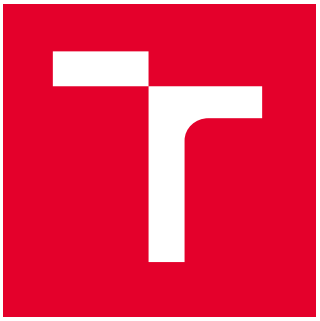


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

APLIKACE NA KLONOVÁNÍ STRÁNEK PRO BEZPEČNOSTNÍ TESTOVÁNÍ

APPLICATION FOR CLONING WEBSITES FOR SECURITY TESTING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Dita Kubíčková**
AUTHOR

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

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Informační bezpečnost**

Ústav telekomunikací

Studentka: Bc. Dita Kubíčková ***ID:*** 202674

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

NÁZEV TÉMATU:

Aplikace na klonování stránek pro bezpečnostní testování

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je návrh a vývoj softwarového nástroje, který umožní naklonovat webovou stránku. Naklonovanou stránku bude následně možné využít při bezpečnostním testování metodami sociálního inženýrství (př. phishingová kampaň v rámci Red Teamingu). V teoretické části analyzujte dostupné nástroje (př. SET v Kali Linux, HTTrack Website Copier nebo GoPhish) na experimentálním pracovišti a identifikujte funkcionalitu pro několik scénářů. Pozornost věnujte také právním aspektům nástroje. Na základě dosažených výsledků navrhněte vlastní aplikaci eliminující nefunkčnost současných řešení. Aplikace by měla být schopná realizovat obfuskaci zdrojového kódu k eliminaci detekčních pravidel antivirových systémů. V rámci praktické části práce implementujte vlastní aplikaci a ověřte její funkčnost, dosažené výsledky přehledně zpracujte.

DOPORUČENÁ LITERATURA:

[1] NAJERA-GUTIERREZ, Gilberto; ANSARI, Juned Ahmed. Web Penetration Testing with Kali Linux: Explore the methods and tools of ethical hacking with Kali Linux. Packt Publishing Ltd, 2018.

[2] RODRÍGUEZ-CORZO, Julio Alexander; ROJAS, Alix E.; MEJÍA-MONCAYO, Camilo. Methodological model based on Gophish to face phishing vulnerabilities in SME. In: 2018 ICAI Workshops (ICAIW). IEEE, 2018. p. 1-6.

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

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

Konzultant: Daniel Hejda

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

ABSTRAKT

Diplomová práce je zaměřena na vytvoření aplikace, s jejíž pomocí je možné naklonovat webovou stránku a využít ji během bezpečnostního testování v rámci phishingové kampaně. V teoretické části je popsán pojem sociálního inženýrství s důrazem na phishing a je představen průběh phishingové kampaně a model útoku. Dále jsou porovnány současné nástroje pro kopírování stránek a zhodnoceny jejich nedostatky. Praktická část je věnována návrhu a vývoji vlastní aplikace v jazyce Python za použití Web Driveru a poskytuje výsledky provedeného testování aplikace v porovnání s nástrojem GoPhish, který je pro klonování stránek často využíván.

KLÍČOVÁ SLOVA

klonování stránek, phishing, sociální inženýrství, model útoku, bezpečnostní testování, GoPhish, Web Driver

ABSTRACT

The master thesis is focused on developing application for website cloning that could be used in phishing campaign during security testing. The theoretical part describes social engineering with emphasis on phishing. It also introduces phishing campaigns steps' and attack framework. Then, it compares currently used tools for website cloning and evaluates their shortcomings. The practical part deals with development of the application in Python using Web Driver and provides the results of application testing in comparison to GoPhish which is often used for website cloning.

KEYWORDS

website cloning, phishing, social enginnering, attack framework, security testing, GoPhish, Web Driver

KUBÍČKOVÁ, Dita. *Aplikace na klonování stránek pro bezpečnostní testování*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2023, 77 s. Diplomová práce. Vedoucí práce: doc. Ing. Zdeněk Martinásek, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Dita Kubíčková
VUT ID autora:	202674
Typ práce:	Diplomová práce
Akademický rok:	2022/23
Téma závěrečné práce:	Aplikace na klonování stránek pro bezpečnostní testování

Prohlašuji, že svou závěrečnou práci jsem vypracovala 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 autorka uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušila autorská práva třetích osob, zejména jsem nezasáhla nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědoma 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 autorky*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Ráda bych poděkovala vedoucímu diplomové práce panu doc. Ing. Zdeňku Martináskovi, Ph.D. za odborné vedení, konzultace a podnětné návrhy k práci. Mé poděkování patří také panu JUDr. MgA. Jakubu Míškovi, Ph.D. za poskytnutou konzultaci k právnímu pohledu na problematiku.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Práce se zabývá problematikou sociálního inženýrství a phishingových kampaní a jejich využití v rámci bezpečnostního testování. Hlavním cílem práce je navrhnout a implementovat aplikaci, která by dokázala vytvořit klon zadané webové stránky, přičemž by eliminovala nedostatky nástrojů, které jsou k tomuto účelu používány v současnosti. Problémy spočívají zejména v chybějících ikonách a zobrazení špatného fontu v případech, kdy jsou tyto prvky nalinkovány v externích souborech, jelikož nástroje tyto soubory neprocházejí. Často také dochází k odepření přístupu na stránku díky nastaveným detekčním pravidlům, které nástroj identifikují jako bota. Vytvoření věrohodné kopie webové stránky je přitom klíčové pro spuštění úspěšné phishingové kampaně. Jejím prostřednictvím je zvyšováno povědomí o hrozbách sociálního inženýrství a snižuje se pravděpodobnost, že se zaměstnanec stane obětí phishingového útoku. Z těchto důvodů si práce klade za cíl vytvořit aplikaci, která by neměla limitace současných nástrojů a klonovala stránky s větší přesností.

V teoretické části práce bude definován pojem sociálního inženýrství a zmapován jeho vývoj od pravděpodobného počátku až do současné podoby. Budou představeny tři různé modely útoků realizovaných sociálním inženýrstvím se specifikacemi jejich jednotlivých kroků. Na nejpodrobnějším z nich pak bude celý proces popsán důkladněji s příklady možného provedení u každé fáze. Dále bude pozornost zaměřena na phishing, jakožto jednu z metod sociálního inženýrství, v níž je mimo jiné využíváno podvodných stránek, a na bezpečnostní testování, které se naopak snaží minimalizovat dopad phishingového útoku. Následně bude analyzován současný stav problematiky nástrojů na klonování stránek a nástroj GoPhish zde bude podroben detailnějšímu testování. Nalezené nedostatky budou demonstrovány na vzorku a budou diskutovány jejich příčiny. Pozornost bude věnována také právnímu pohledu na kopírování stránek, důsledkům vzniku protiprávního stavu a výjimkám ze zákonných ustanovení.

Praktická část práce se bude zabývat návrhem, implementací a testováním vlastní aplikace. Budou zde popsány využité technologie a knihovny a na několika diagramech pak bude zobrazena architektura celé aplikace i jejích jednotlivých součástí a detailně zde budou popsány jednotlivé funkcionality. Dále tato část poskytne popis provedeného testování vyvinuté aplikace na vzorku více než třiceti stránek a porovnání dosažených výsledků s nástrojem GoPhish. Budou zde diskutovány možné překážky bránící kompletnímu naklonování některých stránek a způsoby řešení. Jedno z navrhovaných řešení bude do aplikace implementováno v podobě funkce pro vkládání cookies uživatelem.

1 Teoretická část

Vzestup kyberkriminality je patrný v zemích po celém světě a to i přesto, že velká část trestných činů zůstane neodhalena nebo není nahlášena. Jen v České republice byl v roce 2021 registrován nárůst kriminality páchané v kyberprostoru o 17,9 % oproti předchozímu roku [1]. Nejčastěji je pak nahlašován phishing (nebo některá jeho obdoba, podrobněji bude popsáno v kapitole 1.2), nezaplacení nebo nedoručení objednávky, únik osobních informací a v neposlední řadě krádež identity [2].

Z reportu společnosti Verizon [3] vyplývá, že lidská chyba je stále tím hlavním důvodem, proč je velké množství pokusů o útok úspěšných. Oběťmi jsou mezinárodní společnosti a organizace, ale i malé firmy. A právě této zranitelnosti v lidském faktoru využívá technika sociálního inženýrství.

1.1 Sociální inženýrství

1.1.1 Vývoj sociálního inženýrství

Pojem SI (sociální inženýrství) v dnešním smyslu v souvislosti s informační bezpečností, tedy jako nástroje k získání nějaké útočníkovi utajované informace prostřednictvím oklamání a získání důvěry nositele daného aktiva, se poprvé objevuje ke konci 50. let 20. století. Jednalo se o počátky „phone phreaking“ [4], metody prozkoumávání, jakým způsobem fungují telefonní sítě. John Draper, jeden z prvních „phone phreaks“ té doby, používal sociální inženýrství k získání informací ze zaměstnanců společnosti Bell Telephone tím, že je přiměl uvěřit, že je jedním z nich [4]. Je potřeba podotknout, že samotné zkoumání zapojení telefonních sítí není trestně postihnutelným činem, nicméně někteří z těchto telefonních entuziastů zacházeli o něco dále [5] a získané technické znalosti používali k uskutečňování neplacených hovorů, připojování se k cizím konferenčním hovorům a obecně zneužívání sítí [4]. Již tehdy se však jednalo o odhalitelné skutečnosti, docházelo k četným policejním vyšetřováním se zapojením FBI (Federal Bureau of Investigation – Federální úřad pro vyšetřování) a bylo provedeno několik zatčení a uvěznění [5].

V 60. a 70. letech 20. století nastává prudký vývoj v oblasti informačních technologií a společně s nimi se vyvíjí i metody ochrany. Nově jsou uživatelská data a komunikace chráněny kryptografickými prostředky jako je například asymetrická kryptografie, hashování, bezpečnostní protokoly, ověřování identity hesly nebo implementací access listů [4].

V 80. letech až do poloviny 90. let se útočníci (protože říkat jim v tuto chvíli stále nadšenci by byl eufemismus) v reakci na technologický vývoj spoléhají více na netechnické způsoby dosažení cílů a využívají metod sociálního inženýrství, jako

je přesvědčování, klamání, vydávání se za jinou osobu, manipulace nebo reverzní sociální inženýrství. Ve stejné době, roku 1984, se objevuje pravděpodobně první publikace zabývající se sociálním inženýrstvím, a to v časopise 2600: The Hacker Quarterly v článku Vital Ingredients: Switching Centers and Operators. Sociální inženýrství je v něm popsáno jako technika přesvědčování k získání informací, což je v podstatě princip, pod jakým jej chápeme i dnes [4] [6].

Od druhé poloviny 90. let do roku 2001 dochází k pokroku v oblasti informační bezpečnosti. Autoři práce Defining Social Engineering in Cybersecurity nazývají toto období jako fáze profesionálního hackera, nově se v útocích provedených za pomocí sociálního inženýrství objevuje malware a phishingové e-maily se vyskytují stále častěji. Zároveň si lidé začínají uvědomovat, že jsou to právě oni, kdo jsou nejslabším místem jakéhokoliv zabezpečení [6].

V následujících zhruba deseti letech se útoky vyvíjí z hlediska jejich cílů a využití a stávají se více sofistikované. V odpovědi na to se ve firmách poprvé objevuje bezpečnostní testování zaměstnanců, kdy se systémoví administrátoři snaží skrze sociální inženýrství odhalit zaměstnance, kteří by mohli být tomuto typu útoku náchylní [7].

Další technologický vývoj od roku 2012 umožnil rozšíření sociálního inženýrství do zcela nových rozměrů. V současnosti jsou terči těchto útoků také IoT (Internet of Things) zařízení, senzory a routery. Je využíváno strojového učení a umělé inteligence a do útoků jsou zapojovány nové typy hrozeb, například APT (Advanced Persistent Threat). Takto vytvořený cílený útok se ukazuje jako velmi efektivní a představuje závažný problém pro informační i fyzickou bezpečnost [6], a to nejen pro organizace a firmy, ale i pro jednotlivce. O závažnosti ohrožení kritické infrastruktury státu nemluvě [6].

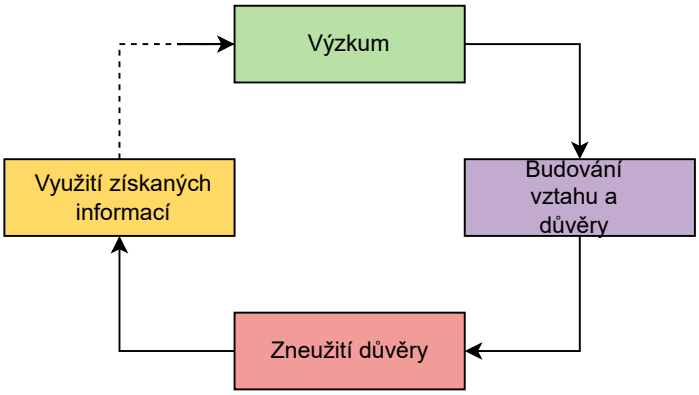
1.1.2 Model útoku

Zajímavý náhled do problematiky sociálního inženýrství nabízí Kevin D. Mitnick, známý jako „the world's most famous hacker“ [8], který se SI začal zabývat už v dospívání, kdy díky informacím, které získal z řidičů veřejné dopravy, dokázal falšovat jízdenky. Později se jeho zájem přenesl v profesi a stal se soukromým vyšetřovatelem. Nicméně fascinace získáváním informací ho neopustila a v průběhu několika let získal neoprávněné přístupy do největších společností a zdrojové kódy telekomunikačních zařízení a operačních systémů [9]. I když nalezené zranitelnosti nikdy nezneužil, jednalo se o porušení zákonů a byl odsouzen k odnětí svobody [8]. Mitnick své zkušenosti publikoval v knize The Art of Deception: Controlling the Human Element of Security [9], kde představuje model cyklu útoku sociálním inženýrstvím. Tento model, zobrazen na obrázku 1.1, se skládá ze čtyř fází: výzkum, budování

vztahu a důvěry, zneužití důvěry a využití získaných informací. Poslední fáze pak přechází zpět do výzkumu, nebo pokud bylo dosaženo cíle, je proces ukončen. Během výzkumu jsou sbírány veškeré informace o cíli, většinou z veřejně dostupných zdrojů technikami OSINT (Open Source Intelligence). Obecně platí, že čím více informací útočník/sociální inženýr získá, tím větší šanci má na úspěšné vybudování vztahu s obětí. Tím je zahájena druhá fáze – vytvoření vztahu a důvěry. Lidé jsou více ochotní sdílet data s těmi, kterým důvěřují. Útočník tedy často při komunikaci využívá identity někoho jiného nebo předstírá, že zná např. spolupracovníka oběti. Poté, co je vytvořen vztah důvěry, dochází k jeho zneužití. Může se jednat o požadavek o poskytnutí nějaké informace, žádost o laskavost, vybídnutí k nějaké akci nebo zmanipulování oběti k tomu, aby požádala o pomoc útočníka. V kladném případě pak nastává poslední fáze, tedy využití získaných aktiv [10]. Ty mohou posloužit jako vstupní parametr dalšího útoku. Tento model je však velmi obecný a neposkytuje vhodné detaily.

Odlišný pohled představuje model autorů Francoise Moutona, Louise Leenen, Mercia M. Malana a H. S. Ventera v práci Towards an Ontological Model Defining the Social Engineering Domain [11]. Dle jejich ontologického modelu se útok skládá z následujících částí: sociálního inženýra, oběti, cíle (finanční obohacení, získání přístupů), média, pomocí kterého dojde ke kontaktování oběti (například e-mailem, přes SMS, prostřednictvím USB disku nebo osobně), principu, na jehož základě oběť vyhoví požadavkům útočníka (přátelství, závazek k něčemu, pocit vzácnosti příležitosti, sociální uznání, autorita) a techniky, jimiž je tohoto cíle dosaženo (phishing, quid pro quo, pretexting). Ve chvíli, kdy útočník stanovil médium, princip a techniku, může pokročit k vektoru útoku a samotnému provedení [11].

Zřejmě nejlepší model je popsán ve studii Social Engineering Attack Framework [10], ve které autoři staví na výše zmíněném Mitnickově cyklu útoku a rozvádějí jej o

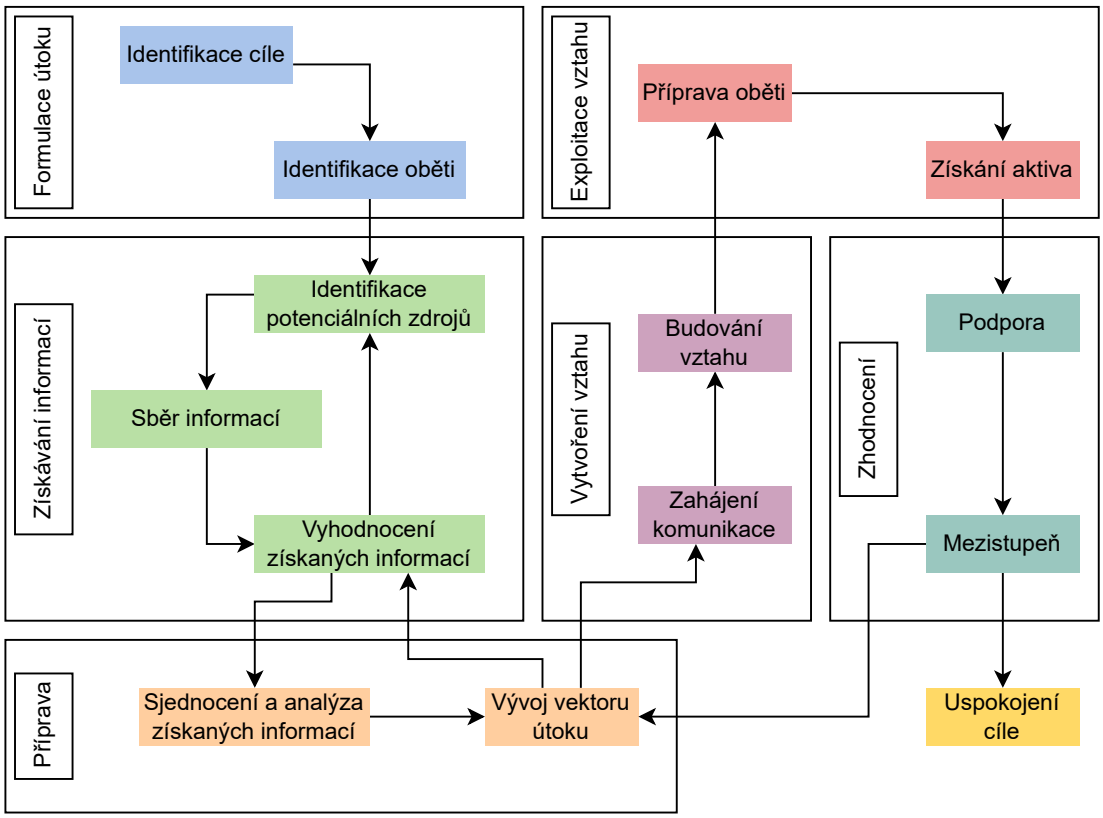


Obr. 1.1: Model cyklu sociálního inženýrství dle Mitnicka.

další fáze a návaznosti. Těmito fázemi jsou: formulace útoku, získávání informací, příprava, vytvoření vztahu, exploitace vztahu a zhodnocení. Jednotlivé kroky jsou více popsány v následujícím textu společně s příklady. Celý framework je pak možné vidět na obrázku 1.2.

