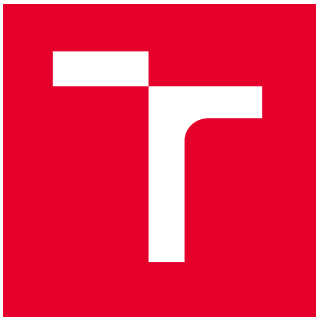


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

ANALYZÁTOR WEBOVÉ APLIKACE

ANALYZER OF WEB APPLICATION

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — **Tomáš Vašíček**

AUTHOR

VEDOUCÍ PRÁCE — **Ing. Zdeněk Martinásek, Ph.D.**

SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Tomáš Vašíček ***ID:*** 221581

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

NÁZEV TÉMATU:

Analyzátor webové aplikace

POKYNY PRO VYPRACOVÁNÍ:

Hlavním cílem bakalářské práce je výzkum a vývoj softwarového nástroje, který realizuje základní analýzu webové aplikace s cílem identifikace operačního systému, webového serveru, programovacího jazyka aj. Nástroj se na základě chování serveru pokusí identifikovat použité technologie a porovná je s informacemi uvedenými v HTTP(S) response hlavičkách. V teoretické části práce realizujte analýzu současného stavu problematiky, zaměřte se na dostupné nástroje (http print, webanalyzer aj.) a analyzujte dostupné databáze. Navrhněte vlastní programové vybavení, které umožní realizovat identifikaci webové aplikace a podporuje protokol HTTPS. Implementovaný nástroj musí být kompatibilní s nástrojem Penterep. Vlastní aplikaci otestujte a přehledně analyzujte dosažené výsledky.

DOPORUČENÁ LITERATURA:

[1] DEBAR, Hervé; TOMBINI, Elvis. WebAnalyzer: accurate detection of HTTP attack traces in web server logs. In: Annales des télécommunications. Springer-Verlag, 2006. p. 682-704.

[2] YONG, Binbin, et al. Web behavior detection based on deep neural network. In: 2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation. IEEE, 2018. p. 1911-1916.

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

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

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

ABSTRAKT

Tato práce se věnuje oblasti identifikace technologií webových aplikací. Hlavním cílem této práce je vyvinout softwarový nástroj pro základní analýzu webových aplikací, jehož cílem je identifikace operačního systému, webového serveru a programovacího jazyka webové aplikace. Dalším cílem práce je možnost integrace vyvinutého nástroje do platformy *Penterep* ve formě rozšiřujícího modulu. V teoretické části je obsažen úvod do problematiky se zaměřením na popis a manuální demonstraci identifikačních metod používaných při analýze webových aplikací. Praktická část prezentuje návrh, implementaci a testování nástroje. Součástí praktické části je i popis experimentálního pracoviště, které bylo vytvořeno s využitím platformy *Docker*.

KLÍČOVÁ SLOVA

dataset, Docker kontejner, identifikace, Penterep, Python, webová aplikace

ABSTRACT

This work focuses on the identification of technologies in web applications. The main goal is to develop a software tool for basic analysis of web applications that aims to identify the application's operating system, web server, and programming language. This work further aims to allow the developed tool to be integrated into the *Penterep* platform as an extension module. The theoretical part of this work is an introduction into the area of web application analysis with emphasis on describing and manually demonstrating the existing identification methods. The practical part presents the design, implementation and testing of the developed tool. This part also includes a description of an experimental environment that was built using the *Docker* platform.

KEYWORDS

dataset, Docker container, identification, Penterep, Python, web application

VAŠÍČEK, Tomáš. *Analyzátor webové aplikace*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2022, 74 s. Bakalářská práce. Vedoucí práce: Ing. Zdeněk Martinásek, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Tomáš Vašíček
VUT ID autora:	221581
Typ práce:	Bakalářská práce
Akademický rok:	2021/22
Téma závěrečné práce:	Analyzátor webové aplikace

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

Děkuji vedoucímu bakalářské práce panu Ing. Zdeňkovi Martináskovi, Ph.D. za odborné vedení, konzultace, a podnětné návrhy k práci. Rovněž děkuji technickému konzultantovi panu Romanovi Kümmelovi za konzultace a odborné rady.

Tato práce vznikla jako součást klíčové aktivity KA6 - Individuální výuka a zapojení studentů bakalářských a magisterských studijních programů do výzkumu v rámci projektu OP VVV Vytvoření double-degree doktorského studijního programu Elektronika a informační technologie a vytvoření doktorského studijního programu Informační bezpečnost, reg. č. CZ.02.2.69/0.0/0.0/16_018/0002575.

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

Projekt je spolufinancován Evropskou unií.

Obsah

Seznam obrázků

Seznam výpisů

Úvod

Webové aplikace se v současné době stávají stále více a více rozšířenými. Jsou využívány téměř ve všech průmyslových odvětvích, jako je například elektronické bankovnictví, internetové obchody, zdravotnictví či různá firemní softwarová řešení. Vzhledem k přirozenému dosahu webových aplikací je nutné efektivně zajišťovat jejich zabezpečení. Na bezpečnosti webových aplikací se podílí mnoho faktorů, ale tím nejdůležitějším, pro účely této práce, je penetrační testování. Smyslem penetračního testování je odhalení zranitelností testovaných systému a navržení jejich nápravy. Zranitelnosti jsou spjaty s konkrétními technologiemi, které daná webová aplikace používá. Identifikace těchto technologií je tedy pro úspěšné odhalení zranitelností zcela zásadní. [1, 2]

Cílem této bakalářské práce je výzkum a vývoj softwarového nástroje pro základní analýzu webové aplikace, jehož primárním cílem je identifikace operačního systému, webového serveru a programovícho jazyka webové aplikace. Podstatným cílem návrhu nástroje, a také cílem jeho implementace, je kompatibilita s platformou *Penterep*, která je nezbytná pro možnost integrace nástroje do této platformy formou rozšiřujícího modulu.

V teoretické části práce jsou uvedeny základní technologie a principy týkající se webových aplikací. Dále jsou zde představeny cíle analýzy webových aplikací následované detailními popisy a demonstracemi identifikačních metod využívaných během analýzy. Závěrem teoretické části jsou popsány a demonstrovány již existující analyzační nástroje společně s jejich identifikačními databázemi.

V úvodu praktické části práce je popsána příprava experimentálního pracoviště. Následující sekce pak obsahují návrh analyzátoru a jeho následnou implementaci. V rámci implementace analyzátoru je popsán i referenční dataset, jenž je základním stavebním prvkem celého procesu analýzy. Praktická část práce je zakončena testováním nástroje v experimentálním pracovišti a analýzou dosažených výsledků.

1 Teoretická část

V této kapitole jsou uvedeny základní pojmy, technologie a principy z oblasti analýzy webových aplikací. Podstatnou částí je zde popis identifikačních metod, které se vyžívají v rámci analýzy. Závěrem kapitoly jsou představeny existující analyzační nástroje.

1.1 Webová aplikace

Webová aplikace je softwarová aplikace, která je uživatelům přístupná standardně skrze webový prohlížeč. Nachází se na *webovém serveru*, který zprostředkovává komunikaci s uživateli. Uživatelské požadavky tedy putují na webový server, který je dále předává aplikaci. Aplikace zpracuje data od uživatele a provede potřebné úkony, jako je například čtení/zápis z/do databáze. Výsledky aplikace jsou následně opět předány webovému serveru, který je pošle nazpět uživateli. Komunikace uživatele se serverem standardně probíhá pomocí protokolů HTTP či HTTPS. [3]

1.1.1 Technologie webových aplikací

Webové aplikace sestávají primárně z klientské a serverové části. Obě části jsou tvořeny odlišnými programovacími jazyky, *frameworky*[1,2] či systémy.

Klientská část zajišťuje zobrazování obsahu webové aplikace a usnadňuje interakci uživatele s aplikací. Je tvořena téměř výhradně pomocí jazyků *HTML*[3], *CSS*[4] a *JavaScript*[5] a frameworky postavenými na těchto jazycích. Často používanými frameworky jsou například *Bootstrap*[6], *React*[7], *Vue*[8] nebo *AngularJS*[9]. [4, 5, 6]

Serverová část zajišťuje například obsluhu uživatelských požadavků, autentizaci uživatelů, správu uživatelských relací nebo databázové operace. Hojně využívanými programovacími jazyky pro realizaci serverové části jsou například *PHP*[10], *C#*[11]

[1] Server-side web frameworks: `https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Web_frameworks`

[2] Client-side web frameworks: `https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Introduction`

[3] *Hypertext Markup Language* (HTML): `https://html.spec.whatwg.org/multipage/`

[4] *Cascading Style Sheets* (CSS): `https://www.w3.org/Style/CSS/`

[5] JavaScript: (`https://www.ecma-international.org/publications-and-standards/standards/ecma-262/`)

[6] Bootstrap: `https://getbootstrap.com/`

[7] React: `https://reactjs.org/`

[8] Vue: `https://vuejs.org/`

[9] AngularJS: `https://angularjs.org/`

[10] *PHP: Hypertext Preprocessor* (PHP): `https://www.php.net/`

[11] C#: `https://docs.microsoft.com/en-us/dotnet/csharp/`

(v rámci frameworku *ASP.NET*[12]), *Java* (v rámci technologie *JSP*[13]) nebo *JavaScript* (v rámci běhového prostředí *Node.js*[14]). Mezi rozšířené frameworky patří například *ASP.NET*, *Express*[15], *Laravel*[16] nebo *Ruby on Rails*[17]. [7, 8, 9]

1.1.2 Webové servery

Pojem „webový server“ pokrývá 2 významy. První význam se vztahuje k hardwarové stránce počítačových systémů, druhý potom ke stránce softwarové.

Webový server (hardware) je počítač, na kterém se nachází operační systém spolu se softwarem webového serveru. Frekventovanými operačními systémy webových serverů jsou *Windows Server*[18] a Linuxové[19] distribuce *Ubuntu*[20], *CentOS*[21] a *Debian*[22]. [10]

Webový server (software) lze také označit jako „HTTP server“ a jedná se o program, který zprostředkovává komunikaci mezi uživatelem a webovou aplikací. Konkrétních webových serverů je mnoho, například *NGINX*[23], *Apache httpd*[24], *LiteSpeed*[25] nebo *Microsoft IIS*[26]. [11, 12]

1.1.3 Protokoly HTTP/HTTPS

Hypertext Transfer Protocol (HTTP) je bezstavový protokol typu klient-server. Protokol má několik verzí, z nichž dlouhodobě nejpoužívanější je verze 1.1. Následující popis protokolu se vztahuje právě k této verzi.

Komunikace mezi klientem a serverem probíhá pomocí jednotlivých zpráv, které jsou označovány jako „HTTP žádosti“ (zprávy klienta) a „HTTP odpovědi“ (zprávy serveru).

[12] ASP.NET: `https://dotnet.microsoft.com/en-us/apps/aspnet`
[13] *JavaServer Pages* (JSP): `https://docs.oracle.com/cd/E13222_01/wls/docs81/jsp/intro.html`
[14] Node.js: `https://nodejs.org/`
[15] Express: `https://expressjs.com/`
[16] Laravel: `https://laravel.com/`
[17] Ruby on Rails: `https://rubyonrails.org/`
[18] Windows Server: `https://www.microsoft.com/en-us/windows-server`
[19] Linux: `https://www.kernel.org/`
[20] Ubuntu: `https://ubuntu.com/`
[21] CentOS: `https://www.centos.org/`
[22] Debian: `https://www.debian.org/`
[23] NGINX: `https://www.nginx.com/`
[24] Apache HTTP Server: `https://httpd.apache.org/`
[25] LiteSpeed: `https://www.litespeedtech.com/products/litespeed-web-server`
[26] Microsoft *Internet Information Services* (IIS): `https://www.iis.net/`

HTTP žádost se skládá z názvu HTTP metody, resource path[27], verze protokolu HTTP a z hlaviček. U některých HTTP metod je součástí žádosti i tělo žádosti, které obsahuje data metody. Nejčastěji používané metody jsou *GET* a *POST*.

HTTP odpověď se skládá z verze protokolu HTTP, stavového kódu a zprávy, hlaviček a těla odpovědi. [13]

Obr. 1.1: Ukázka struktury HTTP zpráv

Na obrázku 1.1 je ukázka HTTP žádosti (nahoře) na server `www.vut.cz` a jeho následné odpovědi (dole). Hlavičky zpráv jsou zvýrazněné zeleným ohraničením, tělo odpovědi modrým ohraničením. Červeným ohraničením jsou zvýrazněny:

- HTTP metoda žádosti: „GET“
- resource path žádosti: „/vut/kontakty“
- stavový kód odpovědi: „200“
- stavová zpráva odpovědi: „OK“
- verze HTTP protokolu žádosti i odpovědi: „HTTP/1.1“

Hypertext Transfer Protocol Secure (HTTPS) je protokol HTTP rozšířený o šifrování přenášených zpráv. Šfirování v současné době zajišťuje protokol *Transport*

[27] „resource path“ lze přeložit do češtiny přibližně jako „cesta ke zdroji“, přičemž pojem „zdroj“ představuje nějaký soubor či službu nacházející se na webovém serveru (anglicky „resource“). Tyto české výrazy se ovšem v praxi moc nepoužívají a v kontextu této práce by byly spíše matoucí. Z těchto důvodů jsou v celém textu práce používány výhradně anglické výrazy „resource“ a „resource path“.

Layer Security (TLS)[28]. [14]

1.2 Analýza webových aplikací

Analýza webových aplikací spočívá v identifikaci konkrétních technologií používaných danou aplikací. Jejím cílem je zjistit programovací jazyk či framework, na kterém je aplikace postavena, webový server, který aplikace využívá, a operační systém serveru, na kterém se aplikace nachází[29]. Pro identifikaci těchto technologií jsou zkoumány charakteristické znaky a chování aplikace, které jsou v této práci označovány jako „indicie“. Indicií je cokoliv, co je charakteristické pro danou technologii aplikace, a tedy umožňuje její rozpoznání. Indicie ovšem nejsou unikátní, tudíž jejich znalost umožňuje technologii identifikovat pouze s určitou pravděpodobností. Proto je zapotřebí indicií nalézt co nejvíce, aby byla identifikace co nejpřesnější. [15, 16]

Ve zbytku této sekce jsou popsány různé indicie a jim příslušné identifikační metody, které lze při analýze využít pro identifikaci daných technologií. Pro demonstraci uvedených metod je využíván nástroj *Burp Suite*[30], jenž je detailněji popsán v sekci 1.3.1.

1.2.1 Identifikace webového serveru

Pro identifikaci webového serveru aplikace lze využít následující indicie [15]:

- Výskyt chybových stránek a jejich obsah
- Hlavičky HTTP odpovědí a jejich pořadí
- Výchozí uvítací stránka serveru
- Přítomnost ikon serveru *Apache httpd*
- Reakce na různé délky resource path
- Reakce na různé délky HTTP hlaviček

Výskyt chybových stránek a jejich obsah

Výchozí chybové stránky se u různých webových serverů liší jejich obsahem a příčinami, které je vyvolávají. Webový server odpoví chybovou stránkou, pokud dojde k nějaké chybě při zpracování HTTP žádosti klienta. Chyby mohou nastat z různých důvodů. Pozorovanými indiciemi jsou potom situace, při kterých chyby nastávají a samotný obsah webových stránek. [15]

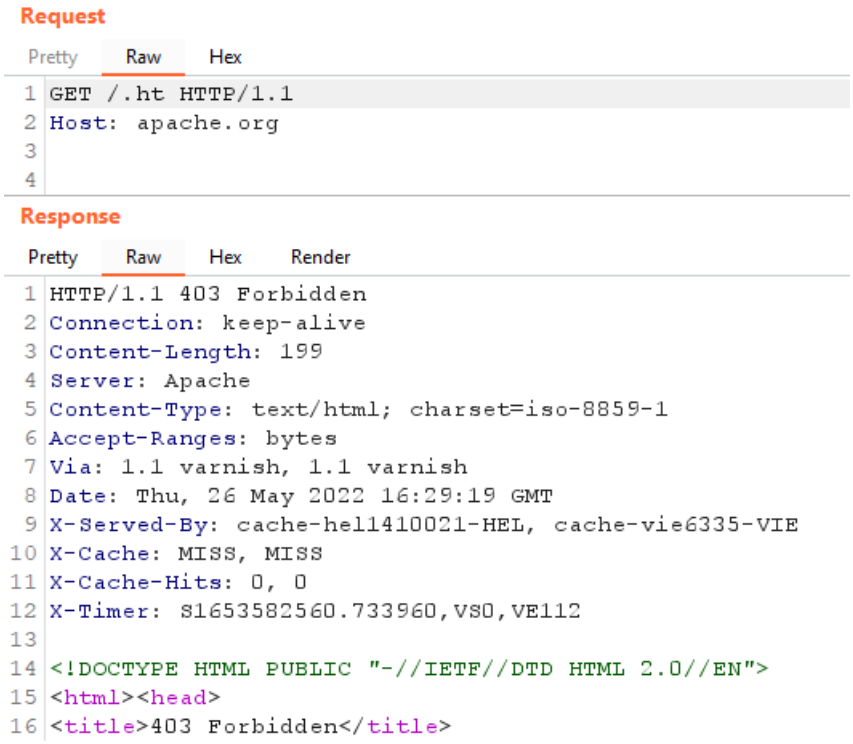
[28] TLS 1.3: `https://datatracker.ietf.org/doc/html/rfc8446`

[29] Výčet prvků, kterými se analýza může zabývat, není úplný. Jedná se pouze o prvky, jejichž znalost je zásadní pro účely penetračního testování.

[30] Burp Suite: `https://portswigger.net/burp`

Pro detailní úpravy HTTP žádostí a následné zkoumání HTTP odpovědí lze využít nástroj *Burp Repeater*.

