VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2021 Bc. Radoslav Heriban

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

MODULY PRO MANUÁLNÍ PENETRAČNÍ TESTOVÁNÍ WEBOVÉ APLIKACE
MODULES FOR MANUAL PENETRATION TESTING OF A WEB APPLICATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Radoslav Heriban

VEDOUCÍ PRÁCE
SUPERVISOR
Mgr. Ing. Pavel Šeda

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Bc. Radoslav Heriban ***ID:*** 197733

Ročník: 2 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Moduly pro manuální penetrační testování webové aplikace

POKYNY PRO VYPRACOVÁNÍ:

Hlavním cílem diplomové práce je návrh a implementace rozšiřujících modulů pro nástroj Burp Suite. Cílem modulů je umožnit spolupráci tohoto nástroje s dalšími externími nástroji k realizaci bezpečnostního testování webových aplikací (např. Penterep a jeho moduly naprogramovanými v Pythonu). V teoretické části práce analyzujte manuální bezpečnostní testování webových aplikací a možnosti rozšíření standardně používaného nástroje Burp Suite. Hlavním cílem je tedy návrh a implementace rozhraní umožňující předávat parametry k realizaci bezpečnostních testů webové aplikace. Vytvořené řešení bude obsahovat knihovnu vytvořenou v jazyce Java pro nástroj Burp Suite a webové rozhraní implementované v knihovně React. Funkčnost aplikace důkladně otestujte na experimentálním pracovišti.

DOPORUČENÁ LITERATURA:

[1] WEAR, Sunny. Burp Suite Cookbook: Practical recipes to help you master web penetration testing with Burp Suite. Packt Publishing Ltd, 2018.

[2] BACUDIO, Aileen G., et al. An overview of penetration testing. International Journal of Network Security & Its Applications, 2011, 3.6: 19.

Termín zadání: 1.2.2021 ***Termín odevzdání:*** 24.5.2021

Vedoucí práce: Mgr. Ing. Pavel Šeda

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

ABSTRAKT

Cieľom práce bolo vytvorenie modulu pre program Burp Suite, vďaka ktorému bude možná výmena dát medzi Burp Suite a inými automatizovanými nástrojmi. Súčasťou práce bol aj vývoj webovej aplikácie zobrazujúcej informácie o nálezoch. V kapitole dva sa nachádza analýza nástrojov, s ktorými sa penetračný testeri bežne stretávajú a s ktorými by mohol modul spolupracovať. Ako programovacie jazyky boli použité Java a JavaScript. Vytvorený modul umožňuje prijímanie dát a ich vkladanie do modulov ako Repeater, Intruder, Scanner. Ďalej modul umožňuje odosielanie dát obsiahnutých v Burp Suite, napr. proxy históriu, sitemapu alebo informácie o nálezoch externým klientom vo formáte JSON pre ďalšie spracovanie. Pri vývoji Webovej aplikácie bola použitá JavaScriptová knižnica React. Webová aplikácia slúži ako informačný panel s grafickou vizualizáciou nálezov.

KĽÚČOVÉ SLOVÁ

Burp, BurpSuite, Java, React, JSON, rozšírenie, modul, penetračné, testovanie, JavaScript

ABSTRACT

The main goal of this master's thesis was development of Burp Suite extension capable of interacting with various other automated tools, accompanied with development of a web application. Chapter two contains analysis of tools commonly used in penetration testing that could benefit from the ability to share Burp Suites data or functionality. The programming languages used were Java and JavaScript. The extension acts as a gateway to inner functionality of Burp Suite. It enables exfiltration of in memory objects such as sitemap, proxy history or found issues in JSON format to other tools, and also listens for incoming data that can be inserted into it's existing modules such as Repeater, Scanner, Spider or Comparer. Frontend application was written using JavaScript library React. The web application offers a graphical visualization of issue data.

KEYWORDS

(Burp, BurpSuite, Java, React, JSON, extension, module, penetration, testing, JavaScript)

HERIBAN, Radoslav. *Moduly pro manuální penetrační testování webové aplikace*. Brno, 2020, 73 s. Diplomová práca. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedúci práce: Mgr. Ing. Pavel Šeda

VYHLÁSENIE

Vyhlasujem, že svoju diplomovú prácu na tému „Moduly pro manuální penetrační testování webové aplikace“ som vypracoval samostatne pod vedením vedúceho diplomovej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej diplomovej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto diplomovej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora

POĎAKOVANIE

Rád by som poďakoval vedúcemu diplomovej práce pánovi Ing. Mgr. Pavlovi Šedovi za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci. Ďalej by som chcel poďakovať svojim rodičom za podporu. V neposlednej rade ďakujem aj spolužiakom Samuelovi Sidorovi, Martinovi Šuličovi, Martinovi Hrdému, Ondřejovi Kupkovi, Antonínovi Bohačíkovi a mnohým ďalším za priateľskú spoluprácu počas celých dlhých piatich rokov štúdia.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam výpisov

Úvod

Táto práca sa zameriava na problematiku internetovej bezpečnosti. V modernej dobe sa kôli rastúcemu trendu digitalizácie dostáva do internetu aj množstvo citlivých dát. Pribúdajú služby cloudového charakteru, ktoré využívajú množstvo technológii. Komplexita moderných systémov však znamená aj vyšší počet bezpečnostných nedostatkov, ktoré môžu byť cieľom kyberkriminality.

Penetračné testovanie je proaktívnou formou zabezpečenia. Ponúka možnosť simulácie reálneho útoku v kontrolovanom prostredí s jasne definovanými pravidlami bez deštruktívnych následkov. Subjektom testovania ponúkajú prehľad o silných a slabých stránkach, ktoré môžu byť zneužité útočníkmi. Na základe týchto poznatkov sú vykonané ochranné opatrenia, aby taká situácia nenastala. V prvej časti práce je popísané penetračné testovanie ako také. Ďalej je zrovnané s podobnými formami prevencie, teda s Red Teamami a Bug Bounty programami. Kapitola úvod do penetračného testovania popisuje aj typové rozdelenie penetračných testov, rozdelené na základe útočníkových vedomostí o systéme, cieli útoku a pozície útočníka v rámci infraštruktúry subjektu.

V druhej kapitole sa nachádza analýza často používaných nástrojov a detailnejšie sa venuje nástroju BurpSuite. Ten spadá do niekoľkých kategórií nástrojov ako HTTP proxy, crawler alebo skener zraniteľností, a je štandardom pre špecialistov na penetračné testovanie so zameraním na webové aplikácie. BurpSuite je známy vďaka svojej HTTP proxy funkcionalite, ktorá umožňuje testerom modifikáciu HTTP žiadostí v reálnom čase. Ďalšou veľmi nápomocnou funkciou je skener zraniteľností, ktorý automaticky testuje žiadosti na známe zraniteľnosti. Práca popisuje prečo je na vývoj modulu vybraný práve BurpSuite.

Cieľom práce je vytvorenie zásuvného modulu pre nástroj BurpSuite, ktorý umožní integráciu funkcionality BurpSuite do automatizovaného systému skriptov a iných nástrojov. BurpSuite poskytuje vývojárom API rozhranie, ktoré umožňuje vývoj zásuvných modulov v niekoľkých programovacích jazykoch. Pre účely tejto práce je použitý programovací jazyk Java.

V tretej kapitole sú popísané požiadavky na funkcionalitu modulu. Tieto požiadavky spĺňajú vlastnosti REST API podľa druhého stupňa Richardsonovho modelu [1], čo je aj navrhovaným riešením.

V kapitole návrhu implementácie je rozobrané možné riešenie na základe limitácií a možností BurpSuite. Taktiež popisuje štruktúry s ktorými BurpSuite pracuje. Okrem samotného modulu práca zahŕňa aj vytvorenie webovej aplikácie, ktorej úlohou je sprehľadnenie informácií o nálezoch. Webová aplikácia je vyvíjaná pomocou knižnice React.

Posledná kapitola obsahuje popis použitých rozhraní BurpSuite API a ich metód,

ktoré modul využíva, ďalej popisuje štruktúry a objekty použité pri implementácii ako napr. použité HTTP výzvy.

1 Úvod do penetračného testovania

Dnešný svet je závislý na internete. Existuje nespočetne veľa technológii, frameworkov, programov, šablón, služieb, webových aplikácii a iných softwarových vymožeností, ktoré poskytujú ľuďom jednoduché riešenia problémov spojených s akoukoľvek prácou s dátami. Avšak všetky z týchto vecí obsahujú aj chyby, či už v softwarovej implementácii alebo chyby v logike aplikácie, ktoré sa dajú zneužiť. Týmito skutočnosťami sa v oblasti informačných technológii zaoberá odvetvie kybernetickej bezpečnosti. Tá sa sústredí na praktické a teoretické zabezpečenie systémov, sietí a programov, ale aj poučovaní ľudských pracovníkov pred útokmi, ktorých cieľom je získanie prístupu k citlivým dátam, ich zničenie, zmena, alebo zamedzenie prístupu k nim. Systémy sú len tak bezpečné ako ich najslabší článok. Preto sa treba zamerať na bezpečnosť všetkých zložiek systému, konkrétne: siete, používané aplikácie, zabezpečenie informácii, riadenie prístupu a ďalej je rovnako nutné vedieť reagovať na incidenty [2].

Jedným z prístupov k zlepšeniu bezpečnosti sú audity, penetračné testy, najatie red teamu, alebo čoraz populárnejšie bug bounty programy [3].

1.1 Čo je penetračné testovanie?

Je proces hľadania a zneužívania zraniteľností v počítačových systémoch formou simulovaného útoku. Útočník však nie je neznámy aktér, ale je ním najatý špecialista z oblasti kybernetickej bezpečnosti, ktorý test vykonáva. Cieľom penetračného testu je preverenie implementovaných defenzívnych opatrení, odhalenie zraniteľností, ich ďalšie zneužitie alebo odhad dopadu, ktorý by zneužitie mohlo zanechať. Výsledkom testu je správa o nájdených nedostatkoch s návodom na ich odstránenie pre testovaný subjekt. Z povahy tejto úlohy vyplýva, že musí byť štruktúrovaná a dôsledná. K tomu slúžia metodológie testovania, napr. zrejme najznámejšia pre testovanie webových aplikácii, OWASP Web Security Testing Guide od komunity odborníkov v oblasti kyberbezpečnosti pod záštitou organizácie OWASP (Open Web Application Security Project) alebo normy ISO/IEC 27001, alebo Penetration Testing Execution Standard (PTES) [2, 4].

1.1.1 Penetračné testovanie vs Red Team

Podobným preventívnym prostriedkom je aj činnosť tzv. Red Teamov. Pomenovanie je z historických príručiek známych ako „Rainbow Series“ zameraných na kybernetickú bezpečnosť, z ktorých prvá bola známa ako „Red Book“, popisujúca útoky na systémy. Nasledovali ju modrá, defenzívna, a neskôr množstvo ďalších, zameraných

na iné témy bezpečnosti. Red Team je tiež útočný tím, ktorý simuluje reálneho útočníka. Jeho ciele sú podobné ako ciele penetračného testu, čo môže vyvolávať mierne zmätenie alebo zamenenie pojmov. Hlavný rozdiel je v tom, že sa jedná o tím zložený z špecialistov na rôzne okruhy bezpečnosti z dôvodu zaistenia maximálne efektívnych útokov. Pozícia Red Teamu je ekvivalentná externému black box penetračnému testu. Časový rozsah operácie je taktiež dlhší [5].

1.1.2 Penetračné testovanie vs Bug Bounty

Novou populárnou formou prevencie proti kybernetickým útokom sú aj Bug Bounty programy. Základnou myšlienkou týchto programov je verejná súťaž v hľadaní chýb a ich hlásenie organizáciám zapojených v týchto programoch s odmenou za objavené kritické zraniteľnosti. Proti penetračným testom sú porovnateľné v rozsahu. Ich cieľom je však objaviť akúkoľvek kritickú zraniteľnosť. Jednej organizácii sa môžu v takomto programe venovať desiatky súťažiacich testerov. Súťaživosť o odmeny tak slúži ako motivácia k odhaleniu a nahláseniu čo najväčšieho množstva zraniteľností [3].

1.2 Prečo penetračné testovanie?

Hlavným cieľom penetračného testovania je odhalenie zraniteľností v systéme skôr ako ich odhalí reálny útočník. Testy prebiehajú v kontrolovanom prostredí, teda nedochádza pri nich k deštrukcii dát, úniku tajných informácii alebo iných aktív. Jedná sa teda o prínosnú a preventívnu aktivitu firiem, spoločností, organizácii a pod.

Z pohľadu firmy je penetračný test investíciou do bezpečnosti, bez ktorej hrozí nebezpečenstvo úniku dát, ktoré môže stáť firmu v konečnom dôsledku o veľa drahšie, alebo v niektorých prípadoch firmu aj zbankrotovať. Je v záujme firmy chrániť dáta a procesy spojené so zamestnancami, zákazníkmi, akcionármi alebo si zachovať dobrú reputáciu.

Kybernetické útoky na firmy vyvíjajú finančnú záťaž spojenú so zotavením sa z útoku, zníženú efektivitu v čase počas a po útoku, čo sa prejavuje ušlým ziskom.

Penetračný test dokáže tieto hrozby včas odhaliť a znemožní útočníkom využiť zraniteľnosti v systéme, čím predíde finančným nákladom a stratám z útoku. Nemusí sa však jednať len o finančnú stratu, ale môžu byť napr. ohrozené životy, ako pri kybernetických útokoch cielených na nemocnice.

V neposlednej rade penetračný test taktiež vyhodnocuje efektivitu už používaných bezpečnostných opatrení. V prípade, že je nejaký prvok zastaralý alebo chýba

úplne, test túto skutočnosť odhalí a poukáže na riešenie, napr. vylepšenia alebo zaobstarania nového prvku ochrany [2].

1.3 Typy penetračných testov

Penetračné testy môžu byť komplexné a zamerané na všetky zložky testovaného systému: software, hardware ale aj ľudí [2, 6, 7]. Na základe ich zamerania ich môžeme deliť na:

- testy mobilných aplikácii,
- testy sieťovej infraštruktúry,
- testy bezdrôtových sietí,
- testy fyzickej bezpečnosti,
- testy sociálneho inžinierstva na personál,

ďalej podľa toho, koľko informácii a zdrojov je testerovi poskytnutých k nahliadnutiu a použitiu od žiadateľa testu na:

- white box testovanie,
- black box testovanie,
- grey box testovanie,

alebo podľa toho, z ktorej strany infraštruktúry žiadateľa prebieha útok na:

- interné,
- externé.

