Platforma pro realizaci a správu penetračních testů – sociotechniky

Bc. Jaroslav Strnad

Diplomová práce
2024

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2023/2024

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Jaroslav Strnad**
Osobní číslo:	**A21178**
Studijní program:	**N0613A140022 Informační technologie**
Specializace:	**Softwarové inženýrství**
Forma studia:	**Kombinovaná**
Téma práce:	**Platforma pro realizaci a správu penetračních testů – sociotechniky**
Téma práce anglicky:	**Platform for Implementation and Management of Penetration Tests – Sociotechnics**

Zásady pro vypracování

1. Specifikujte požadavky na systém s ohledem na jeho zabezpečení.
2. Vyberte vhodné komponenty systému při zaměření na Phishing a Spear Phishing.
3. Zpracujte víceúrovňový přístup do systému.
4. Navrhněte systém pro realizaci Phish a SpearPhis kampaní s jejich vyhodnocením.
5. Navržený systém implementujte v testovacích prosředí a ověřte jeho funkčnost.

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. MICHELE FINCHER. Phishing Dark Waters: The Offensive and Defensive Sides of Malicious Emails. 2015. ISBN 9781118958476. Dostupné také z: https://search.ebscohost.com/login.aspx?direct=true&db=edsebk&an=969390&scope=site.
2. HADNAGY, Christopher. Social engineering: the science of human hacking. Second edition. Indianapolis, IN: John Wiley & Sons, [2018], 1 online resource. ISBN 9781119433729. Dostupné také z: https://proxy.k.utb.cz/login?url=https://onlinelibrary.wiley.com/doi/book/10.1002/9781119433729.
3. SIADATI, Hossein, Toan NGUYEN, Nasir MEMON, et al. X-Platform Phishing: Abusing Trust for Targeted Attacks Short Paper. Financial Cryptography and Data Security: FC 2017 International Workshops, WAHC, BITCOIN, VOTING, WTSC, and TA, Sliema, Malta, April 7, 2017, Revised Selected Papers [online]. 2017, 10323, 587-596 [cit. 2022-11-30]. ISBN 9783319702773. ISSN 03029743. Dostupné z: doi:10.1007/978-3-319-70278-0_37.
4. CHIEW, Kang Leng, Kelvin Sheng Chek YONG a Choon Lin TAN. A survey of phishing attacks: Their types, vectors and technical approaches. Expert Systems with Applications [online]. 2018, 106, 1-20 [cit. 2022-11-30]. ISSN 09574174. Dostupné z: doi:10.1016/j.eswa.2018.03.050.
5. AIBEKOVA, Altynai a Vinesha SELVARAJAH. Offensive Security: Study on Penetration Testing Attacks, Methods, and their Types. 2022 IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE), Distributed Computing and Electrical Circuits and Electronics (ICDCECE), 2022 IEEE International Conference on [online]. 2022, 1-9 [cit. 2022-11-30]. ISBN 9781665483162. ISSN edseee.IEEEConferenc. Dostupné z: doi:10.1109/ICDCECE53908.2022.9792772.
6. ILCA, Florin a Titus BALAN. Phishing as a Service Campaign using IDN Homograph Attack. 2021 International Aegean Conference on Electrical Machines and Power Electronics (ACEMP) [online]. 2021, 338-344 [cit. 2022-11-30]. ISBN 9781665402989. ISSN edseee.IEEEConferenc. Dostupné z: doi:10.1109/OPTIM-ACEMP50812.2021.9590028.

Vedoucí diplomové práce: **Ing. David Malaník, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **5. listopadu 2023**
Termín odevzdání diplomové práce: **13. května 2024**

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 5. ledna 2024

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, 12.5.2024

Jaroslav Strnad, v. r.
podpis studenta

ABSTRAKT

Od začátku internetu byli jeho uživatelé cílem kybernetických útoků. Společně s rostoucí internetovou populací se rozrostla také bezpečnostní rizika. Proti těmto rizikům byla tvořena úspěšná bezpečnostní opatření, ovšem časem přišel kybernetický útok, který se nedá plně zastavit technologickými prostředky. Tento typ útoku, který se zaměřuje na manipulaci s lidmi je zvaný phishing a jeho úspěšnost roste rok co rok. V této práci vysvětlíme, proč jsou tyto útoky tak úspěšné, jak fungují, co znamená sociotechnika a co má společného s phishingem, jak vypadá budoucnost phishingu a proč hlavním způsobem, jak se oproti phishingu bránit, je vědět o něm dostatečně informací a regulerně trénovat na cvičných útocích. Z tohoto důvodu vytvoříme platformu pro tvorbu falešných přihlašovacích portálů a odesílání phishingových kampaní, která bude pro tento trénink použitelná.

Klíčová slova: phishing, spearphishing, sociální inženýrství, manipulace, virtualizace, Debian Linux

ABSTRACT

Since the advent of internet its users have been targets of cyber attacks. Together with the rising internet population, the security risks have also been rising. Successful security measures have been made against these risks, however with time a cyber attack which cannot be fully stopped with technological meassures came. This type of attack, that focuses on manipulation of people, is called phishing and its success has been rising year by year. In this thesis we will explain why are these attack so successful, how they work, what is sociotechnics and how it correlates with phishing, how does the future of phishing look, and why the main way to defend one against phishing is to know enough information about it, and regularily practice on training attacks. Because of that we will create a platform for creation of fake login portals and sending phishing campaigns, which could be used for such training.

Keywords: phishing, spearphishing, social engineering, manipulation, virtualization, Debian Linux

Chtěl bych tímto poděkovat Ing. Davidu Malaníkovi, PhD. za cenné rady, přívětivost a jasné zadání požadavků.

Také bych chtěl poděkovat MUDr. Haně Strnadové, mé matce, za psychickou a další podporu v těchto těžkých časech, a že to se mnou vydržela.

Prohlašuji, že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

V dnešní informační době si jako lidstvo užíváme všelijaké výhody, spojené s faktem, že jsme připojeni ke globální síti prakticky každou vteřinu našeho života. Užíváme si instantní komunikaci s kýmkoliv na světě, již nemusíme chodit do banky, kvůli každé otázce a dokonce jsme schopni pracovat plně z domova. Tento luxus permanentní konektivity si užívá čím dále víc lidí po celém světě, počet lidí, co mají přístup k internetu se znásobil téměř tři a půlkrát od roku 2007, kdy byl vydán první iPhone, a započala tzv. Smartphonová generace [1]. Tento jednoduchý přístup ovšem nepatří pouze lidem, kteří by chtěli internet používat bez temnějších motivů. Kriminálníci na internetu, a jejich kybernetické útoky nejsou zdaleka nic nového, co je ovšem nové, jsou přistupy, které kriminálníci používají, k získání svého cíle. Zatímco kriminálníci vymýšleli nové způsoby prolomení technických ochran, security experti se snažili pravého opaku, snažili se těmto novým útokům předejít, přinejhorším je opravit co nejdříve. Tímto se tyto dvě frakce předháněli, a předhánějí se dodnes. Ovšem kriminálníci také přišli na způsob, jak se dostat ke svému cíly, aniž by security experti mohli chybu napravit. Nelze tuto chybu úplně opravit, protože se pro jednou netýká pouze technický záležtostí. Ona chyba jsou lidé, jejich myšlení, logika, víra v ostatní lidi a technické znalosti. Tímto začal fenomén zvaný phishing, neagresivní (oproti například Rubber hose attack) typ kybernetického útoku, který se specializuje na nejslabší bod veškeré technologické ochrany: lidi. S phishingovými útoky se setkáváme čím dál častěji: nejen že stále rostoucí přístup k internetu umožňuje kriminálníkům získávat nové, potencionálně technologicky neznalé cíle, ale také například změna generací lidí ve firmách (Generace Z, lidé narozeni od roku 1997 (konkrétní rok se liší podle zdroje) v posledních letech nastupovali do pracovního nasazení, také známa jako generace, která vyrůstala s technologiemi jako Smartphony, ovšem chybí jim základní technologické znalosti ohledně počítačů) [2] nepomáhá narůstajícímu počtu phishing útoků. Můžeme se snažit zabránit phishingovým útokům z technické stránky sebevíc, ovšem i když by tato prevence mohla napomoct potenciální phishingový útok zastavit, není jisté, že by tento útok neuspěl na stránce sociotechnické. Z tohoto důvodu je nejlepším způsobem prevence phishingových útoků nejen technická prevence, ale také edukace a trénink ohledně phishingu. Nejedná se o stoprocentní ochranu, přece jen lidé nejsou perfektní, chybuji, či občas věří ostatním až moc. V rámci proti phishingového tréningu je také potřeba držet lidi na nohách, aby si uvědomovali, že se mohou stát obětí phishingového útoku každou chvíli. Z tohoto důvodu, je dobrý nápad lidem jednou za čas zaslat cvičný/penetrační phishingový útok, k prověření

jejich znalostí o subjektu. V této práci Vás podrobně seznámím s phishingem, jeho mnoha formami a jak se jim bránit, se sociálním inženýrstvím (sociotechnikou) a proč je toto sociologické (až psychologické) téma úzce spojené s phishingem a provedu Vás tvorbou platformy na realizaci penetračních phishingových útoků ve formě e-mailových zpráv.

I. TEORETICKÁ ČÁST

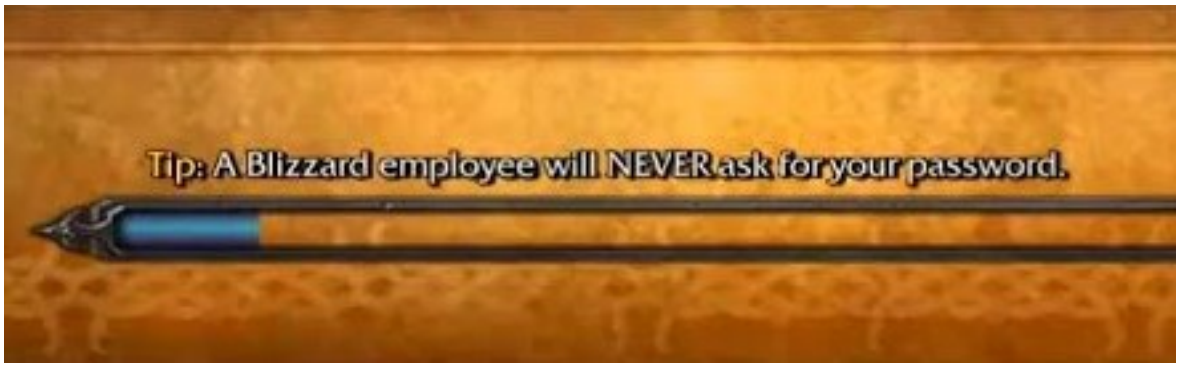
1 ANALÝZA ŘEŠENÉHO PROBLÉMU – PHISHING

1.1 Základní informace a historie

Oficiální definice phishingu zní následovně (přeloženo do češtiny): “Technika pokusu o získání citlivých údajů, jako jsou čísla bankovních účtů, prostřednictvím podvodné žádosti v e-mailu nebo na webové stránce, ve které se pachatel vydává za legitimní firmu nebo osobu s dobrou pověstí. [3] K této definice je nutno dodat, že e-mailové zprávy, či webové stránky nejsou zdaleka jedinnými způsoby, kterými lze obět oklamat. Detailnější popis všech phishingových způsobů bude následovat v podkapitolách. Samotné jméno phishing vychází z anglického “fishing”, neboli rybaření, protože útočník rybaří v rybníku plném lidí, a snaží se najít tu jednu “rybu”, která se chytne na jeho lži. Důvod pro změnu z normálního fishing na phishing, je již téměř mrtvá internetová tradice, kdy jistá slova jsou změněna na “internetovou verzi” onoho slova: například freak = phreak (takto si říkali první hackeři), nebo tzv. leetspeak, kdy písmena bývala nahrazována čísly. [4]

Historicky se první pravé phishingové útoky dají dohledat do devadesátých let, na vrcholu popularity instatního messengeru America Online (AOL), kdy se uživatelé vydávali za administrátory AOL, za cílem získání osobních údajů jiných uživatelů. První užití samotného slova phishing bylo ve Windows programu AOHell, který značně automatizoval samotný phishing za pomoci automatického generátoru nových America Online účtů a automatickým rozesíláním phishingových zpráv náhodným uživatelům, ve kterých se nový účet vydával za administrátora a požadoval po uživateli jeho osobní údaje (uživatelské jméno a heslo). AOHell překvapivě nebyl vytvořen z důvodu obohatit se z ukradnutých účtů. Tvůrce AOHell, tehdejší teenager pod pseudonymem Da Chronic vytvořil AOHell z překvapivě ušlechtilého důvodu: AOL mělo velký problém s chatroomy, které se specializovali na sdílení ilegálního materiálu obsahujícím nezletilé osoby, a reální administrátoři AOL odmítali s tímto problémem cokoliv dělat. AOHell byl tudíž vytvořen aby udělal to největší chaos na AOL, do bodu, kdy by administrátoři museli začít problém řešit. [5] Toto zmiňuji, protože je dobré mít v paměti, že i když phishingové útoky jsou nemorální (ovšem tento bod se může lišit od osoby k osobě) a ilegální (více o legalitě phishingu později v kapitole), ne vždy jsou použity pro nekalé/zlé účely. AOHell byl ve své práci velice úspěšný, navzdory jistým elementům tohoto programu, které by v dnešní době nebyly tak úspěšné. Zejména samotný způsob automatizovaného phishingu s jednou jedinou možnou zprávou ve stylu “jsem administrátor, dej mi své údaje” by byl v dnešní

době jednoduše prokouknut, protože v rámci prevence přesně těchto phishingových útoků téměř všechny internetové služby, od bank po videohry, upozorňují přímo na tento typ útoku, kdy se "administrátor snaží dostat živatelské údaje".



Obrázek 1 Načítací obrazovka videohry World of Warcraft (2004), obsahující varování ohledně phishingu

Zde vidíme fenomén, se kterým se budeme setkávat v těžké většině teoretické části, a jeden z hlavních důvodů, proč vytvořit a používat testovací program na rozesíláni testovacích phishingových kampaní je dobrý nápad. Tento fenomén jsou obecné technologické znalosti demografie lidí, kteří jsou potencionálními cíly phishingu.

Toto extrémně nahrávalo AOHellu v době, kdy byl aktivní. Nejen že obecné technologické znalosti normálního uživatele America Online nebyly dostatečné aby dodržoval základní internetový OPSEC (Operational Security – zajištění, že na internetu nezveřejňujeme citlivé informace, které by potencionálně mohly být použity proti nám), tehdy uživatele ani nenapadlo, že by někdo na internetu mohl být tak "zlý" aby vytvářeli dočasné přátelství za účelem vlastního zisku.

Zde by člověk mohl logicky vyvodit, že po situaci s AOHell by lidé pochopili nebezpečí phishingu a začali se o něm aktivně vzdělávat a bránit. Ať se toto stalo s tehdejšími America Online uživateli nebo ne, zde nastupuje problém lehce zmíněn v úvodu práce. V roce 1996 (kdy byl AOHell nejvíce používán) mělo v severní Americe (domovina AOL) přístup k internetu pouze 46,47 milionů lidí. Když porovnáme s rokem 2020 (poslední rok s dostupnými daty), kdy v severní Americe má přístup k internetu 480,34 milionů lidí, začínáme vidět problém. Lidé, kteří tehdy zažili AOL phishingové útoky, nebo vůbec věděli, že se něco takového stalo, je minimum, oproti extrémní internetové adaptaci, která probíhala v následujících letech. Faktory oblivňující hromadné povědomí ohledně phishingu budou probrány v následují kapitole o sociálním inženýrství.