HTTP žádost může obsahovat specifickou resource path, která na straně serveru způsobí chybu. Například webový server *Apache httpd* je ve výchozím nastavení nakonfigurován tak, aby na všechny HTTP žádosti, jejichž resource path začíná řetězcem „.ht“, odpovídal chybovou stránkou se stavovým kódem `403` a stavovou zprávou `Forbidden`. Příklad tohoto chování je uveden na obrázku 1.2. [17]

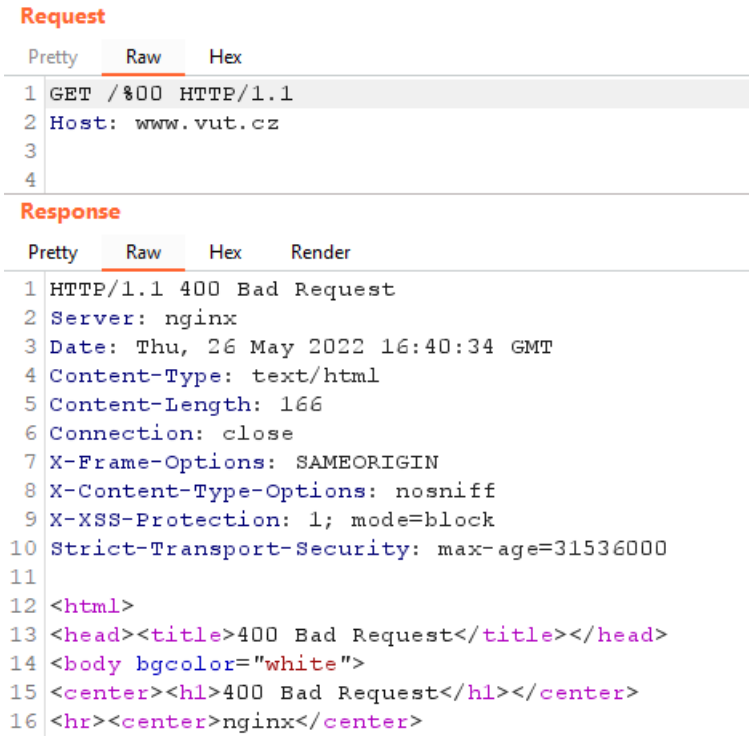


Obr. 1.2: Chybová stránka 403 webového serveru *Apache httpd*

Na obrázku 1.2 je v prostředí Burp Repeater zobrazena HTTP žádost klienta na webovou stránku *apache.org*, která běží na webovém serveru *Apache httpd*. Resource path specifikovaná žádostí je „.ht“. Webový server na takovouto žádost odpověděl chybovou stránkou, která hlásí, že přístup k žádanému resource byl odepřen.

Dalším způsobem, jak lze častokrát vyvolat chybovou hlášku webového serveru, je vložení nulového bajtu do resource path HTTP žádosti. Nulový bajt lze v HTTP žádosti reprezentovat pomocí URL kódování[31], a to konkrétně hodnotou `%00`.

[31] URL kódování: `https://developer.mozilla.org/en-US/docs/Glossary/percent-encoding`



Obr. 1.3: Odpověď webové stránky `www.vut.cz` na nulový bajt v žádosti

Na obrázku 1.3 se opět nachází prostředí nástroje *Burp Repeater*, ve kterém byla na webovou stránku `www.vut.cz` odeslána HTTP žádost s nulovým bajtem v resource path. Server odpověděl chybovou stránkou.

Posledním demonstrovaným způsobem, jak vyvolat chybovou hlášku webového serveru, je zaslání HTTP žádosti s neexistující metodou. Obvyklými HTTP metodami jsou metody GET, HEAD, POST, PUT, DELETE, CONNECT, OPTIONS, TRACE a PATCH. Webové servery jsou na tyto metody připraveny a umí na ně reagovat. Pokud klient zašle serveru HTTP žádost, obsahující neexistující metodu, může tím přimět server odpovědět chybovou hláškou. [18]

Pro zaslání HTTP žádosti s neexistující metodou lze opět využít nástroj *Burp Repeater*.

Request

Pretty Raw Hex

```
1 neexistujici / HTTP/1.1
2 Host: www.vut.cz
3
4
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 400 Bad Request
2 Server: nginx
3 Date: Thu, 26 May 2022 16:45:41 GMT
4 Content-Type: text/html
5 Content-Length: 166
6 Connection: close
7 X-Frame-Options: SAMEORIGIN
8 X-Content-Type-Options: nosniff
9 X-XSS-Protection: 1; mode=block
10 Strict-Transport-Security: max-age=31536000
11
12 <html>
13 <head><title>400 Bad Request</title></head>
14 <body bgcolor="white">
15 <center><h1>400 Bad Request</h1></center>
16 <hr><center>nginx</center>
```

Obr. 1.4: Reakce `www.vut.cz` na neexistující HTTP metodu

Request

Pretty Raw Hex

```
1 neexistujici / HTTP/1.1
2 Host: www.microsoft.com
3
4
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.0 400 Bad Request
2 Server: AkamaiGHost
3 Mime-Version: 1.0
4 Content-Type: text/html
5 Content-Length: 216
6 Expires: Thu, 26 May 2022 16:49:27 GMT
7 Date: Thu, 26 May 2022 16:49:27 GMT
8 Connection: close
9
10 <HTML><HEAD>
11 <TITLE>Bad Request</TITLE>
12 </HEAD><BODY>
13 <H1>Bad Request</H1>
14 Your browser sent a request that this server could not
15 Reference&#32;&#35;7&#46;c76ed417&#46;1653583767&#46;0
```

Obr. 1.5: Reakce `www.microsoft.com` na neexistující HTTP metodu

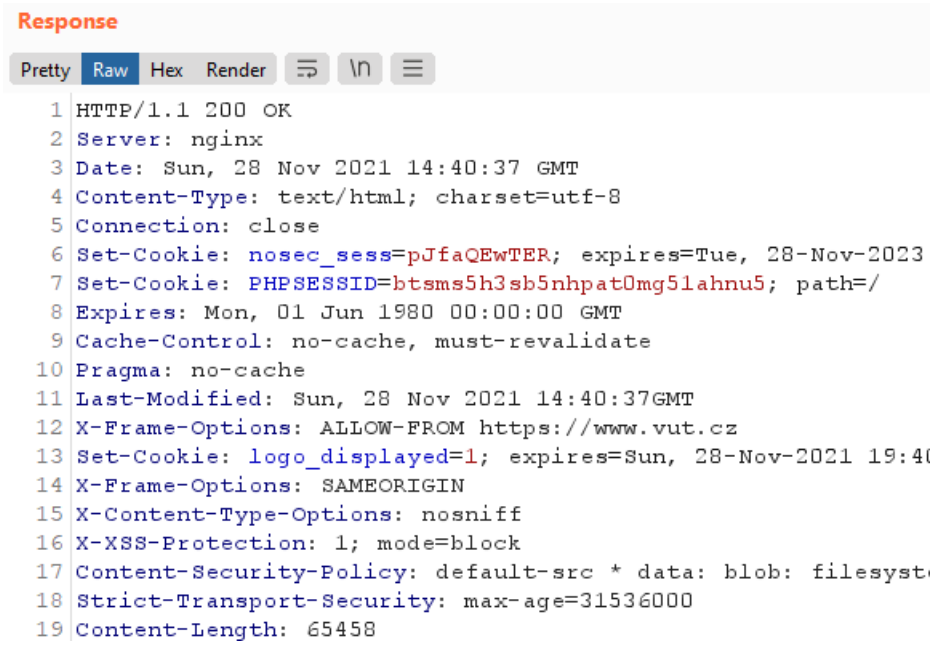
Na obrázcích 1.4 a 1.5 se nachází HTTP žádosti s metodou „neexistujici“ a HTTP

odpovědi serverů na tyto žádosti. Po inspekci obou obrázků je zřejmé, že těla HTTP odpovědí se navzájem liší. To naznačuje, že webové servery webových stránek `www.vut.cz` a `www.microsoft.com` jsou pravděpodobně odlišné.

Obsah chybových stránek může obsahovat informace, které napovídají, o jaký webový server se jedná. Například chybová stránka na obrázku 1.4 obsahuje text „nginx“, který napovídá, že webová stránka `www.vut.cz` běží na serveru *NGINX*[32].

Hlavičky HTTP odpovědí a jejich pořadí

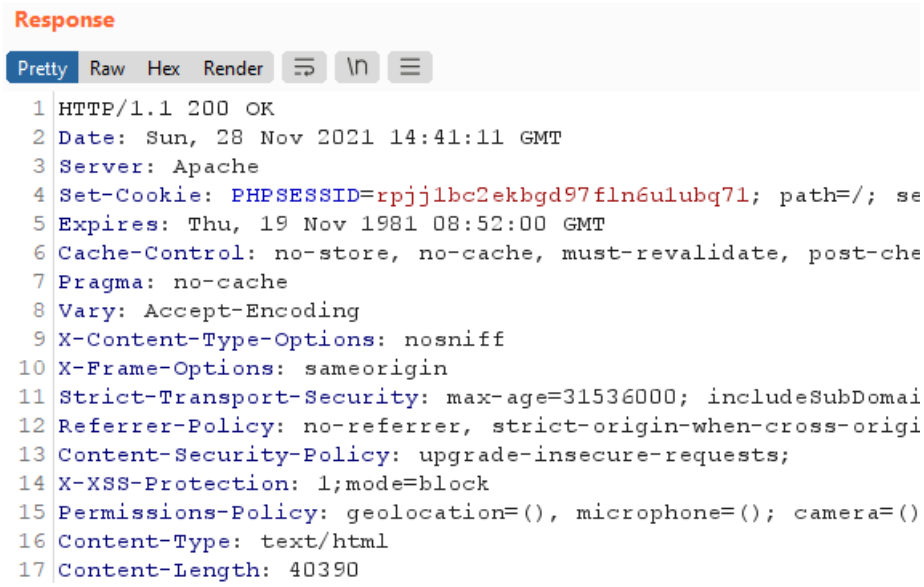
Hlavičky HTTP odpovědí a jejich pořadí se liší v závislosti na konkrétní implementaci webového serveru. Lze je prozkoumat například za použití nástroje *Burp Proxy*. [15]



Obr. 1.6: Hlavičky HTTP odpovědi stránky `www.vut.cz`

Na obrázku 1.6 je zachycena HTTP odpověď webové stránky `www.vut.cz` na HTTP GET žádost. Důležitou hlavičkou je zde hlavička `Server`, která sděluje, o jaký webový server se jedná. Pokud by údaj v této hlavičce byl vždy kompletní a pravdivý, byla by tato hlavička potenciálně jedinou potřebnou indicií k identifikaci webového serveru. Údaj v hlavičce ovšem nemusí být kompletní, lze jej zfalšovat nebo hlavičku zcela vynechat. Z tohoto důvodu se nelze na hlavičku `Server` spoléhat a je nutné zkoumat i další indicie webového serveru. [15]

[32]NGINX: `https://www.nginx.com/`



Obr. 1.7: Hlavičky HTTP odpovědi stránky `www.soom.cz`

Na obrázku 1.7 je HTTP odpověď webové stránky `www.soom.cz`, opět na HTTP GET žádost. Při srovnání s obrázkem 1.6 lze vidět, že některé hlavičky mají HTTP odpovědi společné, ale jejich pořadí je jiné. Společnými hlavičkami jsou například `Server`, `Date` a `Expires`. Odpověď `www.vut.cz` (obr. 1.6) tyto hlavičky obsahuje v pořadí `Server`, `Date`, `Expires`. Odpověď `www.soom.cz` (obr. 1.7) tyto hlavičky obsahuje v pořadí `Date`, `Server`, `Expires`.

Z odlišného pořadí společných hlaviček lze usoudit, že webové stránky `www.vut.cz` a `www.soom.cz` pravděpodobně používají odlišné webové servery.

Výchozí uvítací stránka serveru

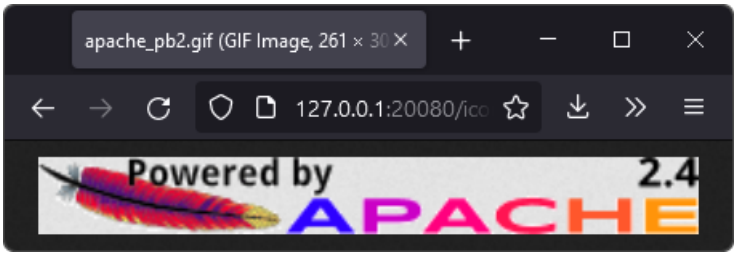
Webové servery mají své výchozí uvítací stránky, které jsou přítomné po čerstvé instalaci serveru a pro účely identifikace slouží jako velmi silná indicie. Jestliže po instalaci není webový server dostatečně nakonfigurován, je možné jeho výchozí uvítací stránku vyvolat zasláním HTTP žádosti, jejíž hlavička `Host` obsahuje IP adresu serveru.

Příklad získání této indicie je demonstrován na webové stránce `www.rhinorealty.com`. Při zaslání neupravené HTTP žádosti server vrátí vlastní očekávaný obsah aplikace. Pokud ovšem je na tento server zaslána žádost s upravenou hlavičkou `Host`, odpoví server výchozí uvítací stránkou webového serveru *Microsoft IIS 7.0* (obr. 1.8).

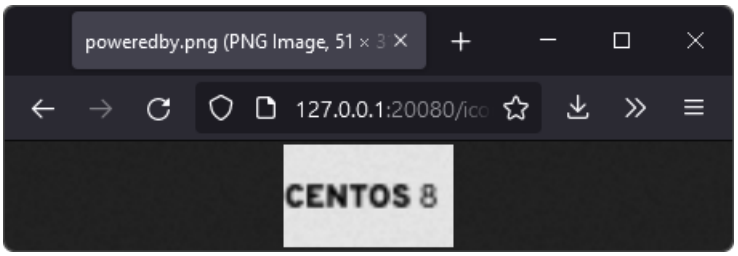
Obr. 1.8: Výchozí uvítací stránka webové stránky `www.rhinorealty.com`

Přítomnost ikon serveru Apache httpd

Webový server *Apache httpd* je charakteristický tím, že se na něm nachází resource `/icons/`. Jedná se o alias pro adresář obsahující výchozí ikony webového serveru a jeho přítomnost je jednou z indicií pro webový server *Apache httpd*. Některé z ikon, které adresář obsahuje, jsou dalšími identifikačními indiciemi pro různé technologie. Adresář dále může obsahovat ikony `apache_pb2.gif` a `poweredby.png`, které, pokud jsou přítomné, slouží jako další identifikační indicie. Ikona `apache_pb2.gif` poskytuje informace o webovém serveru a ikona `poweredby.png` informace o operačním systému. Obrázky 1.9 a 1.10 zobrazují příklady obou těchto ikon.



Obr. 1.9: Ikona `apache_pb2.gif`



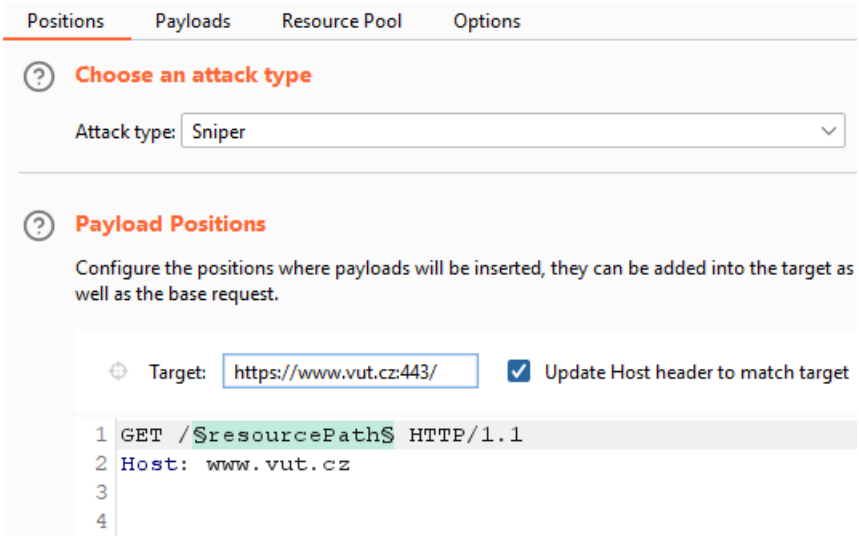
Obr. 1.10: Ikona `poweredby.png`

Reakce na různé délky resource path

Resource path v HTTP žádosti obvykle nemůže být neomezeně dlouhá. Webový server má zpravidla nastavenou určitou hraniční délku, při jejímž překročení odpoví na žádost nějakou chybovou stránkou, či neodpoví vůbec. Tyto hraniční délky se u různých webových serverů liší, a tedy se dají považovat za další pozorovanou indicii při identifikaci webového serveru. [19, 20, 21]

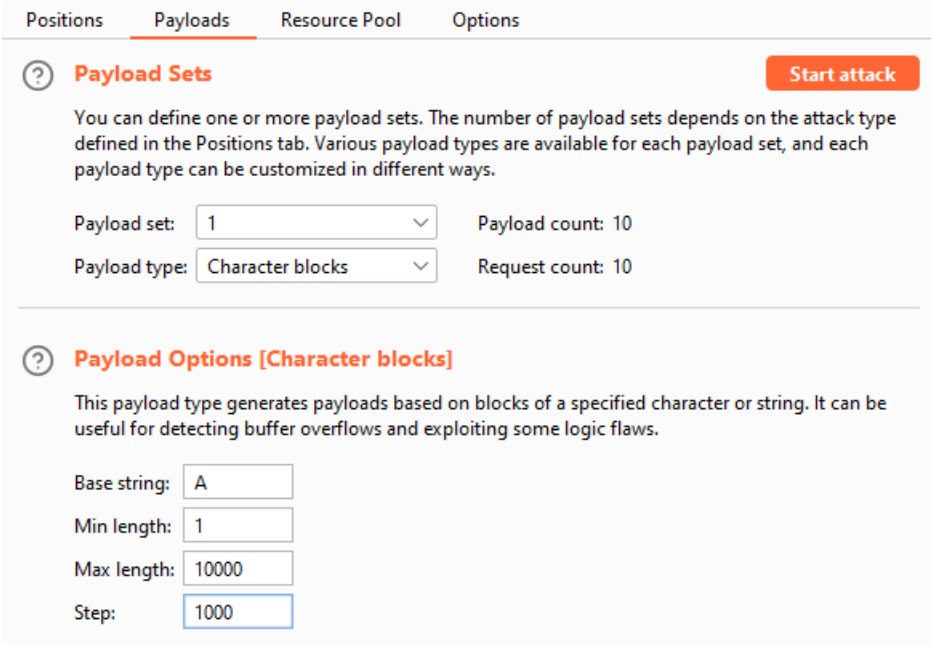
Pro demonstraci nalezení hraniční délky resource path je v následujícím textu použit nástroj *Burp Intruder*, pomocí kterého lze zasílat HTTP žádosti, jejichž resource path řetězce se postupně prodlužují.

Pro spuštění testu je nutné v prostředí *Burp Intruder* specifikovat pravidlo, na základě kterého se budou generovat a odesílat žádosti.



Obr. 1.11: Určení pozice v HTTP žádosti

Na obrázku 1.11 je vidět nastavení cílené pozice pro režim „Sniper“, při kterém jsou na označené místo, to je mezi dvěma symboly $, dosazovány hodnoty specifikované v následujícím kroku nastavení (obr. 1.12).