1.3.1 Rozdelenie penetračných testov podľa zamerania

Testy mobilných aplikácii

Ako už z názvu vyplýva, pozornosť je venovaná mobilným aplikáciám. Mobilné aplikácie sa od klasických desktopových líšia v niektorých kľúčových oblastiach. Priestor na vznik problémov je hlavne vo vývoji mobilných aplikácii. Konkrétne v spôsoboch akým aplikácie ukladajú dáta, komunikujú s inými zariadeniami alebo aplikáciami, alebo vo využívaní slabých kryptografických API, okrem iných problémov ďalej spomínaných v OWASP Mobile Security Testing Guide [8].

Testy sieťovej infraštruktúry

Sú zamerané na odhalenie bezpečnostných problémov v použitých technológiách, návrhu alebo implementácii sietí používaných organizáciou. Cieľom testera je prienik do siete a následne vykonávať ďalšie útoky na pripojené zariadenia, ako administratívne počítače alebo doménový ovládač [9].

Testy bezdrôtových sietí

Súvisia s testami sieťovej infraštruktúry, no dôraz je kladený na routery, prístupové body, kryptografiu použitú na autentizáciu do siete alebo dosah bezdrôtového signálu mimo budovu organizácie.

Testy fyzickej bezpečnosti

V situácii, kedy zariadenia používajú najnovšie verzie softwaru, používajú správne kryptografické API a primitíva, sú správne nakonfigurované stále hrozí nebezpečenstvo, že sa útočník ocitne pri fyzicky nezabezpečenom zariadení. Môže sa jednať o nezamknuté serverovne, kancelárie, verejne prístupné ethernet porty do internej siete, ale aj o chýbajúce kamerové systémy na kritických miestach. Týmto sa zaoberajú fyzické penetračné testy.

Testy sociálneho inžinierstva

Poslednou, často prehliadanou zložkou systémov, sú ľudia, ktorý v nich pracujú. Administrátori, HR, sekretárky, zamestnanci v kanceláriách. Sociálne inžinierstvo útočí práve na nich. Cieľom je znova neoprávnený prístup, získanie hesiel, informácii, alebo iných aktív, za využitia telefonátov na linky podpory, alebo návštevy v prevlečení, pohyb po priestoroch s počítačmi a hľadanie hesiel napísaných na lístočkoch, tabuliach, papieroch na stoloch alebo hľadanie odomknutých počítačov.

1.3.2 Rozdelenie penetračných testov podľa prístupu útočníka

Black box test

Pri black box testovaní sa postupuje spôsobom, akým by postupoval reálny útočník. Jediné dostupné informácie o cieli, ktoré je schopný využiť sú tie, ktoré dokáže sám nájsť. Nevychádza teda zo žiadnych dopredu odhalených informácii a musí začať od nuly.

White box test

Kontrastom k black box prístupu je white box testovanie. V tomto prípade je testerovi vysvetlené ako systém funguje, sú mu poskytnuté zdrojové kódy aplikácií, a útok je tým pádom cielenejší a rozsiahlejší. Organizácia bližšie spolupracuje s testerom.

Grey box test

Prienik vyššie zmienených prístupov je grey box test, ku ktorému dochádza ak je testerovi odhalená napr. len časť zdrojového kódu. Zvyčajne začína ako black box

test, čiže na slepo, ale v niektorej fáze útočník získa prístup napr. k archívom, zálohám zdrojových kódov alebo ich častí.

1.3.3 Rozdelenie penetračných testov podľa polohy útočníka v testovanej infraštruktúre

Externé testovanie

Tento typ testu kladie dôraz na počiatočnú pozíciu testera. V prípade externého testu, je tester v pozícii útočníka mimo sieť organizácie.

Jeho prvým cieľom je prienik do internej siete, kedy sa povaha testu transformuje na interný test. Útok je založený na skenovaní periférii siete organizácie, konkrétne skenovaním portov, testovaním webových aplikácií prístupných verejnosti, použitá môže byť aj phishingová kampaň alebo iné formy sociálneho inžinierstva.

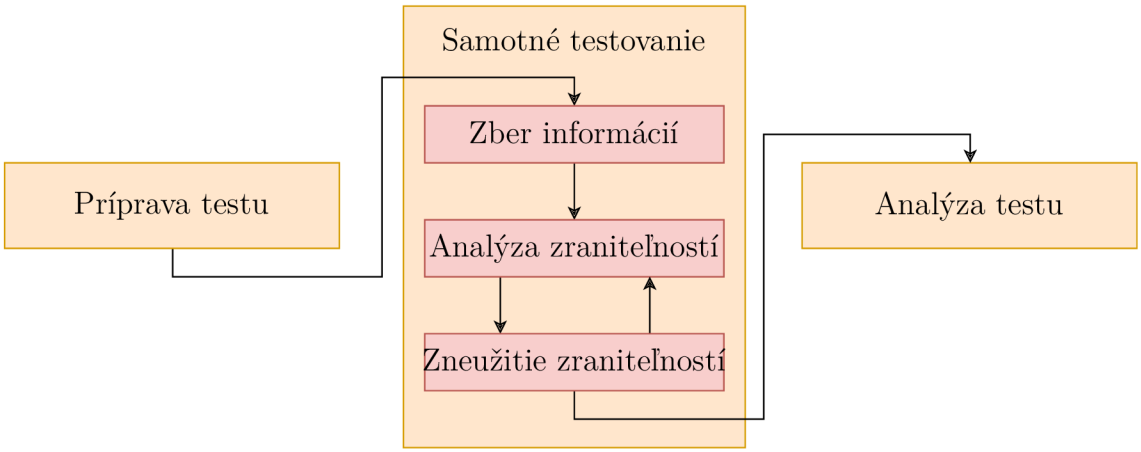
Interné testovanie

Interné penetračné testy slúžia k odhadu rozsahu prístupu, ktorý je útočník schopný získať po úspešnom prieniku do internej siete. V spojení s externým testom, môže útočník ďalej využiť informácie nadobudnuté počas externého testu. Alternatívou je použitie zariadení vnútri siete, napr. USB (Universal Serial Bus) kľúčov s malwarom.

Útok typicky začína získavaním informácii o sieti, teda aké zariadenia sa v sieti nachádzajú, aké služby na nich bežia a pod. Ďalej sa tester sústredí na hľadanie a zneužitie nájdených zraniteľností. Bežným cieľom interného testu je získanie administrátorského prístupu k doménovému kontroléru, prípadne získanie aktív definovaných v príprave na test.

1.4 Fázy penetračného testu

Všetky uvedené typy testov sa riadia rovnakou štruktúrou. Testy nie sú náhodné, ani plne automatizované a ich výsledok musí byť precízny, preto sa riadia metodológiami, teda overenými súbormi krokov, ktoré popisujú ako pri nich postupovať. Penetračný test je z časového hľadiska rozdelený na tri štádiá alebo fázy: na úplnom začiatku príprava na test, nasledovaná samotným testom a výslednou analýzou testu. V prvej fáze sa zákazník dohodne s vykonávateľom na podmienkach, ktoré musia byť dodržané, rozsahu testu, bodoch, na ktoré má byť vynaložené väčšie sústredenie, spíše sa dokument obsahujúci tieto informácie, a pod. Samotný penetračný test pozostáva z piatich hlavných krokov. Sú nimi: prieskum, mapovanie prostredia, odhalenie zraniteľností, využitie zraniteľností a výsledná správa [2, 7]. Ďalej je podrobnejšie popísaný priebeh testu. Priebeh testu je znázornený aj na Obr. 1.1.



Obr. 1.1: Fázy penetračného testu.

Prieskum

V prvej fáze testu si útočník zhromažďuje voľne dostupné informácie o cieli útoku ako použité technológie a ich verzie, IP adresy serverov, doménové mená, subdomény, zmienky developerov na fórach, uniknuté informácie dostupné na internete a pod.

Mapovanie prostredia

V tejto fáze sa útočník zameriava na utvorenie prehľadu funkcionality testovaného objektu, napr. webovej aplikácie. Vytvára si účet, kliká na odkazy, skúša meniť nadstavenia účtu, písanie komentárov, vkladanie vecí do nákupného košíka, odhlasovanie, prihlasovanie, reset hesla a pod.

Odhalenie zraniteľností

Útočník využije všetky doposiaľ nadobudnuté informácie a poznatky o fungovaní aplikácie k tomu, aby odhalil čo najväčšie množstvo zraniteľností manuálne ale aj za pomoci automatizovaných nástrojov. Tie v ďalšom kroku použije na získanie prístupu hlbšie do systému.

Využitie zraniteľností

Táto fáza predstavuje testovanie nájdených slabín v predchádzajúcom kroku na preniknutie do systému. Po úspešnom prieniku vie tester konečne odhadnúť, k čomu sa dá zraniteľnosť využiť.

Výsledná správa

Výstupom penetračného testu je výsledná správa (report). Obsahuje informácie o nájdených zraniteľnostiach, proces ich využitia, reprodukciu útoku, spôsob akým sa dá útok zneužiť a informácie o tom ako útoku zabrániť.

1.4.1 Používané nástroje

V Každej fáze si tester musí zvoliť z rady dostupných nástrojov tie najvhodnejšie. V prípade testovania webovej aplikácie sú najčastejšie potrebné nástroje na skenovanie cieľa, crawlery, fuzzery, nástroje na zachytávanie HTTP výziev a odpovedí alebo nástroje na enumeráciu podadresárov nachádzajúcich sa na serveri [6].

2 Analýza dostupných nástrojov

Táto časť práce sa venuje dostupným nástrojom, ktoré najčastejšie používajú penetrační testeri, hľadači zraniteľností a rovnako aj členovia Red Teamov. Existuje však veľké množstvo nástrojov, ktoré majú využitie v rôznych odvetviach penetračného testovania, preto budú v tejto časti spomenuté skôr nástroje zamerané na penetračné testovanie webových aplikácií [6].

Pri hľadaní zraniteľností je fáza hľadania informácii o cieli základom. Preto väčšina nástrojov v Tab. 2.1 je používaná práve k tomu účelu. Takisto je dôležité podotknúť, že niektoré nástroje majú viacero funkcií a nie sú ľahko kategorizovateľné [13].

Okrem týchto nástrojov sú však používané aj mnohé iné z rôznych dôvodov, napr. preferencie testera, použitie online nástrojov alebo rozšírení v prehliadačoch. Cieľom tejto kapitoly nie je výpis všetkých existujúcich nástrojov, ktoré sa dajú použiť na účely penetračného testovania, ale nástroje, ktoré sú uvedené, sú známe a často využívané napriek tomu, že skoro každý z nich má alternatívy rozličných kvalít. Avšak, na základe týchto informácií je zrejmé, že penetrační testeri pracujú s veľkým množstvom nástrojov. Pre šetrenie času je ideálne repetitívne úkony automatizovať. Rozmanitosť nástrojov však predstavuje problém komunikácie, alebo predávania dát medzi nimi.

Z tabuľky vyplýva, že integrácia HTTP proxy nástrojov môže byť problematická hlavne v dôvodu, že nemá rozhranie príkazového riadku. Nástroje, ktoré pracujú v textovej forme sú ľahšie automatizovateľné. Preto sú v tabuľke 2.2 bližšie porovnané nástroje typu HTTP proxy.

Táto tabuľka poukazuje na viacúčelovosť nástrojov BurpSuite a OWASP ZAP oproti klasickým HTTP proxy ako mitmproxy. Navyše od bežného zachytávania a modifikácie dát v reálnom čase sú viac zamerané na použitie pri penetračnom testovaní. To nasvedčuje aj skutočnosť, že oba nástroje navyše obsahujú aj skenery zraniteľností a možnosť spideringu stránok. V tabuľke 1 však boli spomenuté aj iné skenery zraniteľností, ale problém týchto skenerov spočíva v ich plnej automatizácii a jedno účelovosti. BurpSuite a OWASP ZAP umožňujú testerovi väčšiu interaktivitu a spájajú v sebe niekoľko funkcií na viac účelov v rôznych fázach testovania. Okrem výhody možnosti manuálneho pracovania so žiadosťami a kontrolou odpovedí majú však jeden problém. Neexistuje ľahká cesta ako z nich získať dáta.

Podobnosti medzi nástrojmi BurpSuite a OWASP ZAP sú zrejmé, no napriek tomu má každý svoju unikátnosť. Obidva nástroje sú predinštalované v operačných systémoch Kali Linux a ParrotOS zameraných na penetračné testovanie [10]. Vlastnosti OWASP ZAP a BurpSuite sú porovnané v Tab. 2.3 [11, 12].

Z tabuľky je vidno že oba nástroje sú skutočne podobné. Dá sa hovoriť o substitú-

Tab. 2.1: Prehľad vybraných používaných nástrojov.

Nástroj	**Funkcia**	**CLI**
Nmap	skener portov, detekcia OS, detekcia služieb	Áno
Wafw00f	detekcia firewallov a loadbalancerov	Áno
Nikto	skener zraniteľností	Áno
Wapiti	skener zraniteľností	Áno
hping	skener portov, detektor firewallov, detekcia OS	Áno
Masscan	skener portov	Áno
Zmap	skener portov	Áno
sqlmap	detekcia a exploitácia databázových zraniteľností	Áno
dirb	objavenie obsahu web serveru na základe slovníka	Áno
dirbuster	objavenie obsahu web serveru použitím brute force	Nie
BurpSuite	sken zraniteľností, HTTP proxy, brute force formulárov, spidering obsahu	Nie
OWASP ZAP	HTTP proxy, spidering obsahu, sken zranniteľností, fuzzer	Nie
hydra	online útoky na prihlasovacie údaje, napr. SSH	Áno
Metasploit Framework	skener zraniteľností, spúštač exploitov	Nie
Nessus	skener zraniteľností	Nie
OpenVAS/ Greenbone	skener zraniteľností	Nie
mitmproxy	HTTP proxy	Áno

Tab. 2.2: Porovnanie HTTP proxy nástrojov.

Nástroj/ Funkcia	**BurpSuite Professional**	**OWASP ZAP**	**mitmproxy**
zachytávanie HTTP komunikácie	Áno	Áno	Áno
modifikácia dát v reálnom čase	Áno	Áno	Áno
CLI	Nie	Nie	Áno
Analýza žiadostí	Áno	Áno	Nie
Spidering	Áno	Áno	Nie

Tab. 2.3: Porovnanie OWASP ZAP a BurpSuite pro.