První fáze útoku se zaměřuje na jeho formulaci, tedy jaký je jeho účel, co je cílem a prostřednictvím které vhodné osoby nebo osob je možné ho dosáhnout [10]. Typickým příkladem cíle může být získání přístupu do budovy firmy, přičemž obětí zde může být pracovník ochranky nebo recepční.

Po identifikaci oběti začíná fáze získávání informací. Jedná se o pravděpodobně nejdůležitější část celého útoku, jelikož od kvality informací se odvíjí to, s jakou pravděpodobností bude útočník schopný si utvořit vztah s obětí a vzbudit v ní důvěru. Nejdříve jsou identifikovány potenciální zdroje pro sběr údajů o oběti a cíli. Nejčastěji jde o zdroje, které jsou volně dostupné a často potřebné informace o sobě sděluje oběť sama, ať už skrze příspěvky na sociálních sítích, prostřednictvím zveřejňovaných fotografií či na osobním blogu [10]. Online zdroji to však nekončí. Útočník může přistoupit k dalším metodám sběru informací, jako je třeba „dumpster diving“, kdy je využito nedostatečné opatrnosti firem při vyhazování dokumentů a materi-



Obr. 1.2: Social Engineering Attack Framework.

álů. Není přitom neobvyklé najít interní informace nebo dokonce i hesla. To stejné platí pro vyřazené vybavení. Již několikrát se stalo, že potřebné údaje byly získány z hard disků počítačů, kterých se firmy zbavily. Další používané metody vyžadují více či méně fyzický přístup k člověku, jež je zdrojem informací. Patří sem sledování, odposlouchávání rozhovorů a telefonátů na veřejnosti či ve veřejné dopravě nebo „shoulder surfing“, tedy pozorování, co dotyčný píše [12]. Z těchto důvodů některé firmy zavádějí pravidla, která omezují práci a vyřizování pracovních hovorů na veřejných místech jako jsou kavárny a letiště. Poté, co je dokončen sběr informací, je provedeno jejich zhodnocení z hlediska užitečnosti a relevantnosti a v případě potřeby je celá fáze znovu zopakována.

Následuje fáze přípravy, ve které útočník sjednotí informace do logického celku a na jeho základě vytvoří vhodnou uvěřitelnou záminku pro kontaktování oběti a vektor útoku. Ten se skládá z cíle, kterého chce dosáhnout, určení oběti, použitého média, principů a technik. Vše přitom závisí na kvalitě získaných informací a útočníkově znalosti povahy oběti [10].

Vytvoření vztahu je dalším zásadním krokem útoku. Skrze vybrané médium je zahájena komunikace s obětí a postupně je budován vztah s cílem získat její důvěru [10]. Útočník se přitom vydává za nějakou jinou osobu, často za pracovníka IT podpory. Možností je ale více, může předstírat kolegu, který má nějaký problém a potřebuje pomoct, manažera, který jedná z pozice autority nebo se naopak vydávat za uklízeče nebo údržbáře a dostat se tak téměř kamkoliv [12].

Jakmile si útočník získá důvěru oběti, nastává fáze exploitace vztahu. Nejdříve je oběť vhodnou manipulací uvedena do požadovaného emočního stavu [10], ve kterém se není schopna rozhodovat racionálně, což vede k předvídatelnému chování na základě pocitů, tedy chování, které útočník potřeboval. Existuje několik různých taktik, jak oběť do takového stavu dostat, přičemž výsledné navozené emoce jsou odlišné a útočník musí volit dle toho, jaká je povaha oběti a čeho chce dosáhnout. Příkladem některých z nich může být jednání z pozice autority, navození pocitu vzácnosti příležitosti, vyjádření zalíbení, stejných zájmů nebo lichocení, přesvědčení, že musí oplatit laskavost nebo dostát slibu a nebo jednoduše zahlcení tolika informacemi, že se soustředí jen na jejich vnímání a už je nevyhodnocuje. Uvedeno do praxe, pokud je například oběť vystavena nějakému emočnímu tlaku, jako je výhrůžka od útočníka představujícího autoritu, je menší pravděpodobnost, že bude zpochybňovat oprávněnost požadavku, který na ni má. Na druhou stranu přátelský přístup je také velmi efektivní, po získání důvěry oběti se stačí na požadovanou informaci jednoduše zeptat [12]. Mitnick tuto fázi shrnuje: „That is the whole idea: to create a sense of trust and then exploiting it.“ [13].

Po získání požadovaného aktiva přichází poslední fáze – zhodnocení. Během ní útočník přivede oběť do původního emočního stavu, aby vyloučil, že se oběť bude

cítit provinile kvůli poskytnutí informací, nebo že bude mít pocit, že se stala terčem útoku. Je důležité, aby z interakce měla dobrý či neutrální pocit, jelikož tím se snižuje pravděpodobnost, že o události bude dále přemýšlet. Následuje mezistupeň, kdy se útočník rozhodne, zda má veškeré potřebné informace k uspokojení cíle, nebo zda se potřebuje vrátit do fáze přípravy [10].

1.2 Phishing

Phishing patří mezi nejrozšířenější metody sociálního inženýrství a nejčastěji je realizován skrze e-mail a webové stránky. Princip je poměrně jednoduchý. Útočník zašle oběti legitimně vypadající e-mail, ideálně se spoofovanou adresou odesílatele [14], v němž se ji snaží přesvědčit k nějaké akci. Tou může být stažení anebo otevření přiloženého souboru nebo kliknutí na odkaz [15]. Často se jedná o výzvu ke změně hesla, přihlášení do bankovnictví a potvrzení nějakých informací, nebo informování o aktivitě na uživatelském účtu, které je třeba zkontrolovat. Využívá se zde časového nátlaku na oběť, například že heslo je třeba změnit do 24 hodin, jinak účet bude zablokován.

V případě kliknutí na odkaz je oběť přesměrována na kopii originální stránky a pokud zadá své údaje do přihlašovacího formuláře, jsou tyto zadané informace jako její jméno, heslo nebo číslo kreditní karty odeslány útočníkovi [14].

Phishing obecně není zaměřen na konkrétní jednu osobu. Podvrhnuté e-maily jsou zaslány naráz velkému počtu lidí a útočník počítá s pravděpodobností, že nějaká část z nich své údaje vyplní [16]. Existují však i další druhy phishingu lišící se prostředkem realizace nebo zaměřením. V následujících bodech jsou stručně popsány.

- **Vishing** – oběť je kontaktována útočníkem skrze telefonní hovor,
- **smishing** – oběť je kontaktována SMS zprávou [15],
- **spear-phishing** – cílený phishing zaměřený na konkrétní osobu, bývá úspěšnější než klasický phishing, ale vyžaduje důkladnější přípravu v podobě sběru informací o oběti [17],
- **whaling** – cílený phishing na vysoce postavenou osobu v organizaci nebo firmě,
- **pharming** – manipulace síťového provozu a přesměrování oběti z legitimní stránky na phishingovou, koncepčně velmi podobné útoku MITM (Man-in-the-middle) [15],
- **angler phishing** – útočník se vydává za zákaznickou podporu a skrze falešný účet na sociálních sítích cílí na nespokojené zákazníky, přičemž využívá toho, že skutečným pracovníkům zákaznické podpory trvá delší dobu, než daného zákazníka kontaktují [18].

1.3 Bezpečnostní testování

Důkladné školení zaměstnanců je zcela nezbytné a následné bezpečnostní testování zásadní. Jeho důležitost zřejmě nejlépe vystihuje narůstající počet realizovaných phishingových útoků. Celosvětově bylo k prvnímu čtvrtletí roku 2021 detekováno 611 877 unikátních phishingových stránek, což činí meziroční nárůst o 369 %. [19]. Oblast bezpečnostního testování je poměrně rozsáhlá, avšak pro účely této práce bude popsána jen problematika testování realizovaného pomocí sociálního inženýrství, konkrétně phishingová kampaň.

Vhodně provedená phishingová kampaň, ať už v rámci penetračního testování nebo samostatně, dokáže výrazně zlepšit povědomí zaměstnanců o hrozbách sociálního inženýrství a snížit tak pravděpodobnost, že útok reálného nositele hrozby bude úspěšný. Zároveň organizaci poskytuje představu o zranitelnosti vůči tomuto typu útoku jako celku [20].

Před začátkem samotného testování jsou nejdříve zaměstnanci seznámeni s rozdíly mezi legitimními a podvodnými e-maily, s technikami, které útočníci používají, jak mohou takové zprávy nahlásit a obecně, jak je vhodné postupovat. Je důležité, aby zaměstnanci věděli, že bude probíhat testování a pokud napoprvé neuspějí, nebude to znamenat stigmatizaci a zesměšňování. Že cílem není je nachytat, nýbrž vzdělávat [21].

V rámci testování jsou následně rozesílány e-maily mimikující reálné phishingové útoky a je zaznamenáváno chování uživatelů, jako je kliknutí na odkaz, stažení nebo otevření přiloženého souboru nebo vložení přihlašovacích údajů do formuláře na stránce pod odkazem. Často je obsah zprávy přizpůsoben pracovní náplni zaměstnance, aby lépe simuloval cílený útok, přičemž obtížnost detekovat, že se jedná o phishing, se stupňuje [21].

Po ukončení testování jsou získaná data vyhodnocena a na základě výsledků je vhodně vylepšena bezpečnost [22]. Zaměstnancům, kteří si nevedli úplně dobře, je poskytnuto další školení. Ostatní je vhodné ocenit, pozitivní motivace se zde ukazuje být klíčová a minimalizuje chyby [23].

1.4 Současný stav problematiky

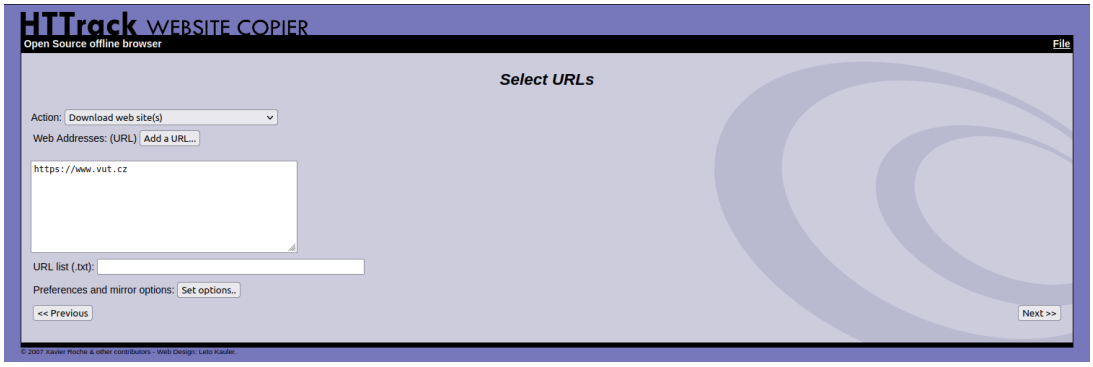
V současné době existuje několik nástrojů sloužících ke klonování webových stránek. Nejpoužívanější a také nejvíce diskutované v literatuře a odborné komunitě jsou programy HTTrack Website Copier, GoPhish a Social Engineering Toolkit. V následující podkapitole 1.4.1 jsou rozvedeny více dopodrobna a jsou porovnány jejich silné a slabé stránky. V podkapitole 1.4.2 jsou pak funkce těchto nástrojů po-

rovnány na testovacím vzorku stránek a příklady jsou demonstrovány na obrázcích, kde je možné názorně vidět nedostatky.

1.4.1 Dostupné nástroje

HTTrack Website Copier je program distribuovaný pod open source licencí GPL[1] a v oficiální dokumentaci je vedený jako „free software offline browser“ [25]. Umožňuje stáhnutí celé stromové struktury webových stránek včetně obrázků a její prohlížení v lokálním adresáři uživatele. Není primárně určen pro klonování stránek za účely phishingu, postrádá tedy jakoukoliv zpětnou vazbu – uživatel nezíská přihlašovací údaje oběti a ani neví, zda se je pokusila na klonované stránce vyplnit. HTTrack lze ovládat z terminálu nebo prostřednictvím poměrně starého GUI (Graphical User Interface – grafické uživatelské rozhraní). Aktuální verze programu je z roku 2017 a při nahlédnutí do githubu [26] lze zjistit, že má spoustu nevyřešených bugů, a že poslední issue, které bylo vyřešeno, je z roku 2018. Nicméně co se týká samotného kopírování stránek, tento nástroj vrátil během testování mnohdy lepší výsledky než novější nástroje, které jsou uvedeny dále. Náhled GUI pro vložení URL (Uniform Resource Locator) adresy klonované stránky je možné vidět na obrázku 1.3.

GoPhish, free open source framework pod licencí MIT[2], je přímo určený pro tvorbu phishingových kampaní. Po spuštění v terminálu se rozběhne server na localhostu uživatele a vytvoří se databáze pro ukládání získaných přihlašovacích údajů [28]. V grafickém rozhraní pak uživatel může vytvořit podvodný e-mail, naklonovat stránku a určit, kam bude oběť přesměrována po vyplnění svých údajů do



Obr. 1.3: HTTrack - náhled vložení URL klonované stránky.

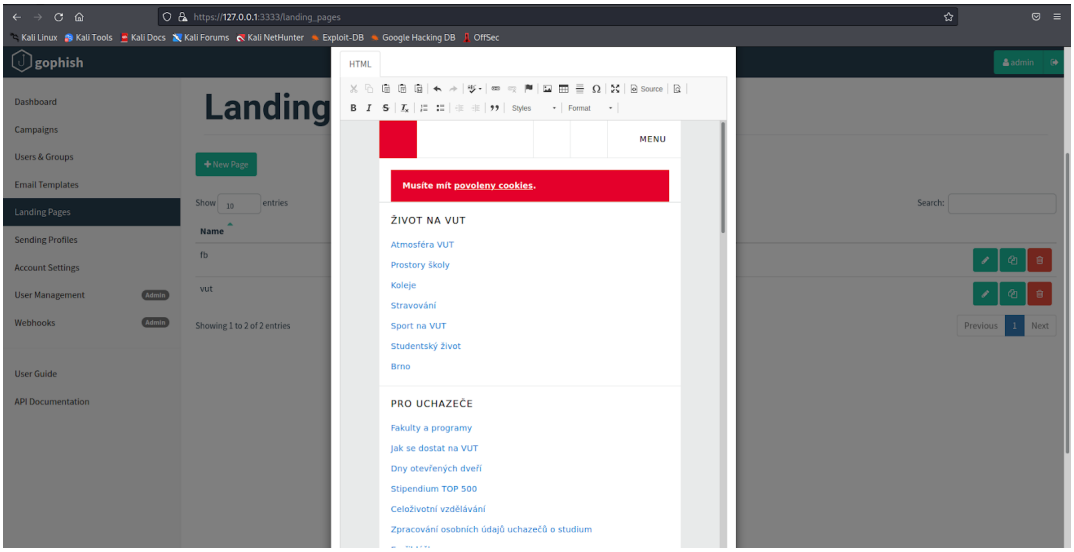
[1]GNU General Public Licence je svobodná copyleftová licence umožňující svobodně sdílet a modifikovat software, více je možné se dočíst zde [24].

[2]Permisivní licence s velmi málo omezeními, umožňuje volné užití, kopírování, modifikaci, slučování, publikování, distribuci, podlicencování a prodávání kopií softwaru, zároveň je kompatibilní s licencí GNU GPL. Celé znění je dostupné na oficiálních stránkách [27].

formuláře falešné stránky. Pro každou takto vytvořenou kampaň lze sledovat, kteří příjemci poskytli svá citlivá data, a kteří například jen otevřeli e-mail nebo klikli na odkaz na podvodnou stránku nebo phishing nahlásili. Program je udržovaný, poslední release proběhl v září minulého roku. Na obrázku 1.4 je zobrazena tzv. „Landing Page“, naklonovaná stránka pro kampaň.