Phishingové útoky nabrali na síle v roce 2001, zejména po tragédii 11. září. Tyto phishingové útoky byly prováděny více organizovanými skupinami, a narozdíl od AOHell cílili na banky, za účelem obohacení se. První phishingový útok na finančního operátora se stal v červnu 2001, útokem na firmu E-Gold, následován podobným útokem na stejnou firmu po 11. září.[6] V následujících 22 letech se objevil velký skandál ohledně phishingových útoků téměř každý rok. Mezi známější patří například útok na ukrajinskou energetickou firmu Kyivoblenergo v roce 2015 za pomocí BlackEnergy malware, za pomoci phishingového e-mailu, čímž způsobili výpadek elektřiny pro 225 tisíc Ukrajinců. [7] Další populární phishing útok byl v roce 2017 s pomocí NotPetya malware, který nejdříve napadl sítě na ukrajině, a poté se rozlezl do mnoha dalších zemí. Útok proběhl ve dvou fázích, první bylo využití chyby v aktualizační funkci účetnického softwaru, následovaného vlnou phishingový e-mailů. I když Petya malware (předchůdce NotPetya) byl navržen jako ransomware (malware šifrující soubory za účelem vydírání uživatele), NotPetya samotná byla pozměněna, aby neobsahovala adresu, na kterou má oběť zaplatit, tudíž se jednalo o čistě destruktivní malware. Celková odhadovaná škoda je extrémní, přes 10 miliard amerických dolarů. [8]

Od roku 2001 se počet phishingových útoků pouze zvyšuje. Mezinárodní konsorcium APWG (Anti-Phishing Working Group), které se snaží eliminovat podvody a krádeže identit způsobené phishingovými útoky ohlásili rok 2023 jako prozatimní nejhorší rok, co se týče počtu phishingových útoků, s téměř 5 miliony phishingových útok, z čehož 42.8% se odehrálo na sociálních sítích. [9]

1.2 Druhy phishingových útoků

Phishing jako samotný jsme si již nastínili, ovšem samotné phishingové útoky spadají do svých vlastních kategorií, charakterizovaných buď podle osob/organizací na které útočíme, nebo podle platformy, ve které phishingový útok provedeme. Mezi nejpoužívanější metody phishingových útoků patří:

1.2.1 Email phishing

Nejznámějším druhem phishingových útoků je email phishing. Email phishing je zasílání spamových emailových zpráv, ve kterých se různými způsoby snažíme z lidí dostat jejich osobní informace, pod záminkou falešné identity, například banka nebo sociální síť. Tyto phishingové emaily mohou obsahovat buď odkaz na hackerem vytvořenou přihlašovací

stránku, která se tváří jak ta reálná (neboli website spoofing), či mohou obsahovat infikované soubory (infikované spustitelné soubory, infikované MS Office dokumenty atp.). Hacker musí zajistit, že obsah emailu, emailová adresa, ze které email posílá a také jím vytvořená falešná stránka se co nejvíce podobá těm reálným. Tímto sníží šanci, že by si obět všimla jistých rozdílů, a odmítla vydat své informace. S vydanými informacemi hacker může naložit mnoha způsoby, od jednoduchého prodání cracklého (prolomeného) účtu například na dark webových marketech, k osobnímu využití cracklého účtu k dalšímu rozesílání phishingových zpráv (zde jde do akce již zmíněná nutnost zajistit autenticitu, pokud máme účet někoho jiného, nemusíme se snažit ten účet okopírovat). Dalšími možnostmi je instalace RATů (remote access trojan) pro distanční přístup do cracknutého zařízení, to ovšem poté vede přes rámec samotého emailového phishingu do jiných potencionálních kybernetických útoků. [10]

1.2.2 Spear Phishing

Spear phishing je druh phishingového útoku, kdy cíleně útočíme na určitou osobu či organizaci. Spear phishing oproti regulernímu emailovému phishingu je mnohem více personalizovaný. Hacker si musí vyhledat dostupné informace o osobě/organizaci, a vytvořit personifikovaný email za pomocí těchto nalezených informací. Obtížnost získávání těchto informací se liší podle důležitosti naší oběti, a na schopnosti oběti udržovat dostatečný, již zmíněný OPSEC. Čím lepší info hacker získá, tím větší šanci má přesvědčit obět. Další útok podobný spear phishingu je následující Whaling. [10]

1.2.3 Whaling a CEO Fraud

Whaling, jak by jméno mohlo napovídat (whale = velryba, v tomto kontextu “velká ryba” neboli někdo důležitý), je phishingový útok podobný spear phishingu, kde personalizovanými phishingovými maily útočíme na vysoce postavenou osobu v organizaci, zejména od manažerů výše. Idea whalingu je jednoduše “víše postavená osoba, větší risk útoku, větší zisk v případě úspěchu”. Časté metody sociálního inženýrství ve whalingových mailech jsou předvolání k soudu, či zákaznické stížnosti. [10]

CEO Fraud je podobný whalingu, ovšem role jsou otočené. Tentokrát se hacker vydává za vysoce postavenou osobu v organizace (například výkonného ředitele = CEO), a útočí na níže postavené osoby v organizaci. Často se hacker snaží přímo přesvědčit obět například k odeslání peněz jinam, v CEO Fraudu nejde tolik o získání osobních dat. [11]

1.2.4 Vishing (Voice phishing)

Druh phishingového útoku, využívající VoIP (Voice over IP, hlasová komunikace přes internet) místo emailových zpráv. Vishing v dnešní době probíhá často automatizovaně, za pomoci hlasových syntetizátorů hackeři nejen automatizují samotný průběh hovoru, také tím skrývají svůj reálný hlas (dobrý OPSEC, který stěžuje práci kriminalistů). Ovšem také často vishing probíhá tak, že hacker, ať už se svým normálním hlasem (například indičtí tech scameři) nebo se softwarem, co mění hlas (všemožné volně dostupné voice changery), provádí hovor osobně. Použití VoIP oproti emailovým zprávám má své vlastní optížnosti, hacker musí zaručit, aby číslo, které se zobrazí oběti, bylo podobné tomu, které hacker používá jako svoji falešnou identitu (toto se dělá za pomoci tzv. VoIP spoofing, služby, díky které hackeři jsou schopni zaměnit identitu čísla, ze kterého volají). [12] Vishing je velice populární forma phishingových útoků mezi hackery, APWG uvádí že v roce 2023 se počet vishing útoků zvyšoval každým čtvrtletím. [9]

1.2.5 Smishing (SMS phishing)

Smishing je phishingový útok, který používá telefonní SMS služby k rozesílání phishingových zpráv. Obsah SMS zpráv je většinou podobný, jako u email phishingu, tudíž jistá žádost buď kliknout na odkaz a přihlásit se, nebo kontaktovat přiložené číslo (na další vishing). Forma SMS zpráv hackerovi pomáhá z několika důvodů: pro člověka bude těžší ověřit celé telefonní číslo oproti emailové adrese, rozdíly mezi jednotlivými smartphone zařízeními mohou pomoci zamaskovat vloženou webovou stránku a také fakt, že lidé na telefonu nejsou tak opatrní, jako u počítače. Často se pro smishing používají tzv. burner phones, což jsou levné, jednorázové telefony se svým vlastním předplaceným číslem. Díky tomu se může hacker lehce zbavit důkazů, poté, co ukončí smishing útok, tak že zbaví svého burner telefonu. Dalším způsobem odesílání smishing útoku jsou email-to-text služby, které umožňují hackerovi odesílat SMS zprávy na telefony jako emailové zprávy z počítače. S tím ovšem také riskuje odhalení své identity, v případě že by byl na radaru kriminalistů, a ti by mohli soudně zažádat o veškerá data od oné email-to-text služby. [13]

1.2.6 Quishing (QR phishing)

Quishing je relativně nový druh phishingových útoků, který namísto regulerních URL odkazů na spoofované stránky, k tomuto účelu používá QR kódy. Využívají faktu, že QR kódy si populace uživatelů smartphonů oblíbila jako jednoduchou zkratku, jak se dostat na

různé stránky, či služby. Jedná se ovšem o dosti nebezpečný nástroj ve správných rukách. Díky QR kódům hacker nemusí hledat dostatečně blízkou doménu k té originální, kterou chce spoofovat. Dnešní smartphony sice ukážou adresu, na kterou QR kód vede, a zeptá se uživatele, zda opravdu na tuto stránku chce jít, ovšem často lidé tuto kontrolu ignorují. Co velice pomáhá quishingu, je způsob distribuce QR kódů. Lze je poslat normálními způsoby, jako je emailová zpráva, ovšem lidé si zvykli na QR kódy všude možně, tudíž hacker potenciálně může QR kódy nastražit kdekoliv na veřejných místech (například pod záminkou QR kódu pro přístup na veřejnou WiFi síť). [14]

1.2.7 Calendar phishing

Zajímavý druh phishingového útoku, kdy hackeři využívají kalendářní služby (například od Googlu), k odeslání phishingových kalendářových událostí za pomocí invitací. Snaží se mimikovat události, které by mohli nastat (narozeniny atp.) a většinou obsahuji URL na infikovanou/spoofovanou webovou stránku. Hackeři využívají defaultních nastavení nových účtů, například u Googlu, k odesílání daných invitací. Naštěstí i když se jedná o velice intruzivní způsob phishingu pro oběť (většinou si toho lidé všimnou, když náhodně otevřou kalendář, a místo prázdného kalendáře vidí extrémní množství událostí, které nikdy nečekal), je také velice jednoduché se tohoto útoku zbavit, změnou nastavení aby kalendář neakceptoval invitace od neznámých kontaktů. [15]

1.3 Jak se bránit po stránce technické

I když se phishing jako samotný specializuje na sociální inženýrství a manipulaci s lidmi, máme pořád způsoby jak phishingové útoky zastavit, než se dostanou k lidskému článku, nebo v případě že lidský článek selhal, máme způsoby jak minimalizovat/nulovat škody. Tyto způsoby nemusí pracovat jako samostatné bezpečnostní články, spíše naopak: kombinací těchto bezpečnostních prvků jsme schopni zaručit co největší technickou bezpečnost před phishingem. Tyto způsoby technické ochrany je dobré popsat si podrobněji, protože v praktické části práce se je budeme snažit co nejlépe obejít.

1.3.1 Filtrování phishingových emailů

Nejčastějším způsobem ochrany proti phishingovým mailům je využití spamových filtrů. Spamové filtry používají několik různých metod k analýze emailu, jeho obsahu a odesílatele, podle kterých rozhodnou, zda email splňuje požadavky k příjmu do normální doručené pošty, spamového koše, nebo rovnou vymazat a nepřijímat. Lepší (a dražší) spamové filtry

obsahují více těchto metod v konjunkci, čímž získávají větší úspěšnost zastavení spamového emailu. Spamové filtry mohou provádět:

1.3.1.1 Analýza obsahu

Filtry v první řadě jsou schopné analyzovat obsah emailu, a vyhledat velice časté typy spamu (například legendární Nigerian prince scam) díky faktu, že tyto časté druhy spamu často neobměňují obsah mezi pokusy. Problémy mohou nastat v případě, že by filtr vzal legitimní obsah a označil jej jako spam, což se může stát, pokud legitimní email obsahuje podobné znění spamovým emailům (nabídky něčeho zadarmo, reklamy atp.). [16]

1.3.1.2 Analýza hlavičky

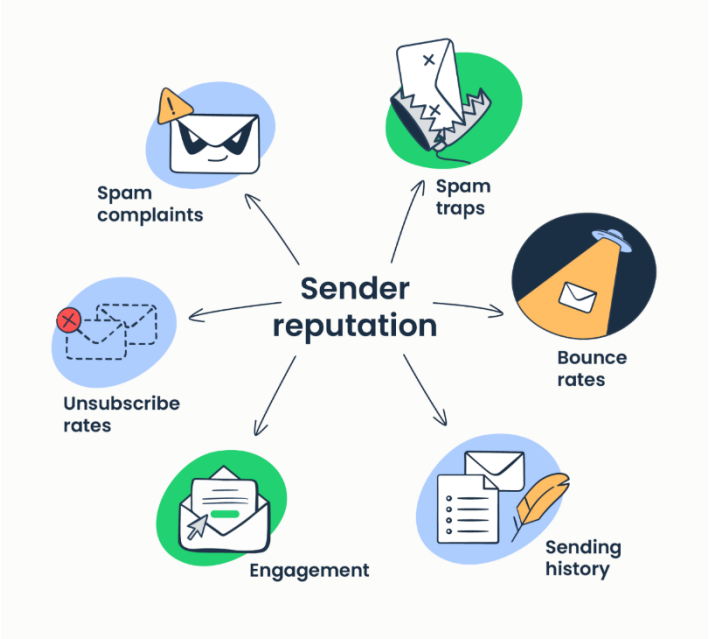
Filtr analyzuje metadata emailu, a hledá v nich nesrovnalosti a falsifikace, například zda doména, která odeslala email, odpovídá jiné, již známé doméně, nebo zda se hacker snaží falsifikovat doménu, za účelem oklamání oběti (například google.com na googIe.com). Zkoumá také kolika lidem byl daný email poslán, čím více lidí zadaných jako příjemci, tím větší šance, že by email mohl být spam. Problémy s analýzou hlavičky emailu začínají v případě, kdy hlavička emailu je správná, filtr poté neví, zda se jedná o legitimní email, nebo zda hacker úspěšně spoofoval svůj email (při email spoofingu se hacker snaží co nejlépe obměnit hlavičku svého spamového emailu aby odpovídala emailu legitimnímu). [16]

1.3.1.3 Blacklisty a Whitelisty

Filtry mohou využívat blacklistů ke kompletnímu zákazu příjmu emailů z IP adres vložených do daného blacklistu, a stejně tak whitelisty pro domény, u kterých je (téměř) 100% šance, že se jedná o legitimní IP adresu, která potřebuje čistou linku komunikace s námi. IP emailové blacklisty jsou v dnešní době velice rozsáhlé, existuje jich mnoho, a někteří jako například BarracudaCentral, nabízí online službu, do které zapíšete IP, a získáte odpověď, zda je dané IP obsaženo v blacklistu. Problém s blacklisty je v případě, když hacker, jehož IP je na blacklistu, změní doménu/IP, čímž do té doby, než tuto novou doménu někdo přidá do blacklistu, je hackerova doména opět brána jako legitimní. [16]

1.3.1.4 Reputace odesílatele

Filtry také analyzovat tzv. reputaci odesílatele, podle předchozích zkušeností s jeho IP adresou a použitou doménou. IP reputace funguje na principu, že každý ESP (email service provider) nabízí uživatelům několik IP adres, které mohou používat na rozesílání emailů. Ovšem tyto adresy jsou většinou sdílené mezi mnoha uživateli, pokud si zákazník nepřiplatí za svoji vlastní. Je velice nutné pro hackera, aby si udržoval co nejlepší IP reputaci, protože ty IP adresy s reputací špatnou většinou bývají odmítnuté mailovými servery. Doménová reputace funguje na podobném principu. Pro hackera je nutné, aby obě tyto reputace držel v pozitivních hodnotách, protože bez jedné z nich by měl pořád problémy s odesíláním svých phishingových mailů. Hlavní faktory, co ovlivňují reputaci odesílatele, jsou: jak často je mail označen jako spam, "Bounce rate" neboli počet emailů, které nebyly schopny bejt odeslány, jak často odesílatel rozesílá emaily (čím více, tím větší šance nekalých úmyslů), a nové pro mě: jak často se příjemce odebral od odběru mailů (v případě, že se jedná o mailing list, and příjemce má možnost odebrat se z tohoto listu). Zajímavým způsobem snižování reputace uživatele jsou tak zvané Spam traps, což jsou prakticky spamové honeypoty, většinou založené internetovými providery, nebo jinými proti-spamovými organizacemi. V případě, že by odesílatel rozeslal svůj spamový email na adresu daného honeypotu, bude automaticky přidán do blacklistu. Je mnoho způsobů jako takový honeypot vytvořit, populární možností je vkládat náhodné emailové adresy po webech, kde je jistota, že hacker scrapuje (škrabe – stahuje veškerá data ze stránky a bere z nich užitečné informace, jako například emailové adresy) pro data, protože je velká šance, že nalezené emailové adresy využije ke spamu, a bude chytnut v honeypotu. [17]