Funkcia/ Nástroj	**CLI**	**API na vývoj modulov**	**Vývoj**	**Cena**	**Jazyk**
OWASP ZAP	Nie	Áno	Komunita	Zadarmo	Java
BurpSuite professional	Nie	Áno	Proprietárny	399€/rok	Java

toch, ale napriek tomu existujú faktory, ktoré môžu pomôcť pri výbere. Pre väčšinu užívateľov je ním cena, kedy je preferovaný OWASP ZAP oproti BurpSuite, pretože poskytuje možnosť použitia web crawlera a obsahuje aktívny skener zraniteľností, ktorý je v BurpSuite dostupný až v Pro edícii. Vzhľadom na to, že ZAP je komunitným projektom, jeho dokumentácia nie je úplná. Na rozdiel od OWASP ZAP, BuprSuite poskytuje väčšiu stabilitu. Z pohľadu vývoja modulov je tiež preferovaný BurpSuite, pretože má lepšiu dokumentáciu.

2.1 Nástroj Burp Suite

V bezpečnostnej komunite je nástroj BurpSuite vyvíjaný firmou PortSwigger veľmi známy. Jeho najčastejšie využitie je takzvaná odpočúvacia webová proxy, teda program ktorý stojí v pomyselnej infraštruktúre webovej komunikácie medzi klientom, v najčastejšom prípade prehliadačom, a vzdialeným serverom a prechádza ním celá komunikácia aplikačnej vrstvy ISO/OSI modelu.

Program BurpSuite je vyvíjaný v troch edíciách, menovite Community, Pro a Enterprise. Tabuľka 2.4 poskytuje prehľad funkcionality každej z edícií [12].

Tab. 2.4: Prehľad funkcií edícií BurpSuite.

Edícia	**Enterprise**	**Pro**	**Community**
Funkcionalita / Cena	3,499€/rok	349€/rok	Neplatené
Aktívny skener zraniteľností	Áno	Áno	Nie
Plánovač a opakovanie skenov	Áno	Nie	Nie
Škálovateľnosť	Áno	Nie	Nie
CI integrácia	Áno	Nie	Nie
Pokročilé manuálne nástroje	Nie	Áno	Nie
Základné manuálne nástroje	Nie	Áno	Áno

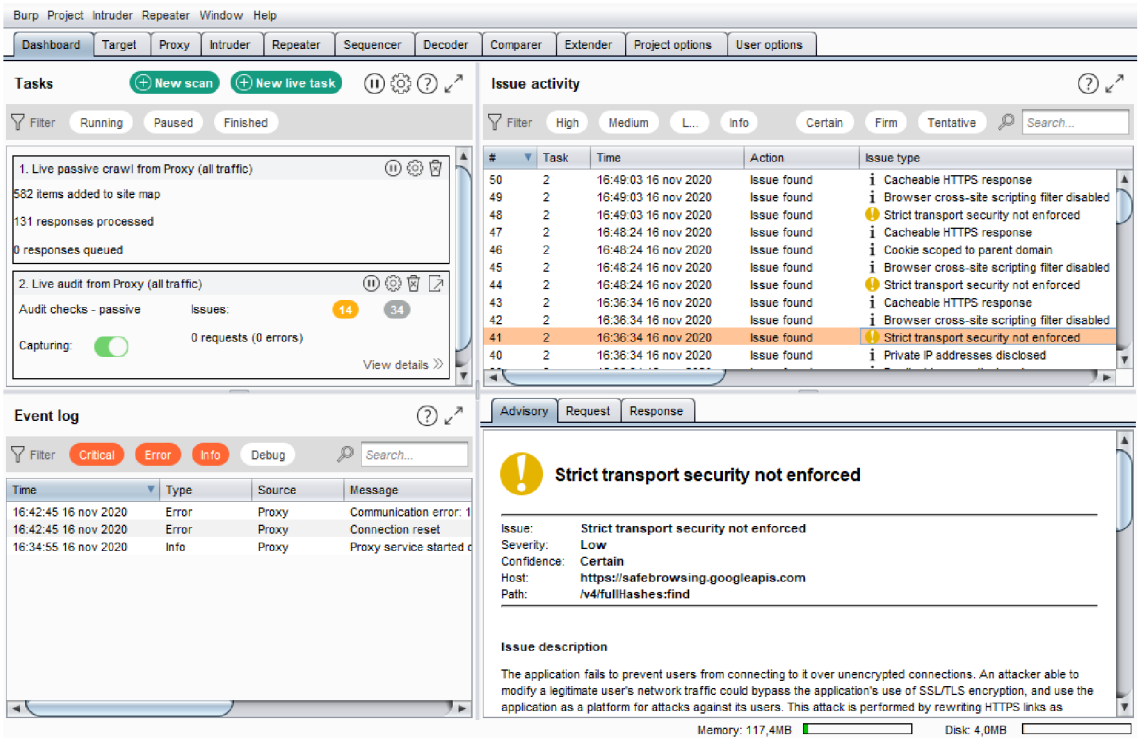
2.1.1 Užívatelské rozhranie

Táto sekcia je zameraná len na Pro a Community edície programu BurpSuite. Pri spustení programu je užívateľ ako prvé vyzvaný ku konfigurácii projektu. V Community verzii je dostupný len dočasný projekt (temporary project). To znamená, že si užívateľ nemôže uložiť výsledky práce na disk. Táto služba je však dostupná v Pro edícií, spolu s načítavaním projektov. Po kliknutí na tlačidlo „Next“ je užívateľovi poskytnutá šanca načítania vlastnej konfigurácie, inak je použitá vstavaná. Po tomto kroku kliknutím na tlačidlo „Start BurpSuite“ sa program spustí.

Pracovné okno na hlavnej obrazovke - dashboard, je rozdelené na štyri sekcie. Ľavé horné okno s názvom Tasks informuje užívateľa o stave skenov. Pod ním vľavo dole je Event Log. Ten ukazuje napr. chybové hlásenia, objavené zraniteľnosti, načítanie alebo odpojenie rozšírení, problémy s pripojením na vzdialený servera pod. Na pravej strane je sa nachádza okno Issue activity, ktoré detailne popisuje nájdené zraniteľnosti. Po kliknutí na niektorý z nálezov sa v pravom dolnom okne vypíšu informácie o náleze a samotná HTTP žiadosť alebo odpoveď, v ktorej bola zraniteľnosť nájdená. V Community edícii je pravá strana v podstate reklamou na Pro verziu.

V hornej časti obrazovky sa nachádza niekoľko záložiek s vstavanými modulmi: Dashboard, Target, Proxy, Intruder, Repeater, Sequencer, Decoder, Comparer, Extender, Project opinions a User options. Užívateľské rozhranie je znázornené na Obr. 2.1.

Dáta, ktoré pomocou BurpSuite môžeme sledovať a napríklad pred ich odoslaním upravovať, sú HTTP výzvy a odpovede. Tie sú v procese penetračného testovania sledované a analyzované pretože obsahujú hlavičky, cookies, php parametre v url a podobné informácie o funkcionalite cieľa. Všetky z týchto parametrov sú pre penetračné testovanie zaujímavé a nástroj Burp umožňuje ľahký prístup a jednoduché pracovanie s nimi. To však nie je jediné čo ponúka. Už aj v Community verzii sa



Obr. 2.1: GUI ukážka.

nachádza množstvo ďalších funkcionalít. Nasledujúca časť popisuje funkcionalitu vstavaných modulov [12, 13, 14].

Target

Modul Target automaticky vytvára stromovú štruktúru navštevovaných stránok. Pre prehľadnosť môžeme obmedziť rozsah mapovania len na konkrétnu doménu. Rovnako umožňuje prehľad HTTP aj SOCKS histórie.

Intruder

umožňuje vybrať miesta v HTTP výzve, na ktoré ďalej aplikuje útok. Intruder použije nami definovaným set payloadov, ktorý je modifikovateľný a môže ním byť napr. slovník, krátky zoznam, jediný prvok alebo .txt súbor. K dispozícii sú štyri typy zabudovaných útokov:

- Sniper – používa jediný set payloadov. Každý vyznačený parameter, na ktorý sa má payload aplikovať, sa v jednej výzve aplikuje len raz. Tento typ útoku je vhodný ak sa zameriavame na jeden konkrétny parameter. Počet žiadostí generovaných týmto typom útoku je počet označených parametrov krát počet prvkov použitého setu.

- Battering Ram – rovnako používa len jeden set. Líši sa v tom, že do všetkých označených parametrov vloží prvok setu naraz. Použiteľný v prípade, kedy treba zadať rovnakú hodnotu do viacerých parametrov. Počet žiadostí generovaný týmto útokom je rovnaký ako počet prvkov v použitom sete.
- Pitchfork – tento typ útoku používa dva sety. Priebeh je nasledovný: v prvej žiadosti je použitý na mieste parametru jedna prvý prvok zo setu 1, na mieste parametru dva prvý prvok zo setu 2. Ďalšia žiadosť použije v prvom parametre prvok zo setu 1, ale v druhom tento krát prvok číslo dva zo setu 2. Počet žiadostí je rovný počtu prvkov v najmenšom sete. Tento útok je vhodný napr. ak je potrebné posielať rovnaké užívateľské meno, ale zároveň meniť iný parameter.
- Cluster Bomb – používa dva sety. Je vhodné ho použiť ak hádame dva parametre, ktoré nemusia spolu súvisieť sú neznáme alebo jednoducho treba hádať dva parametre naraz. Útok funguje tak, že pre každý prvok zo setu 1 bude použitý prvok zo setu 2, potom sa posunie na prvok číslo dva v sete 1, a otestuje ho znova s každým prvkom zo setu 2 a tak ďalej. Počet žiadostí je počet prvkov setu 1 krát počet prvkov setu 2.

Repeater

Ďalší veľmi praktický modul, ktorý umožňuje opakovane odoslať na server HTTP výzvu alebo WebSocket správu, ktorú chceme manuálne viac krát upraviť a sledovať odpoveď serveru. V repeateri môže byť uložených viacero HTTP požiadaviek, každá vo vlastnom okne. Okná sú číslované, ale je možné ich aj pomenovať. Každá záložka má aj vlastnú históriu požiadaviek, v ktorej sa dá pohybovať pomocou tlačidiel označených „<“ a „>“.

Decoder

Tento modul je skôr utilitou. Slúži na rýchle dekódovanie reťazcov, napr. z Base64, URL, hex a iných kódovaní často používaných vo webových aplikáciách. Taktiež dokáže pracovať s hash funkciami. Okrem dekódovania dokáže dáta aj kódovať. Dáta sa do Decoderu dajú odoslať označením v HTTP žiadosti, kliknutím pravým tlačidlom a kliknutím na „Send to decoder“ alebo klasickým kopírovaním. Po vložení dát a kliknutím na decode/encode sa zjaví nové okno s transformovanými dátami, čo umožňuje jednoduché viacnásobné transformácie.

Comparer

Ako už z názvu vyplýva, jedná sa o porovnávač. Porovnáva akékoľvek dve položky dát. Jedným z možných využití je, porovnanie HTTP odpovedí po neúspešnom pri-

hlásení na server a zrovnanie s odozvou obdržanou po úspešnom prihlásení. Ďalší príklad je testovanie slepej SQL injekcie, kedy môžeme pozorovať zmeny v odpovediach po použití rozdielnych vstupov. Dáta sa dajú do Compareru načítať kopírovaním, načítaním zo súboru alebo odoslaním z iných modulov pomocou označenia, kliknutia pravým tlačidlom a voľbou „Send to Comparer“.

Sequencer

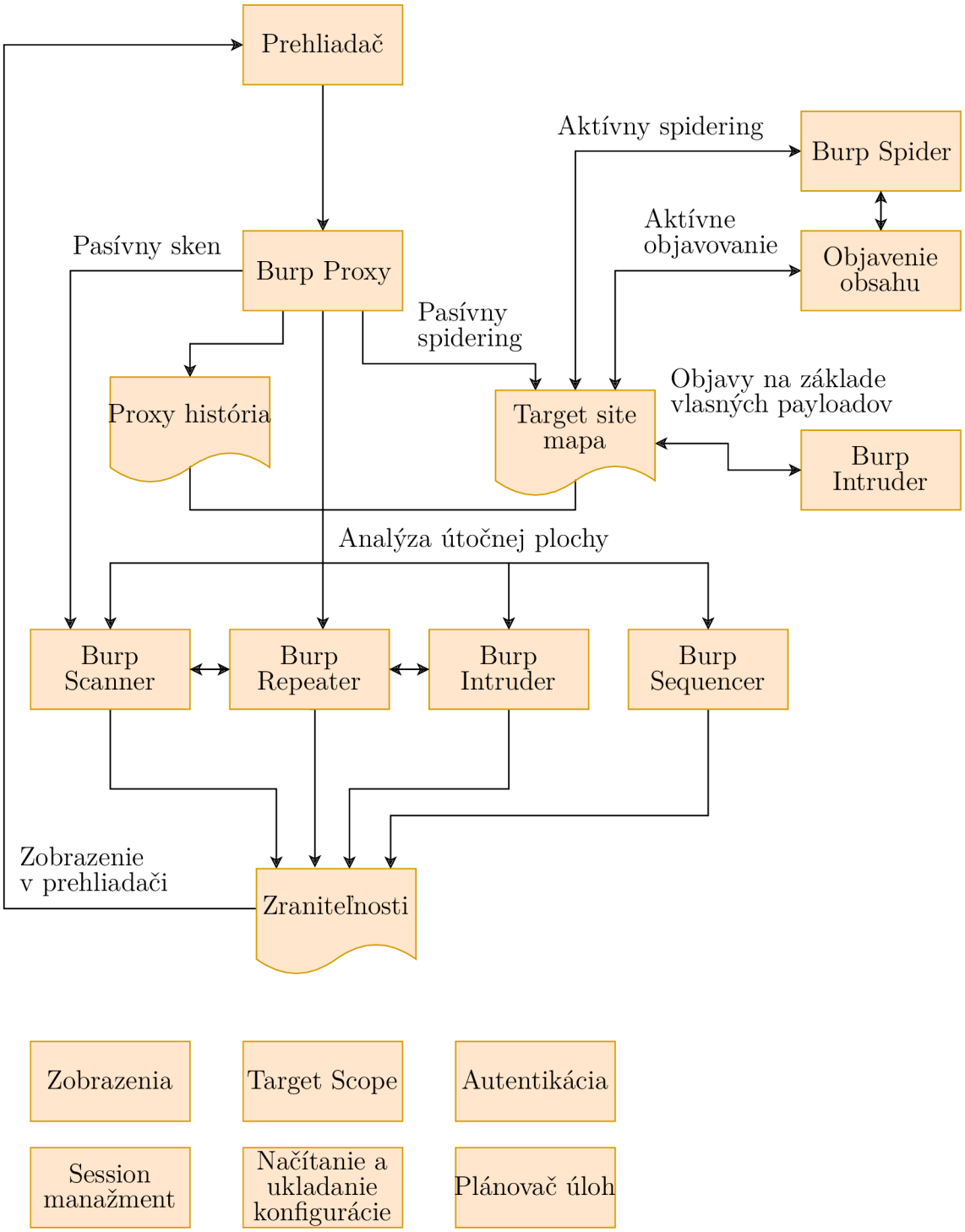
Je analytický nástroj testujúci živý alebo zachytený prúd komunikácie medzi užívateľom a serverom a hľadá napríklad predpovedateľné sekvencie v anti-CSRF tokenoch, cookies, a pod.