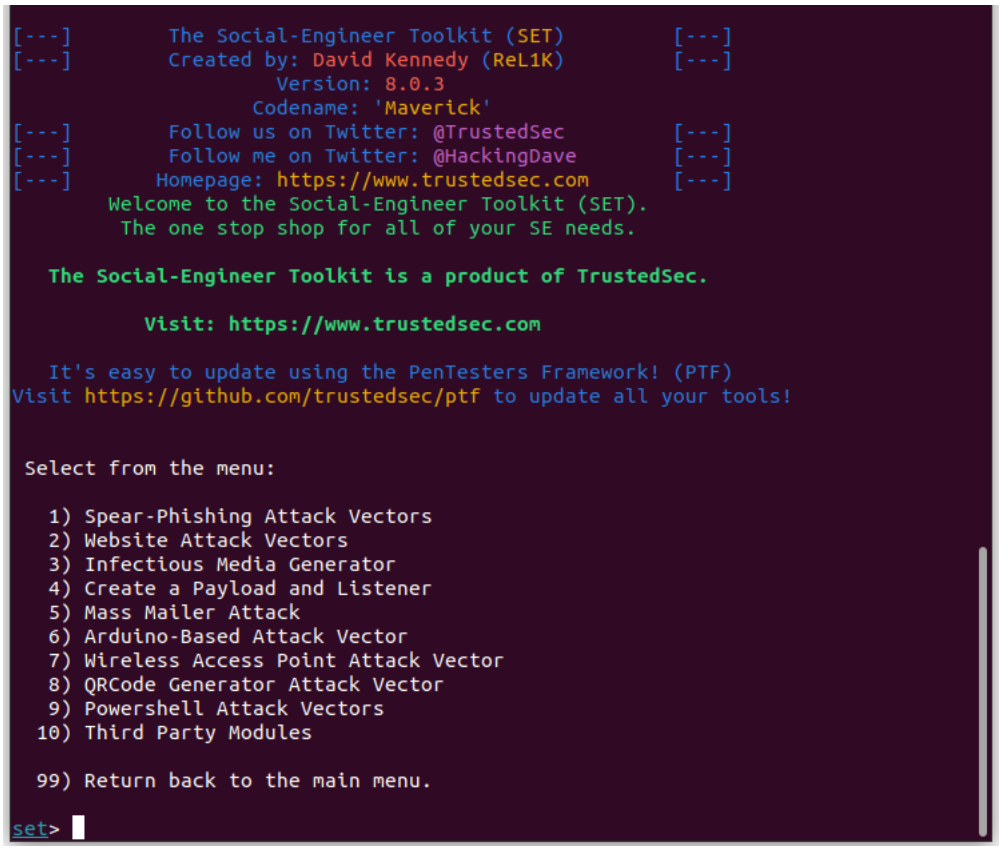
Social Engineering Toolkit, často uváděn jen zkratkou SET, je rovněž open source nástroj, vydaný pod licencí BSD[3]. Slouží k penetračnímu testování metodami sociálního inženýrství a kromě phishingu zahrnuje i další vektory útoku. Například možnost vytvoření payloadu pro útok pomocí Metasploitu, útok Java Applet pro spoofování Java certifikátu, TabNabbing nebo provedení powershell injection skrze naklonovanou stránku otevřenou v prohlížeči oběti. Stejně jako předchozí nástroje, SET lze spustit jak ve Windows, tak na Linuxových distribucích a experimentálně i na Mac OS X [30] (nejedná se však o stabilní verzi). V případě Kali Linux je nástroj už předinstalován. Poslední verze vyšla v roce 2017 a podle dostupných informací v repozitáři [31] nejsou reportované problémy s některými funkcionalitami již zhruba rok řešeny, některé i déle. Níže uvedený obrázek 1.5 zobrazuje hlavní menu nástroje, kde si uživatel volí typ útoku.

V následující tabulce 1.1 jsou shrnuty a porovnány již zmíněné hlavní vlastnosti těchto nástrojů z pohledu uživatele. Je zřejmé, že nejlépe vychází nástroj GoPhish, a to hlavně díky udržovanosti a přehlednému zobrazení získaných údajů. V podkapitole 1.4.2 jsou pak tyto tři nástroje otestovány na několika stránkách a GoPhish je



Obr. 1.4: GoPhish - náhled na vytváření klonované stránky.

[3]Permisivní licence umožňující užití a redistribuci softwaru za jakýmkoli účelem, a to s modifikacemi i bez nich. Podrobněji jsou popsány podmínky například tady [29].



Obr. 1.5: SET - náhled menu volby typu útoku.

dále podroben podrobnějšímu výzkumu z hlediska užívaných webových technologií klonovaných stránek.

1.4.2 Porovnání funkcí nástrojů na vzorku

Nástroje byly testovány na operačním systému Kali Linux verze 2022.3 ve virtuálním prostředí. Jako vhodný vzorek byly vybrány stránky pěti osobních internetových bankovnictví lokalizovaných v české doméně a přihlašovací stránka do Informačního systému VUT (Vysoké učení technické).

V následující tabulce 1.2 lze vidět porovnání, zda se nástroji podařilo stránky naklonovat. Značka fajfky znamená úspěšný pokus, kdy nedošlo ke ztrátě žádných prvků, křížek značí úplný neúspěch ve formě rozbité stránky, chybějících obrázků, posunutých textů, překrývajících se prvků, zobrazení zcela prázdné stránky nebo hlášení chybového stavu aplikace a tylda představuje stav, kdy naklonovaná stránka není stoprocentní, ale za určitých okolností lze předpokládat, že by si toho oběť nevšimla (zvlášť, pokud by na ni byl vyvinut tlak k nějaké okamžité akci) nebo by to považovala za chybu zobrazení prohlížeče.

Tab. 1.1: Porovnání vlastností nástrojů.

	HTTrack	GoPhish	SET
licence	GNU GPL	MIT	BSD
open source	✓	✓	✓
GUI	✓	✓	×
udržovaný nástroj	×	✓	×
zobrazení získaných údajů	×	✓	✓
databáze získaných údajů	×	✓	×
kopírování QR kódu	×	×	×

Tab. 1.2: Porovnání nástrojů na vzorku stránek.

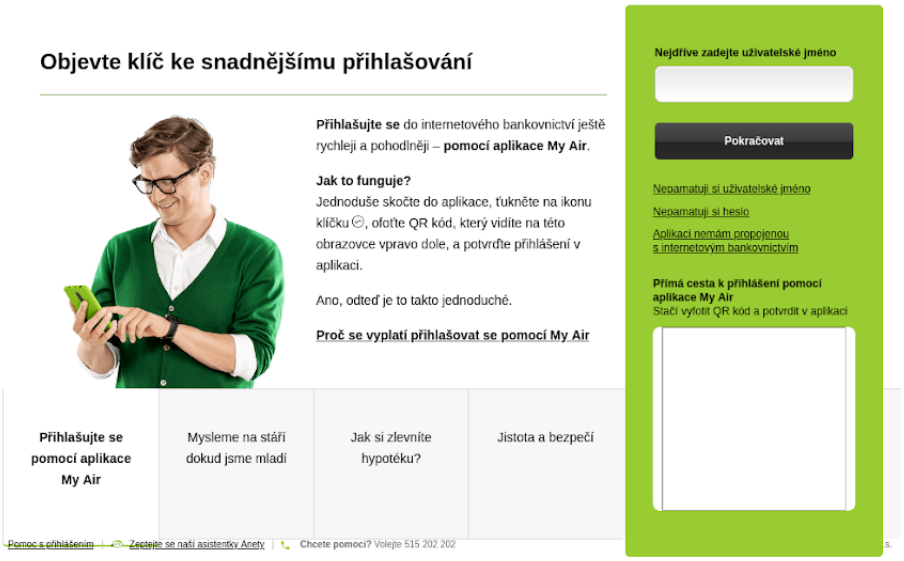
	HTTrack	GoPhish	SET
Bankovnictví 1	~	×	~
Bankovnictví 2	×	×	×
Bankovnictví 3	×	×	×
Bankovnictví 4	×	×	×
Bankovnictví 5	×	×	×
IS VUT	~	×	~

Obrázky uvedené níže demonstrují funkčnosti zmíněných nástrojů. Nejedná se o ukázku všech testovaných stránek, ale spíše o doplnění tabulky o vizuální představu.

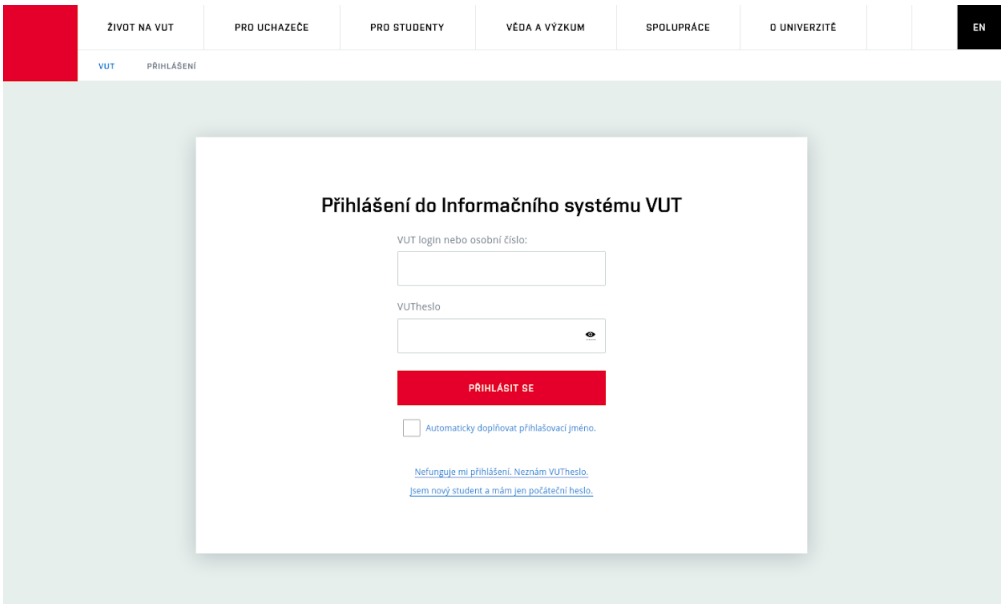
Příkladem téměř úspěšného pokusu je naklonovaná přihlašovací stránka do internetového bankovnictví nástrojem HTTrack. Jediným nedostatkem je zde chybějící QR kód v místě bílého rámečku a mírně posunutý text ve spodní části, jak je vidět na obrázku 1.6.

Druhým příkladem je přihlašovací stránka do VUT IS (informační systém). V případě klonování nástrojem Social Engineering Toolkit, viz obrázek 1.7, je na první pohled zřejmé chybějící logo v levém horním rohu, dále pak chybí dvě ikony vedle přepínače jazyka stránky v pravém horním rohu, otazníky s nápovědou vedle oken formuláře a puntíkované pozadí stránky. Stejný výsledek vrátil i nástroj HTTrack.

A nakonec jako ukázku neúspěšně provedeného klonu lze uvést přihlašovací stránku do dalšího internetového bankovnictví na obrázku 1.8, kdy nástroj GoPhish vrátil chybovou stránku serveru banky.



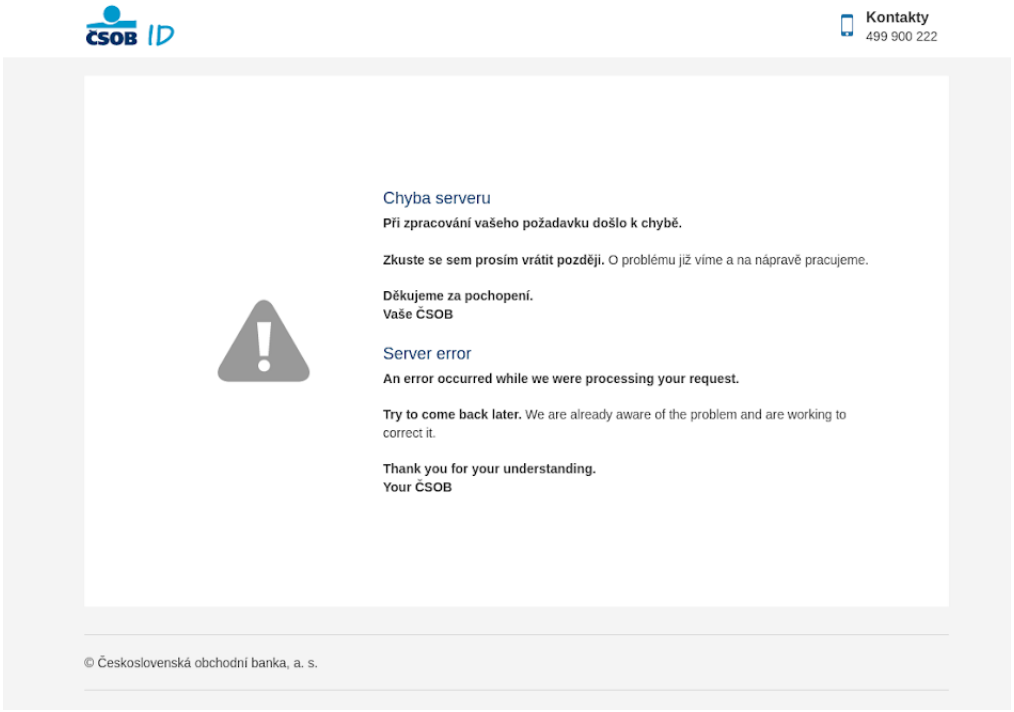
Obr. 1.6: Stránka naklonovaná nástrojem HTTrack - částečně úspěšně.



Obr. 1.7: Stránka naklonovaná nástrojem SET - částečně úspěšně.

Jak bylo zmíněno v textu výše, podrobnějšímu testování byl podroben nástroj GoPhish. Hlavním cílem zkoumání bylo odhalení webových technologií nebo jiných příčin, které by mohly mít na svědomí nekorektní zobrazení naklonovaných stránek. Testovaným vzorkem byly náhodně zvolené stránky napříč různými použitými programovacími jazyky a webovými servery. Jednalo se celkem o patnáct stránek, přičemž o úspěšném pokusu lze hovořit jen u dvou. Tabulka 1.3 zobrazuje u

každé stránky zjištěné technologie v kategoriích programovací jazyk, JavaScript framework, web framework a JavaScript knihovny. Značka v položce klon pak ukazuje, zda se podařilo stránku naklonovat či nikoliv: fajfka znamená úspěšný pokus, tylda částečně úspěšný, kdy chyby nejsou tak markantní, křížek chybějící prvky, rozbitý layout a podobně jako v předchozí tabulce, a dva křížky značí odepřený přístup. Jak je vidět z obsahu všech tří tabulek, nástroje mají své limitace, a proto je zde prostor pro vytvoření vlastní aplikace, která by tyto nedostatky neměla.



Obr. 1.8: Stránka naklonovaná nástrojem GoPhish - neúspěšně.

Tab. 1.3: Webové technologie testovaných stránek.

	Klon	Programovací jazyk	JavaScript framework	Web framework	JavaScript knihovny
Bankovnictví	×	Java	×	Apache Wicket	jQuery jQueryUI
Bankovnictví	×	Java TypeScript	Angular	×	jQuery Bootstrap GSAP ASP.NET Ajax Core-js
Bankovnictví	×	Java	RequireJS React	×	Axios core-js Dropzone
Bankovnictví	×	Java	React	×	core-js Lodash jQuery Bootstrap Popper YUI Library
Bankovnictví	×	JavaScript TypeScript	Angular	×	×
Cloudové úložiště	×	Java ASP.NET JavaScript	×	×	jQuery Lodash Boomerang Core-js Moment.js
Emailová služba	×	JavaScript	Knockout.js	×	×
E-shop	×	JavaScript Node.js Flash	×	Marko	web-vitals jQuery
E-shop	×	JavaScript PHP Ruby	×	×	jQuery

E-shop	××	Node.js Javascript ASP.NET	Knockout.js	Marko ASP.NET	Choices jQuery Modernizr
Sociální síť	✓	AJAX PHP	×	×	×
Fórum	××	GraphQL TypeScript Node.js	Next.js React	Next.js	Lodash Core-js Apollo jQuery
Streamovací služba	✓	Node.js	React Svelte	Next.js	×
Školní IS	×	JavaScript PHP	×	×	Query jQuery Migrate Modernizr Polyfill FancyBox
Školní IS	~	JavaScript	×	×	Boomerang Core-js

Na základě provedeného testování klonování stránek na uvedeném vzorku bylo zjištěno, že problematické nebo neúplné zobrazení naklonované stránky je v jednotlivých případech způsobeno užitím JavaScriptových knihoven a frameworků. Fonty a ikony pak chybějí v případech, kdy jsou nalinkovány v externích CSS (Cascading Style Sheets – kaskádové styly) souborech a nikoliv v hlavním HTML (HyperText Markup Language) stromu.

JavaScript je jeden z nejpoužívanějších programovacích jazyků pro tvorbu webových stránek. Je využíván hlavně při front-end vývoji vedle HTML a CSS a zodpovídá za dynamické a interaktivní chování stránky – animace, našeptávač vyhledávání a podobně [32]. Lze v něm však psát i back-end část webu na straně serveru, která zodpovídá za odpovědi na uživatelské dotazy. Další využití nachází při vývoji mobilních aplikací, 2D a 3D prohlížečových her, virtuální realitě a dokonce v rámci umělé inteligence, kdy lze pomocí JavaScriptových knihoven vytvářet modely pro strojové učení [33]. Kód je psán do samostatného .js souboru, obdobně jako je tomu u kaskádových stylů, nebo jsou tagy skriptu s atributy umístěny přímo na webové stránce. Pro uživatele to nepředstavuje žádný rozdíl, prohlížeče již v sobě mají zabudované nástroje pro renderování JavaScriptového obsahu a není tedy třeba instalovat kompilátor nebo další programy. Na druhou stranu je možné v nastavení prohlížeče

JavaScript zakázat [32]. Důvodů může být hned několik. Obava z možných zranitelností prohlížeče, které by bylo možné exploitovat právě skrze JavaScript (cross-site scripting, cross-site request forgery, zranitelnosti v knihovnách [34]), zabránění načítání reklam a nebo snížení zátěže na paměť a procesor. Obecně to ale není potřeba a ani to není doporučováno [35]. JavaScript je poměrně bezpečný a vzhledem k tomu, že většina dnešních webů jej používá, jeho zablokováním většinou uživatel dosáhne pouze zhoršeného zobrazení a čitelnosti, nefunkčnosti prvků jako jsou formuláře, menu, slideshow nebo v nejhorším případě celé stránky. Jako Vanilla JavaScript se označuje čistý JavaScript bez použití dodatečných knihoven a frameworků, které značně usnadňují práci s kódem a šetří čas. S Vanilla verzí se tedy uživatel setká jen málokdy. Jednou z nejčastěji používaných knihoven, která se také vyskytuje ve vzorku testovaných stránek, je knihovna jQuery. Obsahuje tzv. snippety – předpřipravené bloky kódu s různými funkcionalitami, které stačí vložit na požadované místo v .js souboru [32]. Druhým způsobem ulehčení práce při vývoji webových stránek jsou JavaScriptové frameworky.

