené spojení, při kterém není vypsána chybová hláška, ale pouze varovná hláška.

Další třída je `SlowAttackHelpers`, která je společná a slouží pro operace k vytváření příkazu se zadanými parametry od uživatele. Takže se zde převedou jednotlivé vstupy v uživatelském rozhraní na argumenty nástroje SlowHTTPTest. Tyto argumenty jsou poté vráceny a připojeny při spouštění nástroje SlowHTTPTest jako datový typ `String`.

Každý útok má vlastní třídu implementující *Sampler* a vlastní třídu pro uživatelské rozhraní. Velká část těchto tříd obsahuje podobné části, které jsou upraveny tak, aby splňovaly potřebné parametry pro jednotlivé pomalé útoky.

9.4.1 Uživatelské rozhraní Slowloris

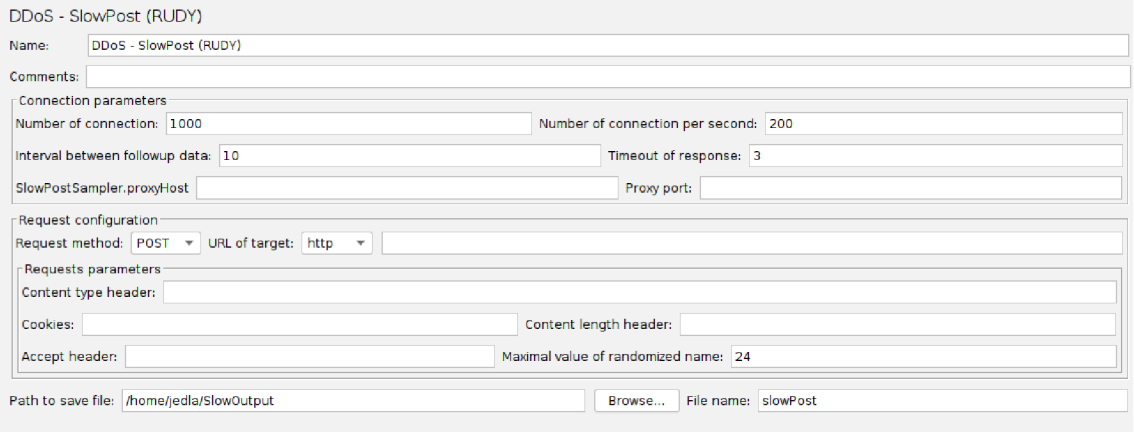
Uživatelské rozhraní lze vidět na obrázku 9.4. Skládá se ze tří hlavních částí. První část obsahuje parametry sloužící pro nastavení spojení. Těmito parametry může být počet opakování dotazu, počet spojení za jednu sekundu, časový limit nebo nastavení parametrů proxy serveru. Druhá část specifikuje parametry dotazu, kterými jsou zejména typ protokolu (HTTP/HTTPS) a IP adresa nebo url cílové webové stránky. Volitelnými parametry jsou cookies, parametry hlavičky a další. Třetí část, která je pro všechny pomalé útoky stejná, obsahuje volbu cesty pro uložení a názvu souboru do kterého se mají výsledky uložit. Volbu cesty lze provést stiskem tlačítka „Browse...“. Po stisku se objeví dialogové okno pro vybrání složky, do které budou výstupní soubory uloženy.



Obr. 9.4: Uživatelské rozhraní *DDoS - Slowloris*

9.4.2 Uživatelské rozhraní slow POST

Uživatelské rozhraní pro slow POST je téměř identické jako rozhraní pro Slowloris. Jediným rozdílem je zde typ použité metody pro dotazy, kde je využita metoda POST místo metody GET. Stejně jako Slowloris obsahuje tři části. První část pro nastavení parametrů spojení, druhou část pro nastavení parametrů dotazu a třetí část pro volbu ukládání. Uživatelské rozhraní lze vidět na obrázku 9.5.