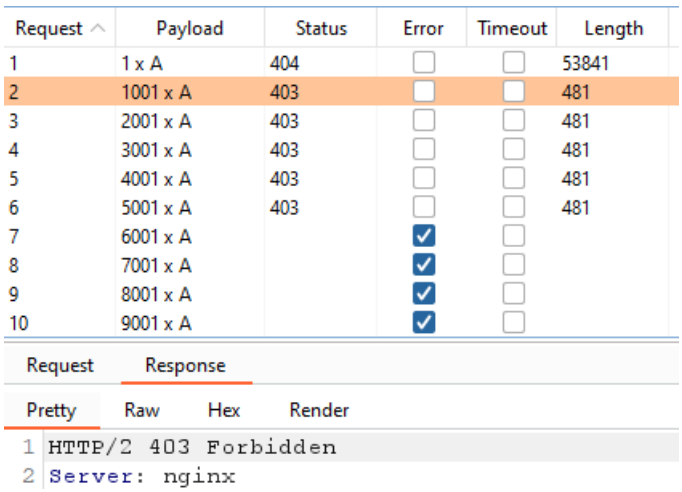


Obr. 1.12: Určení hodnot dosazovaných na pozici v HTTP žádosti

Sekce „Payload Sets“ určuje o jaký typ hodnot se jedná. Možných typů je několik, ale v tomto případě je nastaven typ „Character blocks“, při kterém jsou jako hodnoty použity opakující se textové řetězce.

Sekce „Payload Options“ určuje, jaké textové řetězce budou generovány. V tomto případě budou generována písmena „A“ v postupně zvyšujícím se množství. V první žádosti bude odesláno 1 písmeno „A“, v poslední žádosti bude odesláno 10 000 písmen „A“. Po každé odeslané žádosti se počet písmen zvýší o 1 000.

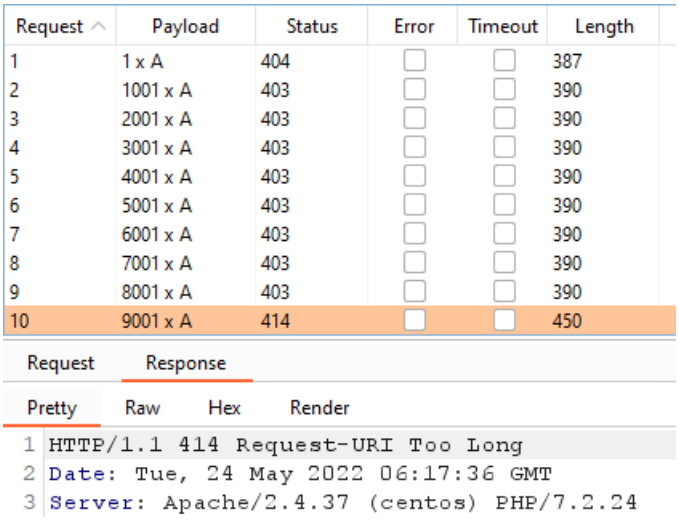
S těmito nastaveními je možné test spustit.



Obr. 1.13: Výsledky testu pro webovou stránku `www.vut.cz`

Na obrázku 1.13 jsou zobrazeny výsledky testu pro webovou stránku `www.vut.cz`. Sloupec „Payload“ specifikuje, jaká hodnota byla v dané HTTP žádosti dosazena na předešle specifikovanou pozici. Sloupec „Status“ obsahuje stavový kód HTTP odpovědi na danou žádost, který je sledovaným ukazatelem pro zjištění hraniční délky resource path. Z obrázku je patrné, že webová stránka `www.vut.cz` odpovídala na různé délky resource path různými stavovými kódy. Přibližnými délkami resource path, při kterých došlo ke změně odpovědi serveru, jsou délky 1 000 a 6 000. Na délku menší než 1 000 server odpověděl kódem `404`. Na délky mezi 1 000 a 6 000 server odpovídal kódem `403`. Na délku větší než 6 000 server neodpověděl vůbec.

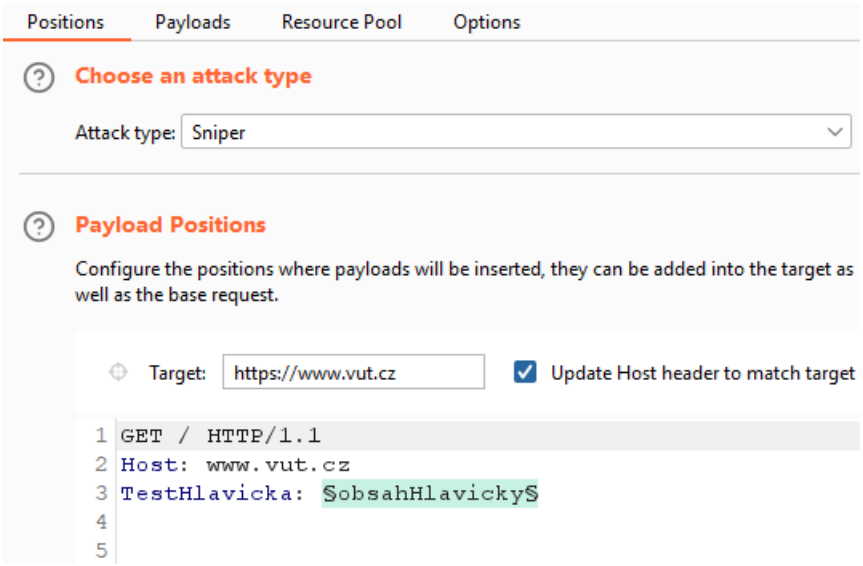
Hraniční délky, při kterých dochází ke změně chování serveru, se u rozdílných webových serverů liší. Stejně tak jsou rozdílné samotné odpovědi, kterými servery reagují. Pro ukázku jsou na následujícím obrázku uvedeny výsledky nástroje pro webový server *Apache httpd*. Nástroj byl spuštěn s identickými nastaveními jako v předešlém příkladu, pouze s odlišnou cílovou webovou stránkou.



Obr. 1.14: Výsledky nástroje pro webový server *Apache httpd*

Reakce na různé délky HTTP hlaviček

Pro délky HTTP hlaviček žádostí platí obdobná pravidla jako pro již zmíněné délky resource path. Opět platí, že existují nějaké hraniční délky hlaviček, při jejichž překročení začne webový server reagovat jinou odpovědí. Toto chování lze opět demonstrovat pomocí nástroje *Burp Intruder*. Následující obrázky zobrazují nastavení a spuštění nástroje, společně s jeho výsledky.



Obr. 1.15: Nastavení délky hlaviček v nástroji *Burp Intruder*

Request	Payload	Status	Error	Timeout	Length
1	1 x A	200	☐	☐	69510
2	1001 x A	200	☐	☐	69510
3	2001 x A	200	☐	☐	69510
4	3001 x A	200	☐	☐	69510
5	4001 x A	200	☐	☐	69510
6	5001 x A	200	☐	☐	69510
7	6001 x A		☑	☐	
8	7001 x A		☑	☐	
9	8001 x A		☑	☐	
10	9001 x A		☑	☐	

Obr. 1.16: Výsledky nástroje *Burp Intruder* pro test délek hlaviček

1.2.2 Identifikace programovacího jazyka

Identifikace programovacího jazyka či frameworku lze docílit nalezením těchto indicií[15]:

- Přípony souborů
- Hlavičky HTTP odpovědí
- Cookies

Přípony souborů

Hodnoty resource path mohou být webovou aplikací mapovány buď přímo na existující soubory systému, anebo na funkce definované ve zdrojovém kódu webové aplikace. Pokud webová aplikace mapuje resource path na existující soubory, mají tyto

soubory většinou nějakou příponu, která je závislá na programovacím jazyce, ve kterém je webová aplikace naprogramována. Zjištění této přípony je jednou z indicií pro identifikaci programovacího jazyka dané aplikace. [22]

Jedním ze způsobů, jak příponu nalézt, je „hrubou silou“ zkoušet pravděpodobné kombinace názvu souboru a jeho přípony. Obvyklými jmény zdrojových souborů jsou například „index“, „default“ nebo „login“. Frekventovanými příponami zdrojových souborů jsou například „.php“, „.aspx“ či „.jsp“. Pokud server při nějaké z žádostí úspěšně nalezne soubor s danou příponou, znamená to, že webová stránka je pravděpodobně vytvořena v programovacím jazyce odpovídajícím dané příponě. [23]

Hlavičky HTTP odpovědí

Některé HTTP hlavičky odpovědí mohou obsahovat informace o programovacím jazyce či frameworku webové aplikace. Takovou hlavičkou je například `X-Powered-By`. Její obsah může napovídat v jakém jazyce, či dokonce v jaké jeho konkrétní verzi, je webová aplikace naprogramována. Tyto informace jsou velice užitečné, avšak nelze se na ně spoléhat, neboť tato hlavička se nemusí vždy nacházet v HTTP odpovědi aplikace nebo její obsah může být webovou aplikací úmyslně zfalšován či jinak pozměněn. Pro zkoumání HTTP hlaviček lze využít nástroj *Burp Proxy*. [24]

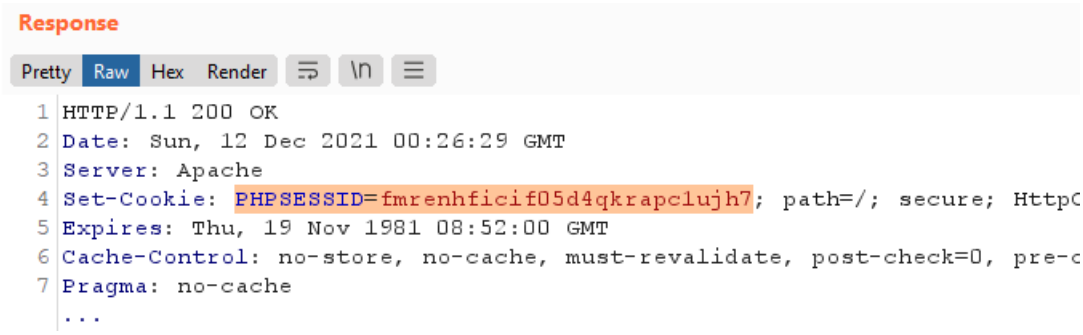
Obr. 1.17: HTTP hlavička `X-Powered-By`

Na obrázku 1.17 se nachází HTTP odpověď webové stránky `www.idnes.cz` v prostředí nástroje Burp Proxy. Odpověď obsahuje HTTP hlavičku `X-Powered-By`. V tomto případě hlavička neobsahuje žádné užitečné informace pro identifikaci programovacího jazyka aplikace.

Cookies

Cookies jsou data vytvořená webovou aplikací a předána klientovi, kde jsou uložena do klientova webového prohlížeče. Následně jsou zasílána zpět webové aplikaci jako součást HTTP žádostí a umožňují personalizovat obsah webové stránky, uchovávat relační informace nebo zaznamenávat a analyzovat chování uživatelů na webové stránce. Každý cookie má svůj název a hodnotu. [25]

Některé technologie webových aplikací používají charakteristické názvy *cookies*. Například programovací jazyk *PHP* používá cookie `PHPSESSID`, redakční systém *WordPress*[33] má cookie `wp-settings` a platforma *Laravel*[34] používá cookie `laravel_session`. Cookies lze opět zkoumat pomocí nástroje *Burp Proxy*. [16]



Obr. 1.18: Cookie `PHPSESSID` na webové stránce `www.soom.cz`

Obrázek 1.18 zobrazuje HTTP odpověď serveru `www.soom.cz` v prostředí Burp Proxy. Na obrázku je zvýrazněný text „PHPSESSID=...“. Tento text představuje cookie vytvořený webovou stránkou `www.soom.cz` a z jeho názvu lze usoudit, že webová stránka používá programovací jazyk *PHP*.

Názvy cookies ovšem také nejsou jistou cestou k identifikaci programovacího jazyka webové aplikace, neboť jejich názvy se dají změnit v konfiguraci webové aplikace. [16]

1.2.3 Identifikace operačního systému

Pro účely základní analýzy webové aplikace je důležité identifikovat, zda operační systém aplikace patří do rodiny operačních systémů Microsoft Windows, nebo se jedná o 1 z distribucí operačního systému Linux. Toho lze docílit sledováním následujících indicií:

- Rozlišování velkých a malých písmen
- Reakce na žádosti o resource s rezervovaným jménem
- Implementace protokolů TCP/IP

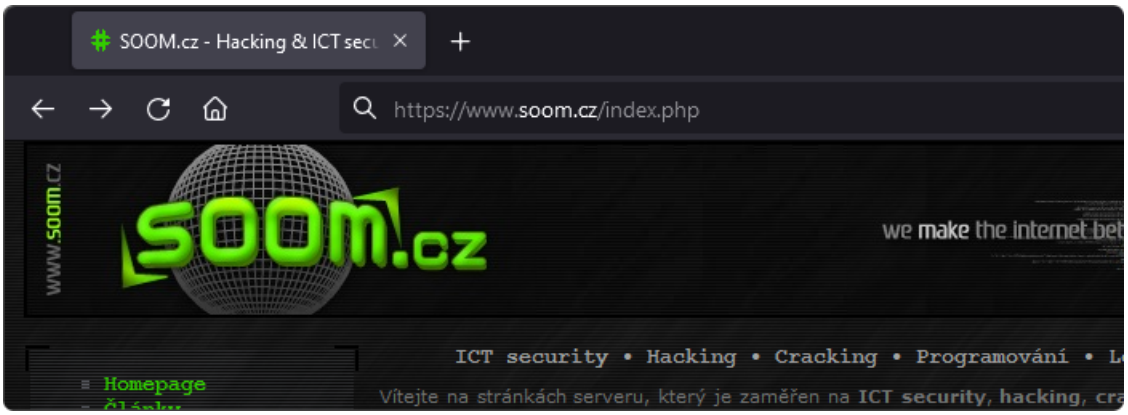
Rozlišování velkých a malých písmen

Souborové systémy operačních systémů Microsoft Windows nerozlišují malá a velká písmena, zatímco souborové systémy Linux/UNIX distribucí malá a velká písmena rozlišují. Na základě tohoto rozdílu lze tedy rozlišit, na jakém operačním systému je

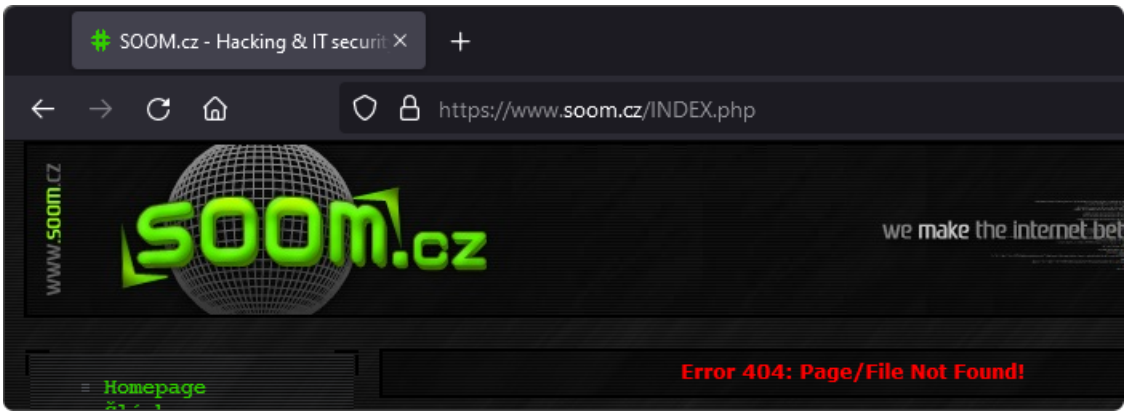
[33] WordPress: `https://wordpress.com/`
[34] Laravel: `https://laravel.com/`

webová aplikace provozována. Pomocí této indicie nelze určit konkrétní verzi operačního systému, neboť se v tomto případě všechny verze chovají stejně. [26]

Pro zkoumání této indicie lze využít standardní webový prohlížeč.



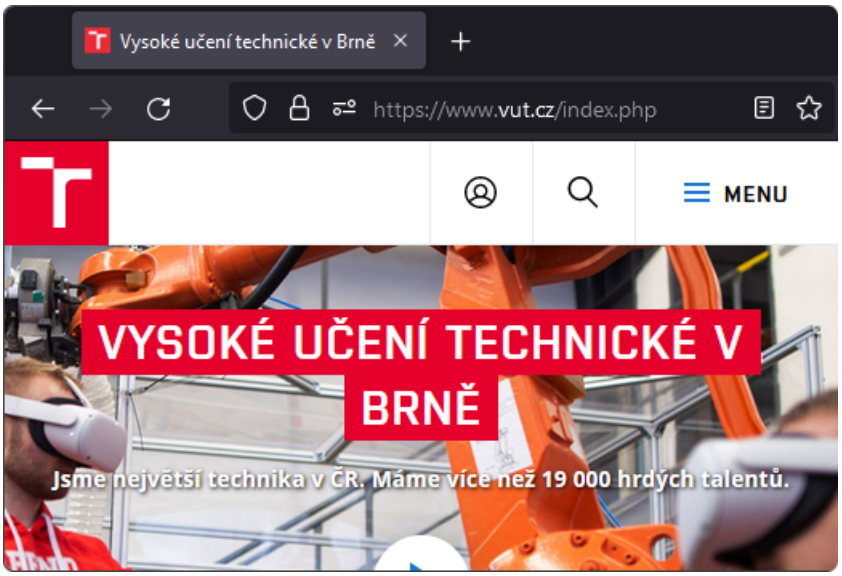
Obr. 1.19: Odpověď stránky `www.soom.cz` na resource `index.php`



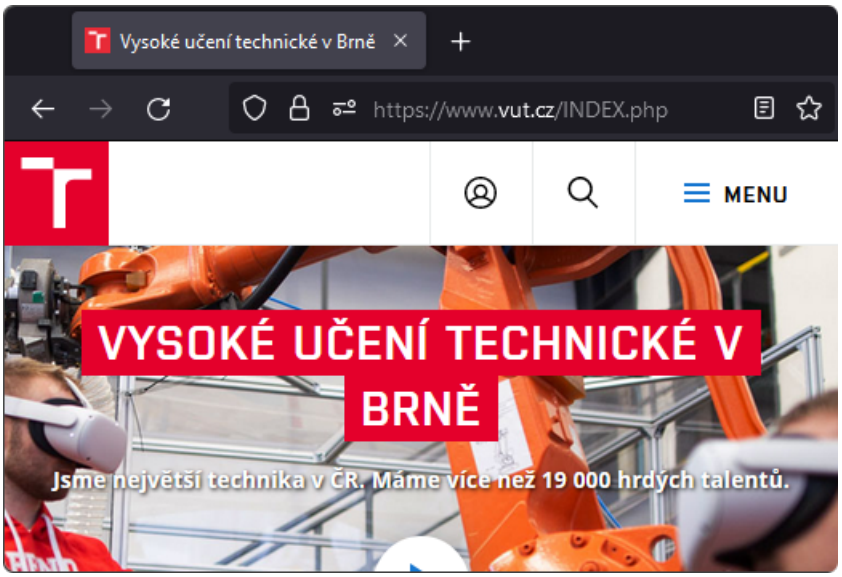
Obr. 1.20: Odpověď stránky `www.soom.cz` na resource `INDEX.php`

Obrázky 1.19 a 1.20 demonstrují citlivost webové stránky `www.soom.cz` na velká a malá písmena. Na obrázku 1.19 jde vidět, že resource `index.php` byl serverem úspěšně nalezen. Při žádosti na resource `INDEX.php` server ovšem odpověděl chybovou stránkou (obr. 1.20), neboť žádaný resource nenalezl.