1.4.3 Právní pohled na kopírování stránek

Kopírování webových stránek je problematické hned z několika hledisek dle toho, co je obsahem stránek a co je kopírováno. Předmětem ochrany zde jsou texty, fotografie, videa i samotný design stránky včetně loga a podobných prvků, přičemž stránka může být chráněná jako celek nebo jen její části. Aby stránka spadala pod ochranu autorským právem jako celek, je nutné, aby naplňovala znaky jedinečného autorského díla, tedy byla výsledkem tvůrčí činnosti autora, byla jedinečná a byla vyjádřena v objektivně vnímatelné podobě. Pokud tyto podmínky nesplňuje, nelze ji považovat za takové dílo a mohou být chráněny jen některé prvky, například vlastní fotografie, pokud jsou původní. Formuláře a uspořádání stránky jsou naopak považovány za standardní součásti a nejsou chráněny. Autorský zákon přitom nerozlišuje, zda se jedná o umělecké vyjádření nebo nikoliv. Dojem zde není zákonný předpoklad [36].

Webové stránky mohou být dále chráněny podle Obchodního zákoníku dle práva nekalé soutěže. Zde se může jednat o úmysl zkopírováním stránky přivodit újmu konkurenci, snahu zmást spotřebitele nebo parazitovat na dobré pověsti konkurenční stránky. Ochrany nabývá také název podnikatele dle § 423 Občanského zákoníku a nakonec je zde relevantní i právo k databázi dle § 88 a následujících Autorského zákona [36].

V případě vzniku protiprávního stavu může poškozená strana po žalovaném požadovat určení svého autorství, zdržení se protiprávního jednání, odstranění závadného stavu a jeho následků, náhradu škody včetně ušlého zisku, vydání bezdůvodného

obohacení, poskytnutí přiměřeného zadostiučinění, ať už omluvou nebo peněžní náhradou a uveřejnění rozsudku [36] [37].

Existuje několik výjimek ze zákonné ochrany, kdy není třeba získávat souhlas oprávněné osoby pro nakládání s dílem. V kontextu kopírování stránek se jedná o volné užití díla dle § 30 Autorského zákona, kde je v odstavci 1 uvedeno: „Za užití díla podle tohoto zákona se nepovažuje užití pro osobní potřebu fyzické osoby, jehož účelem není dosažení přímého nebo nepřímého hospodářského nebo obchodního prospěchu, nestanoví-li tento zákon jinak.“ V dalších odstavcích je pak rozvedeno, že vytvoření rozmnoženiny pro osobní potřebu není zásahem do autorského práva. Musí však být zřetelně označeno, že se jedná o rozmnoženinu a nesmí být použita za jiným než osobním účelem, tedy je vyloučeno zveřejnění [36].

Je zřejmé, že v rámci bezpečnostního testování se omezení z Autorského nebo jiného zákona neuplatní, jelikož vše probíhá s vědomím a souhlasem autora stránky či oprávněné osoby a celý proces je smluvně podložen.

Problematika právní ochrany internetových stránek je poměrně obsáhlá, proto zde nebude zacházeno do dalších detailů. Případné zájemce je možné odkázat na publikaci Internetové právo [36], ze které bylo v této kapitole vycházeno.

2 Praktická část

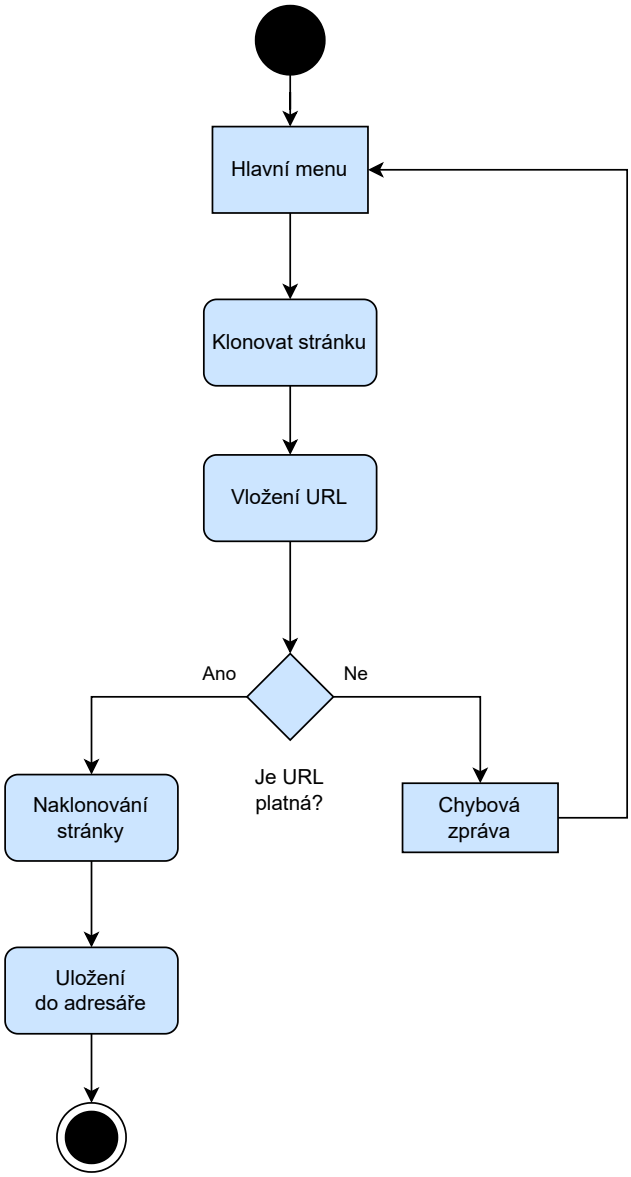
Tato kapitola se věnuje návrhu, implementaci a testování vlastní aplikace pro klonování webových stránek a porovnání jejích výsledků s nástrojem GoPhish. Aplikace bude určena pro použití ve firemním prostředí pro účely bezpečnostního testování odolnosti zaměstnanců vůči phishingu, a to pouze oprávněnými osobami.

2.1 Návrh aplikace

Celá aplikace je z uživatelského pohledu jednoduchá na ovládání, intuitivní a nevyžaduje instalaci. Po spuštění je uživatel vyzván k vložení URL adresy originální stránky, kterou si přeje naklonovat. Aplikace zkontroluje, zda je zadaná adresa platná (neobsahuje-li například překlep ve formátu ve formě mezery). V kladném případě vytvoří klon stránky a uloží jej do adresáře vytvořených kopií na disku uživatele. V opačné situaci se uživateli zobrazí chybová zpráva upozorňující na chybnou adresu a nasměruje jej k opětovnému pokusu o zadání vstupu. Tato struktura aplikace je následně zobrazena v UML diagramu na obrázku 2.1.

Pro implementaci aplikace byl zvolen programovací jazyk Python verze 3, a to z důvodu přívětivé syntaxe, jednoduché údržby kódu a jeho podpory širokého množství knihoven. Hlavní knihovnou, na které program staví, je `Beautiful Soup 4`. Umožňuje vytáhnout kompletní HTML strom stránky a vrátit jej ve formě objektu `BeautifulSoup` – prohledávatelné datové struktury. Co se týče parseru, `Beautiful Soup` podporuje HTML parser, který je součástí standardní Python knihovny, žádná další instalace krom samotné `Beautiful Soup` knihovny tedy není potřeba [38]. Další nezbytnou součástí je knihovna `cssutils: CSS Cascading Style Sheets library for Python` sloužící pro parsování a vytváření CSS souborů. Konkrétní příklady použití jsou uvedeny v kapitole 2.2.4, dokumentace je pak dostupná na stránkách Python Package Index [39].

Většina současných internetových stránek je dynamická nebo kombinací statických a dynamických stránek a trvá delší dobu, než se plně načte. Děje se tak z důvodu načítání skriptů na straně serveru (server-side scripting) a na straně klienta (client-side scripting) [40]. Proto nelze stahovat HTML strom okamžitě. Vhodným řešením je využití `WebDriveru`, nástroje pro automatizované testování webových aplikací [41], jehož architektura je zobrazena na diagramu na obrázku 2.2. `WebDriver` interaguje přímo s prohlížečem a oproti například nástroji `Selenium RC` podporuje spuštění v headless formátu [42]. Díky tomuto nástroji bude možné pozdržet stažení stromu do doby, kdy se stránka kompletně načte včetně veškerých aplikací, které na ní běží. Nákres architektury celé aplikace a jejích součástí je pak zobrazen v diagramu na obrázku 2.3.

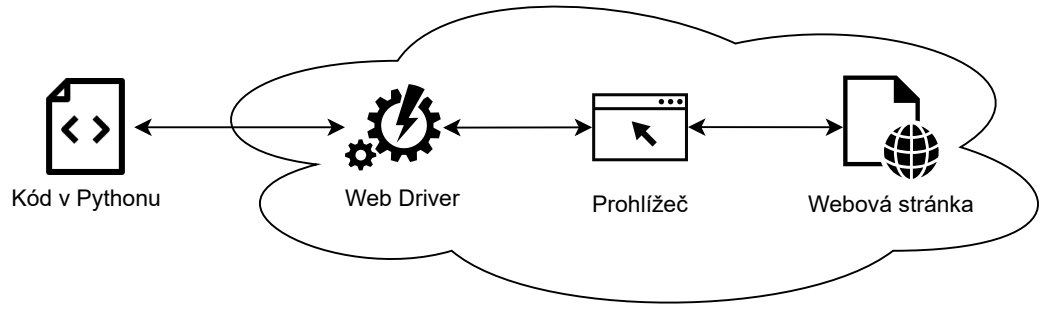


Obr. 2.1: UML diagram aplikace.

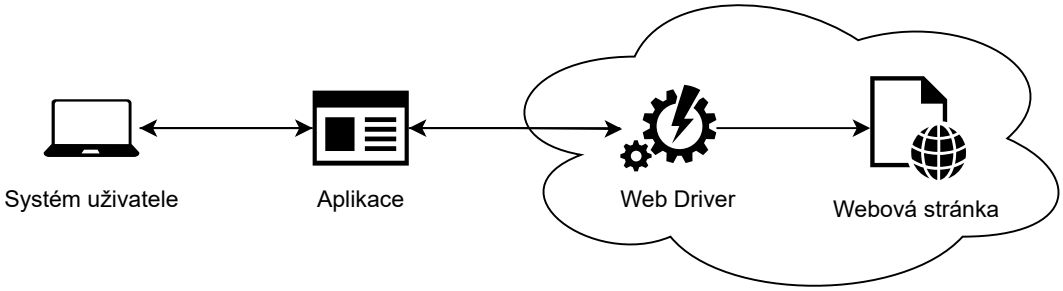
2.2 Implementace aplikace

Před zavoláním hlavní funkce je provedeno spuštění `WebDriveru` kontrolujícího webový prohlížeč Chrome v headless formátu. Aplikace používá `Chrome Driver Manager`, díky kterému je zajištěno, že je vždy po spuštění nainstalována nejnovější verze `Driveru`. S každou aktualizací prohlížeče totiž vychází i nový ovladač a odlišné verze spolu nejsou kompatibilní.

Poté je spuštěna hlavní funkce, kde je pomocí `WebDriveru` a jeho metody `driver.get()` provedena navigace na stránku, jejíž URL poskytl uživatel na vstupu, a kterou si přeje naklonovat. Po kompletním načtení všech dynamických prvků



Obr. 2.2: Diagram architektury WebDriveru.



Obr. 2.3: Diagram architektury aplikace.

stránky jsou postupně volány funkce pro stažení jednotlivých komponent, tedy HTML kódu, CSS stylů, JavaScriptových souborů, obrázků, fontů a ikon. V následujících podkapitolách jsou tyto funkce detailněji popsány, případně vysvětleny na výpisech ze zdrojového kódu aplikace a na příkladech je demonstrováno jejich použití a výstupy. Postup volání funkcí a jejich vzájemnou interakci pak zobrazuje sekvenční diagram na obrázku 2.5.

2.2.1 Nalezení CSS a JavaScriptových souborů

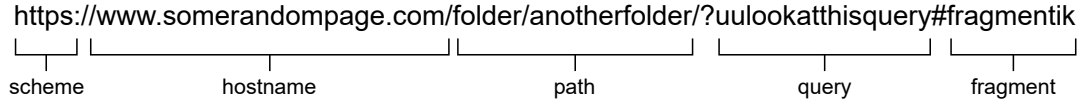
Jako první je volána funkce `find_styles_images(path_url)` se vstupním parametrem vstupu od uživatele. Funkce postupně provádí vyhledávání nalinkovaných JavaScriptových souborů, CSS souborů a obrázků v HTML kódu stránky.

JavaScriptové soubory jsou v HTML kódu uvedeny tagem `<script>`, kde je pomocí atributu `src` odkázáno na externí JS soubor. Aby bylo možné stáhnout obsah těchto souborů, je třeba nejdříve najít veškeré výskyty tohoto tagu a získat hodnoty uložené v jeho atributu. K tomu je využito knihovny `Beautiful Soup`, jíž je

předána načtená stránka, a která následně vrací prohledávatelnou datovou strukturu uloženou v objektu `soup`, v níž je prováděno vyhledávání potřebných HTML prvků.

Po extrahování atributu s adresou z tagu `script` je získaný řetězec otestován, zda je validní a dále je zkoumán jeho formát. Pro formát adresy v HTML dokumentu neexistuje jednotná politika, některé stránky uvádějí absolutní URL adresu souboru, tedy celou adresu včetně `scheme` a `hostname`, některé jen její relativní část. Celá URL syntaxe je pro představu popsána na obrázku 2.4. Z důvodu této nejednotnosti je nejdříve potřeba zjistit, jakým způsobem je vždy konkrétní adresa zadána a případně ji vhodně upravit, než bude uložena do listu `js_files`, se kterým se pak bude dále pracovat. Pokud je adresa zadána absolutně, je rovnou uložena beze změn do listu. Pokud je uvedena relativně, je pomocí série podmínek zjišťováno, na jakou úroveň adresářové struktury se odkazuje a podle toho je sestavena kompletní adresa k uložení do listu. Možnosti a odpovídající řetězce jsou následující.

Cesta je uvedena znakem `/`, například `/styles/utilities.js` nebo jen `/utilities.js`. Tímto způsobem je odkazováno na hlavní složku stránky (označovanou jako `root`). Výsledná absolutní URL je pak vytvořena vložením tohoto řetězce přímo za `hostname` část adresy vložené uživatelem, přičemž pokud uživatelský vstup obsahoval `path` část nebo `query`, jsou tyto odstraněny. Druhým možným případem je adresa začínající znaky `./` a pokud zůstaneme u stejného příkladu, pak taková adresa může být `./utilites.js`. V tomto případě se zůstává ve stejné složce, jako je soubor, ve kterém je tento odkaz uvedený. Adresa uložená do listu pak obsahuje adresu vloženou uživatelem až po `path` část, kterou následuje cesta uvedená za `./`. Třetím případem je situace, kdy cesta odkazuje o jednu nebo dvě složky v hierarchii výše (s posunem o více složek se zpravidla nesetkáme). Taková adresa začíná `../`, respektive `../../`. Výsledná adresa je pak vytvořena odstraněním daného počtu složek, a samozřejmě také `query`, z adresy vložené uživatelem a dosazením adresy souboru na toto místo. Speciálním případem je absolutní adresa začínající dvěma lomítky, tedy takto: `//website.net/folder/utilities.js`. Zde stačí na začátek přidat označení přenosového protokolu HTTPS (HyperText Transfer Protocol Secure), případně HTTP (HyperText Transfer Protocol) dle toho, který klonovaná stránka používá. Nakonec, pokud cesta k souboru nezačíná ani jednou z vyjmenovaných



Obr. 2.4: Syntaxe URL.

kombinací znaků, jedná se o ekvivalentní zápis k `./` a dle toho je s cestou naloženo. Vlastní stažení těchto souborů bude provedeno později.