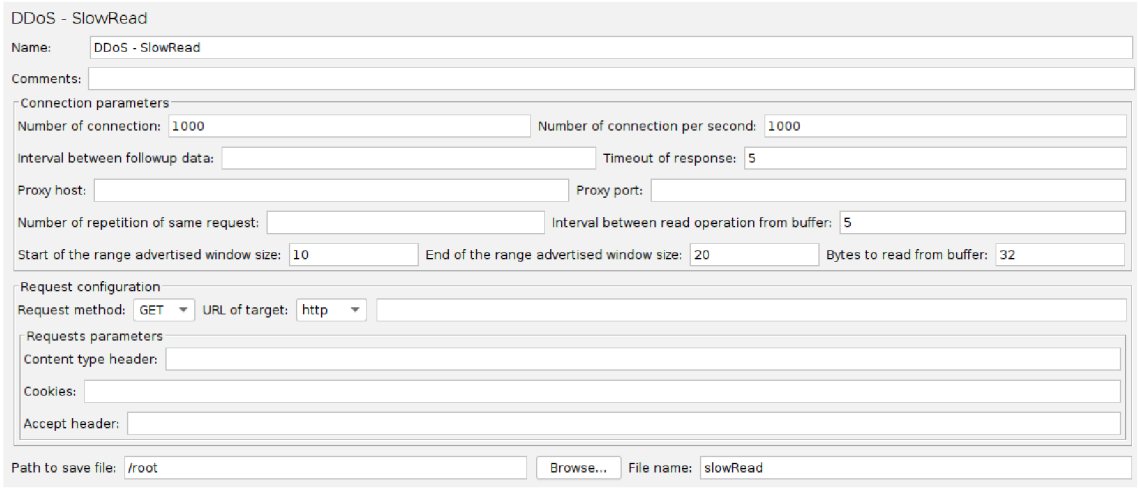


Obr. 9.5: Uživatelské rozhraní *DDoS - SlowPost (RUDY)*

9.4.3 Uživatelské rozhraní slow Read

Uživatelské rozhraní pro slow Read je rozděleno na stejné tři části jako předchozí dvě rozhraní. V případě rozhraní pro tento útok obsahuje první část, sloužící pro nastavení spojení, ale má více volitelných parametrů, jelikož jsou potřebné k vykonání útoku. Kromě již zmíněných parametrů obsahuje počet opakování stejného

dotazu, začátek a konec rozsahu velikosti okna nebo počet bytů, které se mají přečíst z vyrovnávací paměti. Druhá část je téměř identická. Jediným rozdílem je chybějící textové pole pro specifikování maximální délky následujících dat. Toto textové pole zde není potřeba, jelikož slouží pouze pro útoky Slowloris a slow POST. Třetí část se využívá k ukládání výstupních dat a obsahuje stejné uživatelské vstupy jako předchozí dva útoky. Uživatelské rozhraní pro pomalý útok slow Read lze vidět na obrázku 9.6.



Obr. 9.6: Uživatelské rozhraní *DDoS - SlowRead*

Závěr

Cílem diplomové práce bylo aktualizovat operační systém zátěžového testeru, nástroj Apache JMeter a jeho přídavné moduly. Dále opravit případné nalezené chyby vzniklé při aktualizaci a vytvořit skripty pro automatizaci nasazení a aktualizaci nástroje JMeter a rozšiřujících modulů. Poté útokům z přídavného modulu *DDoS* přidat parametry pro nastavení linkové vrstvy a vybrané útoky rozšířit o podporu IPv6. Konečným cílem bylo rozšířit testovací scénáře o nové možnosti a tyto scénáře otestovat s rozšiřujícími moduly tak, aby byly odhaleny případné vzniklé chyby.

Operační systém zátěžového testeru byl aktualizován na distribuci Ubuntu ve verzi 20.04. Nástroj Apache Jmeter byl poté aktualizován na verzi 5.5. Sedm přídavných modulů pro nástroj Apache JMeter bylo převedeno tak, aby byly více modulární a daly se samostatně sestavovat pomocí nástroje Gradle.

Po aktualizaci byl každý modul otestován, jestli lze spustit test bez chyb. U některých modulů se vyskytly chyby, které byly úspěšně opraveny. Pokud chyby byly závažnější, byla jim přiřazena vyšší priorita, jelikož znemožňovaly spuštění testu nebo způsobovaly pád nástroje Apache JMeter.

Skripty pro automatizaci nasazení a aktualizaci nástroje JMeter a rozšiřujících modulů byly vytvořeny a rozděleny na více částí tak, aby byla pro uživatele ovladatelnost co nejjednodušší.