Obrázek 2 Faktory ovlivňující reputaci odesílatele

1.3.1.5 Machine learning algorithmy

Pokročile emailové filtry využívají algoritmů strojového učení k vytvoření co možno nejlepší spamovou ochranu. Nejpoužívanějším (a nejúspěšnějším) z těchto algoritmů je naivní Bayesovské učení, a filtry co jej používají se nazívají Bayesianské filtry. Největšímy výhodamy těchto filtrů je možnost přizpůsobit svým požadavkům díky různým použitelným datamodelům, a také celková úspěšnost filtrů používajících machine learning algoritmy, například Bayesianské filtry mají uváděnou 98% úspěšnost. [16] Ovšem i přes tuto vysokou úspěšnost nejsou tyto filtry perfektní. Mají problémy s rozeznáním spamu u cizojazyčných emailů, mají problém detekovat schválně změněná slova (například google a g00gle), a Bayesianské filtry jsou náchylné na tzv. Bayesiánské otrávení, kdy hacker schválně přidává do obsahu emailu slova, která by se normálně neobjevila ve spamových emailech, čímž rozhodí Bayesianský filtr, který si již nebude tak jistý, že se jedná o spam. Machine learning spam filtry používá například Google ve své emailové službě Gmail. [16]

1.3.1.6 Source authentication

Source authentication filtry zkoumají, zda doména odesílatele má nastavené autentifikační protokoly: SPF (autorizační ověření IP adresy odesílatele), DKIM (ověření autenticity domény, zkoumá zda email, co přišel, se schoduje s emailem odeslaným), a DMARC (další bezpečnostní ověřování, možnost nastavit, jak s mailem bude nacházeno v případě, že by byl

odmítnut: buď neudělá nic, nebo provede "karanténu", ve které email přijme ale pošle ho přímo do spamu, nebo email kompletně odmítne a smaže). Správně nastavený DMARC také znemožní hackerovi spoofovat doménu, pro kterou je DMARC nastaven. [18] / SPF a DKIM jsou v tomto případě nutné, DMARC funguje jako nadstavba pro tyto předchozí. Je i nový autentifikační protokol BIMI Record, který funguje na principu vložení vizuální reprezentace firmy (například logo), jako textový DNS zápis. Díky tomu, vždy kdy by přišel email s dané firmy, uživatel by si to ověřil velice jednoduše podle daného loga. BIMI Record se velice špatně spoofuje, každá firma, co jej používá se musí zaregistrovat, dostat certifikát a udržovat dobrou reputaci. [19] Kvůli tomu se source authentication filtry soustředí na předchozí tři autentifikační protokoly. [16]

1.3.2 Endpoint softwarová ochrana

V rámci technické ochrany před phishingovými útoky je nutné používat dostatečně účinnou endpoint ochranu. Endpointem se v tomto případě myslí veškerá koncová zařízení, která by mohla být afektována phishingovým útokem, tudíž jakýkoliv počítač, smartphone, IoT atd.. Endpoint software může být antivirus, antimalware, antiransomware a podobné bezpečnostní prvky, prodávané jako jeden velký security suite od velkých bezpečnostních firem, jako například Avast, či Eset. Tato endpoint ochrana je prakticky poslední linií obrany, protože většinou jde do akce v případě, že lidský článek selhal rozeznat phishingový útok, a nějakým způsobem s útokem interagoval. Často phishingové útoky obsahují zamaskované infikované soubory či URL na stránky, které po spuštění by mohli koncové zařízení (nebo v horším případě, celou síť) infikovat celou řadou malwarů. Tomuto přesně se snaží endpoint softwarová ochrana zabránit, většinou automatickým zablokováním infikovaného souboru/stránky a jeho karanténou. Je extrémně důležité, aby endpoint security byla pravidelně aktualizována, abychom předešli případům, kdy security neodchytí infekci, protože neměla nejnovější seznam nálezů. I když nám tato ochrana pomůže zamezit katastrofickým scénářům, jako napadení malwarem, nijak nám nepomává s obranou lidského článku. Například v případě, že by hackerovi šlo pouze o osobní informace, je schopen je získat phishingovým útokem bez toho, aniž by spustil endpoint ochranu. [20]

1.3.3 Ochrana webových prohlížečů

Moderní webové prohlížeče, jako Chromium či Firefox, obsahují vprogramované phishingové a malwarové detekční funkce. Tyto detekce fungují tak, že pokud by oběť klikla na spoofovanou stránku z phishingového emailu, podle určitých kritérií je prohlížeč schopen

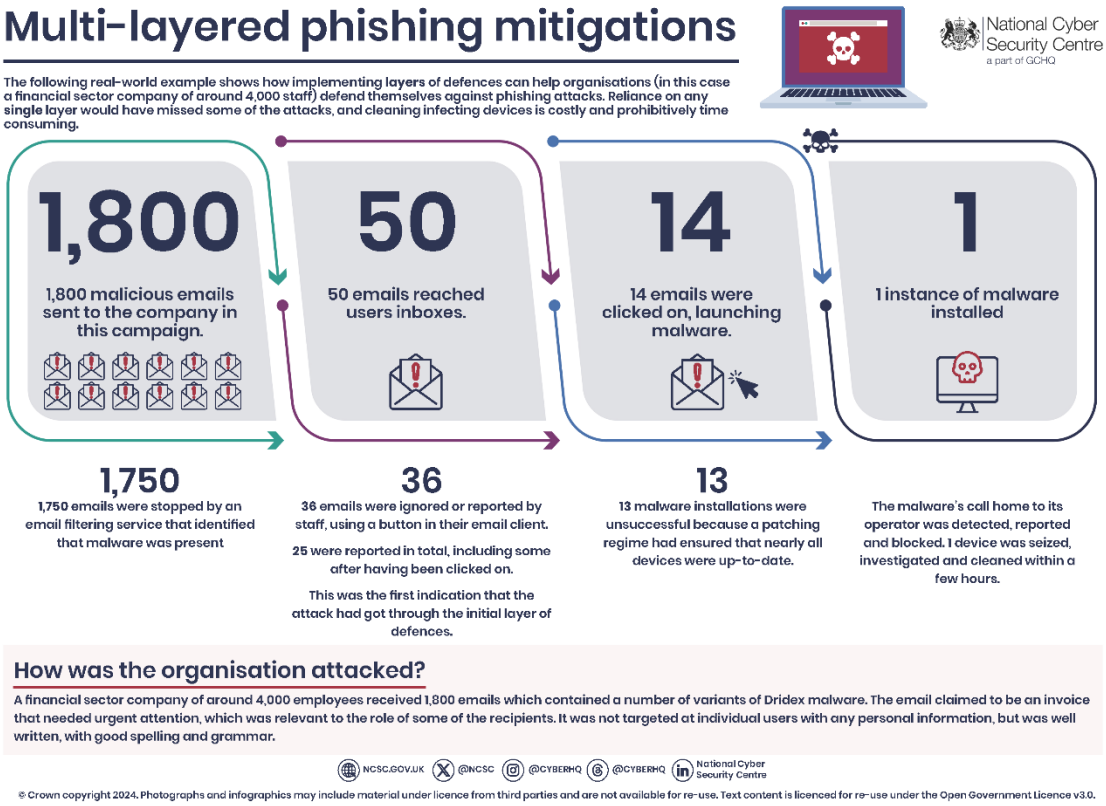
odhalit že by se mohlo jednat o nebezpečnou či deceptivní stránku, a místo načtení zobrazí varovnou zprávu s vysvětlivkou o typu potenciálního nebezpečí dané stránky. Uživatel i přes to má možnost na zablokovanou stránku pokračovat, tudíž je na něm a jeho technických znalostech zda upozornění bude poslouchat, nebo pokud na stránku bude pokračovat.

1.3.4 Multifaktorová autentifikace (MFA)

Další možností technické ochrany, která opět pomáhá spíše až poté, co selže lidský článek, je multifaktorová autentifikace. Principiálně se jedná o generátor tzv. one-time passwords (OTP, heslo na jedno použití), které se zapisují po zapsání jména a hesla při přihlašování, a automaticky se mění po určitém čase. Vždy je k účtu připojen pouze jeden autentifikátor, a je uložen na zařízení, které vlastník účtu má vždy po ruce (ve většině případů mobilní telefon, existují ovšem i MFA klíčenky). MFA také může být automatizována ze strany providera služby bez nutnosti mít určitou MFA aplikaci, například Microsoft a Google mají možnost rozesílat OTP ve formě SMS zpráv nebo emailových zpráv (což ovšem v sobě nese své vlastní bezpečnostní problémy). Ochrana MFA vyplívá z faktu, že i kdyby oběť vydala údaje ke svému účtu, hacker nebude schopen se do účtu přihlásit bez daného OTP. Nejedná se ovšem o 100% ochranu, existují způsoby jak MFA obejít, či prolomit. Nejjednodušším je další, navazující phishing útok, ve kterém se hacker bude snažit získat uživatelovo OTP. Pokud by se mu to povedlo, je schopen se přihlásit do účtu, a MFA přiřazeno k danému účtu vypnout, čímž získá plný přístup kdykoliv. Dalšími možnými útoky jsou například Consent phishing, kdy za pomoci otevřené autorizace (OAuth) si hacker může zažádat o přístup, aniž by potřeboval OTP [21], či SIM swapping, kdy hacker provede phishingový útok na telefonního providera, u které má oběť účet, a snaží se přesvědčit providera, že on je pravý vlastník daného čísla, aby přenesli číslo na jeho SIM kartu, čímž získá přístup k OTP které chodí SMS zprávami. [22]

Jak lze vidět z představených technických bezpečnostních prvků, máme velké množství způsobů, jak se phishingovým útokům bránit, ovšem každý tento prvek má své vlastní nevýhody, které napomáhají potencionálním kriminálníkům je obejít. Naprosto základní nutná ochrana je alespoň nějaký druh emailové filtru proti spamu. Pokud se ovšem bavíme o celé organizaci, která se snaží předejít phishingových útoků, bude potřeba aby začali využívat těchto prvků co nejvíce, a kombinovat je. Například již zmíněná multifaktorová autentifikace by měla být zapnuta pro všechny účty, které ji podporují, nejen například pro emailový účet. Kombinace všech odstupných technických ochran do vrstev nám zaručuje co

nejmenší šanci na úspěšné phishingové útoky. Tímto jsme probrali možnosti technické ochrany vůči phishingovým útokům, jak ale již víme, phishing sám o sobě není cílen na chyby technické, ale chyby lidské.



Obrázek 3 Infografika zobrazující využití technických bezpečnostních prvků ve vrstvách k minimalizaci škod v případě phishingového útoku

1.4 Jak se bránit po stránce lidské

Hlavním cílem phishingu, je přesvědčit lidskou osobu dostatečně, aby provedla, co chceme. Jak by se měl tedy člověk bránit? Znamená to, že již nikdy nebudeme moct věřit druhému člověku na internetu? Naštěstí ještě nežijeme ve světě, kdy by tato situace byla tak černo-bílá, alespoň prozatím (více v kapitole o umělé inteligenci).

Nejlepším způsobem jak se pro člověka bránit od phishingových útoků, je mít technologické zkušenosti o tématu phishingu, vědět na co dávat pozor v případech možného phishingu, a hlavně, tyto zkušenosti využít v praxi, nejen pro sebe ale také lidi okolo, kteří stejné znalosti nemuseli mít. Jednoduše řečeno, nejlepší způsob prevence ze strany uživatelů je edukace, a trénink. Lidmi jsou v této kapitole myšleny osoby, které nemají dostatečné technologické znalosti, například vůbec neví, že phishing existuje (ať už kvůli věku, vyrůstání bez

technologií jako internet, celková nevraživost vůči technologiím a podobné důvody), nebo o hrozbě phishingu zaslechli, ale neví jak jej vypozorovat. Předpokládám, že opačný druh lidí, co regulerně používají internet a zajímají se o technologie by byly schopni buď phishing rozeznat rovnou, nebo si najít známky jak jej rozeznat.

Edukace lidí ohledně phishingu záleží na situaci, ve které se nacházejí ve spojení s citlivými daty. Pokud se jedná o jednotlivce, který například používá pouze smartphone, a nejvíce ceněná data jsou jeho přihlašovací údaje do internetového bankovnictví, spadá nutnost edukace o phishingu pouze na něj. Buď bude mít to štěstí, že ho o existenci phishingu obeznámí kamarád/zprávy/internet, a nebo tu smůlu, že si phishing poprvé zažije na vlastní kůži. Tato demografie lidí, bez pravého technického know-how, je nejvíce náchylná na regulerní spamové phishingové útoky, tudíž ty, které jsou spamovány velkému množství lidí bez velké personalizace.

Pokud se ovšem bavíme o člověku, kterému chybí technické znalosti, ale například získal práci ve firmě, kde je nucen pracovat s počítačem, internetem a citlivými daty, nutnost tuto osobu vyučit v hrozbách phishingu spadá na zaměstnavatele. Přece jen zaměstnavateli jde o ona citlivá data více, než samotnému zaměstnanci. Zaměstnavatel by měl nejen zajistit dostatečnou edukaci ohledně phishingu pro všechny zaměstnance, co jakým koliv způsobem mají přístup k datům (většinou ovšem tuto edukaci dostávají všichni ve firmě), je ohromné plus, když zajistí také dlouhodobý trénink ve formě penetračních phishingových kampaní. Samotná edukace je velice důležitá, aby zaměstnanec pochopil základy phishingu, v jakých případech by se mohl stát obětí takového útoku, a také jak vypozorovat nejčastější známky phishingu. Upřímně si myslím, že edukace ohledně phishingu by měla zdůraznit způsoby, které kriminálníci používají k obejetí technických security prvků (znázorněno v předchozí kapitole), protože by to člověku mohlo usnadnit pochopení jistých známek, značících, že email, či podobná zpráva je phishing (přesněji: když člověka naučíme, že hacker musí měnit náhodná slova ve phishingovém mailu na něco podobného, protože se snaží obejít spam filter, je šance, že si člověk spojí tuto korelaci s divnými emaily, co již dostal, a začně být o to více obezřetný). Po základní edukaci většinou následuje krátký test, obsahující cvičné zprávy, ať již emailové, SMS zprávy, či z instantních messengerů jako WhatsUp nebo Discord, které zaměstnanec musí prohlédnout, a podle bodů, které se naučil v edukaci, správně určit zda se jedná o phishing, nebo legitimní zprávu. Takovýto test je skvělý způsob jak zjistil, kolik si toho zaměstnanec odnesl z phishingové edukace, ovšem pokud bychom skončili s edukací/testováním v tomto bodě, riskujeme že časem zaměstnanci buď

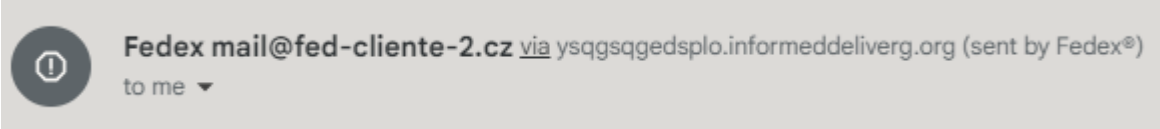
zapomenou všepřítomnou hrozbu phishingu, nebo snad hůř, začnou si myslet, že se nejedná o tak velký problém, že by se jim to nikdy přihodit nemohlo, a začnou si být příliš jistí. Osobně jsem musel interagovat s lidma to chytli tuto myšlenku, je to velice nebezpečná mentalita nejen z důvodu, že se jim to velice špatně rozmlouvá (většinou se jednalo o obecnou tvrdohlavost), ale občas se snaží toto myšlení přenášet na druhé. Bohužel z toho důvodu existují individua, kteří odmítají tyto koncepty kybernetické ochrany, do té doby, než se přímo jim něco přihodí. Zde je vidět, že se bohužel můžeme snažit co nejvíce chceme lidi edukovat, ale někteří lidé zůstanou v této mentalitě. Tudíž musíme počítat s tím, že ať máme sebelepší metody edukace a tréninku ohledně phishingu, nikdy nejsme schopni získat 100% úspěšnost zastavení phishingových útoků, jakmile se dostanou do fáze samotné lidské interakce.