Opačné chování lze názorně předvést na webové stránce `www.vut.cz`.

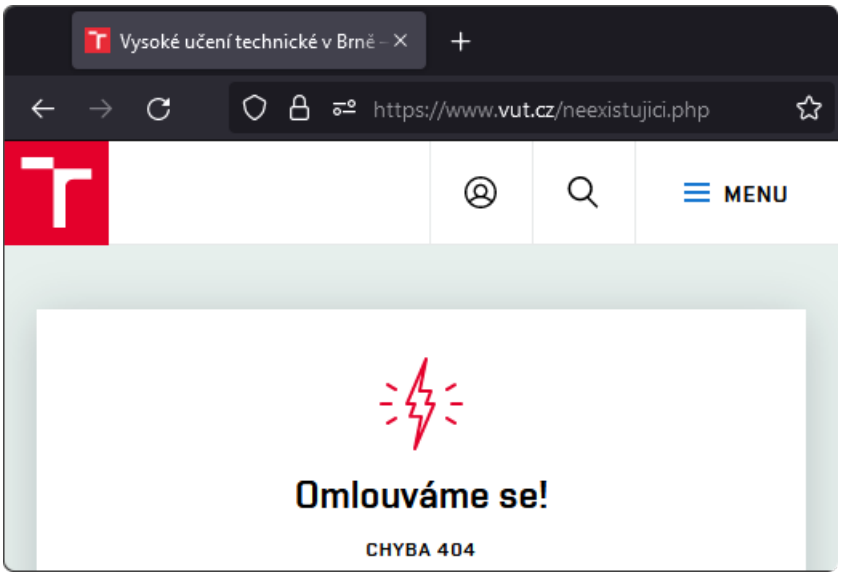


Obr. 1.21: Odpověď stránky `www.vut.cz` na `index.php`



Obr. 1.22: Odpověď stránky `www.vut.cz` na `INDEX.php`

Na obrázcích 1.21 i 1.22 server nalezl žádaný resource, přestože v prvním případě byl vyžádán `index.php` a v druhém případě `INDEX.php`. Kdyby server žádaný resource nenalezl, odpověděl by tak jako na následujícím obrázku.



Obr. 1.23: Odpověď stránky `www.vut.cz` při nenalezení žádaného resource

Demonstrovaná chování stránek `www.soom.cz` a `www.vut.cz` mohou být ukazatelem toho, že tyto stránky jsou provozovány na odlišných operačních systémech.

Reakce na žádosti o resource s rezervovaným jménem

Na operačních systémech Microsoft Windows existuje několik speciálních jmen pro pojmenovávání souborů v souborovém systému, které jsou rezervovány pro systémové účely. Těmito jmény jsou například „CON“, „PRN“, „AUX“, „COM1“ nebo „LPT1“. Pokud na webový server s operačním systémem Windows dorazí HTTP žádost o resource s jedním z těchto rezervovaných jmen, server může zareagovat chybovou hláškou. [27]

Implementace protokolů TCP/IP

Protokoly *Transmission Control Protocol* (TCP) a *Internet Protocol* (IP) jsou protokoly transportní a síťové vrstvy modelů *TCP/IP*[35] a *ISO/OSI*[36] sloužící pro síťovou komunikaci. Tyto protokoly nejsou zcela striktně definované a umožňují, aby se jejich implementace lišily v různých detailech. K těmto detailům patří například hodnota *Time to Live* (TTL) v hlavičce IP paketu, fragmentace IP paketů nebo délka okna TCP segmentu. Takovýchto malých, ba i nepatrných, rozdílů je celá řada a je velmi obtížné je zkoumat bez použití nějakého specializovaného softwarového nástroje.

[35] TCP/IP: `https://www.fortinet.com/resources/cyberglossary/tcp-ip`
[36] ISO/OSI: `https://www.iso.org/ics/35.100/x/`

Takovým nástrojem je například *Nmap*[37]. Bližší popis tohoto nástroje se nachází v sekci 1.3.5. [28] [29, Kapitola 8. Remote OS Detection]

Pro spuštění identifikace operačního systému serveru `www.www.vut.cz` nástrojem *Nmap* slouží následující příkaz (tento příkaz vyžaduje oprávnění správce systému).

Výpis 1.1: Spuštění detekce operačního systému nástrojem *Nmap*

```
nmap www.vut.cz -O
```

Výstupem tohoto příkazu jsou nalezené síťové služby na webovém serveru, ale i kýžená detekce operačního systému.

Výpis 1.2: Výstup nástroje Nmap

```
Nmap scan report for www.vut.cz (147.229.2.90)
Host is up (0.0013s latency).
rDNS record for 147.229.2.90: piranha.ro.vutbr.cz
Not shown: 998 filtered ports

PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https
Warning: OSScan results may be unreliable because we could not find
    at least 1 open and 1 closed port
Device type: specialized|WAP|phone
Running: iPXE 1.X, Linux 2.4.X|2.6.X, Sony Ericsson embedded
OS CPE: cpe:/o:ipxe:ipxe:1.0.0%2b cpe:/o:linux:linux_kernel:2.4.20
   cpe:/o:linux:linux_kernel:2.6.22 cpe:/h:sonyericsson:u8i_vivaz
OS details: iPXE 1.0.0+, Tomato 1.28 (Linux 2.4.20), Tomato
   firmware (Linux 2.6.22), Sony Ericsson U8i Vivaz mobile phone
```

Nástroji se v tomto případě nepodařilo s jistotou určit operační systém webového serveru. Výsledkem je několik operačních systémů, které nástroj vyhodnotil jako pravděpodobné.

[37] Nmap: `https://nmap.org/`

1.2.4 Shrnutí představených indicií

Tab. 1.1: Shrnutí představených identifikačních indicií

Identifikační indicie		
Webový server	Programovací jazyk / Framework	Operační systém
Výskyt chybových stránek	Přípony souborů	Rozlišování velkých a malých písmen
Obsah chybových stránek	Hlavičky HTTP odpovědí	Reakce na žádosti o resource s rezervovaným jménem
Výchozí uvítací stránka	Cookies	Implementace protokolů TCP/IP
Ikony serveru *Apache httpd*		
Odpovědi na různé délky resource path		
Odpovědi na různé délky HTTP hlaviček		

1.3 Dostupné nástroje

V současné době jsou dostupné různé nástroje sloužící k usnadnění či automatizaci analýzy webových aplikací. Některé z nich jsou popsány v této sekci.

1.3.1 Burp Suite

Burp Suite je sada nástrojů od společnosti PortSwigger pro testování bezpečnosti webových aplikací. V současné době je *Burp Suite* v oblasti manuální analýzy webových aplikací prakticky standardem. Obsahuje několik základních nástrojů a je dále rozšiřitelný o komunitní nástroje. Je nabízen ve třech edicích: *Community*, *Professional* a *Enterprise*. Edice Community má určitá omezení a neposkytuje uživateli tolik možností jako ostatní verze, ale je zdarma a pro manuální testování je zcela dostačující. Ostatní edice jsou placené. [30]

Pro účely demonstrace metod analýzy webových aplikací byla v této práci použita edice Community. Konkrétními použitými nástroji byly *Burp Proxy*, *Burp Repeater* a *Burp Intruder*. Názorné ukázky těchto nástrojů se nachází v sekci 1.2.

Burp Proxy funguje jako webový proxy server mezi uživatelem a cílovým webovým serverem. Umožňuje zachytávat, zkoumat a upravovat procházející komunikaci. [31]

Burp Repeater slouží k manuální úpravě a opětovnému odesílání HTTP žádostí. Ve spolupráci s *Burp Proxy* umožňuje manipulaci s předešle zachycenými žádostmi. [32]

Burp Intruder je nástroj pro částečnou automatizaci testů webových aplikací. Umožňuje automaticky odesílat HTTP žádosti, které pozměňuje na základě definovaných pravidel, a následně analyzovat přijaté odpovědi. [33]

Nedostatky nástroje Burp Suite

Plnohodnotná automatizovaná analýza webových aplikací je součástí jen komerčních edicí Professional a Enterprise. Community edice umožňuje částečnou automatizaci analýzy pomocí nástroje *Burp Intruder*, ale ten je v Community edici dostupný pouze ve formě demo verze a jeho rychlost je tak značně omezena.

1.3.2 Wappalyzer

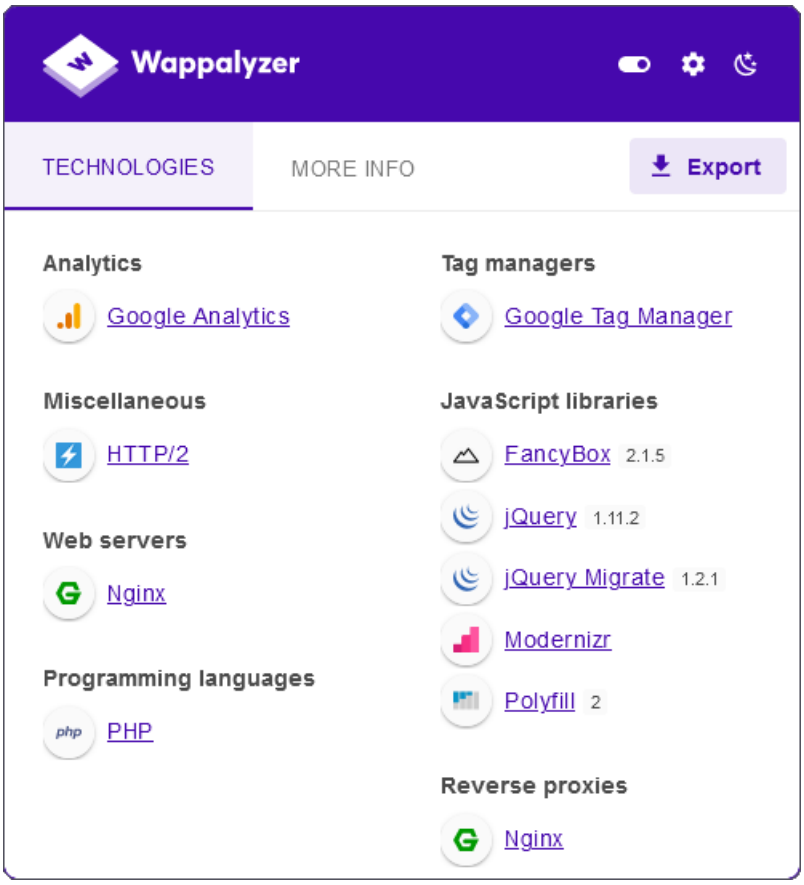
Wappalyzer[38] je nástroj pro automatickou analýzu webových aplikací, který pro analýzu využívá výhradně pasivní inspekci obsahu HTTP komunikace s analyzovanou aplikací. V komunikaci vyhledává textové řetězce a porovnává je s jeho rozsáhlou databází[39] známých ověřených řetězců, kde každý řetězec náleží nějaké konkrétní technologii. Na základě kontroly těchto řetězců je nástroj schopen poměrně úspěšně odhalit technologie, které webová aplikace využívá. [34]

Nástroj je dostupný ve 2 variantách, a to jako rozšíření do webového prohlížeče a jako nástroj pro příkazovou řádku. Varianta webového rozšíření nabízí uživatelsky přívětivé rozhraní. Varianta pro příkazovou řádku poskytuje výsledky ve formátu JSON[40], tudíž je vhodnější pro využití v rámci automatizovaných skriptů či programů. Obrázek 1.24 a výpis 1.3 znázorňují výstupy nástroje pro server `www.vut.cz`.

[38] Wappalyzer: `https://www.wappalyzer.com/`

[39] Wappalyzer databáze: `https://github.com/wappalyzer/wappalyzer/tree/master/src/technologies`

[40] *JavaScript Object Notation* (JSON): `https://www.json.org/json-en.html`



Obr. 1.24: Výstup webového rozšíření *Wappalyzer*

Výpis 1.3: Upravený výstup nástroje *Wappalyzer* pro příkazovou řádku

```
{
    "urls": {
        "https://www.vut.cz/": {
            "status": 200
        }
    },
    "technologies": [{
            "slug": "php",
            "name": "PHP",
            "confidence": 100,
            "version": null,
            ...
                }
            ]
        }, {
            "slug": "nginx",
            "name": "Nginx",
            "confidence": 100,
            "version": null,
```

```
            ...
                }
            ]
        },
        ...
    ]
}
```

Nedostatky nástroje Wappalyzer

Nástroj *Wappalyzer* provádí pouze pasivní analýzu komunikace mezi uživatelem a webovou aplikací. Nevyužívá řady užitečných identifikačních metod, jako jsou například zkoumání reakcí na HTTP žádosti s dlouhou resource path či s dlouhými HTTP hlavičkami.

1.3.3 HTTPrint

HTTPrint[41] se zaměřuje výhradně na identifikaci webového serveru aplikace. Zasílá na server specifické HTTP žádosti a zkoumá odpovědi serveru – jedná se tedy o aktivní analýzu. Zasílané žádosti obsahují například neexistující HTTP metody nebo neexistující verze HTTP protokolu. Zkoumanými částmi HTTP odpovědí jsou například pořadí HTTP hlaviček nebo stavové kódy a zprávy.

Nástroj z poznatků získaných během komunikace se serverem vyhotoví textový řetězec v hexadecimálním ASCII kódování, který porovná se svou databází známých ověřených řetězců. Každý řetězec v databázi odpovídá nějakému konkrétnímu webovému serveru. Nástroj se tedy na základě shody textových řetězců z databáze a z analýzy pokusí identifikovat daný webový server. [35]

Výpis 1.4: Zkrácený výstup nástroje *HTTPrint* pro server `www.postsignum.cz`

```
httprint v0.301 (beta) - web server fingerprinting tool
(c) 2003-2005 net-square solutions pvt. ltd. - see readme.txt
http://net-square.com/httprint/
httprint@net-square.com

Finger Printing on http://www.postsignum.cz:80/
Finger Printing Completed on http://www.postsignum.cz:80/
--------------------------------------------------
Host: www.postsignum.cz
Derived Signature:
Apache
811C9DC56ED3C295811C9DC5811C9DC5811C9DC5505FCFE84276E4BB811C9DC5
0D7645B5811C9DC5811C9DC5CD37187C811C9DC5811C9DC5811C9DC5811C9DC5
```

[41] HTTPrint: `https://www.net-square.com/httprint.html`

```
6ED3C295E2CE6923E2CE6923811C9DC5E2CE6927811C9DC56ED3C295811C9DC5
6ED3C2956ED3C2952A200B4CE2CE6923E2CE69236ED3C2956ED3C295E2CE6923
E2CE69236ED3C295811C9DC5E2CE6927E2CE6923

Banner Reported: Apache
Banner Deduced: Apache/2.0.x
Score: 84
Confidence: 50.60
------------------------
Scores:
Apache/2.0.x: 84 50.60
Apache/1.3.26: 83 48.42
Apache/1.3.27: 82 46.30
Apache/1.3.[4-24]: 81 44.24
Apache/1.3.[1-3]: 76 34.81
TUX/2.0 (Linux): 72 28.28
thttpd: 72 28.28
Microsoft-IIS/6.0: 70 25.32
...
Snap Appliances, Inc./3.x: 0  0.00
Linksys AP1: 0  0.00
Ubicom/1.1: 0  0.00

-------------------------------------------------
```

Nedostatky nástroje HTTPrint

Nástroj *HTTPrint* je v současné době již zastaralý a je využíván spíše výjimečně, jelikož nepodporuje protokol TLS, a tedy je použitelný pouze na webové aplikace používající nešifrovaný protokol HTTP. Navíc se nástroj zaměřuje pouze na identifikaci webového serveru aplikace, a tedy vynechává identifikaci dalších technologií, jako jsou třeba programovací jazyk aplikace nebo operační systém serveru.

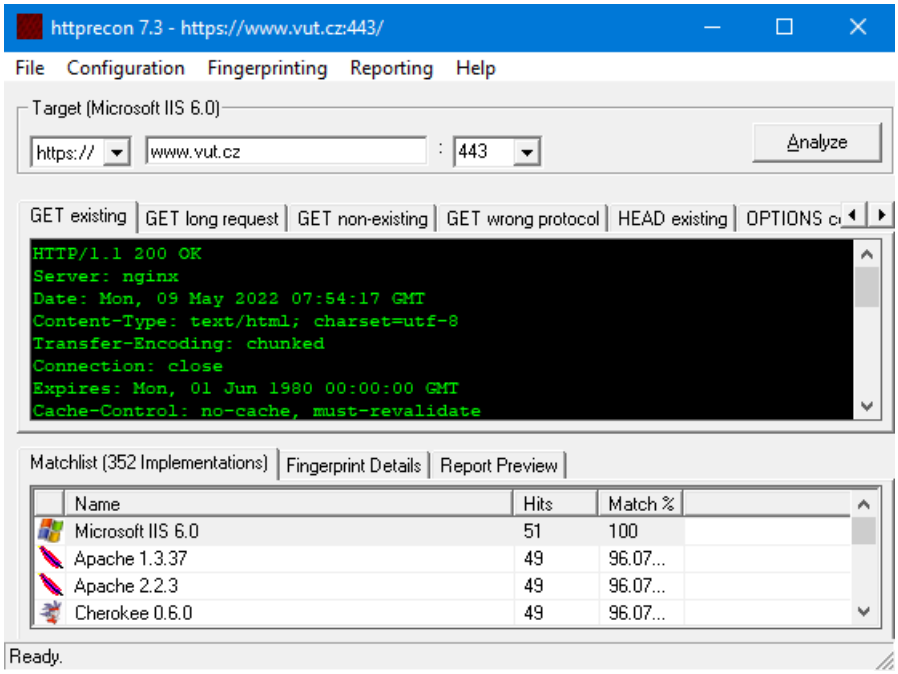
1.3.4 HTTPRecon

HTTPRecon[42] je dalším z nástrojů, které se zaměřují pouze na identifikaci webového serveru aplikace. Nástroj během analýzy pošle serveru 9 HTTP žádostí a zkoumá jeho odpovědi.

Žádosti jsou navzájem odlišné a každá se zaměřuje na vyvolání jiného chování serveru. Mezi typy žádostí patří například žádost s neexistující verzí HTTP protokolu nebo žádost s neexistující HTTP metodou.

[42]HTTPRecon: `https://www.computec.ch/projekte/httprecon/`

Nástroj v každé z odpovědí serveru vyhledává určité prvky, jako jsou například stavový kód či stavová zpráva, pořadí HTTP hlaviček nebo obsah hlavičky „Server“. Hodnoty těchto prvků porovnává se svou databází ověřených záznamů. V databázi jsou uložené hodnoty prvků zvlášť pro každou z odesílaných žádostí. Na základě zjištěných hodnot pro všechny odeslané žádosti nástroj vyhodnotí, do jaké míry se analyzovaný server podobá serverům v jeho databázi, a tyto informace poskytne uživateli jako výsledek analýzy. [36, 37]



Obr. 1.25: Výstup nástroje *HTTPRecon* pro server `www.vut.cz`

Nedostatky nástroje HTTPRecon

Nástroj *HTTPRecon* se zaměřuje pouze na identifikaci webového serveru, a tak vynechává identifikaci ostatních technologií, stejně jako výše popsaný nástroj *HTTPrint*.