Modulu *ServerEmulator* byla doplněna podpora IPv6 pro webový server. Uživateli bylo umožněno vygenerovat webovou stránku menší než 1 kiB, která je využívána na webovém serveru. Vybraným testům z modulu *DDoS* byla přidána podpora zdrojové a cílové IPv6 adresy. Dále byly testy rozšířeny o možnost nastavení parametrů linkové vrstvy a dalších parametrů, které mohou být proměnlivé.

Pro modul *DDoS* byly vytvořeny nové scénáře, kdy pro některé scénáře bylo nutné implementovat nové prvky. Nové prvky přidávají možnost testovat různé scénáře pro 3 typy pomalých útoků a dále možnost testovat HTTP/S požadavky s využitím jedné nebo více IP adres v obou verzích.

V konečné fázi byly všechny testovací scénáře úspěšně otestovány. Tyto scénáře sloužily pro ověření funkčnosti všech parametrů. Při testování nebylo prováděno zátěžové testování, takže je možné, že při maximální zátěži se mohou objevit nečekané chyby. Úspěšným otestováním modulů lze konstatovat, že všechny cíle této diplomové práce byly splněny a výstupem je aktualizovaný zátěžový tester, nástroj JMeter a aktualizované a rozšířené moduly.

Možným rozšířením práce je přidání dalších typů útoků do modulu *DDoS*. Dále se zde naskytuje možnost rozšířit modul *ReportGenerator* o sběr a převod výsledků testování do podoby čitelné pro uživatele.

Literatura

[1] MEIER, J.D. *Performance Testing Guidance for Web Applications: Patterns & Practices*. Microsoft Press, 2007, 288 s. ISBN 9780735625709.

[2] MAČEJOVSKÝ, Oliver. *Zátěžové testování webových aplikací* [online]. Brno, 2016 [cit. 23. 10. 2022]. Dostupné z URL: `https://is.muni.cz/th/gerzd`. Bakalářská práce. Masarykova univerzita, Fakulta informatiky. Vedoucí práce Tomáš REBOK.

[3] Apache JMeter. APACHE SOFTWARE FOUNDATION. *Apache JMeter* [online]. ©1999-2022 [cit. 24. 10. 2022]. Dostupné z URL: `https://jmeter.apache.org/`

[4] Apache JMeter Distributed Testing Step-by-step. APACHE SOFTWARE FOUNDATION. *Apache* JMeter [online]. ©1999-2022 [cit. 24. 10. 2022]. Dostupné z URL: `https://jmeter.apache.org/usermanual/jmeter_distributed_testing_step_by_step.html`

[5] ERINLE, Bayo. *Performance Testing with JMeter 3 - Enhance the performance of your web application*. 3rd edition. Birmingham: Packt Publishing, 2017, 166 s. ISBN 9781787285774.

[6] ERINLE, Bayo. *Performance Testing with JMeter 2.9*. Birmingham: Packt Publishing, 2013, 148 s. ISBN 9781782165859.

[7] SOMMERS, Joel, Kim HYUNGSUK a Paul BARFORD. *Harpoon: A Flow-Level Traffic Generator for Router and Network Tests* [online]. Association for Computing Machinery, 2004, 392 [cit. 27. 10. 2022]. ISSN 0163-5999. Dostupné z URL: doi:`https://doi.org/10.1145/1012888.1005733`

[8] Chapter 8 - Using Simulation for Research. V: EDGAR, Thomas W. a David O. MANZ. *Research Methods for Cyber Security* [online]. Syngress, s. 193-212 [cit. 27. 10. 2022]. ISBN 9780128053492. Dostupné z URL: `https://doi.org/10.1016/B978-0-12-805349-2.00008-X`

[9] Trafgen - a fast, multithreaded network packet generator. *Ubuntu manuals* [online]. ©2019 [cit. 28. 10. 2022]. Dostupné z URL: `https://manpages.ubuntu.com/manpages/bionic/man8/trafgen.8.html`

[10] Packet - packet interface on device level. *Ubuntu manuals* [online]. ©2019 [cit. 28. 10. 2022]. Dostupné z URL: `https://manpages.ubuntu.com/manpages/bionic/man7/packet.7.html`