Extender

Tento modul ponúka užívateľom načítať vlastné zásuvné moduly. Okrem nadstavení sa v ňom nachádza aj output modulov. Pre Pro edíciu sa tu taktiež nachádza zoznam oficiálne podporovaných rozšírení v tzv. BApp store.

2.1.2 Tok dát v BurpSuite

Tok dát v nástroji BurpSuite je popísaný obrázkom 2.2. Dáta prichádzajú typicky z prehliadača. Modul Proxy zachytáva HTTP komunikáciu a automaticky skenuje všetky HTTP žiadosti aj odpovede. Následne sa dáta z žiadostí presunú do modulu Target a začnú tvoriť stromovú štruktúru domény. Site mapy sú aktualizované aj pomocou nástroja Discover content, ktorý hľadá skryté priečinky alebo iný obsah, ku ktorému sa bežný užívateľ nedostane. Ďalej pomocou spideringu, alebo na základe užívateľom definovaných payloadov v module Burp Intruder. Dáta z modulu Proxy aktualizujú aj Proxy históriu. Z modulov Target a Proxy je možné dáta dalej posielať do modulov Scanner, Repeater, Intruder a Sequencer. Všetky nájdené zraniteľnosti sú uložené v BurpSuite, a je možné ich skontrolovať v prehliadači. Moduly, ktoré slúžia skôr na konfiguračné účely a priamo nepracujú s HTTP žiadosťami sú uvedené samostatne pod diagramom [12, 13].



Obr. 2.2: Diagram BurpSuite toku.

2.2 Zásuvný modul

Pod pojmom zásuvný modul si môžeme predstaviť kus softwaru, ktorý rozširuje funkcionalitu programu, ktorého bude súčasťou bez toho aby bol súčasťou jeho jadra. Moduly komunikujú s jadrom programu pomocou verejného API rozhrania. To dáva komunite užívajúcej program príležitosť tvoriť praktické rozšírenia aj bez zverejnenia celého zdrojového kódu hlavného programu. Užívatelia ďalej získavajú príležitosť vybrať si, ktoré moduly si stiahnu a tak si môžu upraviť funkcionalitu presne podľa svojich potrieb. Vývojári jadra aplikácie nemusia nijako moduly integrovať do zdrojového kódu a môžu sa sústrediť na vylepšovanie jadra programu.

2.2.1 Maven

Aplikácie písané v Jave je možné pripraviť na spúšťanie a zdieľanie pomocou nástrojov, ktoré tento proces uľahčujú. Prvým obecne známym z nich je Apache Ant – knižnica, alebo nástroj, ktorý prekladá (build) Java aplikácie. Obsahuje niekoľko vstavaných úloh, napr. kompiláciu kódu, testovanie alebo spúšťanie Java aplikácií. Druhým je Maven, taktiež zastrešený pod Apache software foundation. Maven je nástroj na manažovanie a prekladanie Java projektov. Maven poskytuje vývojárom jednoduchý systém prekladania, zavádza do projektov množstvo užitočných informácii a jeho najväčšou výhodou je jednotný proces prekladania založený na princípe xml súboru nazývaného POM (project object model). Tento súbor obsahuje informácie o projekte a konfiguráciu, ktoré budú použité na build aplikácie. Medzi ne patrí napr. adresár, do ktorého sa aplikácia uloží, zdrojový adresár projektu, adresár s testovacími funkciami, názov projektu, verziu projektu, popis a hlavne knižnice na ktorých je projekt závislý (dependencies). Systém referencii na knižnice je zrejme najcennejšou vlastnosťou Mavenu [15].

2.3 Vývoj modulov v BurpSuite

Okrem základných modulov je však ako naznačuje modul Extender možné BurpSuite rozširovať pomocou verejného API. Komunita má k dispozícií možnosť vyvíjať moduly v jazykoch Java, Python a Ruby. Vzhľadom na to, že táto práca je zameraná na vývoj rozšírení BurpSuite v jayzku Java, sa ďalej o možnostiach rozšírenia v jazykoch Ruby a Python nepíše. Integrácia zásuvných modulov vyvíjaných v jazyku Java prebieha pomocou načítavania .jar súborov v zabudovanom module Extender [12]. Vďaka pestrému API rozhraniu ponúka vývojárom nasledujúce možnosti:

- spracovávať a prepisovať HTTP výzvy a odpovede vo všetkých Burp moduloch

- prístup k proxy histórií, zoznamu nálezov skeneru, sitemapy cieľov
- spúšťanie skenerov a spiderov
- pridávanie vlastných skenovacích kontrol a registráciu vlastných hlásení nájdených zraniteľností
- vkladanie vlastných miest na vloženie útokov v skenovaných výzvach
- používanie vlastných payloadv v module Intruder
- pridávanie, odoberanie prvkov v module Target
- prístup ku cookie jaru a metódy jeho úprav
- implementovať vlastné akcie na správu nadviazaných spojení (session)
- analyzovať HTTP výzvy a odpovede a získavať z nich cookies, parametre, hlavičky a podobné
- pridávať vlastné záložky a kontextové menu a štítky (tab)
- používať HTTP textový editor správ
- rozširovať vstavaný HTTP editor o nové dátové formáty, ktoré nie sú podporované
- posielanie HTTP žiadostí a získavanie odpovedí na ne
- zmenu konfiguračných súborov
- ukladanie stavu BurpSuite

2.3.1 Formát modulu

Ako už bolo spomenuté, modulom je `.JAR` (Java Archive) súbor. BurpSuite vyžaduje, aby moduly obsahovali zložku s názvom „burp“, ktorej obsahom je trieda s názvom „BurpExtender.class“. Okrem toho môže tento `.jar` súbor obsahovať aj svoje závislosti ak bol použitý balíkový manažér ako Maven. V opačnom prípade sa môžu nachádzať priamo uložené niekde na disku, štandardne v OS Windows v zložke `C:\Users\User\.m2` a v Linuxových distribúciach `~/.m2`. V BurpSuite je potom potrebné v záložke Extender/Options/Java Environment zadať cestu k týmto závislostiam. Najjednoduchšia štruktúra modulu za použitia balíčkového manažéra Maven je znázornená v príklade 2.3.

```
modul.jar
├── burp
│   └── BurpExtender.class
└── pom.xml
```

Obr. 2.3: Formát modulu.

Kde:

- `modul.jar` je menom `.JAR` archívu. Toto meno sa nezobrazuje v BurpSuite. miesto toho sa zobrazuje meno rozšírenia definované priamo v zdrojovom kóde volaním metódy `setExtensionName`.
- Zložka `burp` reprezentuje java balíček s rovnakým názvom. Je vyžadovaná konvenciou BurpSuite.
- `BurpExtender.class` je hlavnou triedou obsahujúcou zdrojový kód modulu. Jej meno musí byť `BurpExtender.class`.
- `pom.xml` predstavuje súbor vygenerovaný balíčkovým manažérom Maven. Obsahuje informácie o všetkých použitých závislostiach a knižniciach použitých pri tvorbe modulu a jeho spúšťaní.

Trieda `BurpExtender` musí implementovať rozhranie `IBurpExtender` a metódu `registerExtenderCallbacks`, ktorá zabezpečuje načítanie modulu do BurpSuite jadra [12].

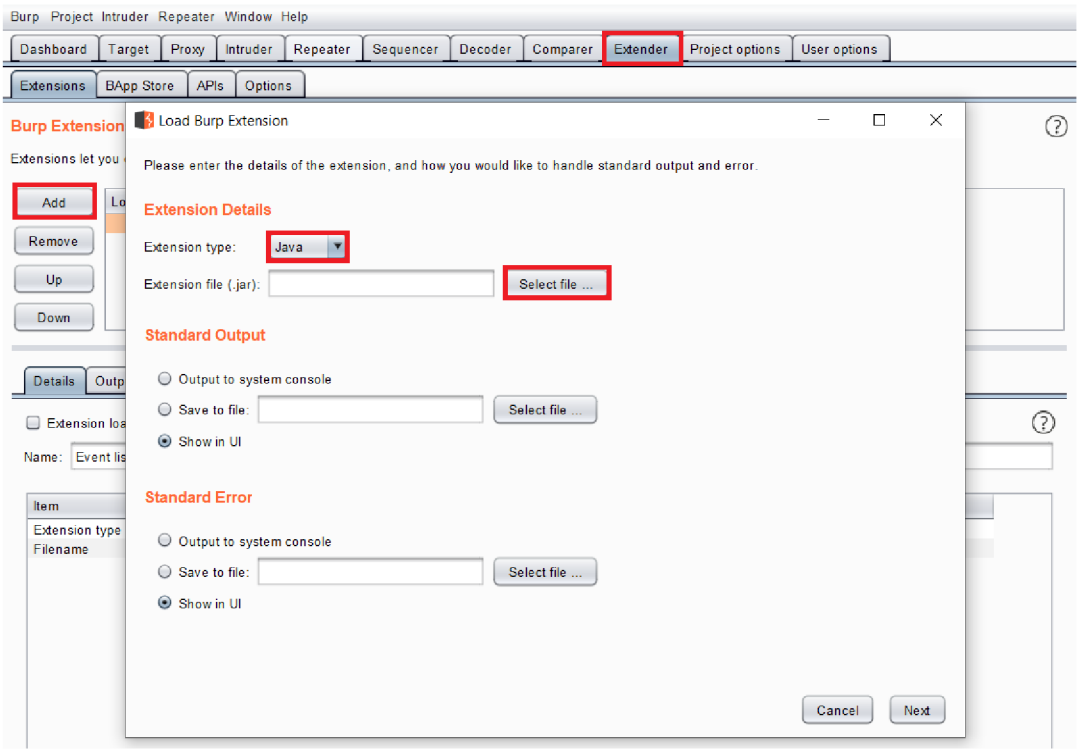
Výpis 2.1: Hlavná metóda modulu.

```
package burp;
public class BurpExtender implements IBurpExtender
{
    public void registerExtenderCallbacks (IBurpExtenderCallbacks
        callbacks)
    {
        // kód modulu
    }
}
```

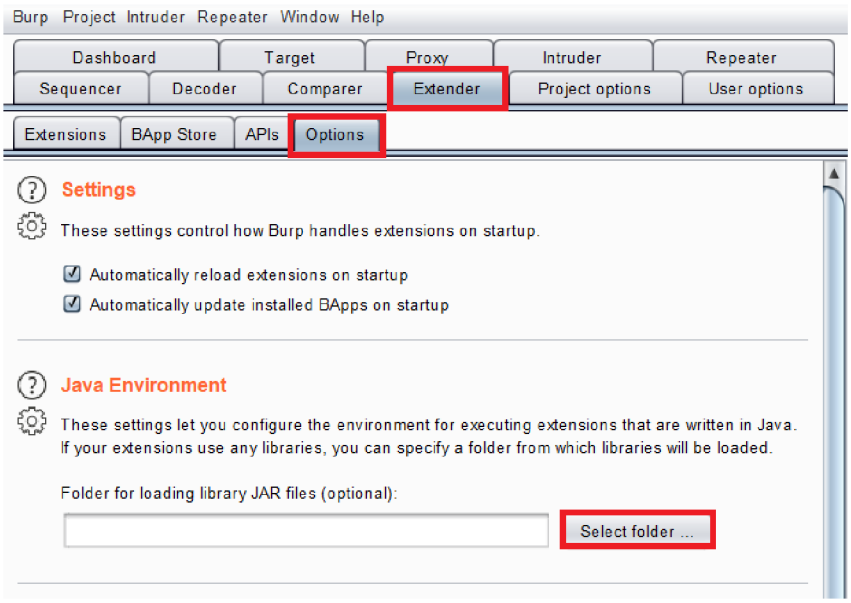
2.3.2 Načítavanie modulu v BurpSuite

Načítanie modulov sa vykonáva priamo v užívateľskom rozhraní nástroja BurpSuite. V záložke Extender sa nachádza okno s prehľadom modulov. Ak je okno prázdne, kliknutím na tlačidlo „Add“ sa otvorí nové okno. V tomto okne je treba vybrať jazyk, v ktorom je modul vyvíjaný a cestu k nemu pomocou tlačidla „Select File“ [13]. Rozhranie je zobrazené na Obr. 2.4. Ďalej je možné zvoliť, kde bude zobrazovaný výstup a chybové hlásenia modulu.

BurpSuite podporuje aj načítavanie externích knižníc. V prípade ak nie je využitý balíčkový manažér ako Maven, je možné zložku s knižnicami definovať v podzáložke Options a pre programovací jazyk Java v sekcii Java Environment použitím tlačítka „Select Folder“. Okná sú znázornené na Obr. 2.5.



Obr. 2.4: Pridanie modulu.



Obr. 2.5: Pridanie knižníc.

3 Požiadavky na funkcionalitu modulu

Cieľom práce je vytvorenie modulu, ktorý umožní integráciu a spoluprácu BurpSuite s inými automatizovanými nástrojmi. Modul má slúžiť ako brána k vnútornej funkcionalite BurpSuite API. Modul má dve úlohy:

- prijímať HTTP žiadosti obsahujúce dáta, ktoré majú byť zaradené do modulov (Repeater, Spider, Scanner atp.) BurpSuite pre ďalšiu manuálnu prácu,
- odpovedať na požiadavky o extrakciu dát, napr. proxy histórie, stav skeneru a pod.