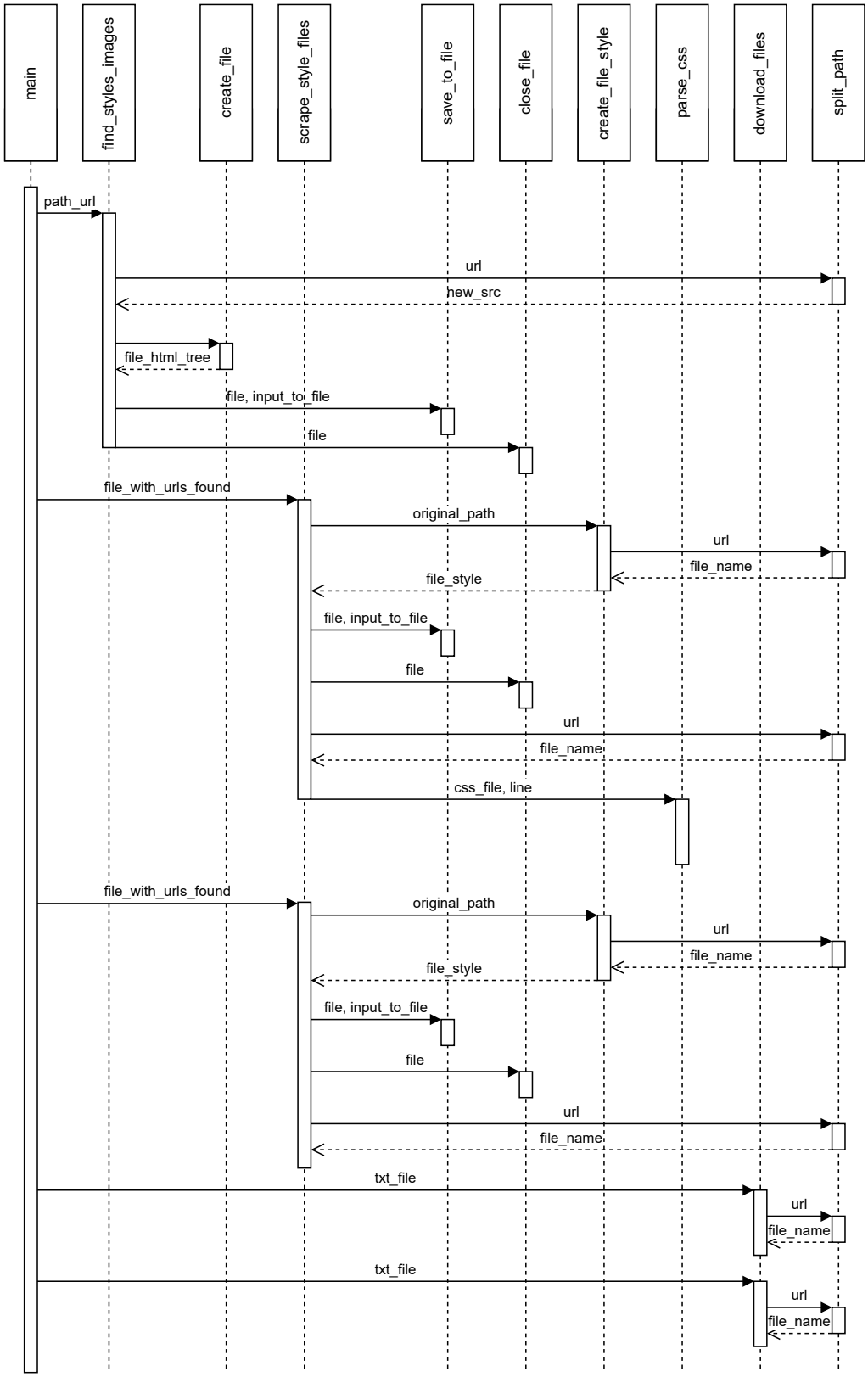
Výše popsané operace nad URL adresou jsou realizovány za pomoci modulu `urllib.parse` [43] obsaženého ve standardní knihovně Pythonu. Tento modul obsahuje funkce pro parsování URL a převod speciálních znaků do formátu vhodného pro použití v URL adresách. Pro potřeby aplikace byla konkrétně použita funkce `urllib.parse.urlparse(urlstring, scheme=", allow_fragments=True)`, na jejíž vstup byla předána URL adresa, kterou zadal uživatel hned po spuštění programu. Funkce vrací jmenný tuple s šesti položkami: `scheme`, `netloc` (jiné označení pro `hostname`), `path`, `parameters`, `query`, `fragment`. Jednotlivé komponenty lze následně libovolně skládat dohromady pro vytvoření požadované adresy. Operace odstraňování složek z adresy stránky je prováděna metodou `os.path.split(path)`, která spadá pod `OS modul` a je určena pro manipulace s cestami. Metoda vrací pár `(head, tail)`, kdy pod `tail` je uložena poslední část vložené cesty a `head` obsahuje zbytek [44]. Pro lepší názornost následuje výpis kódu 2.1, kdy je představenými přístupy odstraněna poslední složka z adresy vložené uživatelem.

```
1 parse_url = urlparse(path_url)
2 just_path = parse_url.path
3 head_tail = os.path.split(just_path)
4 remove_one_folder =  head_tail[0]
```

Výpis 2.1: Odebrání poslední složky z vložené cesty.

Získání odkazů na CSS styly je provedeno obdobně s několika rozdíly. Odkazy na externí CSS jsou v HTML kódu uvedeny tagem `<link>` s atributem `href`, přičemž je u získané cesty navíc kontrolováno, zda obsahuje řetězec `css`. Odkazy na jiné stránky jsou definovány stejným způsobem a bez této kontroly by byly chybně uloženy mezi CSS. Takto ošetřené cesty pak prochází stejným procesem podmínek a úprav jako výše popsané cesty k JavaScriptovým souborům a jsou uloženy k dalšímu zpracování do listu `cs_files`.

Pro každou cestu uloženou buď do listu `js_files` nebo `cs_files` je původní cesta v HTML kódu v `soup` za pomoci modulu `parse` změněna, aby odpovídala čistě názvu daného souboru, včetně odstranění `query` (nejčastěji uvádějící verzi) a shodovala se tak se jménem, pod kterým bude v následující krocích daný soubor uložen. Díky tomu není potřeba po skončení běhu programu manuálně cesty upravovat a zároveň budou platné i při přesunu souborů do jiného adresáře.



Obr. 2.5: Diagram volání funkcí.

2.2.2 Nalezení obrázků a získání HTML kódu

Hledání odkazů na obrázky a ikony je také realizováno ve funkci `find_styles_images(path_url)`. Obrázky mohou být v HTML kódu vloženy dvěma způsoby. Prvním je umístění v tagu `<img>` s atributuem `src`. Pokud není URL vyhodnocena jako neplatná, což se stává v případech, kdy obsahuje mezeru nebo když dokonce v tagu chybí `src`, je předána do stejné struktury podmínek k vyhodnocení umístění v hierarchii adresáře stránky a upravená adresa je pak uložena do listu `img_files`.

Druhým přístupem pro vložení obrázků je stejně jako u CSS souborů vložení do tagu `<link>` s atributem `href`. Aby opět bylo eliminováno zaměnění s odkazem na nějakou externí stránku, stejně jako v případě CSS, je u hodnoty atributu kontrolováno, zda na svém konci obsahuje subřetězec `.png`, `.jpg` nebo `.ico`. Pokud je tato podmínka splněna, proces je stejný, jako je popsáno v předchozím odstavci.

I zde je pro každou uloženou položku upravena originální cesta na relativní odkazující k aktuálnímu adresáři uživatele.

Po skončení běhu těchto tří cyklů pro nalezení JavaScriptových souborů, CSS souborů a obrázků je aktualizovaný obsah `soup` objektu uložen pomocí metody `soup.prettify()` do samostatného souboru `index.html`. Program vytváří také soubor `index_original.html`, který obsahuje původní HTML kód stránky před úpravami, aby bylo možné v případě potřeby jednoduše zjistit provedené změny.

Následně jsou vytvořeny textové soubory pod názvy `javascript_files.txt`, `css_files.txt` a `images.txt`, do kterých jsou uloženy obsahy odpovídajících listů `js_files`, `cs_files` a `img_files` k dalšímu zpracování, které je popsáno v následující kapitole 2.2.3.

2.2.3 Stažení CSS a JavaScriptových souborů

Poté, co jsou odkazy uloženy do textových souborů, je dvakrát za sebou volána funkce `scrape_style_files(file_with_urls_found)`. Jejím účelem je projít tyto odkazy, získat jejich obsah a uložit ho do složky uživatele do souboru pod názvem, jakým je označen v kopírované stránce.

Nejdříve je funkci předán soubor `css_files.txt`. For cyklem je nad každým odkazem spuštěna metoda `driver.get()`, kdy `WebDriver` provede navigaci na stránku, přičemž je zároveň otestováno, zda je adresa validní, a to zachytáváním výjimky `WebDriverException`. Pokud je adresa ze syntaktického hlediska platná, předá `WebDriver` zdrojový kód načtené stránky knihovně `Beautiful Soup`, která jej převede na prohledávatelný `soup` objekt. Jelikož v tomto případě je struktura stránky jednoduchá, stačí vyhledat tag `<pre>`, ve kterém je umístěn celý CSS kód. Pokud obsah tagu není prázdný, je zavolána funkce `create_file_style(original_path)`, přičemž parametr odpovídá danému odkazu. Skrz tuto funkci je dále volána funkce

`split_path(url)`, kde je z cesty modulem `urlparse` oddělen název samotného souboru s koncovkou. Název je vrácen do funkce `create_file_style`, kde je vytvořen prázdný soubor pod tímto jménem, a ten je nakonec vrácen do původní funkce `scrape_style_files`. Zde je do souboru uložen získaný kód originálního CSS souboru.

Při druhém volání této funkce je předáván textový soubor `javascript_files.txt`. Průběh je obdobný, jako bylo popsáno v předchozím odstavci. Jediným rozdílem je to, že nyní jsou stahovány JavaScriptové soubory, princip nicméně zůstává stejný.

2.2.4 Nalezení fontů a ikon

Fonty používané webovou stránkou mohou být v CSS souborech definovány třemi různými způsoby a jen v případě jednoho z nich je třeba je pro naklonování stránky stahovat.

Prvním způsobem je užití předinstalovaných systémových fontů. Jedná se o přímý přístup, kdy je za vlastností `<font-family>` uvedeno konkrétní `<family-name>` nebo obecnější `<generic-name>`. Příkladem konkrétního názvu písma může být zápis `Calibri`, obecně pak `serif`. Lze definovat i více `<font-family>`, prohlížeč pak při sestavování stránky vybírá první jím podporovaný font [45].

Druhou možnost představuje self-hosted font, tedy font uložený v souboru v adresáři stránky. V CSS je vkládán do pravidla `@font-face` a jeho vlastnosti `src`. Opět je možné nalinkovat více fontů, zde ale z jiného důvodu, než tomu bylo v předchozím případě. Fonty mohou mít více různých formátů, například WOFF, EOT, TTF a pokud by byl použit jen jeden, je pravděpodobné, že jej některý prohlížeč nebude podporovat [46].

Třetí způsob není tak běžný jako předchozí popsané, tedy alespoň tak lze usuzovat dle provedeného testování, kterým se později zabývá kapitola 2.3. Jde o využití nějaké třetí strany, skrz níž je importována potřebná konfigurace CSS souboru a zvolený font. Dochází tak však k prodloužení doby načítání stránky z důvodu dotazování na server služby, přičemž se stává, že i tento server se dotazuje dalšího serveru [46].

Ke stahování fontů aplikací je tedy přistoupeno jen v případě self-hosted přístupu. K tomu je třeba nejdříve najít odkazy na soubory s obsaženými fonty, což je realizováno funkcí `parse_css(css_file, line)`. V předchozí kapitole 2.2.3 byl popsán průběh stahování CSS souborů pomocí `scrape_style_files(file_with_urls_found)`. V této funkci je pro každý vytvořený CSS soubor voláno právě `parse_css`, kterému je předán název tohoto souboru a jeho celá URL adresa. Aby bylo možné programově prohledávat CSS kód, musí být proveden jeho parsing, tedy převod do datové struktury. Toho je dosaženo funkcí `cssutils.parseFile(filename,`

`encoding=None)` z knihovny `cssutils`, které je předán daný soubor. Struktura je poté postoupena další funkci z této knihovny, `cssutils.getUrls(sheet)`. Ta provede vyhledání výskytů parametru `url` a obsažené hodnoty uloží do listu. Následně jsou získané řetězce testovány stejným postupem, jako bylo popsáno u adres v kapitole 2.2.1, tedy je ověřena jejich platnost a poté je sestavena jejich absolutní adresa, která je uložena do souboru `found_url_css.txt`. Totožným způsobem jsou získány odkazy na ikony. Vlastní stažení souborů je pak popsáno v následující kapitole 2.2.5.

K nahrazení původních hodnot `url` v tomto CSS souboru novými relativními hodnotami odkazujícími na uživatelský adresář je využito funkce `cssutils.replaceUrls(sheet, replacer, ignoreImportRules=False)`. `Sheet` je datová struktura upravovaného CSS souboru a `replacer` je libovolná funkce s jedním parametrem, jíž je předáván argument `url`, a která vrací řetězec, jenž má nahrazovat původní `url` v dokumentu. V kontextu vyvíjené aplikace je touto předanou funkcí `split_path(url)`, která navrací jen název souboru. Po substituci všech výskytů je původní soubor otevřen v režimu binárního zápisu a je do něj uložena aktualizovaná struktura.

2.2.5 Stažení obrázků, fontů a ikon

Poslední krok představuje stažení všech obrázků, fontů a ikon. Za tímto účelem je dvakrát volána funkce `download_files(txt_file)`, a to pro soubory `images.txt` a `found_url_css.txt`. Zde jsou jednotlivé odkazy předány funkci `urllib.request.urlretrieve(url, filename=None, reporthook=None, data=None)` z modulu `urllib.request`. Tímto jsou soubory staženy do lokálního adresáře uživatele.

Během ladění kódu bylo zjištěno, že v případě některých stránek dochází ke stažení prázdných souborů obrázků, přestože při manuálním přístupu jsou dostupné. Jako vhodné řešení problému se ukázalo nastavení hodnoty `User-Agent` v hlavičce odesílaného GET požadavku na server stránky. V následujícím výpisu kódu 2.2 je zobrazeno toto nastavení a následné poslání dotazu.

```
1 opener = urllib.request.build_opener()
2 opener.addheaders = [('User-agent', 'Mozilla/5.0 (Windows NT 10.0;
     Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome
     /111.0.0.0 Safari/537.36')]
3 urllib.request.install_opener(opener)
4 urllib.request.urlretrieve(line, file_name)
```

Výpis 2.2: Nastavení hlavičky dotazu.

2.3 Testování aplikace

Tato kapitola se zabývá testováním vyvíjené aplikace na vzorku více než třiceti stránek, které byly rozděleny do kategorií:

- finanční sektor,
- státní sektor,
- školství,
- emailové služby,
- zájmy/fóra.

Ve všech případech jde o stránky obsahující přihlašovací formuláře, nikoliv o hlavní stránky, ze kterých se uživatel k přihlášení proklikává. Kategorie finanční sektor zahrnuje stránky internetového bankovnictví u vybraných bank působících v České a Slovenské republice, státní sektor obsahuje některé přístupy pro přihlášení do služeb pro komunikaci s úřady u nás a v zahraničí. Kategorii školství tvoří informační systémy vysokých škol a portály online kurzů, a v sekci zájmy/fóra se nachází diskuzní fóra, weby zaměřené na pokládání a zodpovídání dotazů v rámci nějaké komunity, nebo obecně platformy pro vytváření a sdílení obsahu.

Tento vzorek stránek není vyčerpávající, co se týče možných webových technologií, užitých knihoven nebo frameworků, nicméně poskytuje dobrý základ pro představu o funkcionalitě vyvíjené aplikace. Každá stránka byla zároveň klonována i nástrojem GoPhish a dosažené výsledky byly porovnány na základě přiděleného bodového hodnocení na stupnici 0 až 10 bodů. Kritéria hodnocení pro jednotlivé body jsou popsána v následující tabulce 2.1.

Tabulka 2.2 pak zobrazuje ohodnocený seznam klonovaných stránek a jejich vlastností, jako je web server, na kterém stránka běží, používané JavaScriptové knihovny (případně frameworky), webové frameworky, nástroje pro detekování botů nebo nastavený reverse proxy server. Možné příčiny, proč některé stránky nelze vyvíjenou aplikací naklonovat kompletně, jsou více přiblíženy v dalších odstavcích. Tam, kde je to schůdné, je navíc navrženo možné řešení.

První možnou příčinou je užití nástroje nebo softwaru pro detekci a blokování botů, které se snaží ke stránce přistupovat, ať už je to za účelem scrapingu obsahu, DoS (Denial-of-Service) nebo DDoS (Distributed Denial-of-Service) útoku, nebo útoku hrubou silou s cílem získat přístup k účtu [47]. Techniky detekce lze rozdělit do dvou hlavních skupin. Tou první je behaviorální, kam spadá počet uskutečněných dotazů na stránku nebo například pohyby myší a stisky klávesnice, z čehož je pak vyhodnoceno, zda to odpovídá lidskému chování. Druhá technika je založená na otisku zařízení a prohlížeče, který je tvořen kombinací operačního systému, použitého prohlížeče, jeho verze, obsahu HTTP hlavičky a informacemi, které nástroj obdrží jako odpovědi na výzvy zaslané prohlížeči. V těchto výzvách může zkoušet přítomnost

nebo chování některých konkrétních atributů, které mu v kombinaci se zbytkem otisku pomohou určit, zda se jedná o bota či nikoliv. V kontextu aplikace vyvíjené v rámci této práce se může jednat například o atributy `navigator.webdriver` a `document.__selenium_unwrapped`, jenž značí, že prohlížeč je spouštěn v headless formátu a je kontrolován pomocí softwaru pro automatizaci Selenium [48]. Tato možnost se v testovaném vzorku týká stránky Finanční sektor 2, u níž byla zjištěna přítomnost detekčního nástroje skrze doplněk prohlížeče, a Emailové služby 4, která po dokončeném pokusu o klonování při přístupu přes odkaz ve vyhledávači zobrazila formulář CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) s informací, že dotaz vypadá jako automatizovaný. Z tohoto důvodu lze předpokládat minimálně použití nějakého detekčního nástroje na základě behaviorálních charakteristik v podobě počtu potazů v rámci jedné IP adresy. Posouzení otisku zařízení není prováděno v prohlížeči, ale na straně serveru a stránky na sebe většinou neprozrazují, jestli používají některý takový nástroj [48]. Není proto možné s jistotou říct, že ostatní stránky jej implementovaný nemají. Je jen možné, že vyvíjenou aplikaci nevyhodnotily jako bota.

Druhou možnou příčinou jsou JavaScriptové frameworky a knihovny. Framework vytváří strukturu pro kód webové stránky a poskytuje před-připravené šablony a nástroje [49]. JavaScriptové knihovny naopak obsahují hotové funkce, třídy a rozhraní, které je možné volat dle libosti prostřednictvím API, a to jak z prostředí frameworku, tak bez něj. Rozdíl mezi těmito situacemi je v řízení kontroly. Zatím co bez použití frameworku ovládá volání všech funkcí a komponent knihoven vývojář, v druhém případě řídí veškeré procesy framework, a to včetně volání kódu napsaného vývojářem [50]. Zde se objevuje jádro problému, a tím jsou JavaScriptové moduly. Jedná se o samostatné soubory obsahující skripty, které jsou příliš komplexní na to, aby byly vloženy přímo na místo, odkud mají být spouštěny. Místo toho je k nim přistupováno klíčovými slovy `import` a `export`, přičemž v HTML kódu musí být specifikováno, že se jedná o modul atributem `<script type="module">`. Jednotlivé moduly se mohou také spouštět vzájemně mezi sebou. Pokud je webová stránka s moduly zobrazena v prohlížeči, jsou veškeré moduly nejdříve načteny a poté jsou spuštěny skripty. Pokud je však stejná stránka zobrazena lokálně, import a export modulů neproběhne, protože to je možné jen skrze HTTP/HTTPS protokol, a tudíž nedojde ani k provedení skriptů. Z tohoto důvodu dochází v naklonovaných stránkách k nekorektnímu zobrazení nebo absenci některých prvků, ačkoliv jsou příslušné JavaScriptové soubory staženy. V testovaném vzorku se jedná o většinu stránek s hodnocením 6 nebo 7 bodů, které mají některý z frameworků implementovaný. Jako možné řešení se zde jeví použití lokálního webového serveru.