1.3.5 Nmap

Nmap je nástroj pro průzkum počítačových sítí a testování jejich zabezpečení. Umožňuje nalézt zařízení nacházející se v počítačové síti a odhalovat a analyzovat síťové služby, které se na těchto zařízeních nacházejí. [29, Kapitola 1. Getting Started with Nmap]

Při analýze síťových služeb nástroj *Nmap* zasílá službám předem definované zprávy a zkoumá navrácené odpovědi, ve kterých vyhledává známé sekvence znaků či bajtů. Definice zasílaných zpráv, hledané sekvence i konkrétní služby, které jsou těmito sekvencemi identifikovány, se nacházejí v databázi nástroje, konkrétně

v `nmap-service-probes`[43]. [29, Kapitola 7. Service and Application Version Detection]

Další součástí nástroje je *Nmap Scripting Engine* (NSE). NSE umožňuje používání skriptů (psaných v programovacím jazyce *Lua*[44]) pro automatizaci úloh, jako jsou detailnější identifikace síťových služeb či testování zranitelnosti daných služeb. Příkladem těchto skriptů jsou například `http-favicon`, `http-methods` nebo `http-generator`. [29, Kapitola 9. Nmap Scripting Engine]

Nástroj *Nmap* je také schopen odhalovat operační systém síťových zařízení pomocí analýzy jejich implementace protokolů TCP/IP. Poznatky získané z analýzy porovnává s ověřenými vzorky ze svých databází[45] a na základě shody těchto vzorků určuje technologie nacházející se na daném zařízení. [29, Kapitola 15. Nmap Reference Guide]

Analýzu webové aplikace na serveru `www.vut.cz` lze spustit příkazem uvedeným ve výpisu 1.5. Výpis 1.6 obsahuje výsledek analýzy.

Výpis 1.5: Spuštění analýzy webové aplikace serveru `www.vut.cz`

```
nmap www.vut.cz -p 443 -sC -sV
```

Výpis 1.6: Výsledek analýzy nástrojem Nmap

```
Starting Nmap 7.70 ( https://nmap.org ) at 2022-05-09 11:33 Central
    Europe Summer Time
Nmap scan report for www.vut.cz (147.229.2.90)
Host is up (0.00s latency).
rDNS record for 147.229.2.90: piranha.ro.vutbr.cz

PORT    STATE SERVICE  VERSION
443/tcp open  ssl/http nginx
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
| http-robots.txt: 27 disallowed entries (15 shown)
| /edit/ /media/ /rss/ /ajax/ /login/ /en/login/ /intra/
. . .
|_http-server-header: nginx
|_http-title: Vysok\xC3\xA9 u\xC4\x8Den\xC3\xAD technick\xC3\xA9 v
   Brn\xC4\x9B
| ssl-cert: Subject: commonName=www.vutbr.cz/organizationName=Vysok
   \xC3\xA9 u\xC4\x8Den\xC3\xAD technick\xC3\xA9 v Brn\xC4\x9B/
   stateOrProvinceName=Jihomoravsk\xC3\xBD kraj/countryName=CZ
```

[43]Nmap service detection database: `https://svn.nmap.org/nmap/nmap-service-probes`
[44]Lua: `https://www.lua.org/`
[45]Nmap OS fingerprinting database: `https://svn.nmap.org/nmap/nmap-os-db`

```
| Subject Alternative Name: DNS:www.vutbr.cz, DNS:absolvent.vut.cz,
    DNS:absolvent.vutbr.cz, DNS:absolventi.vut.cz,
...
| Not valid before: 2022-04-25T00:00:00
|_Not valid after:  2023-04-25T23:59:59
|_ssl-date: TLS randomness does not represent time
| tls-alpn:
|   h2
|_  http/1.1
| tls-nextprotoneg:
|   h2
|_  http/1.1

Service detection performed. Please report any incorrect results at
    https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.97 seconds
```

2 Praktická část

V rámci praktické části této práce byl vyvinut vlastní automatizovaný analyzátor webových aplikací a bylo připraveno experimentální pracoviště sloužící pro vývoj a testování analyzátoru. V rámci popisu implementace analyzátoru je uveden i postup tvorby referenčního datasetu. Závěrem této kapitoly je analyzátor otestován v experimentálním prostředí a jsou zanalyzovány dosažené výsledky.

2.1 Experimentální pracoviště

Pro vývoj analyzátoru bylo zapotřebí připravit experimentální pracoviště, aby bylo možné během implementace nástroje průběžně ověřovat a testovat jeho funkčnost. Požadavky na experimentální pracoviště byly následující:

- nízká hardwarová náročnost
- snadná reprodukovatelnost
- přítomnost několika webových aplikací
  - různé programovací jazyky
  - různé webové servery
  - různé operační systémy

Pro řešení experimentálního pracoviště byl zvolen přístup *kontejnerizace* s využitím platformy *Docker*[1].

2.1.1 Kontejnerizace

Virtualizace

Pro vysvětlení a následné srovnání kontejnerizace je nutné nejprve představit pojem *virtualizace*. Virtualizace je vytvoření softwarové abstrakční vrstvy nad hardwarem počítače. Počítač, na kterém se abstrakční vrstva nachází, se nazývá „hostitelský systém“. Abstrakční vrstva umožňuje sdílení hardwarových prvků daného počítače mezi více „virtuálními stroji“. Každý virtuální stroj v sobě obsahuje svůj vlastní plnohodnotný operační systém a veškeré ostatní softwarové vybavení. [38]

Kontejnerizace

Kontejnerizace je formou virtualizace, při které je vytvořena abstrakční vrstva až nad jádrem hostitelského operačního systému. Abstrakční vrstva pak umožňuje sdílení operačního systému mezi více „kontejnery“. Každý kontejner v sobě obsahuje

[1] Docker: `https://www.docker.com/`

softwarové aplikace společně s veškerým softwarem, který dané aplikace potřebují pro svůj běh. Kontejnery je možné spouštět jakožto jednotlivé uživatelské procesy v hostitelském systému pomocí „kontejnerového běhového prostředí“. [39]

Kontejnery jsou tvořeny podle jejich *obrazu* až ve chvíli jejich spuštění. Obraz je neměnným předpisem kontejneru a obsahuje instrukce pro jeho vytvoření. [40]

Virtualizace vs. Kontejnerizace

Kontejnery oproti virtuálním strojům nabízí menší úroveň izolace od hostitelského systému, ale jsou snadněji upravovatelné, zabírají méně prostoru na datovém úložišti a rychleji se spouští. Další výhodou kontejnerizace je existence velkého množství veřejně dostupných předchystaných obrazů, které lze snadno využít. [41]

Windows Subsystem for Linux

Závislost kontejnerů na operačním systému hostitelského počítače má za následek, že kontejnery postavené na operačním systému Windows není možné přímo spouštět na hostitelském operačním systému Linux a naopak. Provoz Windows kontejnerů na operačním systému Linux v současné době není podporován. Provoz Linux kontejnerů na operačním systému Windows je možný s využitím technologií *Hyper-V*[2] nebo *Windows Subsystem for Linux* (WSL)[3].

WSL je nativní prostředí systému Windows pro běh aplikací určených pro operační systém Linux. Funguje na bázi virtualizace minimálního virtuálního stroje s operačním systémem Linux. [42]

2.1.2 Docker

Docker je platforma pro vývoj, distribuci a běh aplikací s využitím kontejnerizace. Na systému Windows je Docker dostupný ve formě uživatelsky přívětivé aplikace *Docker Desktop*. Součástí této aplikace je i kontejnerové běhové prostředí *Docker Engine*[4]. Jednou z velkých výhod použití Dockeru je existence platformy *Docker Hub*[5], která umožňuje sdílet obrazy docker kontejnerů. Docker Hub obsahuje velké množství oficiálních distribucí obrazů přímo od tvůrců daných technologií. [40]

Docker využívá pro obrazy kontejnerů vrstvenou architekturu, která umožňuje obrazy tvořit rozšířením již existujících obrazů. Rozšiřovaný obraz se stává základní vrstvou nového obrazu, na kterou se dalšími úpravami přidávají nové vrstvy. Nové

[2]Hyper-V: `https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/about/`

[3]WSL: `https://docs.microsoft.com/en-us/windows/wsl/`

[4]Docker Engine: `https://docs.docker.com/engine/`

[5]Docker Hub: `https://hub.docker.com/`

vrstvy v sobě uchovávají pouze rozdílové informace oproti své rodičovské vrstvě a jsou tvořeny při operacích, které mění souborový systém obrazu. Při spuštění kontejneru se rovněž tvoří nová vrstva, ale ta je pouze dočasná a je smazána při odstranění kontejneru. [43]

Dockerfile

Docker umožňuje definovat postup pro tvorbu obrazů pomocí textového dokumentu zvaného `Dockerfile`, jehož obsahem jsou příkazy, které se automaticky vykonají při tvorbě obrazu. [44]

Následující výpis je obsahem jednoho z `Dockerfile` souborů, který je použit v rámci experimentálního prostředí.

Výpis 2.1: Ukázka `Dockerfile` syntaxe

```
1  FROM centos
2  WORKDIR /
3
4  EXPOSE 80
5
6  COPY run.sh ./
7
8  RUN sed -i 's/mirrorlist/#mirrorlist/g' /etc/yum.repos.d/CentOS-
       Linux-* && \
9      sed -i 's|#baseurl=http://mirror.centos.org|baseurl=http://
       vault.epel.cloud|g' /etc/yum.repos.d/CentOS-Linux-* && \
10     dnf install httpd php -y && \
11     sed -i -e 's/LoadModule mpm_event/#LoadModule mpm_event/g' -e '
       s/#LoadModule mpm_prefork/LoadModule mpm_prefork/g' /etc/httpd/
       conf.modules.d/00-mpm.conf && \
12     chmod +x ./run.sh
13
14 COPY webapp/. /var/www/html/
15
16 CMD ["./run.sh"]
```

Výpis 2.1 obsahuje 6 odlišných příkazů, a to `FROM`, `WORKDIR`, `EXPOSE`, `RUN`, `COPY` a `CMD`.

`FROM` specifikuje, který obraz se má použít jakožto bázový, tedy jako bázová vrstva nového obrazu. V tomto případě se jedná o obraz operačního systému *CentOS*.

`WORKDIR` nastavuje pracovní adresář pro ostatní příkazy, kterými jsou v tomto případě příkazy `RUN`, `COPY` a `CMD`.

`EXPOSE` slouží pro informování Dockeru (a uživatelů daného obrazu), které síťové

porty [6] budou za běhu kontejneru otevřené. `COPY` vytvoří novou vrstvu, do které zkopíruje zadané soubory.

`RUN` definuje příkazy, které mají být vykonané v operačním systému daného obrazu. Vytvoří novou kontejnerovou vrstvu, provede v ní zadané příkazy a výslednou vrstvu zapíše jako další vrstvu obrazu.

`CMD` určuje právě jeden příkaz, který se má vykonat při spuštění kontejneru. Tento příkaz během tvorby obrazu novou vrstvu nevytvoří. [44]

`Dockerfile` lze použít pro vytvoření obrazu následujícím příkazem:

Výpis 2.2: Tvorba obrazu s využitím připraveného `Dockerfile` souboru

```
docker build -f ./Dockerfile
```

Docker Compose

Docker umožňuje automatizovat spouštění jednoho či více kontejnerů zároveň pomocí nástroje *Docker Compose*. *Docker Compose* využívá konfigurační soubor ve formátu *YAML*[7], který se zpravidla pojmenovává `docker-compose.yml`. V konfiguračním souboru jsou definované jednotlivé *služby*. Služba je abstraktní pojem, který zastřešuje jeden konkrétní obraz kontejneru společně s jeho definovaným nastavením.

Důležitou funkcionalitou *Docker Compose* je možnost využití předem připravených `Dockerfile` souborů. Tímto propojením je možné zároveň automatizovat jak tvorbu potřebných obrazů, tak i následující spouštění daných kontejnerů. [45, 46]

Následující výpis je obsahem jednoho z konfiguračních souborů, který je použit v rámci experimentálního prostředí.

Výpis 2.3: Ukázka konfiguračního souboru `docker-compose.yml`

```
1 version: '3.4'
2
3 services:
4     apache:
5         build: centos-apache-php/
6         ports:
7             - "127.0.0.1:20080:80"
```

Konfigurační soubor ve výpisu 2.3 obsahuje nastavení služby, jejíž obraz využívá Dockerfile obsažený ve výpisu 2.1.

Klíč `services` zastřešuje nastavení jednotlivých služeb. V tomto konfiguračním souboru se nachází nastavení pouze pro jednu službu, ale je možné zde specifikovat nastavení pro více služeb.

[6] Síťový port: `https://www.cloudflare.com/learning/network-layer/what-is-a-computer-port/`

[7] YAML: `https://yaml.org/`

Klíč `build` specifikuje, kde se nacházejí soubory potřebné pro vytvoření obrazu asociovaného s danou službou. Mezi tyto soubory patří již zmiňovaný `Dockerfile`.

Klíč `ports` určuje, jaké síťové porty výsledného kontejneru mají být zpřístupněny na hostitelském systému. [46]

Pro spuštění všech služeb definovaných v konfiguračním souboru `docker-compose-apache.yml` se použije následující příkaz.

Výpis 2.4: Spuštění kontejneru pomocí *Docker Compose*

```
docker compose -f ./docker-compose-apache.yml up
```

Ekvivalentními příkazy pro spuštění daného kontejneru bez využití *Docker Compose* by byly příkazy ve výpisu 2.5. První příkaz vytvoří obraz kontejneru s využitím `Dockerfile` souboru, druhý příkaz z vytvořeného obrazu spustí kontejner s daným nastavením.

Výpis 2.5: Spuštění kontejneru bez *Docker Compose*

```
docker build -t apache ./Dockerfile
docker run -p 127.0.0.1:20080:80 apache
```

2.1.3 Výsledné pracoviště

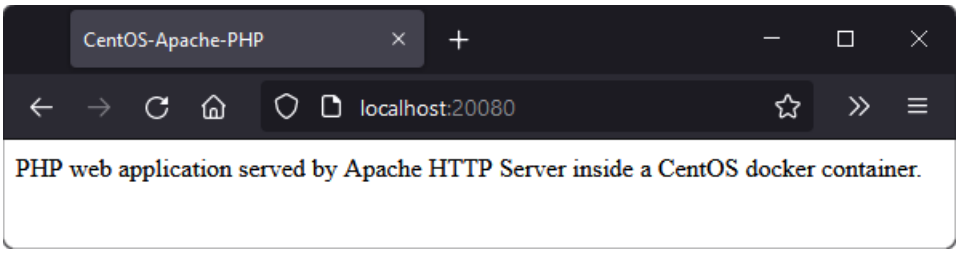
Vytvořené experimentální pracoviště se skládá ze 4 Docker kontejnerů, respektive jejich obrazů. Tři z těchto kontejnerů jsou postaveny na Linuxových operačních systémech, zbylý 1 kontejner je založený na operačním systému *Windows Server*. Každý z kontejnerů v sobě obsahuje webový server a jednoduchou webovou aplikaci, přičemž technologie webových serverů a webových aplikací se mezi jednotlivými kontejnery liší.

Obrazy kontejnerů (kromě Windows kontejneru), ve formě jejich odpovídajících `Dockerfile` souborů, jsou společně s jejich *Docker Compose* konfiguračními soubory obsaženy v elektronické příloze této práce.

Kontejner `centos-apache-php`

Prvním z Linux kontejnerů je `centos-apache-php`. Jedná se o kontejner založený na obrazu operačního systému *CentOS*[8]. Uvnitř se nachází webový server *Apache httpd* a jednoduchá webová aplikace využívající programovací jazyk *PHP*.

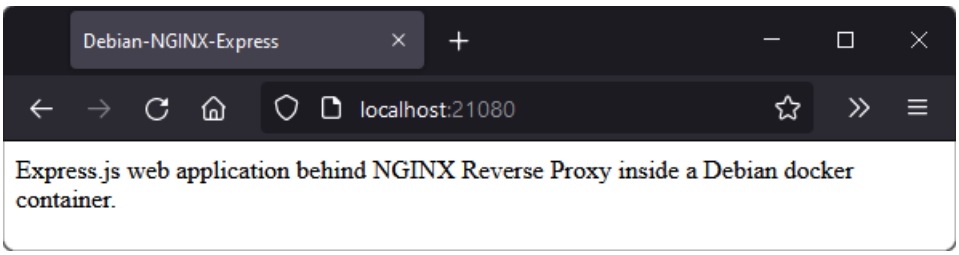
[8] *CentOS* Docker obraz: `https://hub.docker.com/_/centos`



Obr. 2.1: Webová aplikace kontejneru `centos-apache-php`

Kontejner `debian-nginx-express`

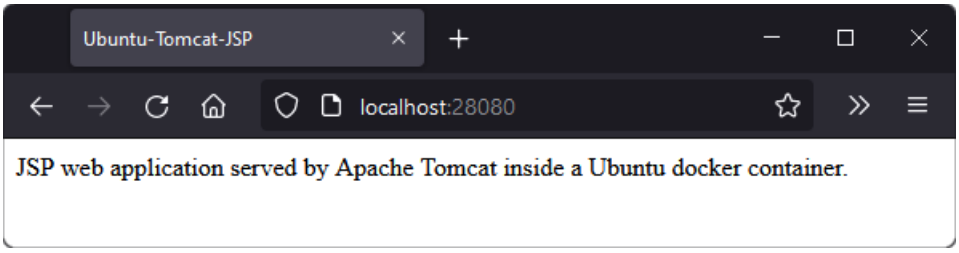
Druhým z Linux kontejnerů je `debian-nginx-express`. Tento kontejner je založený na obrazu běhového prostředí *Node.js*[9], který rozšiřuje obraz operačního systému *Debian*. Uvnitř se nachází webová aplikace využívající *JavaScript* framework *Express*.



Obr. 2.2: Webová aplikace kontejneru `debian-nginx-express`

Kontejner `ubuntu-tomcat-jsp`

Posledním z Linux kontejnerů je `ubuntu-tomcat-jsp`. Tento kontejner je založený na obrazu webového serveru *Apache Tomcat*[10], který je rozšířením obrazu operačního systému *Ubuntu*. Uvnitř se nachází webová aplikace využívající programovací jazyk *Java* s technologií *JavaServer Pages*.