Ako bolo spomenuté v predošlej kapitole, BurpSuite API poskytuje možnosti pracovania so zachytenými žiadosťami. Funkcionalita tohto modulu má umožňovať automatizovaným nástrojom prístup k dátam, ktoré BurpSuite obsahuje, napr. k Proxy histórii a odosielanie žiadostí do BurpSuite na ďalšie spracovanie. Prístup k dátam má byť realizovaný pomocou URL, v ktorej bude ďalej definované aká akcia sa má vykonať.

Získavanie dát z BurpSuite

Vyvíjaný modul má za úlohu odosielanie dát iným programom. Jedná sa o dáta z proxy histórie, informácie o stave skeneru, informácie o tom, aké zraniteľnosti boli nájdené na určitej doméne, kompletné telá požiadaviek, všetky položky z automaticky vytvorenej site mapy vrátane ich subpoložiek, množsto odhalených zraniteľností, a pod.

Odosielanie dát nástroju BurpSuite

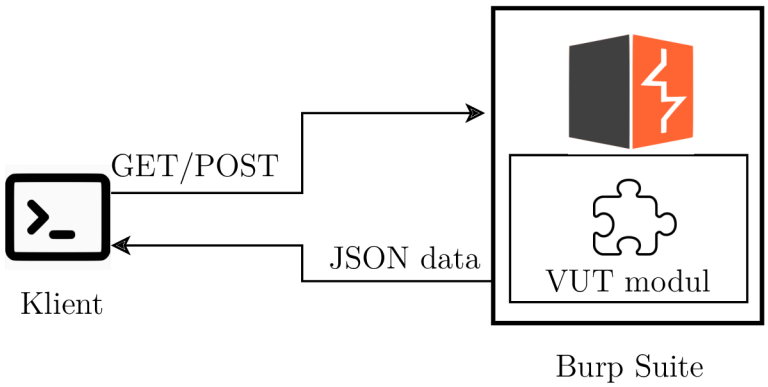
Požiadavky na tento modul zahŕňajú automatizované vkladanie HTTP žiadostí do modulov:

- Repeater
- Intruder
- Comparer
- Target
- Scanner
- Spider

Ďalšou požiadavkou na funkcionalitu je, aby modul bol schopný interagovať s tzv. Scopom (rozsahom) teda zoznamom domén, ktoré majú byť pasívne skenované a zaznamenávané do site mapy. Modul má zabezpečiť odstraňovanie a pridávanie prvkov na základe zadanej domény.

3.1 Návrh implementácie

O spracovanie dát k týmto účelom sa postará BurpSuite API. Avšak časť kódu BurpSuite, ktorá by dovoľovala vytvorenie serveru, alebo možnosť odpovedať na HTTP žiadosti nie je dostupná vo verejnom BurpSuite API. Nástroj BurpSuite dokáže generovať HTTP požiadavky, ktoré sú potrebné napr. na crawlovanie stránok, ale na tieto, ako aj všetky iné výzvy dostáva odpovede od vzdialených serverov. V žiadnej zo svojich akcií BurpSuite nevydáva odpovede na HTTP žiadosti. To znamená, že, súčasťou riešenia bude aj vytvorenie jednoduchého HTTP serveru. Navrhované riešenie je znázornené na Obr. 3.1.



Obr. 3.1: Návrh implementácie.

Štruktúra žiadosti uloženej v tele správy musí zodpovedať štruktúre objektov HttpService a HttpRequestResponse. Polia, ktoré žiadosť obsahuje sú znázornené vo výpise 3.1.

Výpis 3.1: Štruktúra JSON objektu.

```
{
  "comment": "",
  "request": "",
  "response": "",
  "highlight": "",
  "httpService": {
    "port": "",
    "protocol": "",
    "host": ""
  }
}
```

- comment – premenná obsahujúca nepovinný komentár, môže mať hodnotu null
- highlight - hodnota je použitá pre zvýraznenie žiadosti farbou definovanou užívateľom, akceptuje hodnoty red, orange, yellow, green, cyan, blue, pink, magenta, gray, alebo null
- request – samotná žiadosť, ktorá má byť posunutá do ostatných modulov BurpSuite. Žiadost musí byť kódovaná v Base64
- response – rovnako ako žiadosť musí byť v Base64 kódovaní
- host – celá URL žiadosti
- port – port, na ktorý má byť žiadosť odoslaná. Akceptuje hodnoty 80 a 443
- protocol – príznak použitého protokolu. Môže nadobúdať hodnoty http alebo https

Hodnoty, ktoré musia byť vyplnené sú host, port, protocol a request alebo response. Ostatné hodnoty môžu ostať prázdne, teda null.

Server bude spúšťaný zároveň spolu s BurpSuite a týmto rozšírením. Kedže BurpSuite slúži len ako prostredník medzi klientami a servermi, nebude tomu inak ani v tomto prípade. O získavanie a úpravu dát sa postará BurpSuite s využitím jeho API, ale na vytváranie samotných HTTP odpovedí je potrebný nami vytvorený server.

Všetky operácie modulu sú riadené pomocou kombinácie dotazovanej URL v tvare: `http://127.0.0.1/modul/akcia/parameter` a typom HTTP žiadosti. Teda napr. pre získanie sitemapy domény google.com bude použitá žiadosť typu GET na adresu `http://127.0.0.1/target/google.com`. Pre pridanie domény google.com do scope bude použitá metóda POST na adresu `http://127.0.0.1/target/google.com` ako je znázornené v Tab. 3.1. Pre pridanie žiadosti do modulu Repeater budú avšak dáta, ktoré majú byť pridané, v tele žiadosti. Dáta, teda HTTP žiadosti alebo odpovede, ktoré chceme posunúť do BurpSuite sú vo formáte JSON uložené v tele pôvodnej žiadosti.

3.1.1 Backend

O väčšinu práce ako spracovávanie a uchovávanie dát sa stará samotný Burp ale keďže BurpSuite API neposkytuje možnosť spúšťania, alebo iného pracovania s vnútorne použitým serverom, súčasťou modulu musí byť aj vlastný server pre spracovanie príchodzích požiadaviek a zabezpečenie správnych odpovedí. BurpSuite má síce zabudovaný server, ktorý sa stará o spúšťanie proxy poslucháčov či funkcionalitu Burp Collaborator. Vývojár modulov, ktorý má snahu o využitie podobných funkcií však musí rátať s tým, že implementácia podobného serveru je úplne v jeho kompetencii.

Pre prácu s webovými rozhraniami ponúka jazyk java množstvo frameworkov. Framework slúži ako základ, šablóna alebo kostra a ich používanie šetrí vývojárom

Tab. 3.1: Akcie a URL vedúce k ich spusteniu.

Akcia	HTTP metóda	URL
Odoslanie požiadavku do Repeateru	POST	http://localhost:port/repeater
Odoslanie požiadavku do Intruderu	POST	http://localhost:port/intruder
Odoslanie požiadavku do Compareru	POST	http://localhost:port/comparer
Pridanie domény do Scopu	POST	http://localhost:port/target/add/domena
Odstránenie domény zo Scopu	POST	http://localhost:port/target/remove/domena
Získanie celej sitemapy	GET	http://localhost:port/target
Odoslanie Url do Spideru	POST	http://localhost:port/spider/domena
Odoslanie požiadavku k pasívnemu skenu	POST	http://localhost:port/scanner/passive
Odoslanie požiadavku k aktívnemu skenu	POST	http://localhost:port/scanner/active
Získanie zraniteľností domény	GET	http://localhost:port/scanner/issues/domena

čas. Pre účely tohto modulu bolo vyberané z niekoľkých frameworkov.

Zrejme najznámejší java webový framework, Java Spring, je enterprise štandardom pre vývoj webových aplikácii. To však znamená, že je extrémne komplexný a zahŕňa funkcionalitu, ktorá vysoko prevyšuje požiadavky tohto modulu. Podobne robustný je napr. aj Apache Struts a pod. Pre účely tohto projektu bolo vyberané skôr z lightweight frameworkov. Do tejto kategórie spadajú frameworky ako Blade, Play alebo Spark.

Spark Java

Pri implementácii modulu bol nakoniec použitý Spark Java. Jedná sa o veľmi nenáročný framework prezývaný aj ako microframework. Keďže sa pri výbere zohľadňovala hlavne jednoduchosť a backendové zameranie, Spark vyhovel obidvom požiadavkom. Navyše poskytuje dokumentáciu jednoduchú na pochopenie. Spark sa taktiež zameriava na rapídny vývoj REST aplikácii, čim presne vystihuje potreby tohto projektu.

Filozofia, na ktorej je Java Spark postavený je rýchlosť vývoja a prehľadnosť kódu. Zdrojový kód vyvíjaný pomocou tohto frameworku je veľmi expresívny. Jeho hlavnou špecialitou je používanie lambda funkcií pre spracovanie funkcionality priradenej k ceste na server, preto vyžaduje Java verziu 1.8 a vyššiu. Jednoduchá anotácia má tvar znázornený vo výpise 3.2.

Výpis 3.2: Príklad trasovania v Spark Java.

```
get("/", (request, response) -> {
    // funkcia vykonaná po prístupe klienta do cesty
});
```

Týmto spôsobom sú implementované všetky funkcie, ktoré modul zabezpečuje. Spark server sa spúšťa hneď po načítaní modulu do Burp Suite. Dáta sú okamžite dostupné na adrese localhost a porte 4567. Okrem ciest definovaných tabuľkou 3.1, server pre odosielanie dát front end aplikácii využíva sockety.

3.1.2 Frontend

Pri softwari sa pojmom frontend označuje periféria aplikácie s ktorou užívateľ interaguje. Z pohľadu užívateľa sa dá frontend považovať za synonymum užívateľského rozhrania. Z pohľadu vývojára sa frontendom rozumie software, ktorý zabezpečuje funkcionalitu, vzhľad (upravený pomocou CSS (Cascading Style Sheets) súborov) a rozloženie užívateľského rozhrania. Hlavným cieľom frontendového software je vytvorenie intuitívneho ovládania, ale takisto rieši aj otázky toho ako vykresľovať užívateľské rozhranie na rôznych zariadeniach, pretože aplikácia v mobilnej verzii má

menej priestoru a pod. Táto práca sa však zameriava len na vzhľad aplikácie v prehliadači desktopu.

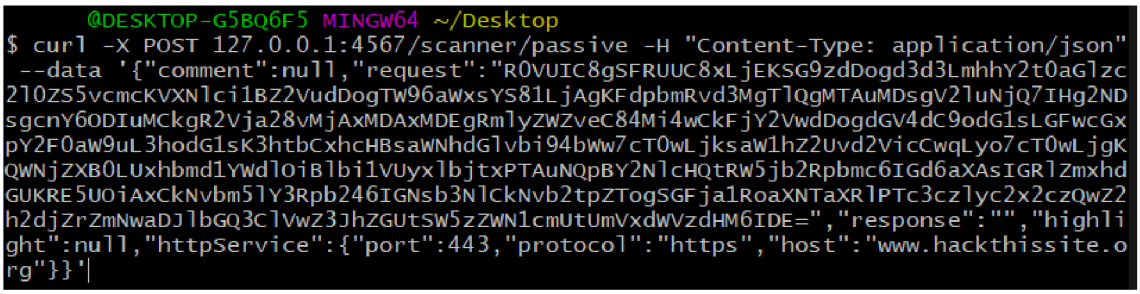
Súčasťou tejto práce je aj frontendová aplikácia, ktorá interpretuje dáta obsiahnuté v nástroji BurpSuite. Okrem klasických prístupov k dizajnu a funkčnosti užívateľských rozhraní sú populárne aj moderné dynamické frameworky ako Angular alebo VueJS. Ďalšou možnosťou je React.

React je JavaScriptová knižnica vyvíjaná spoločnosťou Facebook, a od svojho vzniku stále naberá na popularite. Jeho unikátnou vlastnosťou je zameranie na stav komponentov, čo sa odzrkadľuje do stavu celej aplikácie. Na základe zmien dát React vykresľuje komponenty, ktorých sa zmena dotlka, čo prispieva k jeho rýchlosti odozvy, a celkovej plynulosti.

Vlastnosti zmienených riešení sú podobné no existujú určité výhody a nevýhody každého z nich, na základe ktorých bol výsledný framework zvolený. Zatiaľ čo Vue a Angular zahŕňajú full stack vývoj, React je špecializovaný. Väčšia využiteľnosť frameworkov znamená aj ich väčšiu komplexitu a čas potrebný na učenie práce s nimi. Na rozdiel od nich, React je zameraný čisto na vývoj frontendových aplikácii. React sa ďalej líši od VueJS a Angularu používaním tzv. virtual DOM (Document Object Model) modelu, na rozdiel od klasického prístupu vykonávania zmien v skutočnom DOM modele, čo umožňuje Reactu rýchlejšie načítavanie zmenených dát a prvkov v užívateľskom rozhraní. Najsilnejšou stránkou Reactu je rýchlosť vykresľovania a spracovania skriptov.

3.1.3 Príprava prostredia

Modul je vyvíjaný na operačnom systéme Microsoft Windows 10 Home. Nástroj BurpSuite je vyvíjaný v programovacom jazyku Java, ktorého výhodou je JVM (Java Virtual Machine). Ten zabezpečuje multiplatformovosť a teda operačný systém neobmedzuje funkčnosť programu ani jeho modulov. Ako už bolo spomenuté, modul je takisto vyvíjaný v jazyku Java. Súčasťou pracoviska je IDE (Integrated Developer Environment) Eclipse 2020-09 a samotný nástroj BurpSuite professional v2020.9.2. Ako klient, ktorý sa dotazuje, alebo posiela HTTP správy na ďalšie spracovanie bol použitý Curl 7.73.0. Pre testovanie funkcionality zasielania HTTP žiadosti do modulov Repeater, Intruder, Comparer, Scanner atp. bola použitá žiadosť na Obr. 3.2.



Obr. 3.2: Použitá výzva.

Ako vývojové prostredie pre webovú aplikáciu bol použitý JetBrains Webstorm 2020.3.2. Závislosti ako knižnice sú popísané v súboroch pom.xml pre backend a package.json pre frontend.

4 Implementácia

Táto kapitola detailne popisuje spôsob, akým bola navrhnutá implementácia vytvorená. V časti implementácia frontendu sa nachádza popis React aplikácie, konkrétne jej komponentov a približuje aj výber dizajnu aplikovaného CSS súboru. Ďalej nadväzuje časť popisujúca backendovú implementáciu, spracovanie dát pred odosielaním, použité objekty a použité metódy.