Trochu odlišná situace nastává u modulů, které jsou získávány z externího zdroje, jako je tomu u stránky Emailové služby 4. V tomto případě odesílá prohlížeč po-

žadavek na externí server a v hlavičce dotazu uvádí `Origin`, tedy adresu stránky, která modul požaduje. Tento mechanismus se nazývá CORS (Cross-Origin Resource Sharing). Zdrojový server se poté na základě informací v hlavičce rozhodne, zda data poskytne a odešle zpět `Access-Control-Allow-Origin` nebo zda žádost odmítne [51]. To se děje v případech, kdy není `Origin` specifikován nebo je jeho hodnota `null` [52], jako je tomu u `WebDriveru`, který používá vyvíjená aplikace. U hodnoty `null` je ještě třeba podotknout, že některé stránky tento `Origin` povolují, přestože se jedná o bezpečnostní riziko [52]. Jako řešení by se zde nabízelo přenastavení této hodnoty, ale nejenom, že to `WebDriver` nebo Selenium neumožňuje, taková modifikace je zapovězena [53]. Obcházet toto pravidlo by bylo poměrně komplikované i bez používání nástrojů pro automatizaci a vzhledem k povaze vyvíjené aplikace, která si neklade za cíl narušovat bezpečností pravidla, nebudou takové možnosti dále rozváděny.

Další z možných příčin je WAF (Web Application Firewall – firewall webové aplikace). Jedná se o typ reverse proxy, která má za úkol na základě nastavených pravidel monitorovat a filtrovat provoz a chránit tak server HTTP aplikace. WAF je primárně určen k ochraně proti útokům, jako jsou XSS (Cross-site Scripting) nebo SQL injection [54], ale obvykle obsahuje i základní techniky pro detekci botů, například na základě otisku zařízení [55]. Některé aplikační firewally navíc umožňují nastavení odesílání JavaScriptových výzev prohlížeči k určení, zda se nejedná o bota [56], podobně jako dedikované nástroje pro detekci botů, které byly popsány v textu výše. Je tedy možné, že v případě stránek s nastavenou reverse proxy, což samozřejmě nemusí být jen ty, u kterých byla odhalena, má tato skutečnost vliv na proces klonování.

Poslední možnou příčinou jsou chybějící cookies. Jedná se o užitečná data vztahující se k relaci a svým způsobem z bezestavového HTTP protokolu činí stavový. Po obdržení dotazu na zobrazení stránky od prohlížeče odesílá aplikační server `Set-Cookie` HTTP hlavičku (nebo více hlaviček, dle toho, kolik cookies je třeba poslat) obsahující hodnoty `name` a `value`. Ty dohromady tvoří cookie, kterou si prohlížeč uloží pro použití v dalších dotazech na server. Každá cookie má nastavenou dobu platnosti a v atributu `SameSite` určeno, za jakých podmínek bude posílána při navigaci uživatelem [57]. Cookies s nastavenou hodnotou jako `Lax` budou odeslány, pokud ke stránce uživatel přistupuje skrze odkaz nebo obecně z jiné domény. Cookies s hodnotou `None` jsou odesílány i třetím stranám, zatímco hodnota `Strict` dovoluje cookies odeslat jen v rámci domény, která je v nich uvedena, přičemž pokud neobsahují navíc atribut `Secure`, jsou odmítnuty [58]. Některé stránky ke správnému zobrazení obsahu vyžadují v prohlížeči přítomnost cookies [59], které jsou ale prohlížeči předány serverem při zobrazení jiné stránky domény. Typicky se jedná o přihlašovací formuláře, ke kterým se uživatel proklikne z hlavní stránky webu.

Tab. 2.1: Stupnice hodnocení a popis kritérií.

Body	Kritéria
0	Neproběhne naklonování stránky (prázdná stránka, program zamrzne, stránka detekuje bota a odepře přístup, hlášení o cookies a zablokování).
1	Chybí celý formulář.
2	Stránka je rozsypaná, ale obsahuje formulář a nějakou kostru.
3	Na stránce je nepřehlédnutelná chybová hláška (případně obsahuje i další chyby).
4	Chybí výrazný prvek, nebo stránka obsahuje prvek, který tam být nemá.
5	Chybí prvek formuláře a navíc stránka obsahuje více dalších chyb (chybí ikony / špatné ikony / chybí odkazy / špatný font)
6	Chybí prvek formuláře nebo špatně zarovnané objekty ve formuláři, případně stránka obsahuje maximálně jednu nějakou další chybu, nebo stránka obsahuje vyskakovací okno s informacemi o cookies, nebo na stránce chybí více ikon.
7	Chybí logo (1 ikona) nebo má stránka nedostatky, které je vidět až po sescrolování (nejsou tedy ve většině případů vidět) nebo na stránce chybí obrázek a lze to přehlédnout nebo obsahuje otevřené info v záhlaví, které nejde zavřít a navíc špatný font.
8	Na stránce jsou chyby, které si uživatel zdůvodní a ve většině případů jich nebude dbát (posunutá stránka, proužek v záhlaví).
9	Na stránkce jsou chyby, kterých si uživatel nevšimne (chybějící malá ikona v rohu, jiná šířka proužků v záhlaví, nějaká mezera navíc).
10	Stránka je naklonována kompletně.

V případě, že uživatel takovou stránku otevře například ze záložky nebo odkazu, tyto cookies chybí a zobrazí se varování „Pravděpodobně nemáte povoleny cookies“. Stejná situace nastane při pokusu o klonování těchto stránek, tedy pokus se nezdaří, v testovaném vzorku se konkrétně jedná o Státní sektor 3. Řešením tohoto problému je předat `WebDriveru` chybějící cookies, aktualizovat relaci a až poté přistoupit ke stahování prvků stránky. Možnost vkládat cookies byla do aplikace doprogramována a je detailněji popsána v kapitole 2.4.

Tab. 2.2: Vlastnosti stránek a porovnání klonů.

Stránka (typ služeb)	Web server	JS knihovny / frameworky	Web framework	Detekce botů	Reverse proxy	GoPhish klon	Vlastní klon
Finanční sektor 1	×	jQuery UI jQuery	Apache Wicket	×	×	6	9
Finanční sektor 2	×	Dropzone Core-js Axios React RequireJS	×	Akamai Bot Manager	Nginx	0	2
Finanční sektor 3	Apache	jQuery core-js AngularJS	×	×	×	0	7
Finanční sektor 4	×	Core-js LazySides AngularJS	×	×	×	0	7
Finanční sektor 5	×	Core-js React	×	×	×	0	7
Finanční sektor 6	Microsoft-IIS/10.0	jQuery UI jQuery	Microsoft ASP.NET	×	×	8	10
Státní sektor 1	Apache	jQuery UI jQuery	×	×	×	4	8

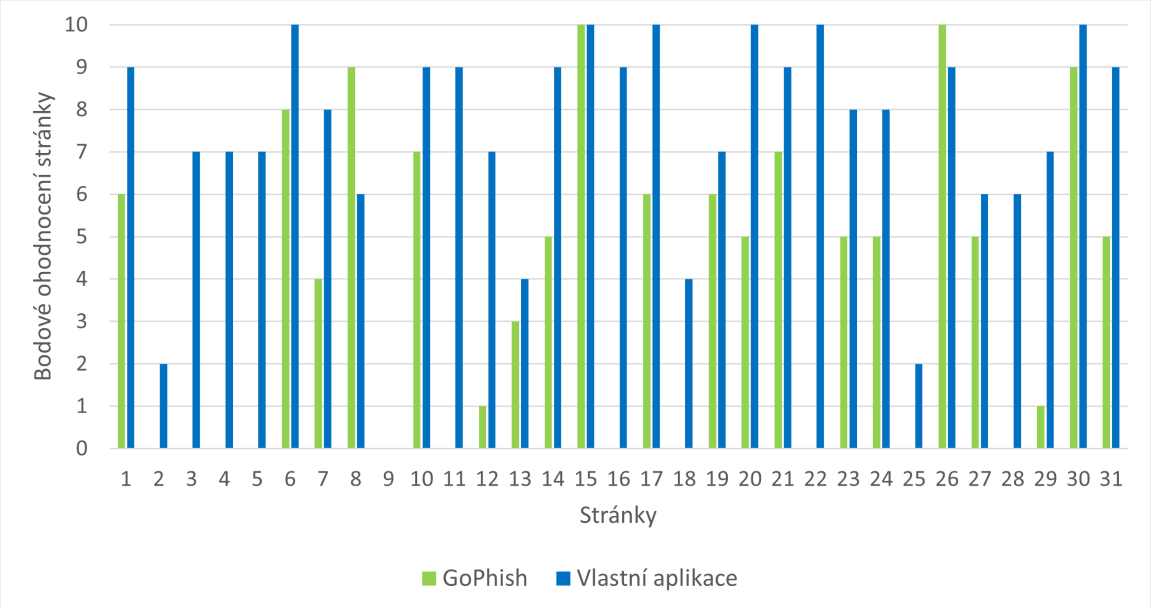
Státní sektor 2	nginx/1.10.3	LightBox jQuery	×	reCAPTCHA	Nginx	9	6
Státní sektor 3	nginx/1.23.4	Core-js jQuery UI jQuery	Django	×	Nginx	0	0
Státní sektor 4	×	×	Ruby on Rails	×	×	7	9
Státní sektor 5	×	jQuery	×	×	×	0	9
Školství 1	nginx	Polyfill Modernizr jQuery Migrate jQuery	×	×	Nginx	1	7
Školství 2	Microsoft- HTTPAPI/2.0	×	×	×	×	3	4
Školství 3	Apache	jQuery UI jQuery	×	×	×	5	9
Školství 4	×	jQuery UI jQuery jQuery Migrate	×	×	×	10	10
Školství 5	×	Underscore.js Core-js Backbone.js	×	×	×	0	9

Školství 6	nginx	Moment.js jQuery UI jQuery DataTables	×	×	Nginx	6	10
Školství 7	IceWarp/12.3.0.3 RHEL6 x64	×	×	×	×	0	4
Školství 8	nginx	Core-js OWL Carousel Modernizr Lightbox jQuery Migrate jQuery toastr	Nette Framework	×	Nginx	6	7
Školství 9	Apache/2.4.54 (Debian)	jQuery	×	×	×	5	10
Školství 10	nginx	Core-js	×	×	Nginx	7	9
Emailové služby 1	×	Knockout.js	×	×	×	0	10
Emailové služby 2	ATS	×	×	×	×	5	8
Emailové služby 3	ATS	×	×	×	×	5	8
Emailové služby 4	nginx	Loadable-Components Core-js jQuery React	×	SmartCaptcha	Nginx	0	2

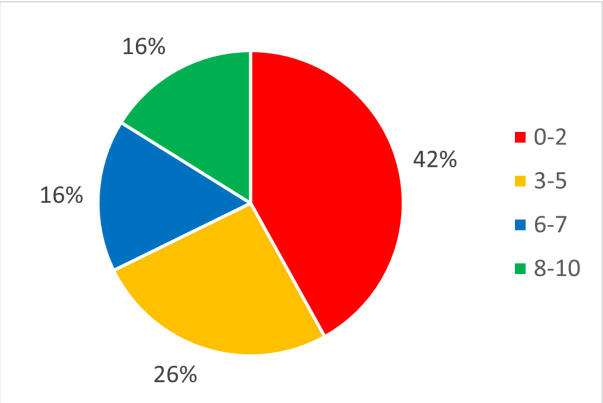
Zájmy/fóra 1	×	jQuery Stimulus	×	×	×	10	9
Zájmy/fóra 2	snooserv	Core-js Google Sign-in authentication	×	×	×	5	6
Zájmy/fóra 3	Next.js	Lodash Core-js Apollo jQuery React Next.js	Next.js	×	×	0	6
Zájmy/fóra 4	Cloudflare nginx	MobX Script.aculo.us jQuery Prototype	×	×	Nginx	1	7
Zájmy/fóra 5	nginx	×	×	×	Nginx	9	10
Zájmy/fóra 6	Apache/2.4.7 (Ubuntu)	jQuery UI jQuery	×	×	×	5	9

Výsledky, které byly získány na základě provedeného testování aplikace, jsou přehledně shrnuty v následujících grafech. V grafu na obrázku 2.6 je pro každou stránku, označenou číslem 0 až 31 podle pořadí v tabulce 2.2, zobrazeno přidělené bodové ohodnocení. Jsou zde porovnány jednotlivé výsledky klonování nástrojem GoPhish (označeno zeleně) s výsledky vyvíjené aplikace (označeno modře). Z celkového počtu 31 stránek dosáhla vlastní aplikace v 27 případech lepšího výsledku, přičemž v 10 z nich se jednalo o stránky, které nástroj GoPhish nedokázal naklonovat ani z části. Dále dvakrát se vyskytl srovnatelný výsledek a ve dvou případech vrátila aplikace výsledek horší.

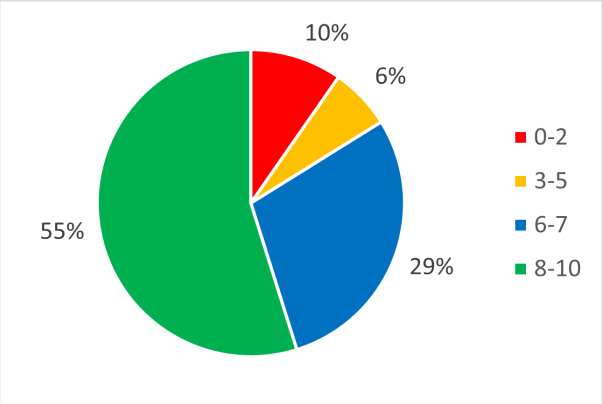
Grafy na obrázcích 2.7 a 2.8 ukazují procentuální zastoupení bodových hodnocení. Na prvním z nich jsou znázorněny výsledky nástroje GoPhish, kdy 42 % stránek se nachází v bodovém rozsahu 0 až 2 body, 26 % v rozsahu 3 až 5 bodů a shodně 16 % odpovídá zbývajícím rozpětím 6 až 7 bodů a 8 až 10 bodů. V případě vlastní aplikace pak nejnižší bodové rozmezí 0 až 2 body představuje 10 % z celkové počtu, 6 % je v rozpětí 3 až 5 bodů, 29 % bylo ohodnoceno 6 nebo 7 body a nadpoloviční většina, tedy 55 % stránek ze vzorku, byla naklonována s hodnocením 8 až 10 bodů. Na základě těchto výsledků je možné říct, že navržená aplikace ve většině případů úspěšně eliminuje nefunkčnost porovnávaného nástroje a poskytuje kompletnější kopie stránek. To znamená, že je následně potřeba méně času pro případné manuální úpravy kódu naklonované stránky.



Obr. 2.6: Porovnání naklonovaných stránek dle bodového ohodnocení.



Obr. 2.7: **GoPhish** – procenta jednotlivých bodových hodnocení.



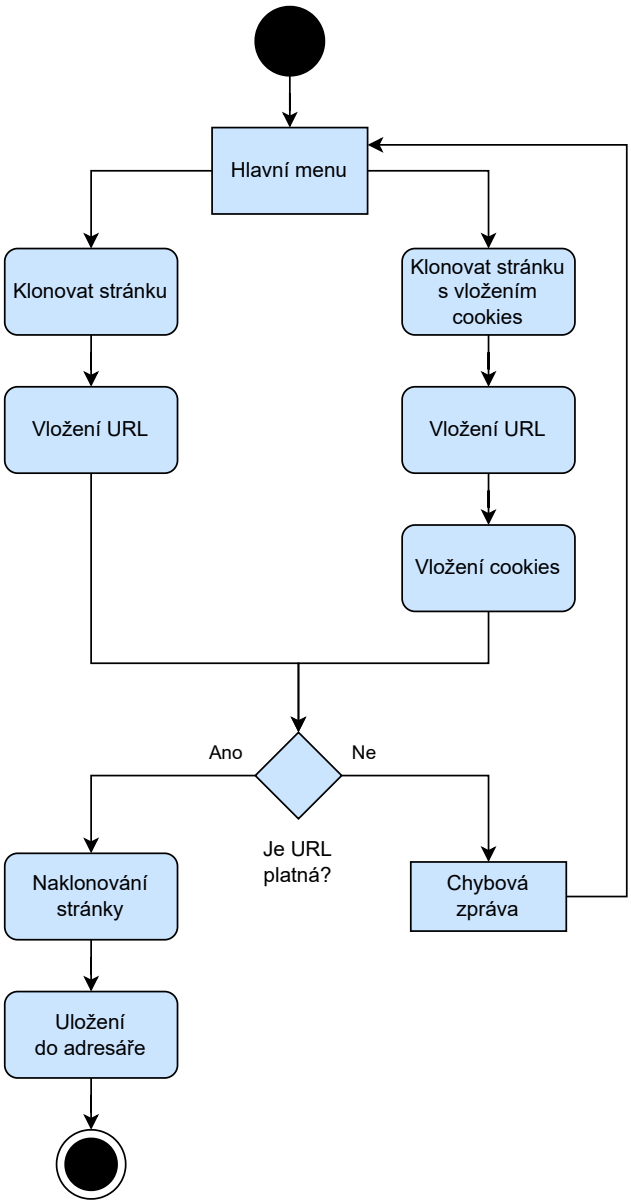
Obr. 2.8: **Vlastní aplikace** – procenta jednotlivých bodových hodnocení.