Z tohoto důvodu edukaci a trénink musíme pravidelně opakovat. Nejlepší by bylo opakovat tento process co nejčastěji, nejlépe pro každého nového zaměstnance v organizaci, ovšem realita situace je, že přece jen někdo musí připravit dostatečnou edukaci, přídavné testy, případně se osobně věnovat lidem, kteří v tématu tápou, a toto všechno stojí peníze. Dalším faktorem, který nám preventuje opakovat tento proces například každý týden, je pracovní morálka. Z osobní zkušenosti vím, že regulerní zaměstnanci podobné edukace a testování nesnáší, protože jim to buď bere čas z regulerní práce, nebo z volného času. Opakováním edukace příliš často bychom nejen snížili potenciální pracovní morálku zaměstnanců, je zde velká šance, že přesně kvůli tomuto opakování by přestali na téma phishingu dávat dostatečnou pozornost, což je přesně to, čemu se edukací snažíme zabránit. Proto nejlepším kompromisem je pravidelná edukace (v rámci možností, a abychom neotravovali zaměstnance moc často), provádět cvičné phishingové kampaně v mezičase.

Způsobů jak odhalit phishingový útok má člověk mnoho. Nás především zajímají známky phishingových emailů, ovšem mnoho těchto nesrovnalostí, co může člověk zpozorovat se opakují v ostatních phishingových útocích na jiných platformách. Je důležité si uvědomit, že ne všechny phishingové útoky jsou stejné kvality. Je možné dostat extrémně kvalitní phishingový útok, který je téměř neprokouknutelný, stejně jak je možné dostat phishingový útok, který člověk pozná při prvním nahlédnutí.

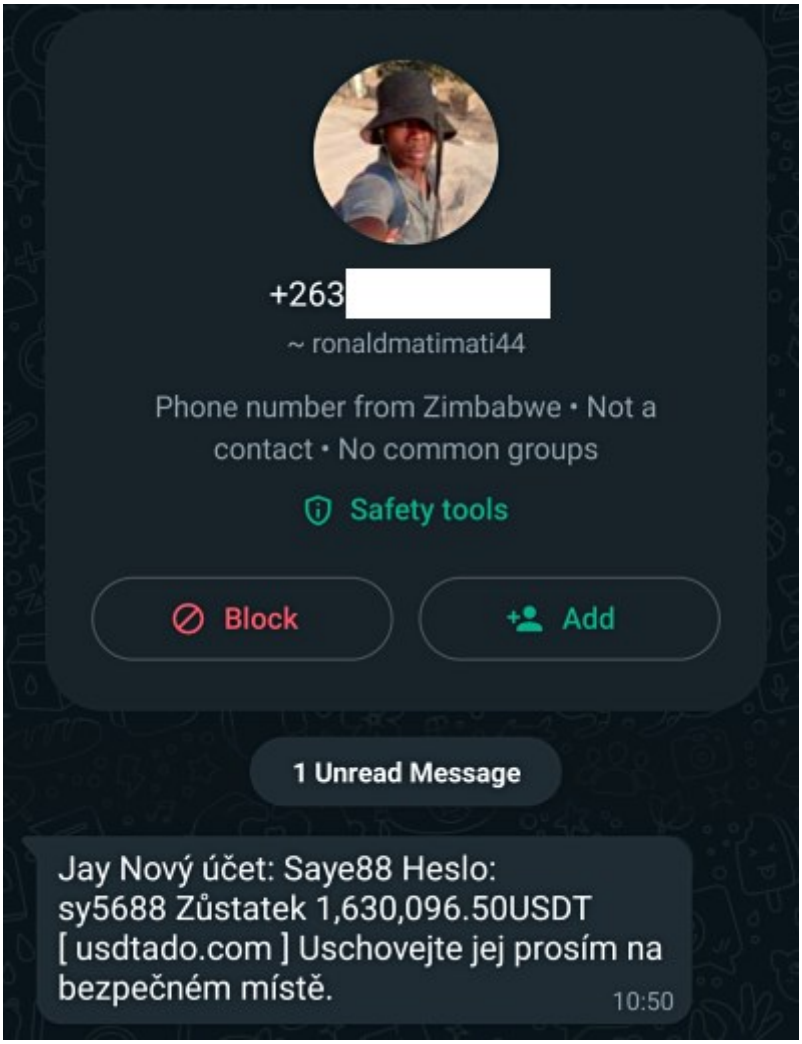
1.4.1 Kontrola odesílatele

První místo, kde by měl člověk hledat známky phishingového útoku v emailu, je odesílatel, a korelace mezi odesílatelem, a obsahem samotného emailu. Samotná emailová adresa je často vytvořena tak, aby co nejvíce odpovídala potenciální reálné adrese, která by mohla tento mail poslat (neboli domain spoofing). Dá se předpokládat, že adresa odesílatele bude ve většině případů obsahovat gramatickou chybu, zejména v doméně. Někdy se ovšem hackeři ani nesnaží dostatečně zkopírovat danou adresu, což vede k lehce prokouknutelným phishingovým útokům.



Obrázek 4 Příklad lehce prohlédnutelného phishingového útoku

Korelací mezi odesílatelem a obsahem emailu, je myšleno, zda dává smysl, že by daný obsah emailu byl odeslán daným odesílatelem. Například, pokud by jste dostali phishingový mail z Gmail domény, s obsahem že Váš Paypal účet byl zablokován, nedávalo by smysl, aby Vám tuto informaci posílal "Paypal" z Gmail domény, a ne ze své Paypal domény. Hackeři často využívají faktu, že člověka spíše zajímá ona urgentní zpráva v těle emailu, a ne samotný odesílatel. Podobná kontrola odesílatele může být provedena i na sociálních sítích. Člověk má v tomto výhodu, pokud je schopen plně ovládat sociální síť na které se nachází. Je tím schopen vyhledat o odesílateli různé informace, které by potenciálně odesílatel chtěl nechat anonymní. Například na facebooku je obrovské množství phishingových útoků, ve kterých lidé s jižní Asie předělávají své osobní profily na profily modelek, a rozesílají tak phishingové zprávy. Ovšem pokud člověk ví jak facebook funguje, může kliknout na profil odesílatele, a nejen že URL jeho profilu bude obsahovat jeho originální jméno před změnou, občas si jsou tak jistí (nebo líní), že ani nezamknou profil a nesmažou staré příspěvky, takže můžete prohlídnout jejich historii přípěvků, a zjistit jak dvacetiletá modelka je ve skutečnosti čtyřicetiletý indonéžan.



Obrázek 5 Phishingový útok, s jasným cizojazyčným jménem a telefonním číslem ze Zimbabwe

1.4.2 Kontrola obsahu

Dále se přesuneme na obsah samotné phishingové zprávy. Zde můžeme pokračovat v metodách, které jsme se zmínili v kontrole odesílatele. Zde se soustřeďujeme na gramatiku samotného textu, než se přesuneme na URL. Záleží na kontextu, v jakém danou zprávu dostáváme. Pokud jsme češi, a předpokládáme, že budeme dostávat zprávu v češtině, jisté gramatické chyby, které by nativní mluvčí neudělal se lehce vyhledají, často i je jasné z textu, že se jedná o špatný strojový překlad. V případě anglických zpráv je to trochu těžší, protože příjemce zprávy sám nemusí být tak dobrý v angličtině, aby vychytal gramatické chyby, ale hackeři jsou také schopni sehnat znění oficiální zprávy od určité organizace, a pouze změnit přiložená URLs nebo přílohy. V tomto případě je dobrým nápadem sledovat

oslovení, které bývá u oficiálních zpráv od organizací na začátku zprávy. Pokud příjemce nemá nikde nastaveno, aby se u jeho účtu, na který zprávu dostává, nezobrazovalo jeho reálné jméno, poté mají hackeři těžší čas jej zjistit a použít v rámci phishingového útoku. Je sice možné, že si reálné jméno najdou jinými způsoby, ale často se ani nesnaží, a jdou na to oklikou. (Například: oficiální email od banky by začínal oslovením "Pane Strnade", protože i když nemám nastaveno zobrazení reálného jména, má banka ví kdo jsem. Pokud hacker nemá jinou možnost jak zjistit reálné jméno, často se snaží toto oslovení obejít, například "Milý pane".) Samozřejmě pouhá gramatika, či oslovení nemusí znamenat, že se na 100% jedná o phishingový útok, proto se přesuneme na nejdůležitější část kontroly obsahu.

Při phishingovém útoku po Vás dotyčný odesílatel vždy něco chce. V těžké většině případů, se jedná o prosbu kliknout na přiložený odkaz a přihlásit se do stránky (která je hackerem spoofovaná), nebo otevřít přiloženou přílohu. V prvním případě koukáme na přiložený URL zkoumáme, zda se liší od reálné stránky. Ve většině případů se hackeři snaží schovat URL za grafický element, například tlačítko (což dělají i zprávy legitimní), a tudíž je potřeba na daný element přejet kurzorem, načež se URL zobrazí po levé spodní straně obrazovky (samozřejmě na URL neklikáme). Získání dobrých (podobných originálu) domén je velice drahé, a tudíž pokud na Vás neútočí celá hackerská organizace, výsledné URL bude rozeznatelné od pravého. Občas hackeři používají na schování lehce prokouknutelných domén tzv. URL shortenery (služba, která převede dlouhé URL do krátkého s doménou oné služby), které sice mohou být použity k normálním účelům, ale ve phishingu je to populární způsob jak zvýšit šanci úspěšného útoku. Osobně bych jakékoliv URL od těchto služeb bral automaticky jako pokus o nějaký phishingový útok.

Phishingové útoky s přílohami jsou též velice nebezpečné, protože se z jednoduchého podívání často nedá přesně zjistit, zda přiložený soubor je bezpečný, či ne. Druhů infikovaných souborů používaných při phishingových útocích je mnoho, od klasických spustitelných souborů, které se vydávají za obrázek, či dokument, k infikovaným Microsoft Office dokumentům. Infikované MS Office dokumenty byly takový problém, že Microsoft přidal funkci do Office Suite, kde kdy byste otevřeli infikovaný dokument, MS Office má zakázáno spustit jakýkoliv kód, který by mohl být obsažen v dokumentu, do té doby, nežli člověk potvrdí, že je dokument bezpečný, a ochranu vypne. Samozřejmě se najdou případy lidí, kteří automaticky odkliknou ochranu, aniž by se jakkoliv ujistili, ovšem celkově tyto typy útoků s MS Office dokumentama mají nízkou úspěšnost.

2 SOCIÁLNÍ INŽENÝRSTVÍ (SOCIOTECHNIKA)

2.1 O co se jedná?

K pravému pochopení tématu phishingu se musíme seznámit s hlavní "zbraní", kterou hackeři používají proti jejich obětem. Sociální inženýrství (také známé jako sociotechnika) je pojem sdílející dvě různé definice podle kontextu, ve kterém je užit. Jako první se dá sociální inženýrství brát z ohledu sociologického. V rámci sociologie se jedná o praktickou společenskovědní disciplínu, ve které se na základě vědeckých poznatků vytváří relevantní systém praktických direktiv a doporučení k dosažení společenských změn, nebo (v užším pojetí) změn v chování a jednání jedinců nebo skupin. [23] Tento typ sociálního inženýrství je především používán v oblastech ekonomie, práva, demografie a politické praxe. Nás ovšem zajímá druhé pojetí sociálního inženýrství, z hlediska bezpečnostního. Z tohoto pohledu sociální inženýrství znamená záměrnou manipulaci lidí za účelem přinutit je provést jistou akci, která povede z jistému obohacení "inženýra", který lidi manipuluje. Toto obohacení nemusí být pouze peněžité, ve většině případů se jedná o získání jistých, normálně tajných informací, které by oběti bez manipulace nebyly ochotné vydat (například přihlašovací údaje k účtu). Tato definice by se teoreticky dala označit také za "podvod", tudíž bylo potřeba vymyslet distinkci mezi těmito dvěma ději. Například jedna taková definice, která se snaží rozlišit sociální inženýrství od podvodu označuje sociální inženýrství jako "jakýkoliv akt, který donutí osobu udělat akci, která by mohla či nemusela být v jejich nejlepším zájmu". [24] Jak z těchto definicí vidíme, jsou si kromě chtěného výsledku podobné, a i bezpečnostní sociální inženýrství využívá mnoha termínů a mechanik ze sociologického sociálního inženýrství, protože se v obou případech prolíná věda sociologická, filozofická a především psychologická. Vědu psychologickou zde vyzdvihuji, protože čím více o ní člověk ví (a tudíž ví, jak s lidmi nejlépe manipulovat), tím úspěšnější bude v útocích používající sociální inženýrství, jako například phishing. V následujících kapitolách se budeme bavit pouze o bezpečnostním sociálním inženýrství.