4.1 Implementácia frontendu

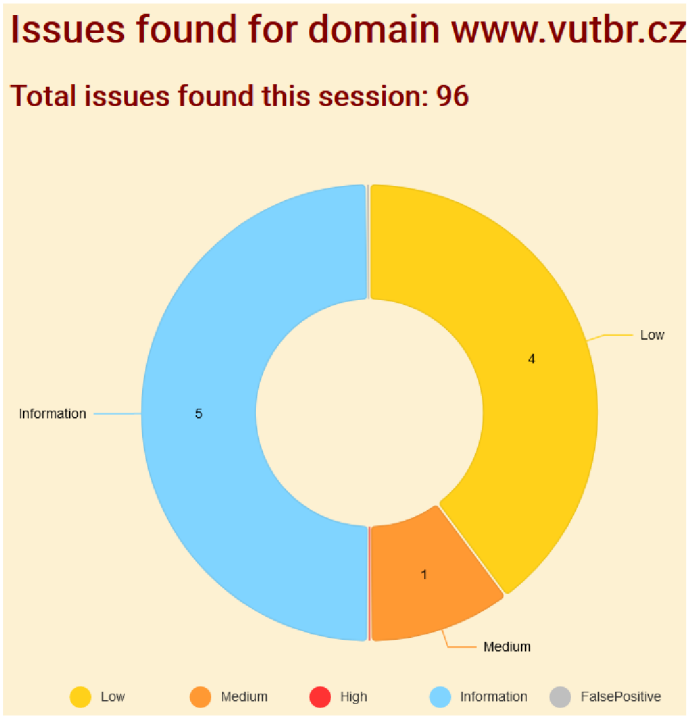
Dáta v užívateľskom rozhraní BurpSuite sú z pravidla textového charakteru s minimálnym grafickým dotykom. Hlavnou dominantou aplikácie je z tohto dôvodu graf. Vizuálna reprezentácia počtu nájdených zraniteľností, alebo tzv. issues dáva na prvý pohľad užívateľovi lepšiu predstavu o stave zraniteľností, ako čistý text. Jedinou závislosťou tejto aplikácie je knižnica Nivo. Výber knižníc na vizualizáciu dát bol opäť širší, no rozdiely sú takisto znova minimálne a Nivo má jednoznačnú výhodu v jeho bezkonkurenčnej interaktívnej dokumentácií. V hornej časti webového rozhrania sa nachádza nadpis s názvom vybranej domény, pre ktorú chceme dáta vizualizovať. Pod ním je uvedený celkový počet zaznamenaných zraniteľností v práve bežiacej inštancií BurpSuite. Náhľad je možný na Obr. 4.1. Nasleduje graf, zoznam domén z ktorých si užívateľ môže vyberať a po kliknutí na graf sa na pravej strane zobrazí detailnejší popis nálezov. Všetky dáta vo webovej aplikácií sa menia v reálnom čase s odozvou jednej sekundy. Pri prechádzanú webu so zapnutým BurpSuite je možné sledovať rastúci zoznam domén a nálezov.

Graf

Komponent `MyResponsivePie` dostáva dáta určené na vykreslenie v dátovom type pole ako svoju vlastnosť (property). Definíciou používania vlastných farieb je nutné aby okrem kategórie závažnosti zraniteľnosti a počtu nájdených zraniteľností každého typu, boli súčasťou predávaných dát aj hodnoty farieb priradené ku každej závažnosti. Graf je responzívny na kliky užívateľa. Po kliknutí na niektorú z kategórií reprezentovanú farebne odlišnou časťou grafu, sa na pravej strane grafu zobrazí detailnejší popis nájdených nedostatkov. Graf je znázornený na Obr. 4.1.

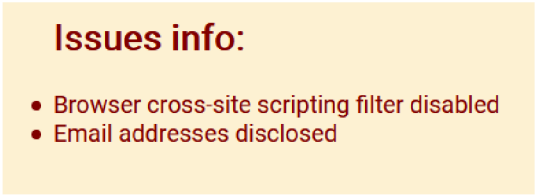
Informácie o nálezoch

K poskytnutiu detailnejších informácií bol použitý zoznam zobrazujúci sa na pravej strane grafu. Keďže počet nálezov a typy nálezov nie sú v exkluzívnom pomere jedna ku jednej, môže sa stať, že doména vykazuje niekoľko nálezov, všety však len jedného



Obr. 4.1: Ukážka grafu v UI.

typu. Často je tomu tak u menej závažných nálezov z kategórie Information alebo Low. Príklad je uvedený na Obr. 4.2.



Obr. 4.2: Ukážka detailu nálezov.

Zoznam domén

Rovnako ako v samotnom BurpSuite, v záložke Target, existuje zoznam domén aj vo webovej aplikácií. Jedná sa o totožný zoznam. Po kliknutí na ľubovoľnú doménu sa v grafe zobrazia dáta spojené s touto doménou, rovnako tak sa aktualizuje popis na pravej strane grafu po kliknutí. Zoznam domén je možné vidieť na Obr. 4.3.

Získavanie dát

Dáta sú z BurpSuite do frontendu zasielané pomocou websocketu. Na backende existuje trasa `/websocket`, kam sa aplikácia pripája. Pri načítaní aplikácie sa vytvorí

Obr. 4.3: Ukážka zoznamu domén.

nový websocket a spojí sa s backendom. Po nadviazaní spojenia začne okamžité preberanie dát. Správa je zložená zo štyroch objektov. Prvý objekt predstavuje počet všetkých nájdených nedostatkov, druhý zoznam domén, o ktorých sú zbierané dáta, tretí objekt obsahuje samotné dáta k vybranej doméne, teda počet každej kategórie zraniteľností, meno kategórie a farbu, ktorá bude použitá pri vykreslení. Posledný objekt obsahuje konkrétne názvy nájdených nedostatkov, ktoré zobrazuje zoznam na pravej strane grafu. Po kliknutí na niektorú z domén sa backendu odošle správa obsahujúca meno zvolenej domény.

CSS

Aplikácia nevyužíva žiadnu knižňicu na úpravu vzhľadu ako napr. Bootstrap. Kaskádový štýl je unikátny pre tento projekt. Farebná schéma bola vybraná s ohľadom na kontrast medzi farbami textu a farbami použitými v grafe. Tradičné označenie kategórií zraniteľností modrou, žltou, oranžovou, červenou a šedou farbou pre informačné, najmenej závažné, stredne závažné, kritické nedostatky a falošné pozitívy, nevyznievajú dobre na podkladoch podobného odtieňa. Biela farba pozadia bola príliš ostrá. Preto bola vybraná jemne bledá telová ako základ s bordovými prvkami. Rozloženie stránky je veľmi intuitívne a vzhľadom na jednoduchosť by sa užívateľ mal ľahko zorientovať v používaní.

4.2 Implementácia backendu

Hlavnou triedou modulu je `BurpExtender.class`. Táto trieda musí implementovať rozhranie `IBurpExtender` a jeho metódu `registerExtenderCallbacks`. Tá ďalej slúži ako hlavná funkcia, ktorá je spustená po načítaní modulu do BurpSuite. V tele tejto metódy sú inicializované objekty pre prácu s dátami aj server. Prvý z objektov je objekt `callbacks`. Volaním metód objektu `callbacks` sa registujú poslucháče. BurpSuite API má niekoľko typov posluchačov, napr. `HttpListener` alebo `ProxyListener`. Tento modul využíva hlavne `HttpListener`. Okrem registrácie posluchačov sa inicializuje aj objekt `helpers`. Metódy tohto objektu poskytujú utility ako napr. dekódovanie dát z `Base64` kódovania do plaintextu, alebo prevody z dátového typu `string` na typ `byte[]`, s ktorým pracujú metódy odosielania požiadavkov do modulov ako je Repeater.

V hlavnej časti kódu sú umiestnené metódy Spark frameworku. Pre pripojenie k serveru je potrebné použiť port 4567. Ako prvý je definovaný websocket na adrese `/websocket`. K práci s websocketom je potrebá trieda `MyWebSocket`. Trieda obsahuje metódy `connected`, `closed` a `message` pre manažment pripojení a metódu `getData`, ktorá sa stará o spracovanie dát pre frontend. Jej výstupom je `JSON` pole obsahujúce štyri ďalšie `JSON` objekty. Ako prvý, celkový počet nájedných zraniteľností zaznamenaných BurpSuitom, ďalej pole obsahujúce zoznam domén z modulu Target, pole detailnejších popisov nálezov a ako posledné pole s dátami potrebnými pre vykreslenie grafu. Spark server je schopný inicializácie aj bez explicitného volania, pri použití websocketu je však odporúčané použiť metódu `init()`. Tá je volaná hneď po vytvorení websocketu. Následujú `get()` a `post()` metódy. Tvar je uvedený v príklade 3.2. Metódy sú vo frameworku Spark definované ako statické pre prehľadnosť kódu. Meno definuje HTTP metódu potrebnú pre prístup. Objekty `request` a `response` slúžia k extrakcii dát z ich tela. Funkcie, ktoré sa majú vykonať po prístupe na definovanú cestu sú v lambda funkcii taktiež pre prehľadnosť. Tabuľka 3.1 z návrhu implementácie obsahuje zoznam ciest a HTTP metód potrebný k prístupu k týmto zdrojom. Navyše okrem týchto ciest boli implementované aj cesty pre získavanie predspracovaných dát z dôvodu pohodlnejšieho prístupu a rýchlosti jazyka java. Sú to ako napr. vytvorenie jednoduchého zoznamu domén odpovedajúcemu zoznamu v záložke Target. Prehľad finálne implementovaných ciest ponúka Tab. 4.1. Z dôvodu prehľadnosti boli v tabuľke uvedené len cesty, ktoré sú inak prefixované adresou `http://localhost:port`.

Príklad implementácie metódy pre spustenie aktívneho skenu zobrazuje výpis 4.1 a akciu spustenú v BurpSuite možno vidieť na Obr. 4.4.

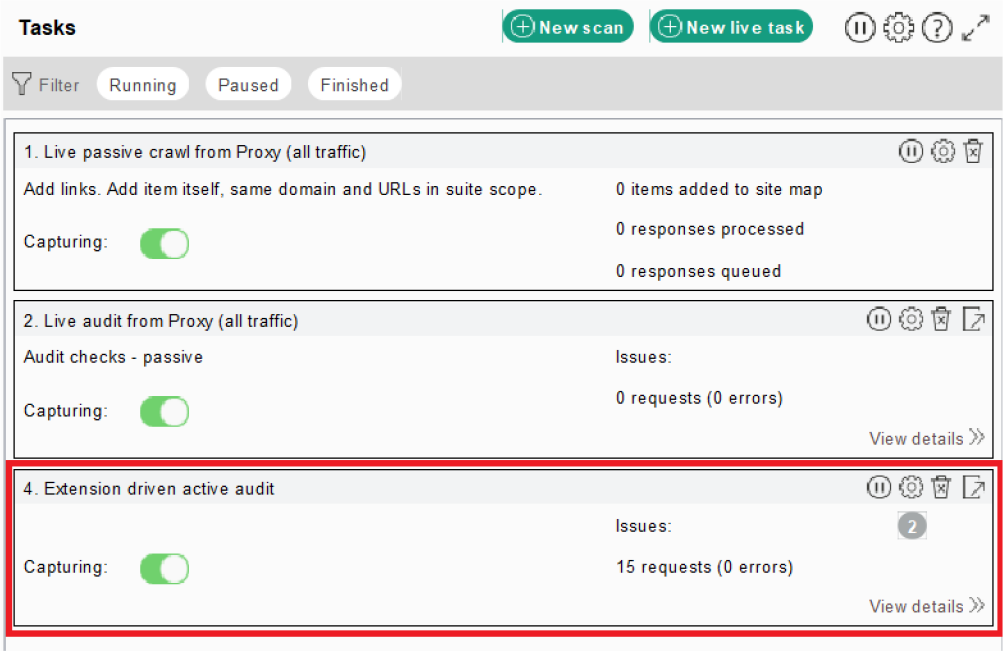
Tab. 4.1: Implementované metódy.

Akcia	HTTP metóda	URL cesta
Odoslanie požiadavku do Repeateru	POST	/repeater
Odoslanie požiadavku do Intruderu	POST	/intruder
Odoslanie požiadavku do Compareru	POST	/comparer
Pridanie domény do Scopu	POST	/target/add/domena
Odstránenie domény zo Scopu	POST	/target/remove/domena
Získanie celej sitemapy	GET	/target
Odoslanie Url do Spideru	POST	/spider/domena
Odoslanie požiadavku k pasívnemu skenu	POST	/scanner/passive
Odoslanie požiadavku k aktívnemu skenu	POST	/scanner/active
Získanie zranitelností domény	GET	/scanner/issues/domena
Získanie všetkých nálezov	GET	/scanner/issues
Získanie zoznamu domén	GET	/targetlist
Získanie počtu nálezov podľa kategórie	GET	/graph/domena
Získanie detailov o nálezoch	GET	/target/issuesinfo/domena
Celkový počet nálezov	GET	/scanner/total

Výpis 4.1: Príklad funkcie v ceste /scanner/active.

```
post("/scanner/active", (request, response) -> {
        rqst = deserializeObject(request.body());
        callbacks.doActiveScan(rqst.getHttpService().getHost(), rqst.
            getHttpService().getPort(),
        rqst.getHttpService().usesHttps(), rqst.getRequest());
        return "Request aktivoval aktívny sken.";
     });
```

V prvej časti kódu sa extrahuje telo prijatej HTTP žiadosti. Pri použití samotného BurpSuite API by bolo ako prvé pre pohodlnejšiu prácu potrebné transformovať dáta z typu `byte[]` na typ `string` za pomoci metódy `bytesToString` objektu `helpers`. Spark tento krok uľahčuje jednoduchým volaním funkcie `request.body()`. Prijatý `JSON` objekt je nutné deserializovať a vytvoriť nový `HttpRequestReponse` objekt. S ním sa dá ďalej pracovať, napr. pomocou metódy `getRequest` sa dostaneme k hodnote uloženej v premennej `request`. Táto procedúra je nutná v prípade, ak chceme spracovanú HTTP žiadosť obsiahnutú v tele obdržanej žiadosti odoslať do modulov ako Repeater alebo Intruder. Celý chod modulu je znázornený na Obr. 4.5.



Obr. 4.4: Ukážka aktívneho skenu spusteného modulom.