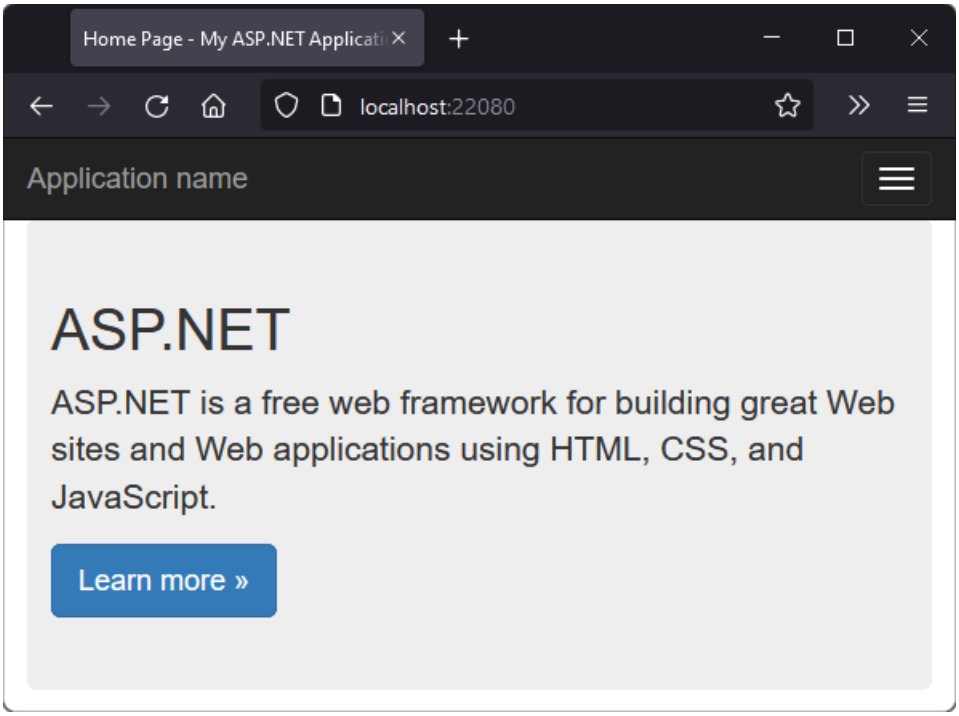


Obr. 2.3: Webová aplikace kontejneru `ubuntu-tomcat-jsp`

[9] *Node.js* Docker obraz: `https://hub.docker.com/_/node`
[10] *Apache Tomcat* Docker obraz: `https://hub.docker.com/_/tomcat`

Kontejner `win_server-iis-aspnet`

Jediným Windows kontejnerem je `win_server-iis-aspnet`. Pro realizaci tohoto kontejneru je využit oficiální demonstrační obraz pro webovou aplikaci s technologií *ASP.NET*[11]. Obraz je založen na operačním systému *Windows Server Core*, obsahuje webový server *Microsoft IIS* a webovou aplikaci využívající technologii *ASP.NET*. Tento obraz byl pro účely experimentálního pracoviště použit bez dalších úprav.



Obr. 2.4: Webová aplikace kontejneru `win_server-iis-aspnet`

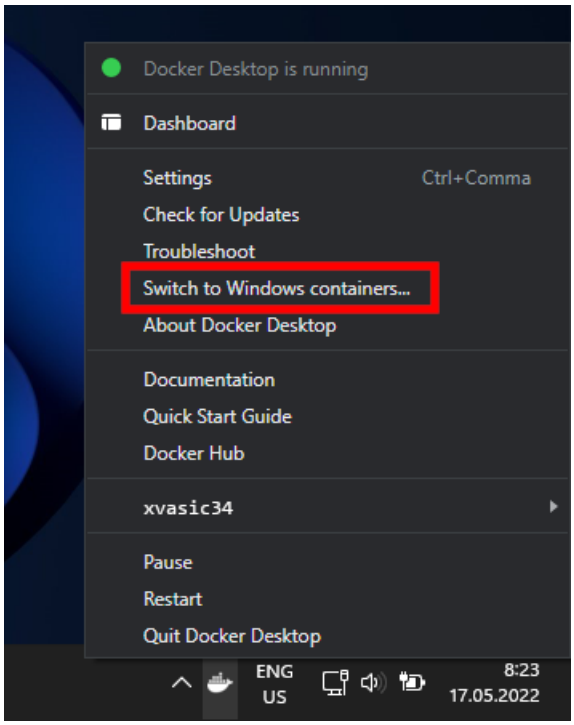
2.1.4 Limitace zvoleného řešení

První limitací tohoto experimentálního pracoviště je fakt, že Windows kontejnery lze spouštět pouze na operačním systému Windows, a to pouze v edicích *Professional* a *Enterprise*. Linuxových kontejnerů se tato limitace nedotýká, protože je lze spouštět na Linuxových distribucích i na operačním systému Windows, s pomocí již zmíněného WSL. [47]

Za další limitaci se dá považovat nutnost manuálně přepínat mezi režimy aplikace *Docker Desktop*. Aplikace má 2 režimy, a to režim pro Windows kontejnery a režim pro Linux kontejnery. Pro správu Windows kontejnerů je nutný režim pro Windows

[11] Obraz *ASP.NET* webové aplikace: `https://github.com/microsoft/dotnet-framework-docker/tree/main/samples/aspnetapp`

kontejnery a pro správu Linux kontejnerů režim pro Linux kontejnery, nelze spravovat oba typy kontejnerů najednou. Je ovšem možné, aby byly spuštěné oba typy kontejnerů zároveň, neboť při přepnutí režimu aplikace nedochází k ukončení spuštěných kontejnerů, dochází pouze k znemožnění jejich správy. Přepnutí režimu aplikace se provádí buď z kontextového menu aplikace *Docker Desktop* (obr. 2.5) nebo příkazem z příkazové řádky (výpis 2.6). Uvedený příkaz je uzpůsoben pro příkazovou řádku `cmd.exe` a předpokládá výchozí umístění aplikace *Docker Desktop*. [48]



Obr. 2.5: Změna režimu aplikace pomocí kontextového menu

Výpis 2.6: Změna režimu aplikace pomocí `DockerCli`

```
"C:\Program Files\Docker\Docker\DockerCli.exe" -SwitchDaemon
```

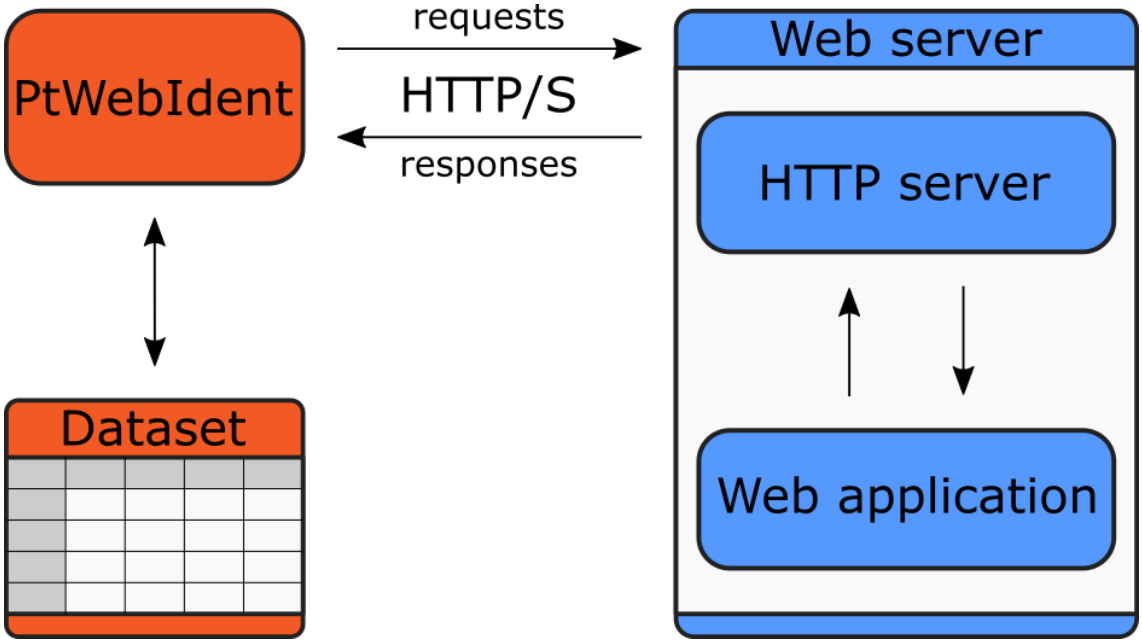
Třetí limitací je síťový přístup ke kontejnerům na hostitelském operačním systému Windows. Windows neumožňuje přímý přístup ke kontejnerům přes jejich vlastní síťové rozhraní. Síťový přístup do kontejnerů umožňuje pouze skrze síťové porty hostitelského systému, jejichž síťový provoz je systémem interně přesměrován na síťová rozhraní kontejnerů. V případě, že je hostitelským operačním systémem některá z Linuxových distribucí, lze přistupovat na síťová rozhraní kontejnerů přímo. [49]

2.2 Návrh analyzátoru

V rámci praktické části této práce byl vyvinut vlastní automatizovaný analyzátor webových aplikací s pracovním názvem *PtWebIdent*. Celkový proces fungování tohoto nástroje se dá rozdělit na 2 primární části.

První částí analýzy je sběr informací o webové aplikaci. Analyzátor využívá identifikační metody popsané v teoretické části práci (sekce 1.2) k tomu, aby odhalil co nejvíce informací o technologiích dané webové aplikace. Tyto informace zpracovává a uchovává během celého procesu analýzy.

Po skončení první části analýzy nástroj ukončí komunikaci s analyzovanou aplikací a přejde k vyhodnocování získaných informací. K jejich vyhodnocení využívá *dataset* ověřených záznamů. Dataset je tabulka záznamů, z nichž každý sestává z výsledků dřívější analýzy jiné webové aplikace a z informací o technologiích, které daná aplikace využívala. Analyzátor porovná informace nasbírané během první části analýzy se všemi záznamy v datasetu a na základě podobnosti těchto informací určí, jaké technologie analyzovaná aplikace využívá.

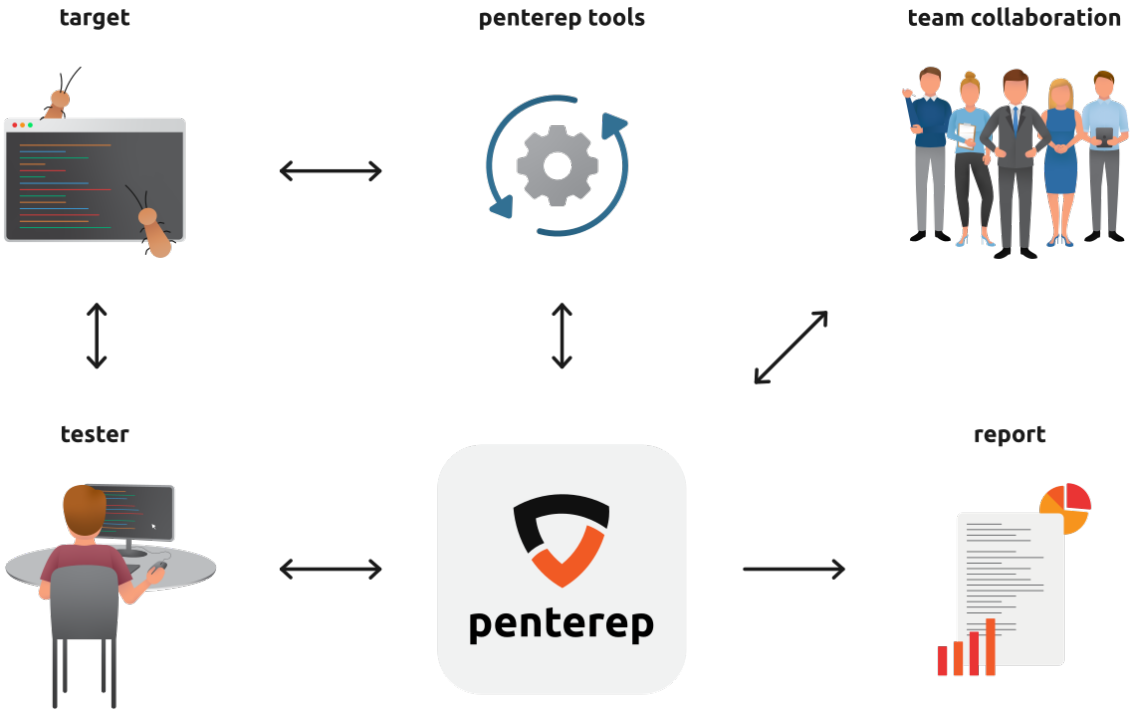


Obr. 2.6: Návrh analyzátoru PtWebIdent

2.2.1 Penterep

Analyzátor *PtWebIdent* je navržen jakožto přídavný modul pro platformu *Penterep*. Penterep je modulární, škálovatelná webová platforma pro penetrační testování. Zaměřuje se na sjednocení všech úkolů penetračních testerů na jednom místě, na auto-

matizaci testovacích procesů a na týmovou spolupráci testerů a dalších zúčastněných osob. Penetračním testerům poskytuje například kontrolní seznamy (dle kterých lze efektivně postupovat při manuálním penetračním testu), automatizované vytváření výsledných zpráv z penetračního testu nebo možnost pomocí webového rozhraní spouštět dostupné testovací nástroje (moduly). [50]



Obr. 2.7: Vizualizace platformy Penterep [51]

2.3 Implementace analyzátoru

Analyzátor *PtWebIdent* je program vytvořený v programovacím jazyce *Python*[12] verze 3.10. Lze ho formou rozšiřujícího modulu integrovat do platformy Penterep, nebo jej spouštět jakožto samostatný nástroj z příkazové řádky.

Analyzátor při svém spuštění přijímá konfigurační argumenty, které lze rozdělit do 2 skupin. První skupinou jsou obecná nastavení, která specifikují například adresu analyzované webové aplikace, prodlevu mezi jednotlivými HTTP žádostmi v rámci analýzy, proxy server pro HTTP komunikaci nebo typ výstupu analyzátoru. Demonstrativní výčet dostupných argumentů se nachází ve výpisu 2.7

[12]Python: `https://www.python.org/`

Výpis 2.7: Příklad obecných nastavení analyzátoru

```
--url      <url>            Test specified URL
--delay    <delay>          Delay between each HTTP request (ms)
--proxy    <proxy>          Set proxy (e.g. http://127.0.0.1:8080)
--headers <header:value>    Set custom header(s)
--json                      Output in JSON formatextension)
--version                   Show script version and exit
```

Druhou skupinou argumentů jsou nastavení identifikačních metod, které má analyzátor během analýzy provést. Příklad těchto nastavení se nachází ve výpisu 2.8

Výpis 2.8: Příklad nastavení identifikačních metod

```
--chk-headers       Check response headers
--det-case-sens     Detect case sensitivity
--bad-method        Invalid HTTP request methods
--long-url          Increasingly long URL
--nmap-os           Nmap OS detection
```

Konfigurace identifikačních metod je zpracována pomocí třídy `IDMethod`, jež je rozšířením bázové třídy `enum.Flag`[13]. `IDMethod` obsahuje výčtové konstanty, které se dají kombinovat s využitím bitových operací. Instance třídy `IDMethod` v sobě uchovává informaci o tom, které výčtové konstanty se v ní nachází. Této funkcionality je v kódu analyzátoru využito pro uložení informace o tom, které identifikační metody mají být spuštěny.

Výpis 2.9: Třída `IDMethod`

```
1  class IDMethod(Flag):
2      """Identification methods configuration"""
3      none             = 0
4      chk_headers      = auto()
5      chk_default_page = auto()
6      chk_icons        = auto()
7      chk_rsvd_names   = auto()
8      chk_nonexistent  = auto()
9      chk              = chk_headers | chk_default_page | chk_icons |
       chk_rsvd_names | chk_nonexistent
10     det_case_sens    = auto()
11     det_file_ext     = auto()
12     det              = det_case_sens | det_file_ext
13     bad_url          = auto()
14     bad_method       = auto()
15     bad_http_vsn     = auto()
16     bad              = bad_url | bad_method | bad_http_vsn
17     long_url         = auto()
```

[13]Třída `enum.Flag`: `https://docs.python.org/3/library/enum.html#enum.Flag`

```
    long_headers      = auto()
    long              = long_url | long_headers
    nmap_os           = auto()
    nmap              = nmap_os
    all               = chk | det | bad | long | nmap
```

Po zpracování argumentů analyzátor postupně spouští zvolené identifikační metody. Všechny identifikační metody jsou obsaženy v třídě `Analysis`, která zajišťuje sběr analyzačních dat, jejich uchovávání a konečné vyhodnocení. Pro uchovávání analyzačních dat slouží datový člen `Analysis.results`, jenž je instancí datové třídy[14] `AnalysisData`, která v sobě pro lepší organizaci dat obsahuje instance dalších vnořených datových tříd, jako jsou například `HeadersData` či `IconsData`.

Výpis 2.10: Datová třída `AnalysisData`

```
@dataclass
class AnalysisData:
    """
    Data class for all analysis data.
    """
    caseSensitive:     bool | None              = None
    fileExtension:     str | None               = None
    defaultPage:       str | None               = None
    headers:           HeadersData | None       = None
    icons:             IconsData | None         = None
    reservedNames:     ReservedNamesData | None = None
    longSequences:     LongSequencesData | None = None
    nmap:              NmapData | None          = None
    responsePages:     ResponsePagesData = ResponsePagesData()
    responsePagesFull: ResponsePagesData = ResponsePagesData()
```

Pro komunikaci s webovou aplikací je využívána metoda `Analysis._request`. Jedná se o pomocnou metodu, která zajišťuje odesílání HTTP žádostí pomocí knihovny *Requests*[15] a následné zpracovávání přijatých HTTP odpovědí pomocí metody `Analysis._process_response`. Při zpracovávání odpovědí je kontrolováno, zda je stavový kód odpovědi prvním výskytem tohoto kódu v rámci komunikace s aplikací. Pokud ano, je spočítán a uložen *hash*[16] těla dané odpovědi.

[14] Uchovávání dat formou datové třídy bylo zvoleno primárně z důvodu jasné definice členů dané třídy. Více informací o datových třídách lze nalézt na webové stránce `https://docs.python.org/3/library/dataclasses.html#dataclasses.dataclass`.

[15] Requests: `https://docs.python-requests.org/`

[16] Hashovací funkce: `https://www.ssl.com/faqs/what-is-a-cryptographic-hash-function/`

2.3.1 Identifikační metody

Každá z implementovaných metod je automatizací některé z již popsaných identifikačních metod (sekce 1.2) a cílí na získání konkrétní identifikační indicie. Indicie hledané v rámci analýzy jsou:

- HTTP hlavičky a cookies odpovědí
- Citlivost na velká a malá písmena
- Ikony serveru *Apache httpd*
- Přípona zdrojových souborů
- Výchozí uvítací stránka webového serveru
- Odpověď na žádost o neexistující resource
- Odpovědi na žádosti o resource s rezervovaným jménem
- Odpovědi na žádosti s neplatnými znaky v resource path
- Odpovědi na žádosti s nepodporovanou HTTP metodou
- Odpovědi na žádosti s neplatnou verzí HTTP protokolu
- Odpovědi na žádosti s velmi dlouhou resource path
- Odpovědi na žádosti s velmi dlouhými HTTP hlavičkami

Kromě těchto metod analyzátor také umožňuje využít nástroj *Nmap* k analýze implementace TCP/IP protokolů webového serveru. Pro spouštění nástroje *Nmap* a zpracovávání jeho výsledků je využit Python modul `python3-nmap`[17].