2.4 Úprava aplikace na základě testování

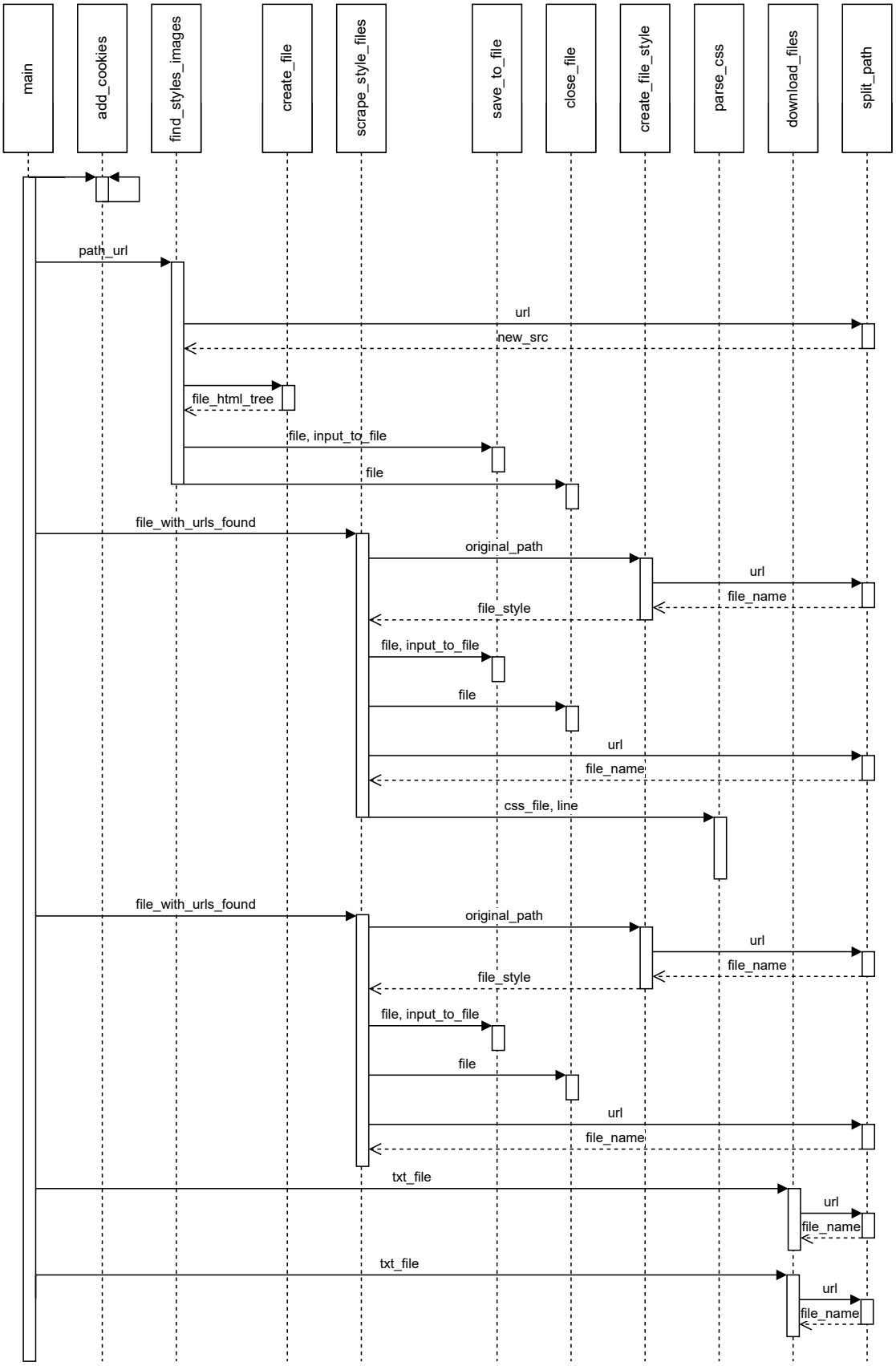
Po provedeném testování a zhodnocení výsledků a příčin, které mohou způsobovat nekompletní naklonování stránek, bylo přistoupeno k implementaci funkcionality pro vkládání cookies, čímž by jedna z těchto příčin byla odstraněna.

Upravená struktura aplikace je zobrazena v UML diagramu na obrázku 2.9. Po spuštění si uživatel v menu zvolí možnost klonovat stránku s vložením cookies a po vložení URL adresy stránky, kterou si přeje naklonovat, je zavolána bezparametrová funkce `add_cookie()`, kde je uživatel vyzván k zadání hodnot atributů `name` a `value`, které cookie tvoří. Funkce je koncipována rekurzivně, takže je možné jednoduše vložit libovolné množství těchto párů. Tyto hodnoty jsou následně předány `WebDriveru` metodou `driver.add_cookie(cookie_dict)`, jež umožňuje vložit soubory cookies do již sestaveného spojení. Z tohoto důvodu je provedena navigace na požadovanou stránku hned poté, co je získána její adresa z uživatelského vstupu,

tedy před zavoláním funkce `add_cookie()`. Po vložení cookies je už jen stránka znovu načtena metodou `driver.refresh()`, aby se změny na stránce projevily, než bude přistoupeno ke stahování jejích prvků. Odtud dále je již běh aplikace bez jakýchkoliv dalších modifikací. Diagram volání funkcí pro scénář s vkládáním cookies je zobrazen na obrázku 2.10. Oproti původnímu diagramu zde není mnoho úprav, avšak pro kompletnost je vhodné ho uvést.



Obr. 2.9: UML diagram aplikace s možností vložit cookies.



Obr. 2.10: Diagram volání funkcí s možností vložit cookies.

2.5 Konečné výsledky

Po implementování funkce pro vkládání cookies bylo znovu provedeno klonování stránky Státní sektor 3, přičemž byly do GET dotazů na stránku vloženy odpovídající cookies soubory. Výstupem byla kopie s jediným nedostatkem, a to mírně posunutým logem o milimetr výše. Nové bodové ohodnocení pro tuto stránku zobrazuje zjednodušená tabulka 2.3, uvádět zde znovu celou tabulku s výsledky testování všech stránek pro ilustraci změny jedné hodnoty by bylo nadbytečné.

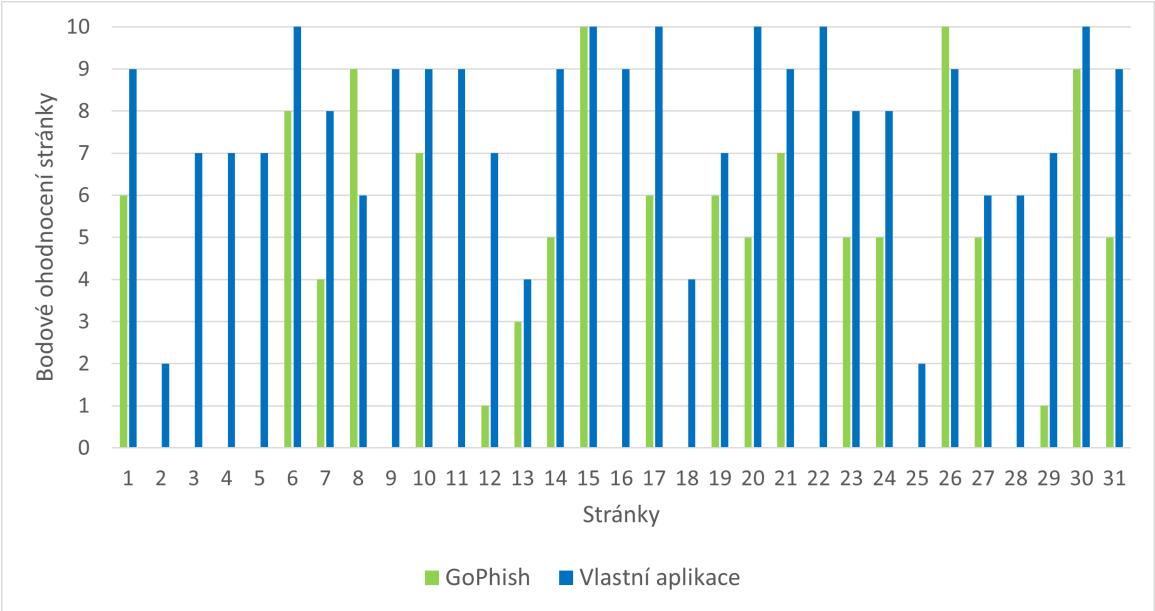
Konečné výsledky porovnání naklonovaných stránek dle bodového ohodnocení po provedené úpravě kódu aplikace jsou shrnuty v grafu na obrázku 2.11. Vyvinutá aplikace dosáhla ve 28 případech z celkových 31 stránek lepšího výsledku než nástroj GoPhish, z čehož v 11 případech šlo o stránky, které porovnávaný nástroj nenaklonoval vůbec. V jednom případě byly výsledky totožné a ve zbývajících dvou šlo o výsledky horší.

Co se týče procentuálního zastoupení bodových hodnocení, rozmezí 0 až 2 body tvoří u vyvinuté aplikace 7 %, což je o 35 % méně než u nástroje GoPhish s 42 %. Do skupiny 3 až 5 bodů spadá u vlastní aplikace 6 % stránek, u porovnávaného nástroje se jedná o 20 % více. Oproti tomu v rozmezí 6 až 7 bodů se nachází více kopií stránek u vlastní aplikace, konkrétně 29 % oproti 16 % v případě nástroje GoPhish. A nakonec v rozmezí 8 až 10 bodů lze u vlastní aplikace nalézt 58 % ze všech stránek z testovaného vzorku, což je o 42 % více než u srovnávaného nástroje, kde se v tomtéž rozmezí nachází 16 %. Graf s výslednými procenty pro jednotlivé skupiny bodů je zobrazen na obrázku 2.12. Pro grafické porovnání s nástrojem GoPhish se lze opět odkázat na obrázek 2.7 v kapitole 2.3 v části shrnující výsledky testování.

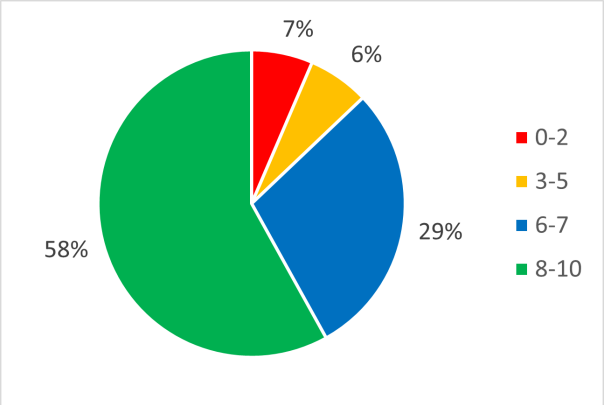
Vyvinutá aplikace tedy úspěšně eliminuje nedostatky současného řešení, a to především v situacích, kdy srovnávaný nástroj stránku nenaklonuje vůbec. Ve většině ostatních případů pak poskytuje kompletnější kopie, ať už se jedná o správný font, zobrazení ikon a odkazů nebo prvků formuláře.

Tab. 2.3: Aktualizovaný výsledek po přidání cookies.

Stránka (typ služeb)	GoPhish klon	Vlastní klon
Státní sektor 3	0	9



Obr. 2.11: Finální porovnání naklonovaných stránek dle bodového ohodnocení.



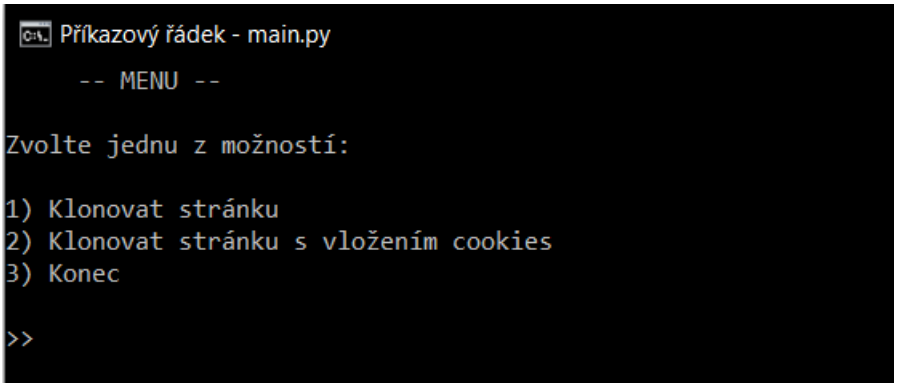
Obr. 2.12: Vlastní aplikace – finální procenta jednotlivých bodových hodnocení.

2.6 Spuštění aplikace a požadavky

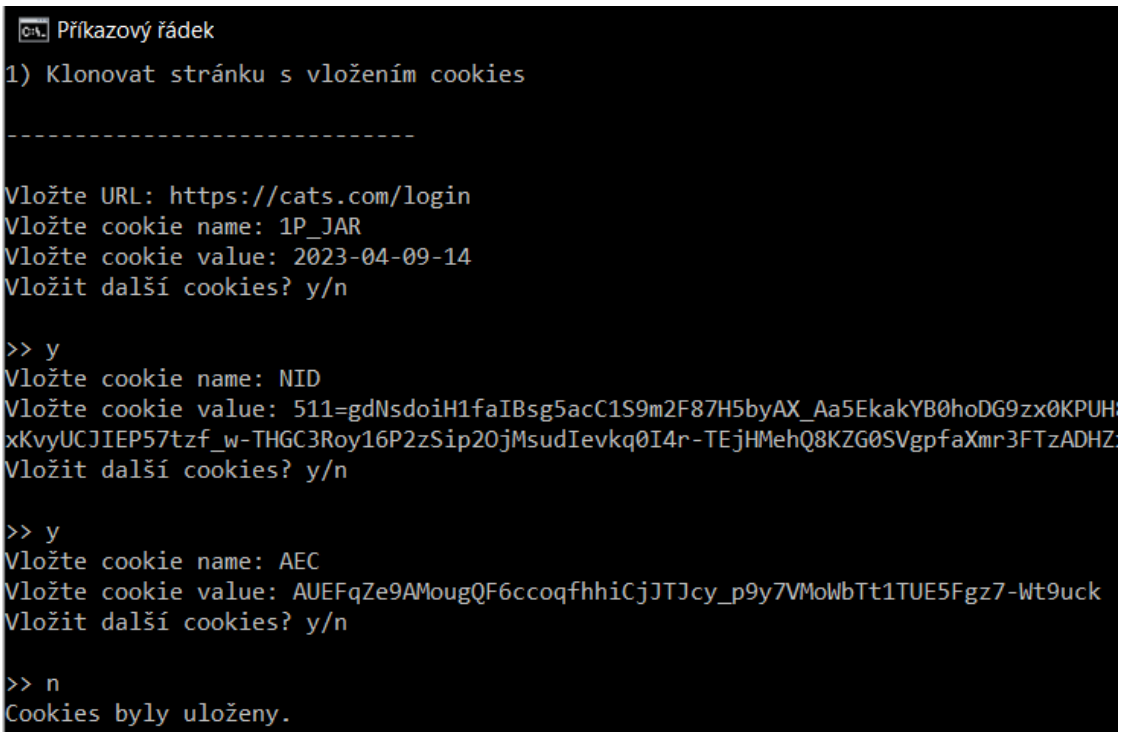
Pro spuštění aplikace je třeba mít nainstalovaný Python 3 a nejnovější verzi prohlížeče Google Chrome. Je to kvůli použitému `Chrome WebDriveru`, jehož aktuální verze je instalována při každém spuštění. Neodpovídající verze prohlížeče a `WebDriveru` spolu nejsou kompatibilní. Veškeré použité moduly a balíčky, které jsou pro funkčnost nezbytné, jsou uvedeny v souboru `requirements.txt`, jež je součástí elektronické přílohy. Jejich instalace je možná příkazem `python setup.py install`.

Aplikaci lze jednoduše spustit z příkazové řádky ze složky obsahující hlavní soubor příkazem `main.py`. Po spuštění se zobrazí menu s možnostmi klonovat stránku přímo a nebo s vložením cookies, jak je vidět na obrázku 2.13. V případě volby první možnosti stačí následně zadat URL adresu stránky, která má být naklonována. V případě druhé možnosti uživatel po vložení URL adresy zadává potřebné cookies, nejdříve hodnotu `cookie name`, poté `cookie value`. Příklad je zobrazen na obrázku 2.14. Během vykonávání programu se do konzole vypisují probíhající úkony, po skončení se objeví oznámení o úspěšném ukončení a program se ukončí. V adresáři, ve kterém se program `main.py` nachází, jsou pak veškeré stažené prvky stránky. Hlavní HTML soubor je označen `index.html`. Soubor `index_original.html` pak obsahuje původní HTML kód před úpravou cest k souborům pro případnou kontrolu.

Co se týče formátu URL adresy, je nutné, aby obsahovala identifikaci HTTP protokolu. Bez ní není možné provést `WebDriverem` navigaci na požadovanou stránku. Dále je potřeba zmínit, že pokud na originální stránce probíhá nějaké přesměrování, aplikaci by měla být předána adresa před přesměrováním. V opačném případě se může stát, že se stránku nepodaří naklonovat.



Obr. 2.13: Náhled menu aplikace.



Obr. 2.14: Náhled menu pro vkládání cookies.

Závěr

Cílem diplomové práce bylo provést analýzu dostupných nástrojů pro klonování stránek, identifikovat jejich nedostatky a na tomto základě navrhnout a implementovat aplikaci, která by je eliminovala. Výslednou aplikaci by mělo být možné využít při phishingové kampani v rámci bezpečnostního testování.

V teoretické části bylo představeno sociální inženýrství s důrazem na phishing, jeho podoby a historický vývoj. Detailně byly také popsány tři modely útoků společně se specifikacemi jednotlivých fází a praktické příklady realizace. Velká část práce byla věnována analýze současného stavu problematiky na poli klonovacích nástrojů, kdy u nejčastěji používaných (HTTrack Website Copier, Social Engineering Toolkit a GoPhish) byly porovnány jejich obecné vlastnosti a jejich funkcionality byly otestovány na vzorku několika stránek. Poté u nástroje GoPhish, který se ukázal být nejlepším ze zmiňovaných, bylo provedeno podrobnější testování na větším vzorku, díky čemuž bylo možné zjistit, kde má nedostatky. Pokryto bylo také právní hledisko kopírování webových stránek a následky protiprávního jednání.