Dáta prístupné GET žiadosťami

Medzi funkcionalitou modulu pri spracovaní požiadaviek typu GET, teda prípadov keď externe vytvárame žiadosť o dáta obsiahnuté v BurpSuite, a spracovaní POST

žiadostí, kedy BurpSuite dáta prijíma je zásadný rozdiel. V prípade GET žiadostí sa všetky objekty, ku ktorým chceme pristupovať nachádzajú v pamäti. Riešením, ako tieto objekty odoslať mimo BurpSuite je transformácia do `JSON` formátu a uložením do tela HTTP odpovede. Na pretváranie objektov do `JSON` tvaru je použitá knižnica `Jackson`. Obsahom knižnice je objekt `ObjectMapper`. Ten obsahuje metódu `writeValueAsString` určenú pre zápis iných objektov do `JSON` formátu. Takto upravené dáta sú ľahko čitateľné iným automatizovaným nástrojom, ktoré o ne žiadajú. Pre vytváranie objektov boli implementované pomocné triedy `jsonHelperObj` a `jsonDOmainInfoObj`. Objekty sú použité pri spracovaní dát vyžiadaných v ceste `/graph/` a `/target/issuesinfo/`. Ich obsahom sú privátne premenné pomenované štýlom kompatibilným s frontend aplikáciou. Metóda vracajúca detailnejšie informácie o nálezoch používa k zbaveniu sa duplicít dátovú štruktúru typu `HashSet`. Porovnvanie prvkov prebieha pomocou preťaženia metódy `equals()` a porovnáva sa hodnota v premennej `name`, čiže označenie typu zraniteľnosti. Prechádzaním všetkých nálezov sa tohto hashsetu zapisujú prvky obsahujúce užívateľom zadanú URL. Väčšina funkcionality spojená s HTTP GET žiadostami používa pre získanie dát zoznamy vracané metódami `getScanIssues` a `getSitemap()`, z ktorých sú ďalej vyberané prvky podľa kritérií ako napr. URL.

Dáta spracovávané POST žiadosťami

V prípade obdržania dát je hlavným krokom rekonštrukcia dát do objektu, s ktorým je BurpSuite schopný pracovať. Takisto ako pri odosielaní dát je k tomu využitý objekt `ObjectMapper` knižnice `Jackson`. Žiadosti, ktoré majú byť posunuté do ostatných modulov prichádzajú vo formáte `JSON` v tele žiadosti, ktorá sa dotazuje. Polia, ktoré tento `JSON` objekt musí obsahovať sa zhodujú s premennými v objektoch `HttpRequestResponse` a `HttpService`. Proces vkladania žiadosti uvedenej v tele originálnej žiadosti začína vytvorením `HttpRequestReponse` objeku. Ten sa iniciuje deserializácie `JSON` objektu pomocou `objectMapperu`. Takto vytvorený objekt môže byť ďalej plnohodnotne používaný. Objekt `callbacks` obsahuje metódy na odosielanie dát príslušným modulom, napr. `doActiveScan()`. Každá cesta volá príslušnú metódu a vkladá dáta do modulov.

4.2.1 Rozhrania BurpSuite API

BurpSuite API poskytuje 42 rozhraní. Pre prehľadnosť sa tieto rozhrania dajú rozdeliť do skupín:

- Rozhrania pre komunikáciu s jadrom BurpSuite
  - IBurpExtender
  - IBurpExtenderCallbacks

  - IExtensionStateListener
  - IHttpListener
  - IScopeChangeListener
  - ISessionHandlignAction
- Utility
  - IExtensionHelpers
  - ITempFile
- Rozhrania pre prácu s GUI
  - IContextMenuFactory
  - IContextMenuInvocation
  - IMenuItemHandler
  - IMessageEditor
  - IMessageEditorController
  - IMessageEditorTab
  - IMessageEditorTabFactory
  - ITab
  - ITextEditor
- HTTP objekty
  - ICookie
  - IHttpRequestResponse
  - IHttpRequestResponsePersisted
  - IHttpRequestResponseWithMarkers
  - IHttpService
  - IParameter
  - IRequestInfo
  - IResponseInfo
  - IResponseKeywords
  - IResponseVariations
- Intruder
  - IIntruderAttack
  - IIntruderPayloadGenerator
  - IIntruderPayloadGeneratorFactory
  - IIntruderPayloadProcessor
- Proxy
  - IIntercepptedProxyMessage
  - IProxyListener
- Scanner
  - IScanIssue
  - IScannerCheck

    - IScannerInsertionPoint
    - IScannerInsertionPointProvider
    - IScannerListener
    - IScanQueueItem
- Collaborator
    - IBurpCollaboratorClientContext
    - IBurpCollaboratorInteraction

Tento modul pracuje hlavne s rozhraniami HTTP objektov a rozhraniami pre komunikáciu s jadrom a utilitami. Najdôležitejším objektom, s ktorým modul pracuje je HTTP výzva, alebo odpoveď dedená z rozhrania `IHttpRequestReponse`. S ňou súvisí objekt `HttpService` z rozhrania `IHttpService`, ktorý každá žiadosť a odpoveď obsahujú. Triedy `HttpRequestReponse` a `HttpService` boli vytvorené z dôvodu vytvárania nových objektov po deserializácii. `MyHttpRequestReponse` a `MyHttpService` sú rozhrania rozšírené o pomocnú metódu na získanie príznaku použitého protokolu. Rozhranie `IRequestInfo` je použité spolu s rozhraniami `IExtensionHelpers` a `IBurpExtenderCallbacks` na manipuláciu s žiadosťami.

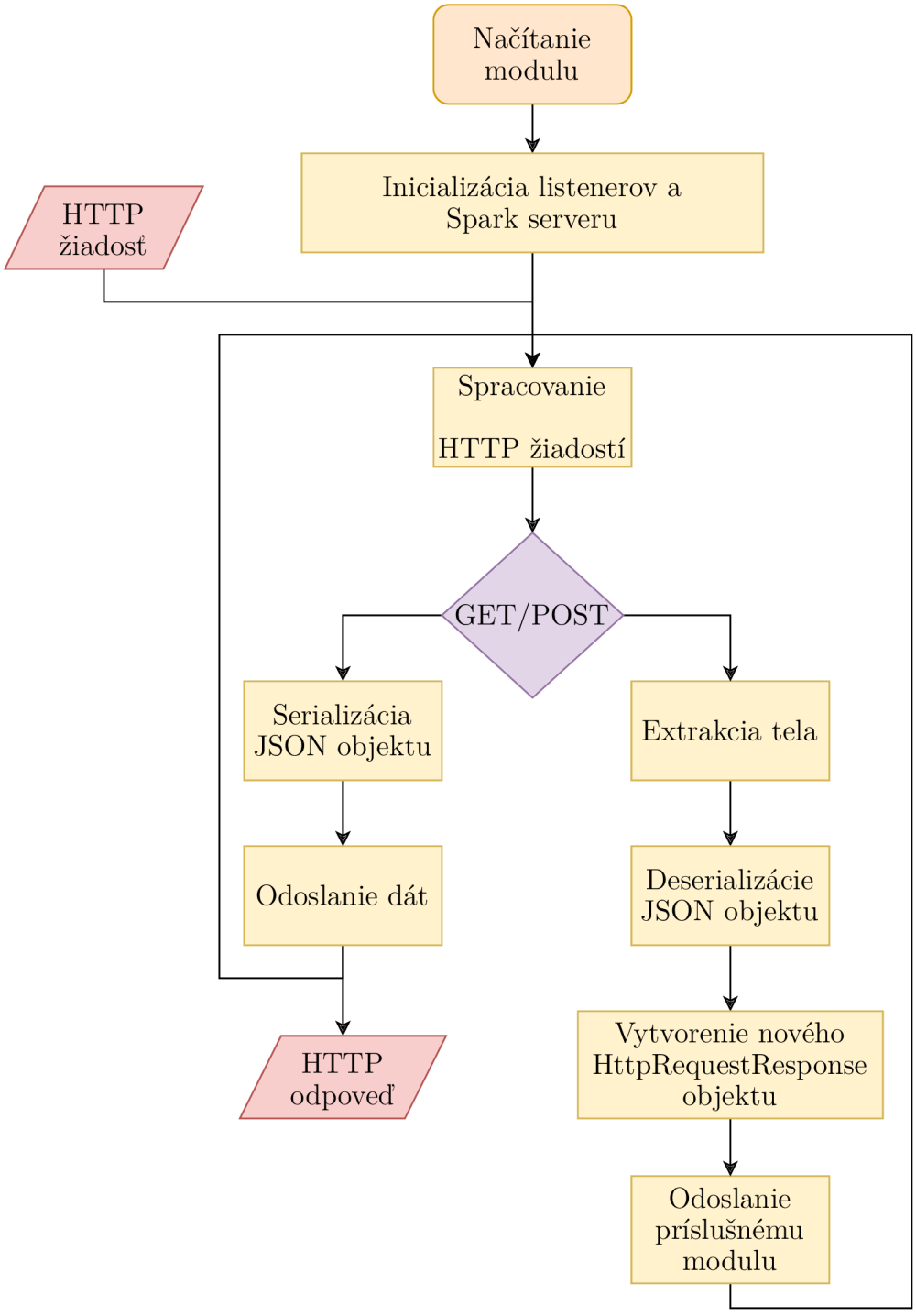
Pomocou rozhrania `IProxyListener` je možné ovládať správanie zachytávania správ, ale v tomto module je zatiaľ nadstavené na rovnaké správanie aké je zadané užívateľom v BurpSuite samotnom. Rozhranie `IExtensionStateListeners` hlási aké poslucháče modul používa do výpisu v záložke Extensions. Pre informácie sa do výpisu pomocou `IScannerListeneru` zapisuje správa pri nájdení nálezu.

4.3 Štruktúra modulu

Celý modul je obsiahnutý v jednom `.JAR` archíve. Pri jeho vytvorení za pomoci balíčkového manažéra Maven budú pribalené aj použité knižnice definované v súbore `pom.xml`.

```
vut-feec-burp-extensions.jar
├── burp
│   ├── BurpExtender.class
│   ├── HttpRequestResponse.class
│   ├── HttpService.class
│   ├── MyHttpRequestResponse.class
│   ├── MyHttpService.class
│   ├── SeverityInfo.class
│   ├── MyWebSocket.class
│   ├── jsonHelperObj.class
│   └── jsonDomainInfoObj.class
└── META-INF
    ├── MANIFEST.MF
    └── maven
        └── com.vutbr.cz
            └── vut-feec-burp-extensions
                ├── pom.properties
                └── pom.xml
```

`vut-feec-burp-extensions.jar` je názov archívu. Zložka `burp` reprezentuje balíček a musí byť prítomná pre správne načítanie modulu. Obsahom sú skompilované triedy vrátane hlavnej triedy `BurpExtender.class` a ostatných, ktoré definujú funkcionalitu modulu. Zložka META-INF je automaticky generovaná a obsahuje súbor `MANIFEST.MF`. Tento súbor obsahuje informácie o verzii použitého Java Developer Kitu, užívateľské meno systému kde bol archív vytvorený, program, ktorým bol vytvorený a verziu manifestu. V zložke `maven` sa nachádza súbor pom.xml a pom.properties. Najdôležitejšie informácie pre správny build modulu sa nachádzajú v `pom.xml`. Tento súbor popisuje jednoduchou `xml` štruktúrou celý projekt. Obsahuje informácie kde patrí unikátny identifikátor projektu `groupId`, `artifactId` zadáva meno výslednému `.JAR` archívu, a blok `dependencies`. Dependencies definujú, ktoré knižnice majú byť pri tvorbe archívu zakomponované do výsledného archívu.



Obr. 4.5: Diagram modulu.

Záver

Cieľom práce bolo vytvorenie zásuvného modulu, ktorý umožní lepšiu automatizáciu procesu výmeny dát medzi nástrojmi používanými pri penetračnom testovaní.

Z kapitoly 2 je viditeľné, že testeri skutočne používajú množstvo nástrojov. V niektorých prípadoch však nástroje neponúkajú možnosť rozhrania príkazového riadku, alebo nie je možné dáta z nich posielať iným nástrojom. Tento modul rieši problém automatizácie úkonov a extrakciu dát nástroja BurpSuite. Ďalej sa kapitola zaoberá porovnaním dvoch štandardne používaných nástrojov, konkrétne BurpSuite pro a OWASP ZAP. Na základe analýzy bol pre účel práce vybraný nástroj BurpSuite.

Kapitola 3 popisuje požiadavky na funkcionalitu modulu a aj návrh možnej realizácie. Modul by mal slúžiť ako brána k dátam spracovaným nástrojom BurpSuite. Na základe výziev sú dáta vkladané, alebo odosielané z BurpSuite. K prijímaniu a odosielaniu dát bol zvolený formát JSON. Modul dokáže prijímať žiadosti a vkladať ich do zabudovaných modulov Repeater, Intruder, Comparer, Spider, Scanner a Target Scope. Na základe domény je ďalej schopný odosielať pole JSON objektov nachádzajúcich sa v Site Mape spolu so všetkými cestami nájdených na danej doméne, alebo pole nájdených zraniteľností spojených s doménou. Užívateľ môže spúšťať ako aktívny sken, tak aj pasívny. Rovnako je pomocou modulu možné spustiť crawlovanie stránok. Tabuľka s prehľdom funkcií je uvedená v kapitole 4, konkrétne v Tab. 4.1. Pre plynulú komunikáciu BurpSuite a klientov pomocou HTTP protokolu bol použitý serverový mikroframework SparkJava, špecializujúci sa na vývoj REST aplikácií.

Súčasťou práce je aj webová aplikácia. Pre realizáciu užívateľského rozhrania bola použitá populárna a moderná JavScriptová knižnica React. BurpSuite je zameraný skôr na textovú reprezentáciu výsledkov, čo však pri ich vyhodnocovaní človekom vyžaduje zvýšené sústredenie. Grafické znázornenie informácií je pre ľudí intuitívnejšie a hneď na prvý pohľad majú predstavu o stave zraniteľností. Aplikácia zobrazuje graf s kategóriami zraniteľností a počtom nálezov v každej z nich. Graf je interaktívny a po kliknutí zobrazuje detailnejšie informácie. Dáta sú doň vkladané v reálnom čase pomocou websocketu.

Literatúra

[1] Justice Will Take Us Millions Of Intricate Moves, Act Three [online]. [cit. 2021-5-21]. Dostupné z: <https://www.crummy.com/writing/speaking/2008-QCon/act3.html>

[2] BACUDIO, Aileen G., et al. An overview of penetration testing. International Journal of Network Security & Its Applications, 2011, 3.6: 19.