2.2 Funkcionalita sociálního inženýrství

Sociální inženýrství využívá psychologického jevu, který znázorňuje specifické způsoby lidského rozhodování, známé jako kognitivní chyby úsudku (cognitive bias, také známo jako kognitivní zkreslení). Kognitivní chyby úsudku jsou systematické odchylky z normálního myšlení, či racionality v rozhodování, které vedou k chybným úsudkům. Lidé si vytváří svou

vlastní subjektivní realitu podle informací, které přijímají. Tato vytvořená realita ovlivňuje jak se chovají, či jak vnímají různé věci. Kvůli tomuto jevu se mohou lidé chovat nelogicky, stereotypně a iracionálně. Důležitým faktem kognitivních chyb úsudků je, že se nejedná o vědomé chování, sám člověk si není vědom toho, že tyto chyby dělá. Z tohoto důvodu se nejedná o záměrné chování, ale o výsledek chybného myšlenkového procesu, při němž dotyčný předpokládá/odmítá nějaký výsledek a svá pozorování proto interpretuje zaujatě. [25] Samotných kognitivních chyb úsudku je mnoho a dají se dělit do kategorií podle části myšlení, které je zkresleno na sociální, paměťové a rozhodovací chyby úsudku/zkreslení. Mezi časté kognitivní chyby úsudku patří například:

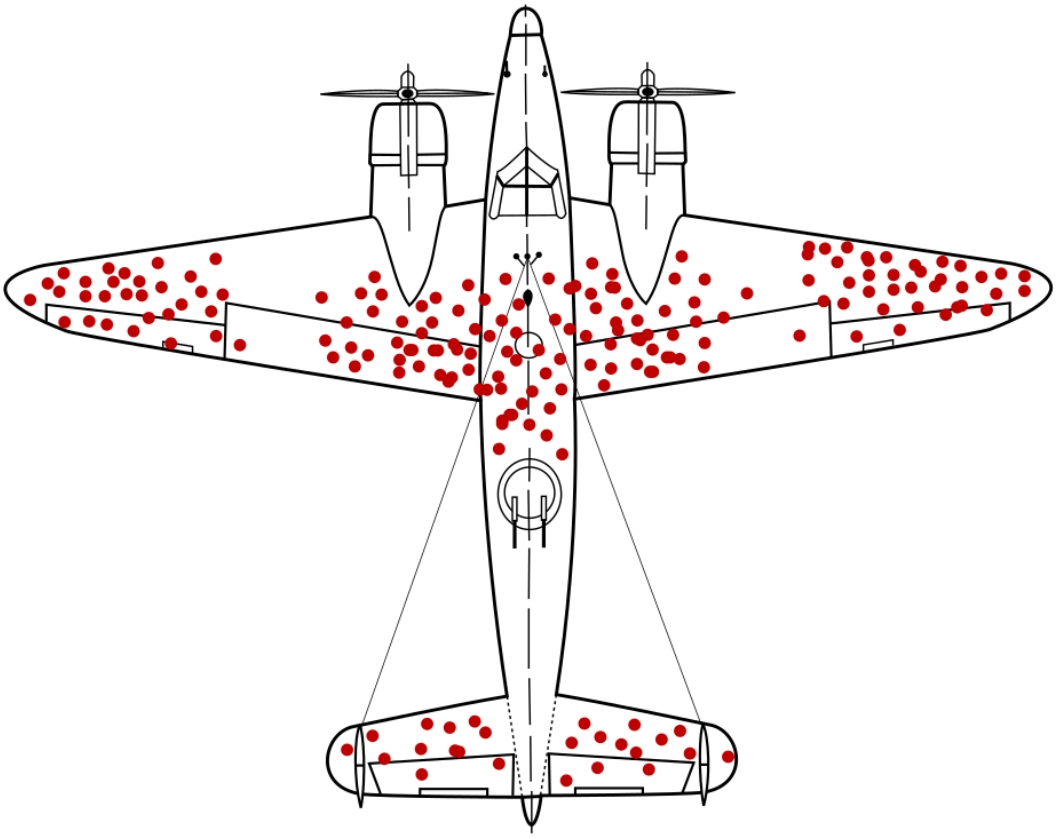
Dunning-Krugerův efekt: Možná nejznámější kognitivní chybou úsudku je Dunning-Krugerův efekt. Ten nám značí, že nekompetentní lidé nejsou schopni pochopit, že jsou nekompetentní, protože nejsou schopni pochopit, co je pravá kompetence. Toto vede k efektu, kdy tito nekompetentní lidé paradoxně přeceňují svou vlastní kompetenci. Je zde i možnost opačného efektu, kdy velice kompetentní jedinci si myslí že jsou velice nekompetentní. Toto je známé jako syndrom podvodníka (imposter syndrom), kdy si kompetentní jedinec myslí, že je podvodník, který si nezaslouží životní pozici, ve které se nachází. [26]

Halo effect: Poněkud chybně přeloženo jako haló efekt, přestože halo v tomto případě znamená svatozář. Halo efekt značí chybu úsudku, kdy je člověk příliš ovlivněn první dojmem z jiného jedince. Tento první dojem může být kladný i záporný, ovšem člověk s halo efektem bude brát tento dojem za velice důležitý v následující interakci s tímto protějškem. [27]

Status quo bias: Pojmenován podle latinského výrazu status quo, znamenajícího "stávající stav". Osoby s touto chybou úsudku preferují stav věcí tak, jak zrovna jsou, a odmítají jakékoliv změny tohoto stavu. Důvodem je, že v jakékoliv změně stavu nachází nejhorší možné katastrofické scénáře, které by se mohly při změně stát. [27]

Klam přeživších: Survivorship bias v angličtině. Klam přeživších je kognitivní chyba úsudku, při které se osoby soustřeďují na věci a lidi, které/kteří něco "přežili". Nejlepším příkladem klamu přeživších je známý obrázek, zobrazující letadlo a zvýrazněné body na něm, které byly nejčastěji poškozeny. Tímto znázorněním se inženýři snažili o zlepšení obrany na onom letadle právě v těchto bodech, protože si mysleli, že tento příklad by fungoval pro všechny letadla. Problém a samotný klam přeživších v tomto případě vychází

z faktu, že jedinný důvod, proč měli data o místech poškození, byl ten, že přeživší, nesestřelená, ale poškozená letadla byla schopná se vrátit zpět na základnu, a tudíž přežily, zatímco mnoho dalších letadel bylo v akci sestřeleno. Z těchto sestřelených letadel není způsob jak získat tato potřebná data, proto graf poškození bral v potaz pouze přežitá letadla. [28]



Obrázek 6 Příklad klamu přeživších na obrázku přeživších letadel, zvýrazňující místa, kde byla nejvíce poškozena

Potvrzovací zkreslení: Anglicky confirmation bias, je kognitivní chyba úsudku, při které člověk značně preferuje informace a vzpomínky, které podporují jeho osobní názor/víru, zatímco s informacemi a vzpomínkami, které jsou proti jeho názoru, dělá pravý opak a nedává jim takovou váhu, i když mohou být faktické. Podle síly této chyby úsudku, je osoba i schopná pozměnit vzpomínky a "přesvědčit" se, že jím preferovaná možnost je ta pravá, která se stala. K potvrzovacímu zkreslení patří také **efekt zpětného rázu** (backfire effect), kdy v případě, že by někdo konfrontoval osobu, která má potvrzovací zkreslení ohledně faktu, že to, co říká nebo si myslí je fakticky nesprávné, daná osoba namísto aby přiznala

svou chybu, naopak začne fabrikacím věřit o to více a začne tyto fabrikace vehementně bránit. [29]

Celkové množství kognitivních chyb úsudku je obrovské, ovšem těchto pár nám již dává představu, jak by se fakt, že tyto chyby myšlení existují, dal použít k phishingu. Dunning-Krugerův efekt napomáhá hackerům faktem, že se jejich oběti mohou cítít kompetentněji, nežli jsou, a tudíž k uchování této iluze by dělali akty, které jsou nad jejich rámec expertízy. Halo efekt napomáhá při phishingu v případě, že hacker udělá velice pozitivní první dojem, a tudíž mu obět bude více věřit. Status quo bias napomáhá například v případě, že by hacker poslal phishingový mail, ve kterém by se vydával za banku s nějakým finančním problémem, který je potřeba vyřešit co nejdříve. Obět trpící status quo biasem bude rozhozena, jedná se o instantní rozhození současného stavu, ve kterém se nachází, a dost možná se jedná o jeden z katastrofických scénářů, které si předtím představoval. Klam přeživších napomáhá hackerům nepřímo, ať již kvůli tomu, že jím trpí buď oběť samotná, nebo například firma, ve které oběť pracuje. Oběť samotná, i když je seznámená s problematikou phishingu, si může začít myslet, že ona sama bude ten "přeživší" phishingu, protože byla obeznámena se základními metodami rozeznání phishingu, a tudíž začne být samolibá. U firmy by to mohl být podobný případ v tom, že uvidí ostatní firmy, které phishing "přežili", začnou věřit, že přesně ví, jak tomu zabránit, a přestanou brát problematiku méně vážně, než by měli. Potvrzovací zkreslení by mohlo pomoct v případě, kdy hacker by začal phishing pozitivního charakteru, tudíž místo vyhrožování finančním problémem, jak tomu bylo u statusu quo, by například obětem namluvil, že něco vyhráli, nebo že získávají kryptoměny (viz. obrázek 5). Takto pozitivní zpráva může oběť potěšit do takové míry, že přestane myslet na možnost, že se ve skutečnosti jedná o phishing.

2.3 Proč je důležité o sociálním inženýrství vědět

Jak vidno z předchozí kapitoly, sociální inženýrství a phishing jdou ruku v ruce. Čím více o této problematice člověk ví, tím lepší bude, ať již phishing dělá, nebo se mu brání. Z hlediska hackerů je jasné, jak tyto informace přispívají ke zvýšení efektivity phishingových útoků, ovšem z hlediska potencionálních obětí je problematika trochu složitější. Osobně bych to vysvětlil takto: lidé, bez těchto znalostí, si často nemusí uvědomovat své chyby a limity. Problematika sociálního inženýrství často chybí v edukačních materiálech ohledně phishingu (u firemních, se kterými jsem se setkal), a upřímně mi to přijde jako velká chyba. Pokud edukační materiály založíme na pouhém rozeznávání emailových adres a

potenciálních gramatických chyb, nejen že je dosti možné, že po určitém čase člověk přestane dávat pozor na tyto body, hlavně se nikdy nenaučí/nespojí si, proč phishingové útoky jsou tak efektivní a časté. Je jednoduché říct, že je to kvůli faktu, že se probourává bezpečnost ne zkrze technická zabezpečení, ale přes lidský prvek, ale toto nevysvětluje, proč je lidský prvek ten nejslabší. Vědomí o sociálním inženýrství toto naopak vysvětluje velice dobře, vysvětluje proč člověk je nejslabší článek jakékoliv ochrany, a to je kvůli lidské psychice, která je chybná a nestálá.

3 PHISHING A AI

V posledních pár letech jsme viděli ohromný vzrůst ve funkcionalitě a popularitě generativních umělých inteligencí používajících LLM (large language models). Většina technologických magnátů začala tvořit své vlastní AI a tudíž máme umělou inteligenci pro ohromné množství činností, ChatGPT na generaci textu, Github Copilot k podpoře při programování, DALL-E ke generaci obrázků a mnoho dalších. Dnes se nedá přímo říct, zda tzv. AI Boom (rapidní progres umělé inteligence)[30], ve kterém se nacházíme, je pouze bublina, která může každou chvíli prasknout (jako v případě NFT, non-fungible tokens), technologie, kterou všichni vzývají do nebes, ovšem postupem času vyprší (jako v případě virtuální reality), nebo zda se opravdu jedná o technologickou revoluci, která si udrží svoji popularitu. Příliš nezáleží, jak tento AI Boom skončí, protože i umělé inteligence, které máme teď, jsou velice kompetentní. Ovšem se všemi zábavnými a morálně dobrými věcmi, které můžeme s těmito inteligencemi dělat, je zde problém, že ne všichni mají s touto technologií dobré úmysly. A některé umělé inteligence jsou jako tvořené pro ulehčení phishingu. Nejjednodušším příkladem pro použití AI k rapidnímu zlehčení phishingu je použití textových generativních umělých inteligencí ke generaci textu phishingového emailu. Místo toho, aby hacker musel vymýšlet manipulativní email pro svoji oběť, stačí mu akorát napsat prompt (výzvu), kde je AI zadán úkol, vypsat text emailu pro specifickou osobu s účelem ji vyděsit a kliknout na přiložený odkaz. Umělé inteligence jsou v poslední době velice limitovány, aby negenerovali nic potenciálně ilegálního nebo imorálního, ovšem tyto restrikce se dají celkem jednoduše obejít lepším zápisem uživatelského promptu. Toto je ovšem pouze vrchol ledovce co se týče věcí, kterých útočníci mohou dosáhnout za pomocí AI. Na začátku dnešní popularity AI jsme se setkali s takzvanými Deepfake, kde AI je schopné ve video záznamech nahradit celé obličeje. Tato technologie samozřejmě postoupila s časem, a společně s ní i další způsoby, jak manipulovat multimediální záznamy. Další podobné instance je například ElevenLabs, AI která generuje velice autentické hlasové audio stopy. [31] Stačí si vybrat daný model, klidně obsahující hlas známé celebrity a vygenerovat si cokoliv chcete aby daná osoba řekla. Toto je obvzlášť populární mezi internetovými uživateli ve formě memes (internetových vtipů), ovšem toto je opět použití této technologie k dobrým účelům. Technologie Deepfake se již dostala do mnoha kontroverzí, kdy lidé generovali pornografický materiál s obličeji celebrit [32], nebo vkládali obličeje politiků do špatných situací. Stejně tak ElevenLabs měla kontroverzi, kde si herci začali stěžovat, že jejich hlas je prakticky kraden bez jakékoliv kompenzace, a poté je jejich hlas použit k

namluvení hrůzných věcí. Těmito příklady se snažím naznačit, jak tato technologie může být používaná k samotnému phishingu. Vygenerovat si text phishingového emailu je oproti ostatním možnostem celkem jednoduchá. Žijeme v době, kdy se lidé regulerně nechávají nachytávají na vishing, například ve Spojených státech amerických je epidemie tzv. Indický tech suppoort scam, kdy se Indové snaží z lidí získat peníze tím, že se vydávají za IT podporu. Tento fenomén je tak rozšířený, že se z něj stal stereotyp a mnoho uživatelů internetu již o něm ví. [33] Co ovšem tito lidé budou dělat v případě, že by tito Indičtí podvodníci použili umělou inteligenci ke generaci hlasu? Nebudou schopni rozeznat přízvuk, a podvodníci získají větší šanci úspěšného phishingu. Co udělá chudák babička, která se stane cílem vishingového útoku, kde si druhá strana vygeneruje hlas jejího oblíbeného herce? Už tak je starší generace náchylnější na takovéto manipulační útoky, s použití AI budou mít miniskulní šanci poznat, že je něco špatně. V případě, že se AI generace obrázků stane čím dál tím realističtější, co by normální člověk mohl udělat, pokud by dostal nerozeznatelný obrázek výčtu jeho účtu, který byl zamknut a druhá strana po něm chce přihlášení? Tyto otázky by se dali pokládat do nekonečna. Nejedná se o problém specifický k phishingu, ale prakticky veškerých aspektů naších životů. Pokud tato technologie bude postupovat aby byly co nejvíce realistické, dostaneme se do éry, kdy lidé nebudou schopni rozeznat co je reálné, a co je AI. Veškeré věci, které jsme do teď mohli brát jako důkazy (audio vizuální materiál) tuto funkcionalitu kompletně ztratí. V tomto případě je budoucnost nejen phishingu velice temná. Vše záleží na tom, jak bude pokračovat postup této technologie. Je možnost, že její použití určitým způsobem by způsobilo takový skandál, že by se umělé inteligence začali řešit na úrovni právní. Toto by mohlo vést k nějaké legislativě, nějakému způsobu limitace těchto technologií. Já jsem osobně normálně proti jakékoliv situaci, ve které by se vláda vkládala do technologických záležitostí (v České Republice se toto často neděje, ovšem ve Spojených státech amerických bylo mnoho situací, kde vláda přišla s legislativami redukujícími technologickou svobodu, například Patriot Act po 11. září [34], nebo nejnovější zákaz TikToku prezidentem Bidenem [35]), ovšem nevidím v tomto bodě mnoho jiných možností. Jedinnou dobrou zprávou je, že AI se dá také použít ke zlepšení technologické ochrany oproti phishingu například za pomocí analýzy emailu, odkazů a podobných. Ovšem ani tak nemohou pomoci ochránit lidský faktor. Z těchto důvodů vidím budoucnost phishingu velice bledě. Phishing bude čím dál jednodušší na provedení, s lepšími způsoby manipulace obětí, a my nemáme příliš možností, jak se proti tomu bránit.

4 POUŽITÉ TECHNOLOGIE

V této kapitole se seznámíme s technologiemi použitými ke tvorbě samotné penetrační platformy a k jakým účelům budou použity.