Většina z výše uvedených metod je přímou automatizací metod popsaných v teoretické části práce, a proto již v této části práce nejsou všechny z nich znovu popisovány. Pro ukázku automatizace identifikačních metod v jazyce Python je v následujícím výpise uvedena metoda pro zjištění odpovědí na žádosti o resource s rezervovaným jménem.

Výpis 2.11: Identifikační metoda `Analysis.chk_rsvd_names`

```
1  def chk_rsvd_names(self) -> None:
2      """Check responses to known reserved resource names"""
3
4      # Apache specific
5      apacheht = self._try_status_code(self.url + '.htaccess')
6
7      # Windows specific
8      com1 = self._try_status_code(self.url + 'COM1')
9      lpt1 = self._try_status_code(self.url + 'LPT1')
10     aux = self._try_status_code(self.url + 'AUX')
11
12     self.results.reservedNames = ReservedNamesData(
13         apacheht=apacheht,
```

[17] `python3-nmap`: `https://pypi.org/project/python3-nmap/`

```
14          COM1=com1,
15          LPT1=lpt1,
16          AUX=aux
17      )
18
19
20  def _try_status_code(self, path: str) -> int | None:
21      """Try to get the status code of a given resource"""
22
23      res = self._request(path)
24      if res is not None:
25          return res.status_code
26      else:
27          return None
```

`Analysis.chk_rsvd_names` se pokusí získat odpovědi webové aplikace pro resource `.htaccess`, `COM1`, `LPT1` a `AUX`. Pro získání odpovědí tato metoda využívá další pomocnou metodu `Analysis._try_status_code`, jejíž návratovou hodnotou je buď stavový kód odpovědi, anebo prázdná hodnota `None`[18] v případě neobdržení odpovědi od aplikace. Získané odpovědi jsou uloženy do instance datové třídy `ReservedNamesData`, jež je jedním z členů již představené třídy `AnalysisData`.

Výpis 2.12: Datová třída `ReservedNamesData`

```
1  @dataclass
2  class ReservedNamesData:
3      """
4      Data class for reserved names response codes
5      """
6      apacheht: int | None
7      COM1:     int | None
8      LPT1:     int | None
9      AUX:      int | None
```

Citlivost na velká a malá písmena

Metoda `Analysis.det_case_sens` zjišťuje citlivost aplikace na velká a malá písmena. Princip odhalení této indicie je velmi jednoduchý, přesto je ale metoda `Analysis.det_case_sens` nejkomplexnější metodou ze všech implementovaných identifikačních metod. Z tohoto důvodu je v této sekci detailněji popsán princip jejího fungování.

[18] `None` je v programovacím jazyce Python klíčovým slovem představujícím prázdnou hodnotu. Datovým typem této hodnoty je `NoneType`. V jiných programovacích jazycích je tato hodnota obvykle reprezentována jako `null`. [52]

Prvním krokem metody je pokus o odhalení citlivosti aplikace na velká a malá písmena pomocí žádosti o resource `favicon.ico`. Pro zjištění citlivosti u konkrétního resource je využívána pomocná metoda `Analysis._url_case_sens`, která porovnává odpovědi aplikace na žádosti s různými variantami velikostí písmen v resource path. Resource `favicon.ico` je dostupný u velkého množství webových aplikací, tedy identifikační metoda častokrát úspěšně končí již po tomto kroku.

Výpis 2.13: identifikační metoda `Analysis.det_case_sens` – `favicon.ico`

```
def det_case_sens(self) -> None:
    """Detect application's case sensitivity"""

    favicon = self._url_case_sens(self.url + 'favicon.ico')

    if favicon is not None:
        self.results.caseSensitive = favicon
        return

        ...
```

Jestliže resource `favicon.ico` není dostupný, je nutné nalézt nějaký jiný existující resource. Metoda proto zašle HTTP žádost na kořenový adresář webové aplikace a zahájí zpracovávání obsahu těla přijaté odpovědi. V těle odpovědi jsou s využitím třídy `URLAttributesHTMLParser` vyhledány URL[19] odkazy, které se nacházejí uvnitř atributů HTML značek. Typickým příkladem takového atributu je atribut `href` HTML značky `a`.

Ze získaných URL odkazů jsou vyfiltrovány a očištěny pouze ty odkazy, které odkazují na doménu dané webové aplikace. Výsledné URL odkazy jsou postupně využívány k pokusům o zjištění citlivosti aplikace na velká a malá písmena. Jestliže ani 1 ze získaných odkazů není validním odkazem, zjištění citlivosti selže.

Výpis 2.14: identifikační metoda `Analysis.det_case_sens` – filtrace odkazů

```
for link in parser.urls_found:
# Remove parameters and URI fragments
link = link.split('?')[0]
link = link.split('#')[0]

# Non-empty after fragment filter
if len(link) != 0:
   link_lower = link.lower()
   # http/s or protocol relative URLs ('//')
   if link_lower.startswith("http") or link_lower.startswith("//"):
       # Same domain (excluding subdomains)
```

[19] *Uniform Resource Locator* (URL): `https://developer.mozilla.org/en-US/docs/Glossary/URL`

```
12          if self._domain.lower() == link_lower.split('/')[2]:
13              links.add(link)
14      # Relative links, that are not webroot ('/')
15      elif ':' not in link and re.match(r'^/+$', link) is None:
16          links.add(link)
```

2.3.2 Tvorba datasetu s vyhledávačem Shodan

Po dokončení všech zvolených identifikačních metod přejde analyzátor k vyhodnocení výsledků, k čemuž využívá dataset ověřených záznamů. Dataset obsahuje 263 záznamů, kde každý záznam sestává ze 2 informačních prvků, ze 39 analyzačních prvků a ze 3 identifikačních značek.

Informačními prvky jsou URL adresa analyzované aplikace a její odpovídající IP adresa. Tyto prvky slouží výhradně pro informační účely a nemají vliv na vyhodnocení výsledků analýzy.

Analyzační prvky představují data získaná pomocí identifikačních metod během analýzy. Tyto jsou prvky jsou využívány při vyhodnocení výsledků.

Identifikačními značkami jsou `serverNameLabel`, `progLangLabel` a `osLabel`. Tyto značky znamenají, jaké technologie se nacházely na analyzované aplikaci v době její analýzy, a tedy vytvářejí spojitost analyzačních prvků s danými technologiemi webových aplikací.

Vytvoření tohoto datasetu bylo zásadním krokem v rámci vývoje analyzátoru, jelikož bez něj je analyzátor schopen pouze sbírat data, ale není schopen tato data vyhodnocovat.

Úvodní sestavení datasetu

Pro sestavení datasetu byl využit nástroj *Shodan*. Shodan je internetový vyhledáváč, který sbírá informace o zařízeních připojených k síti internet. Sbíraná data zahrnují informace o softwaru nacházejícím se na daném zařízení, což v případě webových serverů představuje například obsah HTTP hlaviček jejich odpovědí. Informace zpřístupňuje svým uživatelům formou webové aplikace[20] nebo formou programového API[21]. Pro usnadnění komunikace s API existuje Python modul `shodan`[22]. [53]

Výpis 2.15: Vyhledání webových serverů pomocí Shodan Python API

```
1 import json
2 import shodan
3
```

[20] Shodan: `https://www.shodan.io/`
[21] Shodan *Application Programming Interface* (API): `https://developer.shodan.io/api`
[22] Python `shodan`: `https://github.com/achillean/shodan-python`

```
4  SHODAN_API_KEY = "..."
5
6  api = shodan.Shodan(SHODAN_API_KEY)
7
8  try:
9      r_apache = api.search('product:"Apache httpd"')
10     r_iis = api.search('product:"Microsoft IIS httpd"')
11     r_nginx = api.search('product:"nginx"')
12     r_tomcat = api.search('product:"Apache Tomcat"')
13
14     with open('apache.txt', 'w') as f:
15         json.dump(r_apache, f)
16     with open('iis.txt', 'w') as f:
17         json.dump(r_iis, f)
18     with open('nginx.txt', 'w') as f:
19         json.dump(r_nginx, f)
20     with open('tomcat.txt', 'w') as f:
21         json.dump(r_tomcat, f)
22 except shodan.APIError as e:
23     print('Error: {}'.format(e))
```

Výpis 2.15 obsahuje kód, který byl použit během tvorby datasetu pro získání informací z vyhledávače Shodan. Metoda `api.search()` přijímá, jakožto parametr, textový řetězec, který formou dotazu odešle na Shodan API. API dotazy podporují vyhledávání pomocí filtrů. V tomto případě byl použit filtr `product`, který zužuje výsledky pouze na ta zařízení, u nichž byla nalezena technologie specifikovaná filtrem. Tedy například filtr `product:"Apache httpd"` se dotazuje na zařízení, na kterých se nalézá webový server *Apache httpd*.

Ve výchozím nastavení API vrací pro každý dotaz prvních 100 nejrelevantnějších výsledků. Tímto způsobem bylo tedy získáno dohromady 400 webových serverů, z nichž Shodan rozpoznal 100 jako *Apache httpd*, 100 jako *Microsoft IIS*, 100 jako *NGINX* a 100 jako *Apache Tomcat*. Na všechny tyto servery byl spuštěn analyzátor a výsledky všech analýz byly uloženy do CSV[23] souboru, který v této formě představoval počáteční verzi datasetu. Konečným počet záznamů v datasetu není 400, nýbrž 263, a to proto, že některé z webových serverů, které Shodan nalezl, v době tvorby datasetu nebyly dostupné nebo již vůbec neexistovaly.

Očištění datasetu

Principem fungování analyzátoru je zkoumání výchozího chování technologií webových aplikací, tedy například zkoumání výchozích chybových stránek webového serveru. Jestliže zkoumaná aplikace využívá technologii, která byla cíleně upravena,

[23] *Comma-separated values* (CSV): `https://datatracker.ietf.org/doc/html/rfc4180`

a tedy již neprojevuje známky svého výchozího chování, analyzátor ji není schopen identifikovat a ani to není jeho primárním cílem. Z tohoto důvodu bylo nutné manuálně očistit získaný dataset, tedy ho prohlédnout a odstranit všechny analyzační prvky záznamů, které do datasetu jakožto celku nezapadaly.

Kritériem pro ponechání prvku záznamu v datasetu byl celkový počet výskytů jeho hodnoty napříč celým datasetem.

Pokud se hodnota prvku v datasetu nacházela alespoň 3krát, byla v datasetu ponechána. Pokud se jednalo o jediný výskyt dané hodnoty v rámci celého datasetu, byla odstraněna.

Pokud se hodnota prvku v datasetu nacházela právě 2krát, bylo nutné tuto hodnotu manuálně ověřit a dle vlastního uvážení rozhodnout, zda se jedná o výchozí projev některé z technologií webové aplikace, či nikoliv. Pokud ano, hodnota byla v datasetu ponechána, pokud ne, byla odstraněna.

Tímto postupem byl očištěn celý dataset. Byly odstraňovány pouze jednotlivé prvky záznamů, ne celé záznamy. Důvodem pro to je, že prvky záznamů jsou projevy různých technologií využívaných webovou aplikací. Jeden nepadnoucí prvek tedy neznamená, že všechny ostatní prvky rovněž padnout nebudou. Pokud by se ale po očištění v datasetu nacházely záznamy, jejichž analyzační prvky by byly všechny prázdné, byly by záznamy zcela odstraněny. Taková situace při tvorbě datasetu ovšem nenastala.

Označení datasetu

Konečnou úpravou datasetu bylo vyplnění identifikačních značek u jednotlivých záznamů.

Značka `serverNameLabel` byla vyplněna dle úsudku vyhledávače Shodan. Tedy například záznamy odpovídající webovým serverům, které byly z Shodan API navráceny pro filtr `product:"Apache httpd"`, byly označeny hodnotou `apache`.

Zbývající značky `progLangLabel` a `osLabel` byly vyplněny na základě vlastního uvážení po manuální inspekci hodnot analyzačních prvků.

2.3.3 Vyhodnocení výsledků analýzy pomocí datasetu

Finální vyhodnocení výsledků probíhá v metodě `Analysis.identify`. Princip vyhodnocení spočívá v porovnávání dat získaných během analýzy s hodnotami všech záznamů z celého datasetu. Při shodě hodnot s některým ze záznamů datasetu jsou zaznamenány identifikační značky daného záznamu. Po porovnání všech hodnot s datasetem se vypočítá procentuální zastoupení jednotlivých identifikačních značek, které představuje konečný výsledek analýzy.

Nejprve jsou pomocí metody `csvformat.analysisToCsv` data, nasbíraná během analýzy, převedena do formátu datové třídy `csvformat.CsvEntry`. Tato třída představuje 1 záznam v datasetu, neboť každý datový člen této třídy odpovídá jednomu z prvků záznamu. Všechny datové členy této třídy jsou typu `csvformat.Labeled`. Třída `csvformat.Labeled` v sobě obsahuje informaci o tom, ke které z identifikačních značek daná hodnota náleží – například přípony zdrojových souborů aplikace náleží značce `progLangLabel` a citlivost na velká a malá písmena náleží značce `serverNameLabel`.

Dále je pomocí modulu `pandas`[24] načten dataset z CSV souboru a je z něj vytvořen objekt typu `pandas.DataFrame`, který umožňuje snadno a efektivně pracovat s tabulkovými daty.

S takto připravenými daty započne porovnávání hodnot získaných během analýzy se záznamy v datasetu. Pro každou z hodnot výsledků analýzy se z datasetu vyfiltrují pouze ty záznamy, jejichž odpovídající prvek obsahuje totožnou hodnotu. Jestliže takové záznamy existují, zaznamenají se hodnoty jejich identifikačních značek.

Výpis 2.16: Porovnání hodnot s datasetem

```
matches: dict[DatasetLabel, dict[str, int]] = {
    DatasetLabel.server:   {},
    DatasetLabel.progLang: {},
    DatasetLabel.os:       {}
}

fields = dataclasses.fields(csvResults)
for i in range(2, len(fields) - 3):

    element: Labeled = getattr(csvResults, fields[i].name)

    if element.value is None:
        continue

    filter = [str(row) == element.value for row in dataset.iloc[:,
   i]]
    filtered_dataset: pd.DataFrame = dataset[filter]

    if not filtered_dataset.empty:
        match = matches[element.label]

        for value, count in filtered_dataset[element.label.value].
   value_counts().iteritems():
            if match.get(value) is None:
                match[value] = count
```

[24]pandas: `https://pandas.pydata.org/`

```
24                  else:
25                      match[value] += count
```

Pro každou ze 3 identifikačních značek je nakonec spočítán celkový počet jejích výskytů společně s počty konkrétních hodnot, kterých tato značka během porovnávání nabývala. Následně jsou výpočítány poměry počtů konkrétních hodnot značky ku celkovému počtu zaznamenání dané značky. Tyto poměry jsou metodou navráceny jakožto konečné výsledky.

2.3.4 Kompatibilita s platformou Penterep

Pro možnost integrace analyzátoru do platformy Penterep stačí, aby analyzátor poskytoval svá výstupní data v definovaném JSON formátu. Pro vytvoření a úpravy výstupní JSON struktury je v analyzátoru použit pomocný modul `ptlibs`[25], jenž je jedním z nástrojů platformy Penterep.

Pro demostraci formátu výstupu slouží následující 2 výpisy. Výpisy byly upraveny, aby byla usnadněna jejich čitelnost.

Výpis 2.17: JSON výstup analyzátoru pří úspěšné analýze

```
[
    {
        "testcode": "ptwebident",
        "status": "null",
        "vulnerable": "null",
        "data": {
            "serverHeader": "Apache/2.4.37 (centos) PHP/7.2.24",
            "server": {
                "Apache": 0.67,
                "Nginx": 0.23,
                "Tomcat": 0.04
            },
            "progLang": {
                "PHP": 0.97,
                "ASP.NET": 0.03
            },
            "os": {
                "Linux": 0.72,
                "Windows": 0.28
            }
        }
    }
]
```

[25] `ptlibs`: `https://pypi.org/project/ptlibs/`

Výpis 2.18: JSON výstup analyzátoru pří neúspěšné analýze

```
{
    "testcode": "ptwebident",
    "status": "error",
    "message": "'http://127.0.0.1:21080/' does not respond"
}
```

2.4 Testování analyzátoru

Testování vyvinutého analyzátoru bylo provedeno s využitím připraveného experimentálního pracoviště, jehož popis se nachází v sekci 2.1. Z důvodů limitací pracoviště popsaných v sekci 2.1.4 byly v rámci testování analyzátoru Linux kontejnery spouštěny a analyzovány na hostitelském operačním systému *Kali Linux*[26], zatímco Window kontejner byly spouštěn a analyzován na systému *Windows 11 Professional*.

Prvním analyzovaným kontejnerem byl kontejner `centos-apache-php`, jehož technologiemi jsou operační systém *CentOS*, webový server *Apache httpd* a programovací jazyk *PHP*.

Výpis 2.19: Výstup analyzátoru pro kontejner `centos-apache-php`

```
[
    {
        "testcode": "ptwebident",
        "status": "null",
        "vulnerable": "null",
        "data": {
            "serverHeader": "Apache/2.4.37 (centos) PHP/7.2.24",
            "server": {
                "Apache": 0.67,
                "Nginx": 0.23,
                "Tomcat": 0.04
            },
            "progLang": {
                "PHP": 0.97,
                "ASP.NET": 0.03
            },
            "os": {
                "Linux": 0.75,
                "Windows": 0.25
            }
        }
    }
]
```

[26]Kali Linux: `https://www.kali.org/`

Ve výpisu 2.19 jde vidět, že analyzátor úspěšně odhalil webový server i programovací jazyk aplikace. U operačního systému analyzátor správně určil, že se jedná o Linuxovou distribuci.

Dalším kontejnerem v pořadí je `debian-nginx-express`. Jeho technologiemi jsou operační systém *Debian*, webový server *NGINX* a framework *Express*, jenž je postaven na programovacím jazyce *JavaScript*.

Výpis 2.20: Výstup analyzátoru pro kontejner `debian-nginx-express`


```
[
    {
        "testcode": "ptwebident",
        "status": "null",
        "vulnerable": "null",
        "data": {
            "serverHeader": "nginx/1.18.0",
            "server": {
                "Nginx": 0.51,
                "IIS": 0.18,
                "Tomcat": 0.15
            },
            "progLang": {
                "JavaScript": 1.0
            },
            "os": {
                "Linux": 0.75,
                "Windows": 0.25
            }
        }
    }
]
```


Dle výpisu 2.20 je zřejmé, že analyzátor opět úspěšně odhalil webový server i programovací jazyk aplikace a v případě operačního systému opět správně určil, že se jedná o Linuxovou distribuci.