[11] SlowHttpTest simulate a DOS attack!. *Medium* [online]. [cit. 27. 4. 2023]. Dostupné z URL: https://medium.com/@4ag2/slowhttptest-simulate-a-dos-attack-69a0d854dba

[12] BELOKRYLOV, Alexander. Why And How Java Continues To Be One Of The Most Popular Enterprise Coding Languages. *Forbes* [online]. Forbes Media LLC., ©2022 [cit. 30. 10. 2022]. Dostupné z URL: https://www.forbes.com/sites/forbestechcouncil/2022/04/06/why-and-how-java-continues-to-be-one-of-the-most-popular-enterprise-coding-languages/

[13] What is Java?. AMAZON. *Amazon Web Services* [online]. Amazon Web Services, ©2022 [cit. 30. 10. 2022]. Dostupné z URL: https://aws.amazon.com/what-is/java/

[14] The art of long-term support and what LTS means for the Java ecosystem. SMITH, Donald. ORACLE. *Java Magazine* [online]. Oracle, ©2022, 9. 9. 2021 [cit. 30. 10. 2022]. Dostupné z URL: https://blogs.oracle.com/javamagazine/post/java-long-term-support-lts

[15] Differences between Java 8 and Java 9?. ORACLE. *Tutorial Spoint* [online]. ©2022, 23. 3. 2020 [cit. 30. 10. 2022]. Dostupné z URL: https://www.tutorialspoint.com/differences-between-java-8-and-java-9

[16] "Hello, World!". In: OGIHARA, Mitsunori. *Fundamentals of Java Programming* [online]. Springer Cham, 2018, s. 3-24 [cit. 30. 10. 2022]. e-ISBN 9783319894911. Dostupné z URL: doi:https://doi.org/10.1007/978-3-319-89491-1

[17] BEHLER, Marco. Java Versions and Features. *Unconventional Guides. For Programmers.* [online]. ©2021, 28. 10. 2022 [cit. 30. 10. 2022]. Dostupné z URL: https://www.marcobehler.com/guides/a-guide-to-java-versions-and-features

[18] MUSCHKO, Benjamin. *Gradle in Action.* New York: Manning Publication Co., 2014, 480 s. ISBN 9781617291302.

[19] SYED, Shahbaz Ali a Tariq Rahim SOOMRO. *Achieving Software Release Management and Continuous Integration using Maven, Jenkins and Artifactory* [online]. 2018, **3**(2) [cit. 31. 10. 2022]. ISSN 25204475. Dostupné z URL: https://www.researchgate.net/publication/330008070_Achieving_Software_Release_Management_and_Continuous_Integration_using_Maven_Jenkins_and_Artifactory

[20] Gradle vs Maven Comparison. *Gradle* [online]. Gradle, ©2022 [cit. 2. 11. 2022]. Dostupné z URL: https://gradle.org/maven-vs-gradle

[21] What is Gradle?. *Gradle User Manual* [online]. Gradle, ©2022 [cit. 2. 11. 2022]. Dostupné z URL: https://docs.gradle.org/current/userguide/what_is_gradle.html

[22] DOULIGERIS, Christos a Aikaterini MITROKOTSA. DDoS attacks and defense mechanisms: classification and state-of-the-art. *Computer Networks* [online]. 2004, **44**(5), 643-666 [cit. 4. 11. 2022]. ISSN 13891286. Dostupné z URL: doi:https://doi.org/10.1016/j.comnet.2003.10.003

[23] Dos vs DDoS. In: *Cloudflare* [online]. Cloudflare, ©2022 [cit. 4. 11. 2022]. Dostupné z URL: https://www.cloudflare.com/img/learning/ddos/glossary/dos-attack/dos-vs-ddos-attack.png

[24] SHETH, Chirag a Rajesh THAKKER. Performance Evaluation and Comparison of Network Firewalls under DDoS Attack. *International Journal of Computer Network and Information Security* [online]. 2013, **5**(12), 60-67 [cit. 4. 11. 2022]. ISSN 20749090. Dostupné z URL: doi:https://doi.org/10.5815/ijcnis.2013.12.08

[25] XU, Rui, Wen-li MA a Wen-ling ZHENG. Defending against UDP Flooding by Negative Selection Algorithm Based on Eigenvalue Sets. *2009 Fifth International Conference on Information Assurance and Security* [online]. IEEE, 2009, 342-345 [cit. 5. 11. 2022]. ISBN 9780769537443. Dostupné z URL: doi:https://doi.org/10.1109/IAS.2009.280

[26] ŠVEHLÁK, Milan. *Rozšíření nástroje JMeter* [online]. Brno, 2017 [cit. 5. 11. 2022]. Dostupné z URL: http://hdl.handle.net/11012/65888. Diplomová práce. Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací. Vedoucí práce Zdeněk Martinásek.