4.1 HTML

HyperText Markup Language, neboli HTML je značkovací jazyk (markup language, principielně jím označujeme čistý text, čímž dáváme najevo, co daný text znamená) vydaný v roce 1993. Již od vydání se stal extrémně populárním, zejména kvůli jeho jednoduché a pochopitelné syntaxy. Funguje na principu přidávání značek (tags) do textu, čímž definujeme textový obsah na různé struktury (HTML elementy), jako jsou nadpisy, obrázky a podobné. Ke správnému zobrazení .html souborů je potřeba webový prohlížeč, který soubor načte a renderuje (vykresluje) jej do výsledné chtěné podoby. I když se HTML dá používat samotné na tvorbu rudimentárních webových stránek, ve většině případů se používá v konjunkci se skriptovacím jazykem (jako PHP, či Javascript) a jazykem CSS, který zajišťuje více možností pro zkrášlení front-endu. HTML v tomto projektu se objevuje buď v rámci PHP kódu samotné aplikace, nebo jako součást spoofovaných webových stránek. [36]

4.2 PHP

PHP, poprvé vydáno v roce 1995, je skriptovací programovací jazyk používaný ke tvorbě webových stránek a aplikací. Díky PHP jsme schopni do projektu přidat veškerou logiku nutnou ke správné operaci platformy a propojit ji s dalšími nutnými komponenty, například databází. Toto PHP umožňuje díky zabudovaným modulům/knihovnám pro mnoho dalších technologií, jako již zmíněné databáze, FTP servery a internetové protokoly. Ohromnou výhodou je kompatibilita HTML a PHP, kdy programátor je schopen vkládat HTML kód do PHP kódu a naopak, což programátorovi zjednodušuje práci a umožňuje lehčí přehlednost kódu, než kdyby musely kódy být oddělené. PHP funguje nezávisle na platformě, na které se nachází, tudíž se nemusíme obávat nekompatibility mezi například operačními systémy. PHP je stále velice populární, i v dnešní době, kdy v roce 2024 používá PHP přibližně 76.5% webových stránek po celém světě. [37]

4.3 CSS

CSS (Cascading Style Sheets) je tzv. style sheet language, neboli jazyk, který se stará o vizuální prezentaci daného HTML souboru. HTML samo o sobě má vbudované možnosti, jak graficky upravit daný obsah, ovšem jsou limitované. CSS zajištuje výběr z mnoha různých stylů, které mohou být aplikovány na všechny front-endové prvky výsledného projektu. CSS v našem projektu je důležité, zejména kvůli spoofaným stránkám, a jejich autenticitě. [38]

4.4 VMware Workstation/Vmware ESXi

V rámci našeho projektu potřebujeme testovací virtualizované prostředí s Unixovým operačním systémem. K účelu programování a testování platformy používám k virtualizování prostředí program VMware Workstation (Pro). WMware workstation, oficiálně vydáno v roce 1999, je velice populární hypervizor (virtualizační software se schopností spustit mnoho operačních systémů na jednom stroji) druhého typu (hypervizor je spuštěn v rámci hlavního operačního systému počítače), který je schopen virtualizovat 64 bitové operační systémy (předchozí verze byly schopné virtualizovat i 32 bitové, neboli x86-32, ovšem celkově se technologie posunula k 64 bitovým systémům, zejména kvůli nutnosti mít více nežli 4GB RAM paměti, a tudíž nebyl důvod udržovat kompatibilitu s 32 bity). WMware Workstation má schopnost bridgingu hostového síťového adaptéru (kdy na místo virtuálního síťového adaptéru, který WMware Workstation nastaví, hypervizor začne používat síťový adaptér hostového počítače jako i ten virtuální), což se osvědčilo jako nutnost pro správný chod platformy (více v praktické části).

WMware ESXi je podnikový hypervizor první typu, tudíž se nejedná o hostovaný hypervizor, jako v případě Workstation, kdy hypervizor je software pracující na hostovském počítači. Namísto toho je ESXi tzv. bare-metal hypervizor, neboli hypervizor, který běží přímo na hostovském hardwaru, obsahuje a intergruje důležité komponenty operačních systémů, jako kernel a přímo kontroluje hardware na kterém je a stará se o hostované operační systémy. WMware ESXi zmiňuji, protože finální projekt, jak jsme se domluvili s vedoucím práce Ing. Malaníkem, poběží na tomto hypervizoru. Ovšem programovat a testovat projekt takovýmto způsobem by bylo obtížné a zdlouhavé, tudíž je jednodušší prostředí připravit ve WMware Workstation, který se chová více jako regulerní hypervizor, podobný například Hyper-V od Microsoftu. [39]

Zajímavá poznámka ohledně společnosti WMware: společnost WMware byla v roce 2023 odkoupena americkým gigantem Broadcom Inc. Hlavní problém této akvizice ze strany uživatele jejich softwaru přišel velice nedávno (únor 2024), kdy byly ohlášeny změny cen licencí pro všechny jejich produkty. Oznámili, že cena licence, která se do teď odvíjela podle počtu procesorů se změní na cenu stanovenou podle počtu procesorových jader. Tímto krokem veškeré ceny licencí vzrostli astronomicky, kdy firmám, které platili například 60 tisíc dolarů ročně, se výdaje na WMware zvýšili až desetkrát. [40] Tento velice korporátní čin naštěstí neovlivňuje náš projekt, v případě nutnosti vyměnit v budoucnu hypervizor, by bylo akorát potřeba předělat virtuální prostředí.

4.5 Debian

Debian (také známý jako Debian GNU/Linux) je Linuxová distribuce, používající balíčkovací systém APT (Advanced Packaging Tool). Jako ostatní Linuxové distribuce je soustřeďuje na používání volného a open-source softwaru, ovšem uživatel je schopen nainstalovat i proprietární software, například ovladače grafických karet NVIDIA. Narozdíl od odnoše Debianu, Ubuntu, které je myšleno spíše pro začínající uživatele Linuxu, se Debian soustředí spíše na pokročilejší uživatele, protože neobsahuje tolik user friendly funkcí, které by pomohli člověku, který přichází z Windows, se zorientovat v novém operačním systému. Debian, jako většina pokročilejších distribucí Linuxu, se velice založená na používání terminálu (příkazové řádky), k plné kontrole nad operačním systémem. Debian je velice populární distribuce Linuxu pro použití na serverech, protože je proslulá svoji stabilitou. Díky tomuto faktu jsme schopni nechat běžet naší platformu s vysokým uptimem (čas, kdy program běží). [41]

4.6 PhpStorm

PhpStorm od české společnosti JetBrains, je IDE (integrated development environment) zaměřené na programování webových stránek a aplikací pomocí jazyků HTML, PHP, CSS, JavaScript atp. Obsahuje podporu mnoha různých frameworků, jako je Laravel či Symfony (nepoužité v naší platformě), integraci s mnoha různými druhy databází a podporu verzovacích služeb jako Git. Podporuje debugging všech kompatibilních nástrojů za pomocí debuggerů Xdebug a Zend Debugger. Největším problémem s PhpStorm je jeho licence, PhpStorm je prodáván jako subscription, tudíž nutnost platit každý měsíc (respektive platba musí být na celý rok), na místo jedné dlouhodobé licence. Naštěstí mají vyjímky pro různé

případy, jako jsou například studenti, profesoři či developeři open-source projektů, kteří si tento program mohou stáhnout a používat zadarmo (pokud stále splňují podmínky). [42]

4.7 PHPMailer

PHPMailer je podpurná knihovna pro PHP, umožňující jednoduché odesílání emailových zpráv. PHP samo o sobě obsahuje pár základních funkcí k odeslání emailových zpráv, ovšem PHPMailer tuto funkcionalitu značně zlepšuje do bodu, kdy jsme za pomoci PHP kódu odesílat emailové zprávy se všemi elementy emailu z regulerních emailových klientů. Hlavní výhodou pro naši platformu je podpora SMTP, kterým podle požadavků budeme odesílat naše phishingové kampaně. Navíc přidává podporu emailových příloh, automatickou validaci emailových adres, podporu několika různých kódování zpráv a v případě samotného programování platformy je velice jednoduchá na používání. [43]

4.8 PNServer

K tvorbě webových push notifikací, jsem použil knihovnu PNServer, distribuovanou na githubu pod MIT licencí, vytvořena Stefanem Kintzlerem. Knihovna, která je především napsaná v PHP s trochou JavaScriptu, poskytuje možnost založit Service Workera (funkcionálně se jedná o proxy server sedící mezi webovou aplikací, prohlížečem a sítí ve webovém prohlížeči, který naslouchá síťovým žádostem, a poté dělá chtěnou akci), který naslouchá specifickým příkazům na platformu (v našem případě poslouchá příjem ukradených údajů z úspěšného phishingu) a poté vytvoří a vyvolá webovou notifikace uvnitř prohlížeče. PNServer tyto webové notifikace provádí za pomocí tzv. VAPID (Voluntary Application Server Identification) klíče, který si musí developer vygenerovat sám, a dodává celému systému web push notifikací další vrstvu ochrany. PNServer také obsahuje funkcionalitu, která umožňuje napojit jej na databázi (buď SQLite, nebo náš MySQL), a uložit odebrání notifikací (žádost, že chceme notifikace dostávat) do databáze, do své vlastní tabulky, čímž si platforma bude pamatovat, zda jsou odebírané notifikace i po vypnutí prohlížeče. [44] Společně s PNServerem pan Kintzler vydal i skvělý tutoriál k web notifikacím, který byl velice pomocný při jejich implementaci. [45]

4.9 JavaScript

JavaScript je programovací jazyk, extenzivně používán ve webových stránkách, a webových aplikacích. Společně s HTML a PHP tvoří hlavní pilíře dnešního internetu, ovšem na rozdíl

od PHP, je jednodušší najít aplikace, které JavaScript nepoužívají. [46] S vedoucím práce Ing. Malaníkem jsme se dohodli, že JavaScriptu v platformě mám dát co nejméně. Hlavním problémem JavaScriptu (krom osobních, jako například nehezká syntax, či fakt, že je na téměř všech stránkách, i když by nebyl teoreticky potřeba a akorát používá více hardwarových sil, nežli by bylo pro webovou stránku normální) je fakt, že jeho inkluze přidává další potenciální ochranná rizika do platformy, což je celkem špatné, vzhledem k tomu, že práce je z velké části založená na internetové ochraně. Ve výsledku bylo JavaScriptu použito v platformě co nejméně, především v rámci PNServeru a dalších malých případů, kdy je JavaScript použit z hlediska UX (user experience).

II. PRAKTICKÁ ČÁST

5 PLATFORMA NA REALIZACI PHISHINGOVÝCH PENETRAČNÍCH TESTŮ

5.1 Tvorba testovacího/programovacího prostředí

5.1.1 Úvod do testovacího prostředí

Jedním z požadavků úspěšného dokončení této práce, je vytvoření virtualizovaného Unixového prostředí na testování samotné platformy. Po dohodě s vedoucím práce Ing. Malaníkem se jedná o virtualizovaný Debian (Debian GNU/Linux), který je ideální k implementaci takového programu, jako je naše platforma, zejména kvůli již zmíněné stabilitě. Tento Debian bude virtualizován v hypervizoru VMware Workstation Pro, velice obsáhlý (a drahý) hypervizor, jehož mnohé funkcionality budeme používat ve tvorbě a testování naší platformy. Hlavním důvodem, proč praktickou část diplomové práce začínám touto kapitolou, je fakt, že mnoho naprogramovaných funkcionalit platformy by nefungovalo na jiném, než Unixovém operačním systému. Tudíž i kdybych platformu programoval na Windows, nebyl bych plně schopen otestovat funkčnost programovaných částí bez Unixového prostředí, na které jsem schopen platformu přeposlat.

U distribuce Debian si můžeme vybrat z několika verzí, jak tomu většinou bývá u Linuxových distribucí. V případě Debianu se jedná o tři různé verze: Stable, Testing a Unstable. Stable, jak název napovídá, je hlavní stabilní produkční distribuce Debianu. Jedná se nejnovější verzi, na které všechny komponenty operačního systému byly plně otestovány, a splňují zadané požadavky (zejména, co se týče stability a bezpečnosti). Testing je modernější verze Debianu, která obsahuje balíčky, které ještě nebyly potvrzeny pro přidání do Stable distribuce. Unstable je tzv. cutting-edge distribuce, neboli distribuce, na které probíhá aktivní development. Obsahuje více balíčků nežli předchozí verze, ovšem nezaručuje jejich bezpečí, či stabilitu. [47]

Pro tvorbu našeho prostředí jsem vybral Debian Stable 12.5 (kódové označení Bookworm), zejména kvůli garanci bezpečnosti a stability. Naše platforma je designována na dlouhodobý uptime za pomocí NGINX webserveru, tudíž stabilita operačního systému nám vážně pomáhá.

5.1.2 Instalace virtualizovaného Debianu

Začít instalaci operačního systému ve VMware Workstation je triviální. Stačí vybrat možnost "New Virtual Machine", vybrat si zda chceme typickou, či vlastní instalaci (v našem případě potřebujeme vlastní instalaci), vybrat hardwarovou kompatibilitu virtuální mašiny (zde jsem vybral Workstation 16.x, kvůli kompatibilitě s VMWare ESXi), vybrat systémovou cestu ke stáhnutému Debian obrazu (ve formátu .iso), pojmenovat novou virtuální mašinu a vybrat, kolik síly hostovského počítače chcete virtuální mašině dát. V mém případě jsem zadal dva jednojádrové procesory (platforma není příliš náročná procesor) a 8GB RAM paměti (hostovký počítač má 64GB RAM paměti). Tyto hodnoty lze kdykoliv změnit, i po instalaci operačního systému.

Před instalací Debianu musíme vybrat heslo pro root (administrátorský účet). Toto heslo budeme používat velice často, tudíž doporučuji něco jednoduše zapamatovatelného a lehce napsatelného. Poté máme potřebu vytvořit uživatelský profil pomocí uživatelského jména a hesla. Tento profil bude použit po instalaci v případě, že nebudeme používat root. Samotná instalace Debianu na virtuální mašině je velice jednoduchá, zejména protože nemusíme manuálně nastavovat diskové oddíly, které bývají u Linuxu matoucí, zejména pro nové uživatele, nebo když máte více disků. Díky tomu, že máme pouze jeden (virtuální) disk, můžeme v instalaci vybrat metodu "Guided – use entire disk", kde nás instalátor provede tvorbou oddílů. Další možnosti tohoto výběru nám nabízí nastavení normálního, či zašifrovaného LVM (Logical Volume Management), což je jiný způsob správy diskového prostoru, ovšem nám budou stačit normální diskové oddíly. Po výběru (jediného) disku dostaneme možnost rozdělit různé části Linuxového filesystému, jako například /home či /var do různých oddílů, ovšem pro naše účely vystačí jeden oddíl obsahující veškeré soubory. Tímto nastavením se nám vytvoří jeden hlavní oddíl / a malý swap oddíl (poskytuje Linuxové distribuci virtuální paměť a napomáhá různým procesům, jako například hibernace systému). V rámci samotné instalace nám instalátor nabídne, zda máme externí instalovatelná média, které by balíčkový manažer apt mohl nainstalovat, a poté získáme možnost přizpůsobit apt výběrem (nejlépe námi blízkého) serveru a možnost přidat HTTP proxy, kterou nepotřebujeme. Kromě možnosti účastnit se odesílání statistik o našem systému developerům (tuto možnost z bezpečnostních důvodů odmítneme), získáme možnost vybrat si grafické prostředí našeho Debianu, a případný, často používaný software, který by potenciálně uživatel chtěl v nové instalaci. Osobně jsem vybral KDE Plasma jako grafické prostředí, jedná se o dobře vypadající prostředí, které se funkcionálně podobá

Windows, a na rozdíl od jistých jiných možností (GNOME), nebere příliš hardwarových prostředků. Dobrá volba na této stránce je "standard system utilities", ušetří nám několik terminálových příkazů. Posledním bodem instalace je instalace GRUB bootloaderu. GRUB zařizuje nahrání operačního systému po startu systému. Instalace GRUBu bývá u nových Linuxových uživatelů problematická, především pokud mají více disků, nebo se snaží o dualboot operačních systémů, v našem případě ovšem stačí vybrat jedinný nabídnutý disk a počkat na konec instalace. Po konci instalace nám instalátor nabídne restart systému, a samotný VMware Workshop nám nabídne možnost, jak po restartu "odpojit" instalační médium, ekvivalent vytáhnutí instalačního flashdisku pokud bychom instalovali na fyzický stroj, čímž předejdeme nechtěnému bootu zpět do instalačního média.