Posledním z Linux kontejnerů je `ubuntu-tomcat-jsp`. Technologiemi tohoto kontejneru jsou operační systém *Ubuntu*, webový server *Apache Tomcat* a programovací jazyk *Java* dohromady s techonologií *JSP*.

Výpis 2.21: Výstup analyzátoru pro kontejner `ubuntu-tomcat-jsp`


```
[
    {
        "testcode": "ptwebident",
        "status": "null",
        "vulnerable": "null",
        "data": {
```

```
            "serverHeader": null,
            "server": {
                "Apache": 0.27,
                "IIS": 0.25,
                "Nginx": 0.23
            },
            "progLang": {
                "JSP": 1.0
            },
            "os": {
                "Linux": 0.73,
                "Windows": 0.27
            }
        }
    }
]
```


Ve výpisu 2.21 jde vidět, že analyzátor tentokrát nebyl schopen odhalit webový server, kterým byl *Apache Tomcat*. Avšak programovací jazyk *Java* s technologií *JSP* analyzátor opět odhalil. Stejně tak operační systém správně určil jakožto Linuxovou distribuci.

Posledním kontejnerem, který byl využit pro testování analyzátoru, byl Windows kontejner `win_server-iis-aspnet`, jehož technologiemi jsou operační systém *Microsoft Windows Server 2016*, webový server *Microsoft IIS* a webový framework *ASP.NET*.

Výpis 2.22: Výstup analyzátoru pro kontejner `ubuntu-tomcat-jsp`


```
[
    {
        "testcode": "ptwebident",
        "status": "null",
        "vulnerable": "null",
        "data": {
            "serverHeader": "Microsoft-IIS/10.0",
            "server": {
                "IIS": 0.61,
                "Tomcat": 0.14,
                "Nginx": 0.12
            },
            "progLang": {
                "ASP.NET": 1.0
            },
            "os": {
                "Linux": 0.58,
                "Windows": 0.42
            }
```

```
        }
    }
]
```

Z výpisu 2.22 lze vyčíst, že analyzátor úspěšně odhalil webový server a webový framework. Tentokrát ale selhal, byť poměrně těsně, při identifikaci operačního systému kontejneru, kde převažujícím poměrem rozhodl, že se jedná o Linuxovou distribuci.

2.5 Analýza dosažených výsledků

Analyzátor v rámci testování v experimentálním pracovišti úspěšně odhalil 3 ze 4 webových serverů, 4 ze 4 programovacích jazyků či frameworků a u 3 ze 4 operačních systémů správně určil, zda se jedná o operační systém Windows či některou z Linuxových distribucí.

Analyzátor v současném stavu identifikuje názvy technologií webových aplikací, ale již se nezaobírá identifikací konkrétních verzí daných technologií. V tomto směru nabízí prostor pro rozšíření jeho funkcionality.

Referenční dataset, který analyzátor využívá k vyhodnocování výsledků analýzy, obsahuje 263 záznamů. Mezi těmito záznamy byly v rámci vývoje pozorovány určité vzorce, které napovídaly existenci hlubších spojitostí mezi analyzačními daty a některými identifikačními značkami. Tyto pozorované vzorce nabádají k dalšímu výzkumu v oblasti rozšiřování datasetu a jeho užití při vyhodnocování výsledků.

Závěr

Cílem této bakalářské práce byl výzkum a vývoj softwarového nástroje pro základní analýzu webové aplikace s cílem identifikace operačního systému, webového serveru a programovacího jazyka webové aplikace. Identifikace technologií se nespoléhá pouze na obsah HTTP hlaviček odpovědí webové aplikace, ale zaměřuje se spíše na zkoumání chování aplikace.

V teoretické části práce byl obsažen úvod do problematiky a detailní popis a manuální demonstrace identifikačních metod používaných v rámci analýzy webových aplikací. Závěrem teoretické části práce byly představeny existující analyzační nástroje a byly zde popsány jejich identifikační databáze.

V praktické části práce byla zdokumentována tvorba experimentálního pracoviště s využitím kontejnerizace s platformou *Docker*. Dále byl představen návrh a konkrétní implementace analyzátoru s odkazem na platformu *Penterep*, jíž se analyzátor nakonec stane součástí. Implementované identifikační metody reflektují metody popsané v teoretické části práce. V rámci implementace analyzátoru byl rovněž uveden proces tvorby referenčního datasetu. Závěrem praktické části práce byl nástroj otestován v předešle vytvořeném experimentálním pracovišti a byly přehledně zanalyzovány jeho dosažené výsledky. Všechny cíle této bakalářské práce byly splněny.

Literatura

[1] FINGENT. *Web Application Development: A Detailed Guide for 2022*. [online]. 2022-05-16, ©2022 [cit. 2022-05-25]. Dostupné z: https://www.fingent.com/blog/web-application-development-a-detailed-guide/.

[2] NÁRODNÍ ÚŘAD PRO KYBERNETICKOU A INFORMAČNÍ BEZPEČNOST. *PENETRAČNÍ TESTOVÁNÍ – ÚVOD DO PROBLEMATIKY*. [online]. 7. března 2022 [cit. 2022-05-25]. Dostupné z: https://www.nukib.cz/download/publikace/podpurne_materialy/2022-03-07_Penetracni-testovani_v1.0.pdf.

[3] RIVERBED TECHNOLOGY. *How Does a Web Application Work?* [online]. 2021-09-28, ©2022 [cit. 2022-05-25]. Dostupné z: https://www.riverbed.com/faq/how-does-web-application-work.html.

[4] MOZILLA a komunita přispěvatelů. *Introduction to client-side frameworks*. [online]. May 2, 2022 [cit. 2022-05-25]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Introduction.

[5] Wappalyzer. *UI frameworks*. [online]. [cit. 2022-05-25]. Dostupné z: https://www.wappalyzer.com/technologies/ui-frameworks.

[6] WAPPALYZER. *JavaScript frameworks*. [online]. [cit. 2022-05-25]. Dostupné z: https://www.wappalyzer.com/technologies/javascript-frameworks.

[7] MOZILLA a komunita přispěvatelů. *Introduction to the server side*. [online]. Apr 19, 2022 [cit. 2022-05-25]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Introduction.

[8] WAPPALYZER. *Programming languages*. [online]. [cit. 2022-05-25]. Dostupné z: https://www.wappalyzer.com/technologies/programming-languages.

[9] WAPPALYZER. *Web frameworks*. [online]. [cit. 2022-05-25]. Dostupné z: https://www.wappalyzer.com/technologies/web-frameworks.

[10] WAPPALYZER. *Operating systems*. [online]. [cit. 2022-05-25]. Dostupné z: https://www.wappalyzer.com/technologies/operating-systems/.

[11] MOZILLA a komunita přispěvatelů. *What is a web server?* [online]. Mar 5, 2022 [cit. 2022-05-25]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server.

[12] WAPPALYZER. *Web servers*. [online]. [cit. 2022-05-25]. Dostupné z: https://www.wappalyzer.com/technologies/web-servers.

[13] MOZILLA a komunita přispěvatelů. *An overview of HTTP*. [online]. May 20, 2022 [cit. 2022-05-25]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview.

[14] MOZILLA a komunita přispěvatelů. *HTTPS*. [online]. Oct 8, 2021 [cit. 2022-05-25]. Dostupné z: https://developer.mozilla.org/en-US/docs/Glossary/https.

[15] OWASP [Open Web Application Security Project] a komunita přispěvatelů. *Fingerprint Web Server*. [online]. Dec 3, 2020 [cit. 2022-05-25]. Dostupné z: https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/01-Information_Gathering/02-Fingerprint_Web_Server.

[16] OWASP [Open Web Application Security Project] a komunita přispěvatelů. *Fingerprint Web Application Framework*. [online]. Dec 3, 2020 [cit. 2022-05-25]. Dostupné z: https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/01-Information_Gathering/08-Fingerprint_Web_Application_Framework.

[17] THE APACHE SOFTWARE FOUNDATION a komunita přispěvatelů. *httpd.conf.in*. [Výchozí konfigurační soubor pro Apache HTTP Server] [online]. Aug 9, 2018 [cit. 2022-05-25]. Dostupné z: https://github.com/apache/httpd/blob/trunk/docs/conf/httpd.conf.in.

[18] MOZILLA a komunita přispěvatelů. *HTTP request methods*. [online]. May 13, 2022 [cit. 2022-05-25]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods.

[19] NGINX. *Module ngx_http_core_module: large_client_header_buffers*. [Dokumentace HTTP serveru NGINX] [online]. [cit. 2022-05-25]. Dostupné z: http://nginx.org/en/docs/http/ngx_http_core_module.html#large_client_header_buffers.

[20] THE APACHE SOFTWARE FOUNDATION. *Apache Core Features: LimitRequestLine Directive*. [Dokumentace HTTP serveru Apache] [online]. ©2018 [cit. 2022-05-25]. Dostupné z: https://httpd.apache.org/docs/2.2/mod/core.html#limitrequestline.

[21] LITESPEED TECHNOLOGIES. *Server Tuning: Max Request URL Length (bytes)*. [Dokumentace HTTP serveru LiteSpeed] [online]. ©2013–2022 [cit. 2022-05-25]. Dostupné z: https://www.litespeedtech.com/docs/webserver/config/tuning#maxReqURLLen.

[22] MICROSOFT a komunita přispěvatelů. *URL Routing*. [online]. 02/19/2020 [cit. 2022-05-25]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/web-forms/overview/getting-started/getting-started-with-aspnet-45-web-forms/url-routing.

[23] MIESSLER, Daniel; HADDIX, Jason; g0tmi1k a komunita přispěvatelů. *raft-small-files.txt*. [Seznam častých souborů webový aplikací] [online]. Aug 23, 2017 [cit. 2022-05-25]. Dostupné z: https://github.com/danielmiessler/SecLists/blob/master/Discovery/Web-Content/raft-small-files.txt.

[24] MOZILLA a komunita přispěvatelů. *HTTP headers: Security [odstavec X-Powered-By]*. [online]. May 13, 2022 [cit. 2022-05-25]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers.

[25] MOZILLA a komunita přispěvatelů. *Using HTTP cookies*. [online]. Apr 27, 2022 [cit. 2022-05-25]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies.

[26] MICROSOFT a komunita přispěvatelů. *Adjust case sensitivity: Differences between Windows and Linux case sensitivity*. [online]. 04/28/2022 [cit. 2022-05-25]. Dostupné z: https://docs.microsoft.com/en-us/windows/wsl/case-sensitivity.

[27] MICROSOFT a komunita přispěvatelů. *Naming Files, Paths, and Namespaces: File and Directory Names: Naming Conventions*. [online]. 05/05/2022 [cit. 2022-05-25]. Dostupné z: https://docs.microsoft.com/en-us/windows/win32/fileio/naming-a-file.

[28] FORTINET. *What is a Transmission Control Protocol TCP/IP Model?*. [online]. ©2022 [cit. 2022-05-25]. Dostupné z: https://www.fortinet.com/resources/cyberglossary/tcp-ip.

[29] LYON, Gordon "Fyodor". *Nmap Network Scanning*. [online]. ©2008–2022 [cit. 2022-05-25]. Dostupné z: https://nmap.org/book/toc.html.

[30] PORTSWIGGER. *Burp Suite*. [online]. ©2022 [cit. 2022-05-25]. Dostupné z: https://portswigger.net/burp.

[31] PORTSWIGGER. *Using Burp Proxy*. [online]. May 17, 2022 [cit. 2022-05-25]. Dostupné z: https://portswigger.net/burp/documentation/desktop/tools/proxy/using.

[32] PORTSWIGGER. *Using Burp Repeater*. [online]. May 17, 2022 [cit. 2022-05-25]. Dostupné z: https://portswigger.net/burp/documentation/desktop/tools/repeater/using.

[33] PORTSWIGGER. *Using Burp Intruder*. [online]. May 17, 2022 [cit. 2022-05-25]. Dostupné z: https://portswigger.net/burp/documentation/desktop/tools/intruder/using.

[34] WAPPALYZER a komunita přispěvatelů. *wappalyzer.js*. [Zdrojový kód nástroje Wappalyzer] [online]. Jan 24, 2022 [cit. 2022-05-25]. Dostupné z: https://github.com/wappalyzer/wappalyzer/blob/master/src/wappalyzer.js.

[35] SHAH, Saumil. *An Introduction to HTTP fingerprinting*. [online]. 19th May, 2004 [cit. 2022-05-25]. Dostupné z: https://www.net-square.com/httprint_paper.html.

[36] RUEF, Marc. *HTTPRecon Documentation*. [online]. ©2007-2022 [cit. 2022-05-25]. Dostupné z: https://www.computec.ch/projekte/httprecon/?s=documentation.

[37] RUEF, Marc. *HTTPRecon Database*. [online]. ©2007-2022 [cit. 2022-05-25]. Dostupné z: https://www.computec.ch/projekte/httprecon/?s=database.

[38] IBM CLOUD EDUCATION. *Virtualization*. [online]. 19 June 2019 [cit. 2022-05-25]. Dostupné z: https://www.ibm.com/cloud/learn/virtualization-a-complete-guide.

[39] IBM CLOUD EDUCATION. *Containerization*. [online]. 15 May 2019 [cit. 2022-05-25]. Dostupné z: https://www.ibm.com/cz-en/cloud/learn/containerization.

[40] DOCKER INC. a komunita přispěvatelů. *Docker overview*. [online]. Feb 6, 2022 [cit. 2022-05-25]. Dostupné z: https://docs.docker.com/get-started/overview/.

[41] BUCHANAN, Ian. *Containers vs. virtual machines*. [online]. ©2022 [cit. 2022-05-25]. Dostupné z: https://www.atlassian.com/microservices/cloud-computing/containers-vs-vms.

[42] MICROSOFT a komunita přispěvatelů. *Windows Subsystem for Linux Documentation*. [online]. 12/29/2021 [cit. 2022-05-25]. Dostupné z: https://docs.microsoft.com/en-us/windows/wsl/.

[43] DOCKER INC. a komunita přispěvatelů. *About storage drivers*. [online]. Jan 21, 2022 [cit. 2022-05-25]. Dostupné z: https://docs.docker.com/storage/storagedriver/.

[44] DOCKER INC. a komunita přispěvatelů. *Dockerfile reference*. [online]. Sep 3, 2020 [cit. 2022-05-25]. Dostupné z: https://docs.docker.com/engine/reference/builder/.

[45] DOCKER INC. a komunita přispěvatelů. *Overview of Docker Compose*. [online]. May 17, 2022 [cit. 2022-05-25]. Dostupné z: https://docs.docker.com/compose/.

[46] DOCKER INC. a komunita přispěvatelů. *Compose specification*. [online]. May 22, 2022 [cit. 2022-05-25]. Dostupné z: https://docs.docker.com/compose/compose-file/.

[47] MICROSOFT a komunita přispěvatelů. *Windows container requirements*. [online]. 10/22/2021 [cit. 2022-05-25]. Dostupné z: https://docs.microsoft.com/en-us/virtualization/windowscontainers/deploy-containers/system-requirements.

[48] DOCKER INC. a komunita přispěvatelů. *Docker Desktop for Windows user manual*. [online]. May 5, 2022 [cit. 2022-05-25]. Dostupné z: https://docs.docker.com/desktop/windows/#switch-between-windows-and-linux-containers.

[49] DOCKER INC. a komunita přispěvatelů. *Networking features in Docker Desktop for Windows*. [online]. Sep 13, 2021 [cit. 2022-05-25]. Dostupné z: https://docs.docker.com/desktop/windows/networking/.

[50] PENTEREP. *Web Platform for Penetration Testing*. [online]. [cit. 2022-05-25]. Dostupné z: https://www.penterep.io/.

[51] PENTEREP. [online vektorová grafika]. [cit. 2022-05-25]. Dostupné z: https://www.penterep.io/img/platform.svg.

[52] PYTHON SOFTWARE FOUNDATION. *Built-in Constants: None*. [online]. May 25, 2022 [cit. 2022-05-25]. Dostupné z: https://docs.python.org/3/library/constants.html#None.

[53] SHODAN. *What is Shodan?*. [online]. [cit. 2022-05-25]. Dostupné z: `https://help.shodan.io/the-basics/what-is-shodan`.

Seznam symbolů a zkratek

API Application Programming Interface

CSS Cascading Style Sheets

CSV Comma-separated values

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

IIS Internet Information Services

IP Internet Protocol

JSON JavaScript Object Notation

JSP JavaServer Pages

NSE Nmap Scripting Engine

PHP PHP: Hypertext Preprocessor

TCP Transmission Control Protocol

TLS Transport Layer Security

TTL Time to Live

URL Uniform Resource Locator

WSL Windows Subsystem for Linux

A Obsah elektronické přílohy

Elektronická příloha obsahuje bakalářskou práci ve formátu PDF, adresář se zdrojovým kódem vyvinutého nástroje a adresář se soubory experimentálního pracoviště.

Adresář zdrojového kódu nástroje obsahuje zdrojové soubory nástroje a soubory potřebné pro instalaci všech jeho softwarových závislostí.

Adresář experimentálního pracoviště obsahuje návod k použití pracoviště, 4 konfiguračního soubory pro *Docker Compose* a 3 další adresáře. Každý z těchto adresářů náleží některému z kontejnerů experimentálního pracoviště. Jejich obsahem jsou odpovídající `Dockerfile` soubory, adresáře se soubory webových aplikací, případně další konfigurační soubory či skripty.

- `/` ..... Kořenový adresář přiloženého archivu
  - `ptwebident` ..... Kořenový adresář nástroje
    - `ptwebident` ..... Zdrojové soubory
      - `__init__.py`
      - `analysis.py`
      - `csvformat.py`
      - `dataset.csv`
      - `ptwebident.py`
      - `results_data.py`
      - `urlattributes_htmlparser.py`
      - `utils.py`
    - `LICENSE`
    - `MANIFEST.in`
    - `README.MD`
    - `requirements.txt`
    - `setup.py`
  - `testing_environment` ..... Kořenový adresář testovacího prostředí
    - `centos-apache-php` ..... Soubory kontejneru `centos-apache-php`
      - `webapp` ..... Zdrojové soubory webové aplikace
        - `index.php`
      - `Dockerfile`
      - `run.sh`
    - `debian-nginx-express` ..... Soubory kontejneru `debian-nginx-express`
      - `webapp` ..... Zdrojové soubory webové aplikace
        - `views`
          - `index.ejs`
        - `package.json`
        - `server.js`
      - `Dockerfile`
      - `run.sh`
      - `server.conf`
    - `ubuntu-tomcat-jsp` ..... Soubory kontejneru `ubuntu-tomcat-jsp`
      - `webapp` ..... Zdrojové soubory webové aplikace

```
    │   │   └── index.jsp
    │   └── Dockerfile
    ├── docker-compose-apache.yml
    ├── docker-compose-iis.yml
    ├── docker-compose-nginx.yml
    ├── docker-compose-tomcat.yml
    └── readme.txt
└── BP_Vasicek.pdf ..... Bakalářská práce (PDF)
```