[3] HackerOne [online]. [cit. 2020-12-09]. Dostupné z: <https://www.hackerone.com/>

[4] OWASP Web Security Testing Guide [online]. [cit. 2020-12-09]. Dostupné z: <https://owasp.org/www-project-web-security-testing-guide/stable/2-Introduction/>

[5] REHBERGER, Johann. Cybersecurity Attacks – Red Team Strategies. Packt Publishing, March 2020. ISBN ISBN 978-1-83882-886-8.

[6] ORIYANO, Sean-Philip. CEH™ Certified Ethical Hacker Study Guide. Version 9. Indianapolis, Indiana: John Wiley, 2016. ISBN 978-1-119-25224-5.

[7] HERZOG, Pete. The Open Source Security Testing Methodology Manual [online]. 3. ISECOM, 2010 [cit. 2020-12-09]. Dostupné z: <www.osstmm.org>

[8] OWASP Mobile Security Testing Guide [online]. [cit. 2020-12-09]. Dostupné z: <https://mobile-security.gitbook.io/mobile-security-testing-guide/>

[9] DAVIS, Royce. The Art of Network Penetration Testing. Manning Publications Co., 2020. ISBN 9781617296826.

[10] ParrotSec. ParrotOS tools [online]. [cit. 2020-12-09]. Dostupné z: <https://github.com/ParrotSec/parrot-tools/blob/master/debian/control>

[11] OWASP Zed Attack Proxy [online]. [cit. 2020-12-09]. Dostupné z: <https://www.zaproxy.org/>

[12] Burp Suite documentation [online]. [cit. 2020-12-09]. Dostupné z: <https://portswigger.net/burp/documentation>

[13] LOZANO, Carlos A., Dhruv SHAH a Riyaz Ahemed WALIKAR. Hands-On Application Penetration Testing with Burp Suite. Packt Publishing, 2019. ISBN-13 978-1788994064. ISBN-10 178899406X.

[14] WEAR, Sunny. Burp Suite Cookbook. Packt Publishing, 2018. ISBN 978-1-78953-173-2.

[15] What is maven? [online]. [cit. 2020-12-09]. Dostupné z: <https://maven.apache.org/what-is-maven.html>

Zoznam symbolov, veličín a skratiek

API	programovateľné rozhranie aplikácie – Application programming interface
BApp	Burp aplikácia – Burp Application
CLI	rozhranie v príkazovom riadku – Command Line Interface
CSS	kaskádové typy – Cascading Style Sheets
CSRF	medzidoménovy falzifikát výzvy – Cross Site Request Forgery
DOM	dokumentový model objektov – Document Object Model
GUI	grafické uživateľské rozhranie – Graphical User Interface
hex	hexadecimálne – Hexadecimal
HR	ľudské zdroje – Human resources
HTTP	Hypertextový transportný protokol – Hypertext Transport Protocol
IDE	integrovavé vývojové prostredie – Integrated Developer Environment
IP	interenetový protokol – Internet Protocol
JAR	Java archív – Java archive
JSON	JavaScriptový popis objektov – JavaScript Object Notation
JVM	virtuálny stroj Java – Java Virtual Machine
OWASP	Open Web Application Security Project
POM	model projektových objektov – Project Object Model
PTES	štandard výkonu penetračného testu – Penetration Testing Execution Standard
REST	reprezentatívny stav prenosu – Representational State Transfer
SOCKS	bezpečné sokety – Socket Secure
SQL	štrukturovaný jazyk dotazov – Structured Query Language
URL	jednotný lokátor zdrojov – Uniform Resource Locator
USB	univerzálna sériová zbernica – Universal Serial Bus

ZAP	HTTP proxy nástroj k účelom penetračného testovania – Zed Attack Proxy

Zoznam príloh

A Príklady použitia modulu

Pre skúšku funkčnosti modulu je v tejto prílohe testovacia žiadosť v správnom formáte. Je dôležité podotknúť, že žiadost obsahuje na samom konci dva krát znak nového riadku, čo je potrebné pre správne fungovanie odosielania z BurpSuite. Preto ak chceme vytvoriť novú žiadosť, pred kódovaním do Base64 formátu je treba pridať dva znaky nového riadku na koniec žiadosti. Tento jav je vidieť aj na Obr. A.1.

Výpis A.1: HTTP žiadosť v Base64 kódovaní.

```
R0VUIC8gSFRUUC8xLjEKSG9zdDogd3d3LmhhY2t0aGlzc2l0ZS5vcmcKVXNlci1BZ2VudDogTW
    96aWxsYS81LjAgKFdpbmRvd3MgTlQgMTAuMDsgV2luNjQ7IHg2NDsgcnY6ODIuMCkgR2
    Vja28vMjAxMDAxMDEgRmlyZWZveC84Mi4wCkFjY2VwdDogdGV4dC9odG1sLGFwcGxpY2F0
    aW9uL3hodG1sK3htbCxhcHBsaWNhdGlvbi94bWw7cT0wLjksaW1hZ2Uvd2VicCwqLyo7cT
    0wLjgKQWNjZXB0LUxhbmd1YWdlOiBlbi1VUyxlbjtxPTAuNQpBY2NlcHQtRW5jb2Rpbmc6
    IGd6aXAsIGRlZmxhdGUKRE5UOiAxCkNvbm5lY3Rpb246IGNsb3NlCkNvb2tpZTogSGFja1
    RoaXNTaXRlPTc3czlyc2x2czQwZ2h2djZrZmNwaDJlbGQ3ClVwZ3JhZGUtSW5zZWN1
    cmUtUmVxdWVzdHM6IDE=
```

čo zodpovedá žiadosti na Obr. A.1.

```
GET / HTTP/1.1
Host: www.hackthissite.org
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:82.0) Gecko/20100101 Firefox/82.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Cookie: HackThisSite=77s9rslvs40ghvv6kfcph2eld7
Upgrade-Insecure-Requests: 1

|
```

Obr. A.1: Výzva v RAW formáte.

Pre testovanie je odporúčaný klient ako napr. cURL. Vybrané ukážkové príkazy sú uvedené nižšie. Užívateľ môže jednoducho požiadať o SiteMapu generovanú programom Burp pre doménu `en.wikipedia.org` pomocou príkazu:

Výpis A.2: Príklad žiadosti o poslanie Site Mapy domény en.wikipedia.org.

```
curl 127.0.0.1:4567/target/en.wikipedia.org
```

Kedže cURL vo východzom nadsavení používa metódu GET, nieje treba nič špecifikovať. Pre obdržanie všetkých nálezov pre doménu `en.wikipedia.org` bude použitý príkaz:

Výpis A.3: Žiadosť o nájdených zraniteľnostiach domény en.wikipedia.org.

```
curl 127.0.0.1:4567/scanner/issues/en.wikipedia.org
```

Pre poslanie užívateľom definovaného požiadavku do modulu Repeater bude použitý príkaz:

Výpis A.4: Príklad odoslania žiadosti do modulu Repeater.

```
curl -X POST 127.0.0.1:4567/repeater -H "Content-Type: application/json" --
    data '{"comment":null,"request":"ROVUIC8gSFRUUC8xLjEKSG9zdDogd3d3LmhhY2
    t0aGlzc2l0ZS5vcmcKVXNlci1BZ2VudDogTW96aWxsYS81LjAgKFdpbmRvd3
    MgTlQgMTAuMDsgV2luNjQ7IHg2NDsgcnY6ODIuMCkgR2Vja28
    vMjAxMDAxMDEgRmlyZWZveC84Mi4wCkFjY2VwdDogdGV4dC9odG1sLGFwcGxpY2F0aW9uL3
    hodG1sK3htbCxhcHBsaWNhdGlvbi94bWw7cT0wLjksaW1hZ2Uvd2VicCwqLyo7cT0
    wLjgKQWNjZXB0LUxhbmd1YWdlOiBlbi1VUyxlbjtxPTAuNQpBY2NlcHQtRW5jb2Rpbmc6
    IGd6aXAsIGRlZmxhdGUKRE5UOiAxCkNvbm5lY3Rpb246IGNsb3NlCkNvb2tpZTogSGFja1
    RoaXNTaXRlPTc3czlyc2x2czQwZ2h2djZrZmNwaDJlbGQ3ClVwZ3JhZGUtSW5zZWN1
    cmUtUmVxdWVzdHM6IDE=","response":null,"highlight":null,"httpService":{"
    port":443,"protocol":"https","host":"www.hackthissite.org"}}'
```

Prepínač -X nadstavuje ďalej definovanú HTTP metódu, v tomto prípade POST. Nasleduje adresa, v tomto prípade buď `127.0.0.1` alebo `localhost` spolu s portom a cestou k modulu, ktorý chceme použiť. Prepínač -H definuje vlastný hlavičku, v tomto prípade Content-type. Prepínač `--data` definuje telo žiadosti.

Podobne ako pri posielaní žiadosti do Repeateru, aj pri začatí aktívneho skenu je potrebné definovať POST metódu. Príkaz je obdobný, treba len zmeniť cestu.

Výpis A.5: Príklad odoslania žiadosti do modulu Scanner pre aktívne skenovanie.

```
curl -X POST 127.0.0.1:4567/scanner/active -H "Content-Type: application/
    json" --data '{"comment":null,"request":"ROVUIC8gSFRUUC8xLjEKSG9zdDogd3
    d3LmhhY2t0aGlzc2l0ZS5vcmcKVXNlci1BZ2VudDogTW96aWxsYS81LjAgKFdpbmRvd3
    MgTlQgMTAuMDsgV2luNjQ7IHg2NDsgcnY6ODIuMCkgR2Vja28
    vMjAxMDAxMDEgRmlyZWZveC84Mi4wCkFjY2VwdDogdGV4dC9odG1sLGFwcGxpY2F0aW9uL3
    hodG1sK3htbCxhcHBsaWNhdGlvbi94bWw7cT0wLjksaW1hZ2Uvd2VicCwqLyo7cT0
    wLjgKQWNjZXB0LUxhbmd1YWdlOiBlbi1VUyxlbjtxPTAuNQpBY2NlcHQtRW5jb2Rpbmc6
    IGd6aXAsIGRlZmxhdGUKRE5UOiAxCkNvbm5lY3Rpb246IGNsb3NlCkNvb2tpZTogSGFja1
    RoaXNTaXRlPT c3czlyc2x2czQwZ2h2djZrZmNwaDJlbGQ3ClVwZ3JhZGUtSW5zZWN1
    cmUtUmVxdWVzdHM6IDE=","response":null,"highlight":null,"httpService":{"
    port":443,"protocol":"https","host":"www.hackthissite.org"}}'
```

Podotýka sa, že rozdiel medzi aktívnym a pasívnym skenom je v poli `response`. Zatiaľ čo u aktívneho skenu môže byť aj hodnota `null`, u pasívneho skenu, ak nexistuje odpoveď, treba použiť prázdny reťazec, teda `""`. Príkaz pre pasívne preskenovanie užívateľom definovanej žiadosti je rovnaký ako príkaz pre aktívny sken, s jedniným rozdielom v ceste a poli `response`.

Výpis A.6: Príklad odoslania žiadosti do modulu Scanner pre pasívne skenovanie.

```
curl -X POST 127.0.0.1:4567/scanner/passive -H "Content-Type: application/
    json" --data '{"comment":null,"request":"R0VUIC8gSFRUUC8xLjEKSG9zdDogd3
    d3LmhhY2t0aGlzc2l0ZS5vcmcKVXNlci1BZ2VudDogTW96aWxsYS81LjAgKFdpbmRvd3
    MgTlQgMTAuMDsgV2luNjQ7IHg2NDsgcnY6ODIuMCkgR2Vja28
    vMjAxMDAxMDEgRmlyZWZveC84Mi4wCkFjY2VwdDogdGV4dC9odG1sLGFwcGxpY2F0aW9uL3
    hodG1sK3htbCxhcHBsaWNhdGlvbi94bWw7cT0wLjksaW1hZ2Uvd2VicCwqLyo7cT0
    wLjgKQWNjZXB0LUxhbmd1YWdlOiBlbi1VUyxlbjtxPTAuNQpBY2NlcHQtRW5jb2Rpbmc6
    IGd6aXAsIGRlZmxhdGUKRE5UOiAxCkNvbm5lY3Rpb246IGNsb3NlCkNvb2tpZTogSGFja1
    RoaXNTaXRlPTc3czlyc2x2czQwZ2h2djZrZmNwaDJlbGQ3ClVwZ3JhZGUtSW5zZWN1
    cmUtUmVxdWVzdHM6IDE=","response":"","highlight":null,"httpService":{"
    port":443,"protocol":"https","host":"www.hackthissite.org"}}'
```

B Obsah přiloženého ZIP súboru

Táto príloha obsahuje informácie o priloženom `.zip` súbore a inštalácii modulu. Vytvorenie `.jar` súboru vyžaduje balíčkový manažér Maven. Potom stačí priložený .zip súbor stiahnuť, rozbaliť a otvoriť príkazový riadok v zložke VUTmodul. Potom sa v príkazovom riadku sa použije príkaz:

```
mvn clean package
```

Po dokončení procesu sa objaví nová zložka s názvom target. V nej sa bude nachádzať archív s názvom `vut-feec-burp-extensions-1.0.0.jar`, ktorý predstavuje samotný modul. Potom podľa návodu v kapitole 2, v sekcií 2.3.2 sa načíta modul do BurpSuite.

```
zdrojové kódy ..... koreňový adresár priloženého .zip súboru
├── backend ..... zdrojové kódy modulu
│   └── VUTmodul
│       ├── src
│       │   └── main
│       │       └── java
│       │           └── burp
│       │               ├── BurpExtender.java
│       │               ├── HttpRequestResponse.java
│       │               ├── HttpService.java
│       │               ├── MyHttpService.java
│       │               ├── MyHttpRequestResponse.java
│       │               ├── MyWebSocket.java
│       │               ├── jsonHelperObj.java
│       │               ├── jsonDomainInfoObj.java
│       │               └── SeverityInfo.java
│       └── pom.xml ..... súbor obsahujúci informácie o závislostiach
└── frontend ..... zdrojové kódy webovej aplikácie
    ├── src
    │   ├── Results.js
    │   ├── index.js
    │   ├── index.css
    │   └── App.js
    ├── public
    │   ├── favicon ..... zložka obsahujúca ikony
    │   ├── robots.txt
    │   └── index.html
    ├── package.json ..... súbor obsahujúci informácie o závislostiach
    └── package-lock.json
```