[27] VISHNU, N. S., Ranbir SINGH BATTH a Gursharan SINGH. Denial of Service: Types, Techniques, Defence Mechanisms and Safe Guards. *2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)* [online]. IEEE, 2019, 2019, 698 [cit. 5. 11. 2022]. ISBN 9781728137780. Dostupné z URL: doi:https://doi.org/10.1109/ICCIKE47802.2019.9004388

[28] RAMADHAN, Gilang, Yusuf KURNIAWAN a CHANG-SOO KIM. Design of TCP SYN Flood DDoS attack detection using artificial immune systems. *2016*

6th International Conference on System Engineering and Technology (ICSET) [online]. IEEE, 2016, 2016, 72-76 [cit. 5. 11. 2022]. ISBN 9781509050895. Dostupné z URL: doi:https://doi.org/10.1109/ICSEngT.2016.7849626

[29] GUPTA, Neha, Ankur JAIN, Pranav SAINI a Vaibhav GUPTA. DDoS attack algorithm using ICMP flood. *International Conference on Computing for Sustainable Global Development (INDIACom)* [online]. 2016 [cit. 6. 11. 2022]. Dostupné z URL: https://ieeexplore.ieee.org/document/7725026

[30] CHARVÁT, Ondřej. *Zátěžový tester* [online]. Brno, 2017 [cit. 7. 11. 2022]. Dostupné z URL: http://hdl.handle.net/11012/65877. Diplomová práce. Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací. Vedoucí práce Václav Zeman.

[31] What is a DNS amplification attack?. *Cloudflare* [online]. Cloudflare, ©2022 [cit. 7. 11. 2022]. Dostupné z URL: https://www.cloudflare.com/learning/ddos/dns-amplification-ddos-attack/

[32] DAMON, Evan, Julian DALE, Evaristo LARON, Jens MACHE, Nathan LAND a Richard WEISS. Hands-on denial of service lab exercises using SlowLoris and RUDY. *Proceedings of the 2012 Information Security Curriculum Development Conference on - InfoSecCD '12* [online]. New York, New York, USA: ACM Press, 2012, 2012, 21-29 [cit. 7. 11. 2022]. ISBN 9781450315388. Dostupné z URL: doi:http://dl.acm.org/citation.cfm?doid=2390317.2390321

[33] R U Dead Yet? (R.U.D.Y.) attack. *Cloudflare* [online]. Cloudflare, ©2022 [cit. 7. 5. 2023]. Dostupné z URL: https://www.cloudflare.com/learning/ddos/ddos-attack-tools/r-u-dead-yet-rudy/

[34] PARK, Junhan, Keisuke IWAI, Hidema TANAKA a Takakazu KUROKAWA. Analysis of Slow Read DoS attack. *2014 International Symposium on Information Theory and its Applications* [online]. 2014 [cit. 7. 5. 20230]. e-ISBN 978-4-8855-2292-5. Dostupné z URL: https://ieeexplore.ieee.org/abstract/document/6979803

[35] What is an HTTP flood DDoS attack?. *Cloudflare* [online]. Cloudflare, ©2022 [cit. 10. 11. 2022]. Dostupné z URL: https://www.cloudflare.com/learning/ddos/http-flood-ddos-attack/

[36] HTTPS Flood. *Radware* [online]. Radware, ©2022 [cit. 10. 11. 2022]. Dostupné z URL: https://www.radware.com/security/ddos-knowledge-center/ddospedia/https-flood/

[37] LANŽHOTSKÝ, Karel. *Zátěžový tester* [online]. Brno, 2022 [cit. 10. 11. 2022]. Dostupné z URL: `http://hdl.handle.net/11012/205533`. Bakalářská práce. Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací. Vedoucí práce Václav Zeman.