5.1.3 Příprava Debianu na programování/testování

Po bootování do nainstalovaného Debianu náš čeká rozsáhlá příprava, při které musíme nainstalovat všechny potřebné balíčky k eventuálnímu spuštění platformy. I když Debian společně s KDE Plasma je velice přátelský na kontrolu systému za pomoci GUIs (Graphical User Interface, grafické prostředí) a myši, ve výsledku je nutné používat terminál na drtivou většinu věci, které budeme na Debianu dělat.

První záležitostí hned po instalaci by měl být update systému. Ten provedeme v terminálu za pomoci příkazu `sudo apt update`, kdy sudo znamená použití administrátorského root profilu. Získáme chybovou hlášku, že námi vytvořený uživatelský profil není součástí "sudoers" souboru, neboli náš normální profil nezapadá do skupiny administrátorů. V tomto případě můžeme udělat dvě věci: buď se za pomoci `sudo -i` dostaneme do interaktivního root shellu (kde budeme mít plné administrátorské privilegia), nebo přidat náš uživatelský profil do onoho "sudoers" souboru. Vzhledem k tomu, že ne všechny terminálové příkazy podporují administrátorská práva a tudíž bychom museli pořád vlízat a vylézat s root shellu, doporučuji druhou možnost. Tohoto dosáhneme tak, že se nejdříve dostaneme do root shellu, a poté za pomoci příkazu `usermod -aG sudo X` kde X je naše uživatelské jméno přidáme náš účet do "sudoers" souboru. Díky tomu, kdykoliv potřebujeme administrátorská práva (jako v případě již zmíněné aktualizace systému) postačí nám před chtěný příkaz přidat slovo `sudo`.

Po první aktualizaci je chvíle, kdy si uživatel může přizpůsobit svoji Debian instalaci jeho požadavkům. Já jsem například změnil defaultní Unixový shell bash a zaměnil ho za modernější (z hlediska funkcionality) zsh. Největší výhodou zsh oproti bash je napovídání

příkazů při jejich psaní do terminálu, což velice pomáhá v případě, že jakkoliv pokročilý uživatel zapomene nějaký příkaz či cestu. Obzvlášť pro vypisování cest v Linuxu je toto napovídání extrémně pomocné, protože obsahuje i možnost vyvolat příkaz `ls` (vypsání souborů v adresáří) aniž by uživatel musel smazat příkaz, který zrovná píše. Zsh také podporuje mnoho pluginů, jeden, který jsem nainstalovat na naši Debian instalaci je oh-my-zsh, plugin, který zkrášluje a zlepšuje přehlednost shellu v terminálu.

Obrázek 7 Terminál s zsh shellem, pluginem oh-my-zsh a tématem Agnoster

Další změnu, kterou jsem provedl, byla změna terminálového emulátoru. Debian instalace s KDE Plasma prostředím je nainstalována s emulátorem Konsole, který je pomalejší, než jiné možnosti, a není zdaleka tak přizpůsobitelný. Namísto něj jsem nainstalovat emulátor Kitty, který je velice rychlý, přizpůsobitelný a s podporou GPU akcelerace. Možností jak přizpůsobit Linuxovou instalaci je mnoho, tolik že dokonce lidé s Linuxovými distribucemi nedělají nic jiného, než je zkrášlují svůj operační systém. My ovšem potřebujeme prostředí schopné chodu a testování naší platformy, a tudíž se přesuneme na nutné instalace k tomuto účelu.

Vzhledem k tomu, že platforma je prakticky webová aplikace, tak k hlavnímu chodu platformy potřebujeme jazyk PHP, MySQL databázi a webový server. Instalace PHP je lehce komplikovanější na naši Stable verzi Debianu. Stable Debian má ve svých repozitářích PHP verzi 8.2 jako svoji nejnovější dostupnou. Ovšem celkově nejnovější verze PHP je verze 8.3, která by z bezpečnostního hlediska byla lepší. V případě verze 8.2 bychom získali PHP jednoduchým příkazem `sudo apt install php php-common`. Ovšem v případě verze 8.3 musíme obejít oficiální apt repozitáře, přidat externí repozitář a podepsat ho (stáhnutím a ověřením veřejného GPG klíče), poté jsme schopni nainstalovat PHP verzi 8.3 za pomoci `sudo apt install php8.3 php-xdebug php8.3-cli php8.3-fpm php8.3-{X,Y,Z}` kde X,Y,Z a podobné jsou externí PHP moduly (plný postup v citaci) [48]. Pro nás potřebné budou moduly curl (spojení a komunikace se servery za pomoci známých protokolů), gmp (integerová aritmetika), mbstring (operace na string, které byly zakódované více-bytovým kódováním), mysqli (operace s MySQL databází) a zip (operace se ZIP archívy). Xdebug je PHP debugger, který nám vážně pomůže při samotném

programování. PHP8.3-cli nám dává možnosti využití PHP v příkazové řádce (nutné k debugování). PHP8.3-fpm (FastCGI Process Manager) je alternativa tradičního PHP-CGI (Common Gateway Interface), a funguje jako procesový manažer mimo webserver. Užívá si vysoké oblíbenosti v dnešní době, díky větší rychlosti, stabilitě a šetřivosti s prostředky webserveru. Po instalaci všech těchto komponentů jsme schopni vyvolat příkaz `php -i`, který nám vypíše velice dlouhý dump všech nainstalovaných/nastavených komponentů PHP. Nemusíme se zkrz něj prodírat sami, ale budeme jej potřebovat později. Abychom zkusili, zda bylo alespoň PHP-fpm nainstalováno správně, můžeme jej zkusit zapnout/zjistit status. Toto můžeme udělat, protože PHP-fpm jako jedinná část všech PHP komponentů, co jsme právě nainstalovali, funguje jako služba na Linuxu. Debian, jako většina dnešních Linux distribucí používá systemd, notoricky známý a nesnášený (v Linuxové komunitě) démon pro správu systému. Zkrze něj jsme schopni ovládat veškeré služby běžící na našem Debianu. Veškeré systemd příkazy se řídí vyvoláním `systemctl`. Pokud bychom například chtěli zjistit stav PHP8.3-fpm, získáme jej vyvoláním `sudo systemctl status php8.3-fpm`. Stejně tak zaměněním `status` za `start` nebo `stop` jej instantně vypneme, a použitím `enable` či `disable` zapneme nebo vypneme automatický start po bootu systému.

Dalším potřebným komponentem je databáze. Pro tyto účely jsem vybral MySQL, se kterým mám trochu zkušeností ze studia. Na našem Debianu ovšem nejsme schopni nainstalovat MySQL. MySQL byl celkově odebrán z repozitářů Debianu již v Debianu verze 9, a byl nahrazen MariaDB. Ovšem toto nám vůbec nemusí vadit, protože MariaDB (jednoduše řečeno) je open-source varianta MySQL (který je developován společností Oracle). Dokonce v mnoha benchmarcích MariaDB vyhrála nad MySQL, v rychlosti a výkonu na slabším hardwaru. Pro instalaci MariaDB vyvoláme příkaz `sudo apt install mariadb-server mariadb-client`, kde MariaDB-Server je samotný databázový server, se kterým komunikujeme za pomocí příkazů z MariaDB-Client. Po instalaci vyvoláme příkaz `sudo mysql_secure_installation`, který nás navede na interaktivní nastavení čerstvé MariaDB instalace. Hlavní částí tohoto nastavení je možnost změnit, respektive nastavit heslo pro MariaDB root účet, protože čerstvá instalace nemá root heslo. Funkčnost MariaDB jsme stejně jako u PHP-fpm schopni zkontrolovat za pomocí `sudo systemctl status mariadb`. Do samotného MariaDB se můžeme zkrz terminál přihlásit příkazem `mysql -u root -p`, který nás přihlásí jako root po zadání hesla. Zde bych chtěl zvýraznit dvě věci: jedná se o prozatím první příkaz, který nepožadoval `sudo`, a i když jsme instalovali

MariaDB namísto MySQL, MariaDB pořád používá MySQL terminologii v příkazech, díky čemu si nemusíme plést terminologii mezi jednotlivými databázemi. Když už jsme přihlášeni v MariaDB jako root, můžeme si rovnou vytvořit naší databázi. Zde nastupují klasické SQL příkazy tudíž vyvoláním `CREATE DATABASE X;` , kde X je jméno databáze ji vytvoříme. Osobně jsem ji pojmenoval phisherfrienddb, podle jména platformy. Ven z MariaDB se dostaneme klávesovou zkratkou CTRL+C, zkratka, která je používaná v terminálu k terminaci jakékoliv procesu, který právě probíhá.

Posledním nutným komponentem k implementaci platformy je webserver. Možností pro webservery je mnoho, ovšem vedou dva hlavní, velice populární webservery: Apache a NGINX. Apache je stálice ve světě webserverů, býval nejpopulárnějším webserver přibližně 20 let, a i dnes si užívá velkou popularitu. V půlce desátých let 21. století ho ovšem sesadil z trůnu rivalský webserver: NGINX. NGINX se stal velice populární možností vzhledem k mnohem menším hardwarovým nárokům, a tudíž schopnosti běžet perfektně na slabších serverech. NGINX oproti Apache je modulární, bez externích knihoven je schopen zobrazovat pouze statický obsah (ovšem velice rychle oproti Apache). Tyto knihovny jsou například námi již nainstalovaný PHP-fpm, který zajišťuje kompatibilitu s PHP. Apache na druhou stranu má kompatibilitu s PHP a dalšími jazyky přímo v sobě, čímž ovšem ztrácí na rychlosti. [49] Před touto prací jsem měl pouze lehké zkušenosti s Apache, a i když Apache je pro naši platformu dostačující, rozhodl jsem se raději vybrat NGINX. Instalace NGINXu je velice jednoduchá, stačí pouhé vyvolání příkazu `sudo apt install nginx`. Hlavní konfigurační soubor NGINXu najdeme v adresáři /etc/nginx/ pod jménem nginx.conf. Zde prozatím nemusíme nic měnit. Pokud se přesuneme do souboru default, v adresáři /etc/nginx/sites-available/, uvidíme jak NGINX naslouchá pro webové stránky.

```
server {
        listen 80 default_server;
        listen [::]:80 default_server;

        root /var/www/html;

        index index.html index.htm index.nginx-debian.html;

        server_name _;

        location / {
                try_files $uri $uri/ =404;
        }
}
```

Obrázek 8 Příklad konfiguračního souboru NGINX pro webové stránky

Zde vidíme, že NGINX naslouchá na port 80 (HTTP), této stránce servíruje soubory z adresáře /var/www/html, a tam hledá indexový soubor končící na specifické přípony. Deklarace `server_name _;` má stejnou funkcionalitu, jak kdyby tento příkaz chyběl, tudíž neposlouchá pro žádné specifické domény. Tudíž s tímto nastavením poslouchá pouze pro localhost. První věc, kterou musíme změnit, respektive přidat v tomto souboru, je index.php do pole index, protože naše první stránka bude ve formátu PHP, a bez této změny by ji NGINX odmítal zobrazit. Po uložení této změny v souboru můžeme vyvolat příkaz `sudo nginx -t`, který projde nastavení NGINX a řekne nám, zda vše sedí, nebo pokud jsme někde udělali chybu. Pokud otevřeme webový prohlížeč a zadáme adresu "localhost", zobrazí se nám cvičná titulní stránka vytvořená NGINXem, pouhá konfirmace, že NGINX je nainstalován správně a funguje, jak má. Ovládání NGINXu opět funguje zkrze systemd, tudíž pokud by byl nějaký problém, a my chtěli vědět, co se s NGINXem děje, vyvolali bychom `sudo systemctl status nginx`.

S těmito komponentama nainstalovanýma, máme připravený Debian na implementaci a testování platformy. V tomto bodě, se můžeme rozhodnout, jakým způsobem programovat samotnou platformu: buď přímo ve virtualizovaném Debianu, nebo na jiném (například v hostovském) operačním systému. Já osobně pro velkou část programování platformy používal hostovké PC s Windows 10. Výhody byly především rychlost a reaktivita nativního systému oproti virtualizovanému s menšími hardwarovými možnostmi, lepší funkcionalita PhpStorm IDE na Windows oproti Debianu (toto ovšem může být pouze můj problém), a možnost celkem jednoduše přendat celou platformu zpět na virtualizovaný Debian, díky funkci VMware Workstation "Shared Folders", kdy si můžeme zvolit složku na hostovském

PC, která bude automaticky připojena (mounted) na virtualizovaný Debian. Musím ovšem zmínit, že tato funkce je bezpečnostní riziko, protože je zde možnost, že by něco mohlo proniknout z virtuální mašiny na hostovský počítač, tudíž by se tato funkce měla používat pouze při testování. Programování na Windows bylo pohodlnější, ovšem pouze do bodu, kdy jsem přendaval aktualizovanou platformu zpět na virtualizovaný Debian. Byla zde nutnost přehodit mnoho proměnných, vyexportovat databázi z Windows a naimportovat zpět na Linuxu (což samo o sobě bylo problematické, kvůli kódování, které používá MySQL Workbench) a také se objevilo několik bugů, které se na Windows v platformě nevyskytovalo. Ať ovšem člověk vybere jakýkoliv postup, je potřeba začít s tvorbou samotné platformy.

5.2 Programování platformy

5.2.1 Příprava k programování

K samotnému programování je potřeba mít nějaké prostředí, kompatibilní s vybranými technologiemi. Prvně jsem přemýšlel o Neovim, moderní fork editoru Vim (který je vylepšená verze originálního Vi editoru, který je předinstalován téměř na všech Linuxových distribucích). Editory vycházející z Vi jsou extrémně obsáhlé textové editory s téměř esoterickým ovládáním a velice unikátními funkcemi oproti klasickým editorům, jako například Notepad. Proč ovšem použít pouhý textový editor, když by bylo příjemnější používat plné IDE (Integrated Development Environment)? Zde nastupuje Neovim, který pokračoval ve šlépějích Vimu a jeho podpoře pro pluginy. Neovim (a jeho předchůdci) mají téměř kultovní status na internetu, a tudíž i mnoho lidí, co pro něj tvoří pluginy. Toto se dostalo do takového stavu, kdy je možné přetvořit klasický textový editor jako čistý Neovim na plnohodnotné IDE pro prakticky jakýkoliv jazyk. Hlavním problémem s tímto postupem je ovšem nutnost vědět, jak tuto transformaci provést. Po velkých bojích, zejména z důvodu, že podpora Neovimu na Windows je hrozná, jsem vzdal sny o programování celé platformy na Neovimu, a přesunul jsem se k plným IDE (povedlo se mi to ovšem na naší testovací platformě, kde jej používám do teď).

Díky studentskému statusu, a Github Student Developer Pack programu, jsem získal licenci k PhpStorm od české společnosti JetBrains. Veškeré předchozí boje s programovacím prostředím rázem skončili. PhpStorm bylo na naší platformu jako dělané, obsahuje podporu všech použitých jazyků a všemožných databází a integruje verzování pomocí Gitu a

podobných verzovacích nástrojů. Za pomoci PhpStorm jsem dokončil celou platformu, s lehkou pomocí Neovimu na testovací platformě.

Máme tudíž prostředí, ve kterém platformu naprogramovat, je zde ovšem další program, který by nám značně zjednodušil manipulaci s jednou částí platformy, a tou je databáze. MySQL Workbench je GUI program pro tvorbu, editaci a manipulaci (migraci) s MySQL databází. Značně zjednodušuje jakékoliv akce, které je potřeba s databází udělat, za pomoci přehledného UI. Je rovněž schopen načtenou databázi převést na diagramy, čímž jednoduše získáme vizuální reprezentaci celé databáze, čímž jsem ušetřil čas a papíry.