V praktické části byla navrhnuta a implementována vlastní aplikace v jazyce Python za využití `WebDriveru` a knihoven `Beautiful Soup` a `cssutils`. Testování bylo realizováno na více než třiceti stránkách, přičemž každá z nich byla klonována i nástrojem GoPhish a vytvořené kopie byly ohodnoceny na bodové stupnici od nuly do deseti bodů dle míry kompletnosti nebo absence zásadních prvků. Následně byly popsány možné příčiny způsobující neúplné naklonování některých stránek, a kde to bylo možné, bylo navrženo řešení. Pro jednu z těchto příčin, konkrétně chybějící cookies při přístupu na stránku `WebDriverem`, byla doprogramována funkcionalita pro vložení daných souborů uživatelem.

Finální testování ukázalo, že vlastní aplikace dosáhla v 90,3 % případů lepších výsledků než porovnávaný nástroj a obzvláště efektivní je v případech, kdy GoPhish danou stránku nedokáže naklonovat vůbec. Pro většinu ostatních stránek vrací kompletnější kopie, především pak co se týče fontu a zobrazených ikon. Aplikaci bude možné využít v rámci red team testování a školení zaměstnanců na metody a útoky sociálním inženýrstvím, čímž bude zvyšována kybernetická bezpečnost státu. Komplexní implementací vlastní aplikace byly veškeré definované cíle diplomové práce splněny.

Literatura

[1] MORAVČÍK, Ondřej. Vývoj registrované kriminality v roce 2021. *Policie České republiky* [online]. c2020 [cit. 2022-11-27]. Dostupné z: <https://www.policie.cz/clanek/vyvoj-registrovane-kriminality-v-roce-2021.aspx>

[2] Cyber crime categories that were reported most often in 2021, by number of victims.*Statista* [online]. [cit. 2022-11-27]. Dostupné z: <https://www.statista.com/statistics/184083/commonly-reported-types-of-cyber-crime/>

[3] BASSETT, Gabriel, C. David HYLENDER, Philippe LANGLOIS, Alex PINTO a Suzanne WIDUP. 2022 Data Breach Investigations Report. *Verizon* [online]. c2022 [cit. 2022-11-27]. Dostupné z:<https://www.verizon.com/business/en-gb/resources/reports/dbir/2022>

[4] HATFIELD, Joseph M. Social engineering in cybersecurity: The evolution of a concept. *Computers & Security* [online]. 2018, (73), 102-113 [cit. 2022-11-27]. ISSN 0167-4048. Dostupné z: <https://www.sciencedirect.com/science/article/pii/S0167404817302249>

[5] LAPSLEY, Phil. *The History of Phone Phreaking* [online]. c2005-2009 [cit. 2022-11-27]. Dostupné z: <http://historyofphonephreaking.org>

[6] WANG, Zuoguang, Limin SUN a Hongsong ZHU. Defining Social Engineering in Cybersecurity. *IEEE Access* [online]. 2020, 8, 85094-85115 [cit. 2023-05-01]. ISSN 2169-3536. Dostupné z: doi:10.1109/ACCESS.2020.2992807

[7] MOTYL, Ivo, Jan PALKA a Jiri PALKA. Advanced Methods for Securing the Information Systems. *Annals of DAAAM and Proceedings of the International DAAAM Symposium* [online]. 2010, **21**(1) [cit. 2023-05-02]. ISSN 1726-9679. Dostupné z: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84904279122&partnerID=40&md5=403bbcea2778ada3479210a311e3195f>

[8] Who is Kevin Mitnick?: Everything You Need to Know About Kevin Mitnick, the Number 1 Authority on Hacking, Social Engineering and Security Awareness Training. *Mitnick Security: Kevin Mitnick and the Global Ghost Team* [online]. Mitnick Security Consulting, c204-2022 [cit. 2022-11-29]. Dostupné z: <https://www.mitnicksecurity.com/about-kevin-mitnick-mitnick-security>

[9] MITNICK, Kevin D. *The Art of Deception: Controlling the Human Element of Security* [online]. John Wiley, 2003 [cit. 2022-11-29]. ISBN13: 9780764542800.

Dostupné z: <https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnx1Y3M0MDJ8Z3g6MmZiYzA4YjdmYmM4NDgwOA>

[10] MOUTON, Francois, Mercia M. MALAN, Louise LEENEN a H.S. VENTER. Social engineering attack framework. *2014 Information Security for South Africa* [online]. IEEE, 2014, 2014, 1-9 [cit. 2022-11-29]. ISBN 978-1-4799-3384-6. Dostupné z: doi:10.1109/ISSA.2014.6950510

[11] MOUTON, Francois, Louise LEENEN, Mercia M. MALAN a H.S. VENTER. Towards an Ontological Model Defining the Social Engineering Domain. *IFIP Advances in Information and Communication Technology* [online]. 2014, July 2014, (431) [cit. 2022-11-30]. ISBN: 978-3-662-44207-4. Dostupné z: doi:10.1007/978-3-662-44208-1 22

[12] HEIKKINEN, Seppo. Social engineering in the world of emerging communication technologies. *Proceedings of Wireless World Research Forum* [online]. 2006 [cit. 2023-04-29]. Dostupné z: <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=be5a68ba31989b6d224dd5666a6b2392b067b886>

[13] LEMOS, Robert. Mitnick teaches 'social engineering'.*ZDNET* [online]. c2023, July 18, 2000 [cit. 2023-04-30]. Dostupné z: <https://www.zdnet.com/article/mitnick-teaches-social-engineering/>

[14] ALAZRI, Aisha Suliaman. The awareness of social engineering in information revolution: Techniques and challenges. *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)* [online]. IEEE, 2015, 2015, 198-201 [cit. 2023-04-30]. ISBN 978-1-9083-2052-0. Dostupné z: doi:10.1109/ICITST.2015.7412088

[15] SYAFITRI, Wenni, Zarina SHUKUR, Umi Asma' MOKHTAR, Rossilawati SULAIMAN a Muhammad Azwan IBRAHIM. Social Engineering Attacks Prevention: A Systematic Literature Review. *IEEE Access* [online]. 2022, 10, 39325-39343 [cit. 2023-04-30]. ISSN 2169-3536. Dostupné z: doi:10.1109/ACCESS.2022.3162594

[16] GIANDOMENICO, Nena. What is Spear-phishing? Defining and Differentiating Spear-phishing from Phishing. *Digital Guardian* [online]. February 28, 2023 [cit. 2023-04-30]. Dostupné z: <https://www.digitalguardian.com/blog/what-is-spear-phishing-defining-and-differentiating-spear-phishing-and-phishing>

[17] KROMBHOLZ, Katharina, Heidelinde HOBEL, Markus HUBER a Edgar WEIPPL. Advanced social engineering attacks. *Journal of Information Security and Applications* [online]. 2015, 22, 113-122 [cit. 2023-04-30]. ISSN 22142126. Dostupné z: doi:10.1016/j.jisa.2014.09.005

[18] KHACHUNTS, Hasmik. What is Angler Phishing and How Can You Avoid It?. *Security Boulevard* [online]. c2023, January 26, 2022 [cit. 2023-04-30]. Dostupné z: <https://securityboulevard.com/2022/01/what-is-angler-phishing-and-how-can-you-avoid-it/>

[19] Number of unique phishing sites detected worldwide from 3rd quarter 2013 to 1st Quarter 2021. *Statista*[online]. June 2021 [cit. 2022-10-02]. Dostupné z: <https://www.statista.com/statistics/266155/number-of-phishing-domain-names-worldwide/>

[20] Social engineering penetration testing. *TechTarget* [online]. c2023 [cit. 2023-05-03]. Dostupné z: <https://www.techtarget.com/whatis/definition/social-engineering-penetration-testing>

[21] How to Run an Effective Phishing Test at Work. *Dashlane* [online]. c2023, March 7, 2020 [cit. 2023-05-03]. Dostupné z: <https://www.dashlane.com/blog/phishing-test>

[22] SOUPPAYA, Murugiah P. a Karen A. SCARFONE. Technical Guide to Information Security Testing and Assessment. *Special Publication (NIST SP), National Institute of Standards and Technology* [online]. Gaithersburg, MD, 2008, September 30, 2008 [cit. 2023-05-03]. Dostupné z: <https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=152164>

[23] WRIGHT, Ryan a Bennett THATCHER. Phishing Tests Are Necessary. But They Don't Need to Be Evil. *Harvard Business Review* [online]. April 01, 2021 [cit. 2023-05-03]. Dostupné z: <https://hbr.org/2021/04/phishing-tests-are-necessary-but-they-dont-need-to-be-evil>

[24] GNU General Public License. *The GNU Operating System and the Free Software Movement* [online]. [cit. 2022-11-27]. Dostupné z: <https://www.gnu.org/licenses/gpl-3.0.en.html>

[25] ROCHE, Xavier. *HTTrack Website Copier* [online]. c2022 [cit. 2022-11-27]. Dostupné z: <https://www.httrack.com/>

[26] ROCHE, Xavier. Httrack. *GitHub* [online]. c2022 [cit. 2022-11-27]. Dostupné z: <https://github.com/xroche/httrack>

[27] *The MIT License (MIT)* [online]. c2022 [cit. 2022-11-27]. Dostupné z: <https://mit-license.org/>

[28] *GoPhish User Guide* [online]. [cit. 2022-11-27]. Dostupné z: <https://docs.getgophish.com/user-guide/>

[29] BSD licenses. *CodeDocs* [online]. c2022 [cit. 2022-11-27]. Dostupné z: <https://codedocs.org/what-is/bsd-licenses>

[30] KENNEDY, David. The Social-Engineer Toolkit (SET). *GitHub* [online]. c2022 [cit. 2022-11-27]. Dostupné z: <https://github.com/trustedsec/social-engineer-toolkit>

[31] Social Engineer Toolkit - issues. *GitHub* [online]. c2022 [cit. 2022-11-27]. Dostupné z: <https://github.com/trustedsec/social-engineer-toolkit/issues>

[32] MORRIS, Scott. Tech 101: What is JavaScript?: Everything you wanted to know about JavaScript and then some!. *Skillcrush* [online]. Skillcrush, c2012-2022 [cit. 2022-11-27]. Dostupné z: <https://skillcrush.com/blog/javascript/>

[33] MILLER, Stephen. What Is JavaScript Used For?. *Codecademy* [online]. NYC, c2022, 26 May 2021 [cit. 2022-11-27]. Dostupné z: <https://www.codecademy.com/resources/blog/what-is-javascript-used-for/>

[34] MONUS, Anna. JavaScript security: Vulnerabilities and best practices. *Raygun* [online]. c2022, 15 Dec 2021 [cit. 2022-11-27]. Dostupné z: <https://raygun.com/blog/js-security-vulnerabilities-best-practices/>

[35] HOFFMAN, Chris. What Is NoScript, and Should You Use It to Disable JavaScript?. *How-To Geek* [online]. LifeSavvy Media, c2022 [cit. 2022-11-27]. Dostupné z: <https://www.howtogeek.com/138865/htg-explains-should-you-disable-javascript/>

[36] JANSA, Lukáš, Petr OTEVŘEL, Jiří ČERMÁK, Petr MALIŠ, Petr HOSTAŠ, Michal MATĚJKA a Ján MATEJKA. Internetové právo. Brno: Computer Press, 2016. ISBN 978-80-251-4664-4.

[37] JANSA, Lukáš. Kopírování internetových stránek. *Právo IT: Odborný portál o právu informačních technologií* [online]. c2016, 1. 11. 2009 [cit. 2023-04-22]. Dostupné z: <https://www.pravoit.cz/novinka/kopirovani-internetovych-stranek>

[38] RICHARDSON, Leonard. *Beautiful Soup Documentation* [online]. c2004-2015 [cit. 2022-11-27]. Dostupné z: <https://beautiful-soup-4.readthedocs.io>

[39] HÖKE, Christof. *Cssutils 1.0 documentation* [online]. c2004-2013 [cit. 2023-04-09]. Dostupné z: <https://pythonhosted.org/cssutils/>

[40] What is a Dynamic Web Page?. *Packetlabs* [online]. c2022, 6 Aug 2022 [cit. 2022-11-27]. Dostupné z: <https://www.packetlabs.net/posts/dynamic-pages/>

[41] ChromeDriver. *ChromeDriver - WebDriver for Chrome* [online]. [cit. 2022-11-27]. Dostupné z: <https://chromedriver.chromium.org/>

[42] Selenium - Webdriver. *Tutorials Point: Simply Easy Learning* [online]. c2022 [cit. 2022-11-27]. Dostupné z: <https://www.tutorialspoint.com/selenium/selenium_webdriver.htm>

[43] Urllib.parse — Parse URLs into components. *Python* [online]. c2001-2023 [cit. 2023-04-09]. Dostupné z: <https://docs.python.org/3/library/urllib.parse.html>

[44] Os.path — Common pathname manipulations. *Python* [online]. c2001-2023 [cit. 2023-04-09]. Dostupné z: <https://docs.python.org/3/library/os.path.html>

[45] Font-family. *MDN Web Docs* [online]. c1998–2023 [cit. 2023-04-09]. Dostupné z: <https://developer.mozilla.org/en-US/docs/Web/CSS/font-family>

[46] MEGIDA, Dillion. Self-hosted fonts vs. Google Fonts API. *LogRocket* [online]. Boston, c2023, August 14, 2020 [cit. 2023-04-09]. Dostupné z: <https://blog.logrocket.com/self-hosted-fonts-vs-google-fonts-api/>

[47] Bot Detection – How to Detect Bots on Your Website, Apps, & APIs. *DataDome* [online]. c2023, Sep 21, 2022 [cit. 2023-04-09]. Dostupné z: <https://datadome.co/bot-management-protection/bot-detection-how-to-identify-bot-traffic-to-your-website/>

[48] VASTEL, Antoine. Bot detection 101: How to detect web bots?. *Antoine Vastel Blog* [online]. c2023, February 9th, 2020 [cit. 2023-04-09]. Dostupné z: <https://antoinevastel.com/javascript/2020/02/09/detecting-web-bots.html>

[49] MORRIS, Scott. Tech 101: What Is a JavaScript Framework? Here's Everything You Need to Know. *Skillcrush* [online]. c2012-2023 [cit. 2023-04-09]. Dostupné z: <https://skillcrush.com/blog/what-is-a-javascript-framework/>

[50] KIDDER, T. J. What is a JavaScript Framework?. *LEARN academy* [online]. San Diego, c2023, August 23, 2019 [cit. 2023-04-09]. Dostupné z: <https://learnacademy.org/blog/what-is-a-javascript-framework/>

[51] Fetch: Cross-Origin Requests. *JAVASCRIPT.INFO* [online]. c2007—2023, October 18, 2022 [cit. 2023-04-09]. Dostupné z: <https://javascript.info/fetch-crossorigin>

[52] SALIBA, Simon. Exploiting CORS Misconfiguration Vulnerabilities on Web Servers. *Medium* [online]. Feb 6, 2021 [cit. 2023-04-09]. Dostupné z: <https://medium.com/swlh/exploiting-cors-misconfiguration-vulnerabilities-2a16b5b979>

[53] Origin. *MDN Web Docs* [online]. c1998–2023 [cit. 2023-04-09]. Dostupné z: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Origin>

[54] Web Application Firewall. *OWASP* [online]. California, c2023 [cit. 2023-04-09]. Dostupné z: <https://owasp.org/www-community/Web_Application_Firewall>

[55] Bot Manager vs. WAF: Why You Actually Need Both. *Radware blog* [online]. c2023, February 8, 2022 [cit. 2023-04-09]. Dostupné z: <https://blog.radware.com/security/applicationsecurity/2022/02/bot-manager-vs-waf-why-you-actually-need-both/>

[56] RICHABADAS, Tushar. Barracuda Web Application Firewall protects against web scraping. *Barracuda: Your journey, secured.* [online]. Campbell (California), c2003-2022, Aug. 1, 2016 [cit. 2023-04-09]. Dostupné z: <https://blog.barracuda.com/2016/08/01/barracuda-web-application-firewall-protects-against-web-scraping/>

[57] Using HTTP cookies. *MDN Web Docs* [online]. c1998–2023 [cit. 2023-04-09]. Dostupné z: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies>

[58] Set-Cookie. *MDN Web Docs* [online]. c1998–2023 [cit. 2023-04-09]. Dostupné z: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>

[59] JOHANSEN, Alison Grace. Should you accept cookies? 5 times you definitely shouldn't. *Norton* [online]. c2023, August 15, 2022 [cit. 2023-04-09]. Dostupné z: <https://us.norton.com/blog/privacy/should-i-accept-cookies>

Seznam symbolů a zkratek

APT	Advanced Persistent Threat
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
CORS	Cross-Origin Resource Sharing
CSS	Cascading Style Sheets – kaskádové styly
DDoS	Distributed Denial-of-Service
DoS	Denial-of-Service
FBI	Federal Bureau of Investigation – Federální úřad pro vyšetřování
GUI	Graphical User Interface – grafické uživatelské rozhraní
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IoT	Internet of Things
IS	informační systém
MITM	Man-in-the-middle
OSINT	Open Source Intelligence
SI	sociální inženýrství
SET	Social Engineering Toolkit
URL	Uniform Resource Locator
VUT	Vysoké učení technické
WAF	Web Application Firewall – firewall webové aplikace
XSS	Cross-site Scripting

A Obsah elektronické přílohy

Elektronická příloha práce je dostupná ve veřejném GitHub[1] repozitáři a obsahuje soubor `main.py` se zdrojovým kódem aplikace, seznam potřebných knihoven a modulů v dokumentu `requirements.txt` a soubor `setup.py`, který uvedené požadavky po spuštění příkazem `python setup.py install` nainstaluje do systému uživatele. Soubor README.md obsahuje návod ke spuštění a uvádí příklad použití pro stránku VUT.

- `/` ..... kořenový adresář přiloženého archivu
  - `DP-Application-for-cloning-websites` ..... adresář obsahující zdrojový soubor
    - `main.py` ..... zdrojový kód aplikace
  - `README.md` ..... návod ke spuštění
  - `requirements.txt` ..... požadavky
  - `setup.py` ..... soubor pro instalaci balíčků

[1] <https://github.com/dikubi/DP-Application-for-cloning-websites.git>