[38] Comparing Centos Linux and CentOS Stream. *The CentOS Project* [online]. The CentOS Project, ©2022 [cit. 13. 11. 2022]. Dostupné z URL: `https://www.centos.org/cl-vs-cs/`

[39] O Ubuntu. *Ubuntu CZ/SK* [online]. OpenAlt, 2022 [cit. 13. 11. 2022]. Dostupné z URL: `https://wiki.ubuntu.cz/faq/o_ubuntu`

Seznam symbolů a zkratek

API Aplikační programovací rozhraní – Application Programming Interface

CIDR Beztřídní směrování – lassless Inter-Domain Routing

DoS Odepření služby – Denial of Service

DDoS Distribuované odepření služby – Distributed Denial of Service

DNS Systém doménových jmen – Domain Name System

DE Desktopové prostředí – Desktop Environment

FTP Protokol přenosu souborů – File Transfer Protocol

GUI Grafické uživatelské rozhraní – Graphic User Interface

HTTP Hypertextový přenosový protokol – Hypertext Transfer Protocol

HTTP/S Hypertextový přenosový protokol/zabezpečený – Hypertext Transfer Protocol/Secure

HTTPS Zabezpečený hypertextový přenosový protokol – Hypertext Transfer Protocol Secure

ICMP Internetový protokol řídicích zpráv – Internet Control Message Protocol

IMAP Protokol pro přístup k internetovým zprávám – Internet Message Access Protocol

IP Internetový protokol – Internet Protocol

JDBC Připojení k databázím Java – Java Database Connectivity

JDK Vývojová sada pro jazyk Java – Java Development Kit

JRE Prostředí Java Runtime Environment – Java Runtime Environment

JVM Virtuální stroj Java – Java Virtual Machine

LDAP Protokol lehkého adresářového přístupu – Lightweight Directory Access Protocol

LTS Dlouhodobá podpora – Long-term support

MAC Řízení přístupu k médiu –Media Access Control

OSI Propojení otevřených systémů – Open Systems Interconnection

POM Model objektů projektu – Project Object Model

POP3 Poštovní protokol verze 3 – Post Office Protocol version 3

PYPL Popularita programovacích jazyků – PopularitY of Programming Language

RAM Paměť s náhodným přístupem – Random Access Memory

RegExr Regulární výraz – Regular Expression

RFC Žádost o vyjádření – Request For Comments

RHEL Red Hat Enterprise Linux – Red Hat Enterprise Linux

SELinux Systém Linux s vylepšeným zabezpečením – Security-Enhanced Linux

SSH Zabezpečený shell – Secure Shell

SSL Vrstva bezpečných socketů – Secure Sockets Layer

TCP Protokol řízení přenosu – Transmission Control Protocol

TLS Zabezpečení transportní vrstvy – Transport Layer Security

UDP Protokol uživatelských datagramů – User Datagram Protocol

USB Univerzální sériová sběrnice – Universal Serial Bus

XML Rozšiřitelný značkovací jazyk – Extensible Markup Language

A Obsah přílohy

Kapitola obsahuje popis příloh. Ve složce *modules* se nachází zkompilované moduly pomocí openJDK 17. Složka *jmeter-deployment* obsahuje soubory pro automatizaci nasazení a aktualizaci nástroje JMeter a přídavných modulů. Tyto skripty byly popsány v kapitole 7 Automatizace pro nasazení a aktualizaci nástroje JMeter.

```
/ ..... kořenový adresář
├── README.md ..... popis příloh
├── modules ..... zkompilované moduly v podobě jar souborů
│   ├── AdvancedThreadGroups.jar
│   ├── Ddos.jar
│   ├── NetAnalyzer.jar
│   ├── NetEmulator.jar
│   ├── ReportGenerator.jar
│   ├── ServerEmulator.jar
│   └── WebGenerator.jar
└── jmeter-deployment ..... skripty pro automatizaci nasazení a aktualizaci modulů
    ├── README.md ..... návod k jednotlivým skriptům
    ├── build_and_deploy.sh
    ├── prepare_server.sh
    └── start_jmeter.sh
```