5.2.2 Tvorba databáze

Prvním bodem, kterým jsem začal, bylo rozmyšlení a design databáze. Začal jsem tímto bodem, protože se od něj praktický odvíjí zbytek celé platformy. Začal jsem jedním z požadavků, kterým je multi-user přístup do platformy, tudíž potřeba uložit účty do databáze. Vytvořil jsem tabulku users obsahující základní nutná data uživatelských profilů: uživatelské jméno, heslo, email (v případě, že by administrátor chtěl uživatele kontaktovat), permise (zda se jedná o uživatelský, nebo administrátorský účet) a čas vytvoření účtu. Do této tabulky, stejně jako všech dalších, jsem přidal i identifikační záznam id v případě, že bych přes něj mohl udělat relaci na jinou tabulku. Bylo nutné při tvorbě této tabulky nezapomenout na fakt, že hesla nesmíme ukládat v cleartextu, tudíž použitý VARCHAR datový typ jsem limitoval volněji, na 255 charakterů max. Uživatelskému jménu a emailové adrese jsem dal nutnost být unikátní, u uživatelského jména je to samozřejmé, a u emailu mi nedávalo smysl dávat možnost více účtů udělaných na stejné emailové adrese. Čas tvorby účtu je defaultně nastaven na CURRENT_TIMESTAMP, čímž se při tvorbě účtu automaticky zapíše přesný čas a datum ve formě rok-měsíc-den čas do sekund.

Je potřeba vytvořit tabulku obsahující základní nastavení naší platformy, kam uložíme informace nutné ke správnému chodu všech funkcí naší platformy. Po tvorbě tabulky config jsem začal přidávat nutné položky: doménu, na které se platforma nachází, Slack webhook, Slack Bot token a Slack kanál. Zadaná doména, i když platforma bude schopna si ji vyhledat sama, se bude hodit v obou hlavním funkcionalitách naší platformy. Jinak bude platforma schopna odeslat notifikace do instantního messengeru Slack, pokud si člověk založí svého Slack Bota (velice jednoduchá záležitost, pokud má uživatel již Slack účet) a získá nutné hodnoty: Slack webhook je URL, na které bude Slack Bot naslouchat pro příkazy k notifikaci, a Slack Bot token je kódové označení specifického bota. Při tvorbě Slack Bota

si uživatel nastaví, do jakých kanálu ve Slacku má bot přístup a uvnitř platformy si napíše, do kterého pokoje ve Slack kanálu chce, aby dostával notifikace. Tyto notifikace budou dostupné jak pro otevřené phishing maily, tak pro příjem ukradených údajů. Posledním políčkem k budoucí tvorbě relace je user_id, kterým se můžeme odkázat na specifického uživatele a tudíž zařídit, aby každý uživatel si mohl platformu nastavit, jak chce.

Platforma bude obsahovat dvě hlavní funkcionality: tvorbu emailových kampaní a tvorbu spoofovaných webových stránek pro krádež údajů. Pro tyto funkcionality musíme připravit dostatečné tabulky.

Nejdříve jsem začal kampaní. Vytvořil jsem tabulku campaigns, která bude obsahovat základní informace o vytvořené kampani: její jméno (unikátní), obsah mailu, titulek, reply-to adresu, seznam emailových adres, na který kampaň odesíláme a počet odeslaných emailů v kampani. Ke tvorbě relací jsem také přidal položku pro odkaz na id uživatele, který kampaň založil a položku pro odkaz na SMTP profil, který bude použit k odeslání samotné kampaně. Protože ještě nemáme na jaký SMTP profil tímto odkazovat, další tabulkou je nastavení SMTP profilu, který obsahuje URL SMTP serveru, uživatelské jméno a heslo, a emailovou adresu spojenou s účtem u tohoto SMTP serveru, ze které se odešla ona kampaň v případě výběru tohoto profilu. Důvodem k tomuto rozhodnutí, je v předchozích kapitolách zmíněná reputace. Snažíme se vypadat s našimi kampaněmi co nejvěrohodněji, a pokud bychom prakticky spoofovaly email, ze kterého posíláme kampaň (tudíž jiný, než který je spojený s SMTP profilem), naše reputace by šla dolů velice rychle. Následují informace o tom, zda byla kampaň již zkontrolovaná administrátorem, a zda byla již poslaná. Při dodělávání se tato tabulka značně rozrostla: přidal jsem odkaz na projekt/spoofovanou stránku, která byla použita v rámci kampaně, políčka pro zprávu od uživatele administrátorovi a zpět (v rámci potvrzování kampaně), zda se jedná o phishingovou, či spearphishingovou kampaň a v případě, že jde o spearphishingovou, tak který seznam obětí byl použit. Poslední položkou je cesta k uploadované příloze, pokud existuje.

Poslední věc, která nám chybí ohledně kampaní, jsou výsledky dané kampaně. Výsledky se v tomto případě myslí lidé, kteří phishingový email v rámci kampaně otevřeli. Pro tyto výsledky jsem vytvořil tabulku resultscampaign, která bude obsahovat základní informace o oběti, která otevřela phishingový email, které získáme za pomocí skrytého trackeru vloženého do těla phishingového mailu. Těmito informacemi jsou datum a čas otevření, email oběti (abychom věděli, který z potenciálně mnoha odeslaných emailů byl otevřen), IP adresa a user agent (HTTP hlavička se základními informacemi o oběti, jako je operační

systém, a prohlížeč, ve kterém byl mail otevřen). K tvorbě potenciální relace jsem také přidal pole campaign_name, které jsme schopni navázat na hlavní kampaňovou tabulku.

Druhá funkcionalita je tvorba spoofovaných stránek z templátů. Pro tento účel jsem začal tvorbou tabulky stolencredentials, která bude obsahovat ukradená data od obětí phishingu. Bude obsahovat položky pro uživatelské jméno, heslo, datum a čas kdy údaje byly odeslány, IP adresu, ze které údaje přišly, jméno "projektu" (neboli vytvořené spoofované stránky), token (jedná se o CSRF token v případě, že jej originální stránka obsahuje), a políčko HIBP, neboli HaveIBeenPwned (kam uložíme hodnotu, kolikrát bylo dané heslo použito opakovaně, podle HaveIBeenPwned databáze).

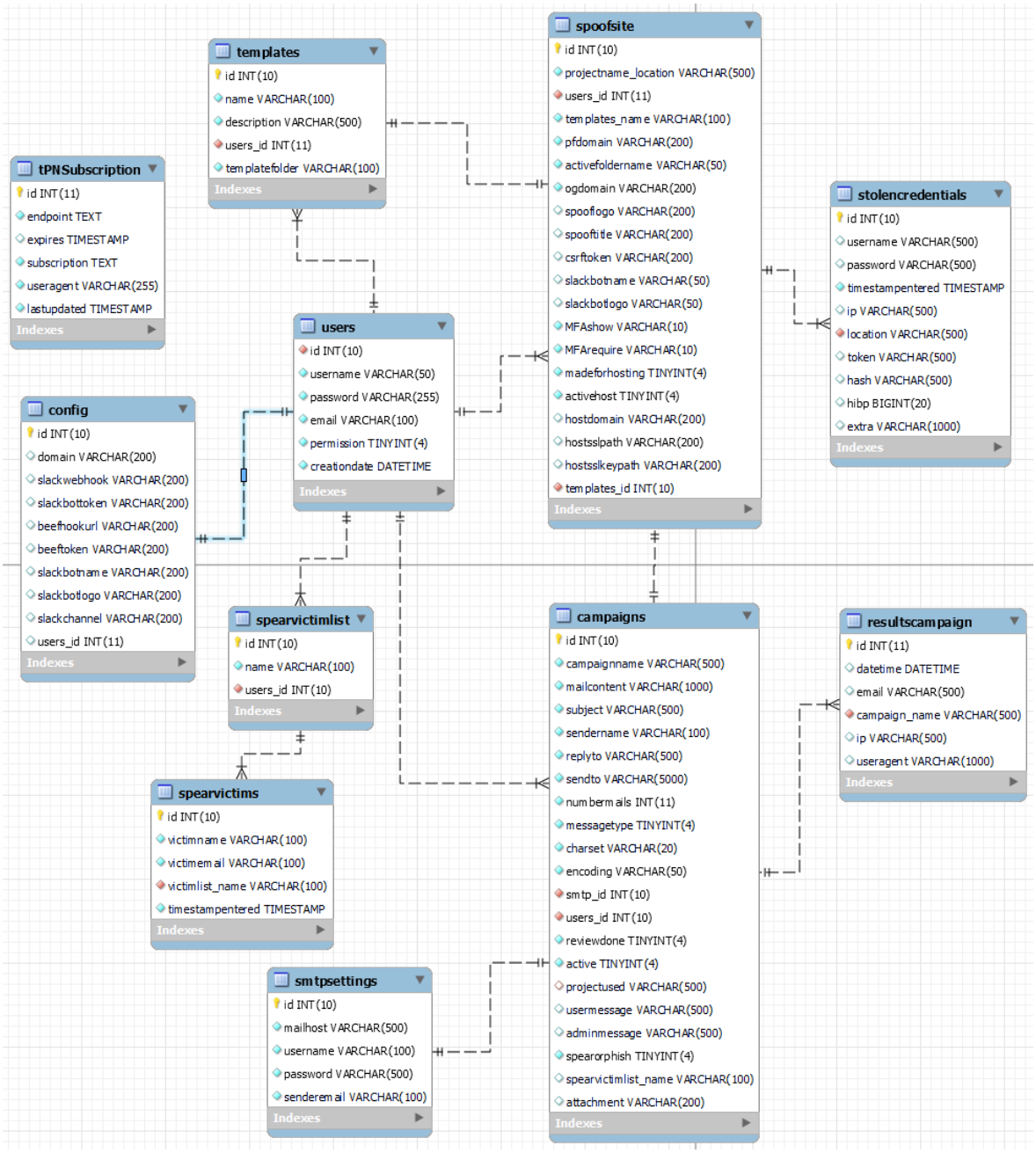
Po domluvě s vedoucím práce Ing. Malaníkem jsem doplnil možnost přidávat nové templáty do PhisherFrienda, tudíž jsem pro ně přidal jejich tabulku templates. Templates obsahuje všechny informace, které uživatel zadává při uploadu: jméno templátu a popis. Není potřeba zapisovat celou cestu templátu, protože se všechny nachází ve složce templates ve své vlastní podsložce, pojmenované stejně, jako templát.

Pro samotný projekt/spoofovanou stránku, jsem vytvořil tabulku spoofsite. Tabulka, která ze začátku byla jedna z nejmenších, se postupně při tvorbě platformy velice rozrostla. Máme zde všechna data, která uživatel zadává při tvorbě spoofované stránky z templátu, a pokud chce, hostování na stejném webserveru, co PhisherFriend. Spoofsite obsahuje jméno projektu (unikátní), jméno použitého templátu, doména PhisherFriend (odkazující navíc na složku /receive/), originální doména stránky kterou spoofujeme (na eventuální redirect), odkaz na uživatele, který spoofovanou stránku vytvořil a zda chceme na spoofované stránce políčko s multifaktorovou autentifikací (pokud to templát podporuje). Máme i několik políček, která nejsou nutná k vyplnění: logo a titulek spoofované stránky (bez změny se vezme z templátu), CSRF token (název HTML elementu držící tento token) a jméno + logo Slack bota, který pošle notifikaci. S podporou hostování spoofovaných stránek na stejném webserveru co PhisherFriend, se tabulka spoofsite rozrostla o jméno aktivní složky (což je složka uvnitř uživatelovy individuální složky uvnitř /active/, která bude obsahovat samotné soubory spoofované stránky), boolean zda byla spoofovaná stránka vytvořena k hostování na našem webserveru, a zda je zrovna aktivní, uživatelova doména, na kterou navede spoofovanou stránku a cesty k SSL certifikátu a klíči.

Kampaně mohou být buď phishingové, či spearphishingové. V případě spearphishingových bude potřeba, aby si uživatel vytvořil specializovaný seznam obětí, do kterého přidá

emailové adresy svých obětí společně se jmény obětí. Databázově jsem toto udělal dvěmi tabulkami: první je spearvictimlist pro samotné seznamy, kde jediná pole jsou jméno a odkaz na ID uživatele, který seznam založil a druhá je spearvictims, kde jsou vypsány samotné oběti s jejich plným jménem a emailovou adresou, odkazem na seznam, ve kterém se nacházejí a přesný čas, kdy byly vloženi do databáze (bude použité u individuálního mazání).

Poslední tabulkou je tPNSubscription, která je vygenerována automaticky jako část notifikačního PNServeru, a ukládají se zde odběry Web Push notifikací.



Obrázek 9 Finální model PhisherFriend databáze

Prvním krokem k tvorbě samotné platformy byl výběr jména. Osobně jsem na vymýšlení jmen velice špatný, a po dlouhém dumání jsem ji pojmenoval PhisherFriend. Přece jen se jedná o phishing, hrůznou aktivitu, ale naše platforma je kamarád, a pomůže nám s tím. První věc, co jsem udělal, bylo hrubé rozložení složek. Základ PhisherFrienda je v /html/, ovšem další funkcionalitu/"moduly" jsem chtěl ve svých vlastních složkách. Podrobné rozložení složek bude v následujících kapitolách. Poslední osobní poznámka, kterou bych rád dodal předtím, než se dostaneme do samotného programování platformy: nikdy jsem nebyl graficky nadaný (byl jsem spíše přes hudbu), takže neumím ani kreslit. Z tohoto důvodu se omlouvám, pokud by Vám PhisherFriend přišel po grafické/UI stránce poněkud základní. Já upřímně nevím, jak by z hlediska UI měla vypadat "hezká" platforma, vzhledem k tomu že osobně preferuji když stránky a webové aplikace jsou jednoduché, a namísto UI se zaměřují na přehlednost a UX. Jediný UI element, pokud by se to dalo tak nazvat, který ovšem očekávám od jakékoliv stránky, je dark mode, jednoduše protože je s ním stránka mnohem příjemnější na oči, ať se na ní uživatel dívá ve dne, či v noci. Z tohoto důvodu ve PhisherFriendu používám Bootstrap CSS Darkly, který celé platformě dá velice příjemný dark mode.

5.2.3 db.php

Vzhledem k tomu, že v průběhu celé tvorby PhisherFrienda potřebujeme přístup k databázi, vytvořil jsem ve složce /config/ náš první soubor db.php. Tento soubor obsahuje přihlašovací informace do databáze včetně názvu samotné používané databáze (z předchozí kapitoly phisherfrienddb), definici proměnné $DBConnect, která funguje jako naše připojení do databáze a budeme ji používat při každém databázovém úkonu uvnitř PhisherFrienda a jednoduchý test připojení. Na tento soubor budeme odkazovat v každém jiném, kde děláme nějaké akce s databází. Veškeré databázové příkazy při tvorbě PhisherFrienda jsou vytvořený za pomocí mysqli (MySQL Improved Extension) modulu, který jsme nainstalovali společně s PHP při nastavování Debianu. Největší výhodou používání mysqli je možnost zapisovat tzv. Prepared Statements (připravené výpisy), což je druh zapisování databázových příkazů v PHP, který nám umožní vyhnout se velice častému typu kybernetického útoku, který by mohl PhisherFrienda postihnout: SQL Injection. SQL injekce je druh kybernetického útoku, kdy útočník se snaží vepsat do programu svůj vlastní SQL kód. Toto se děje například u neochráněných input fieldů (políček, do kterých se může psát), které jsou napojené na databázi. Pokud program není ošetřen proti SQL injekcím, útočník by mohl být například schopen vytáhnout uživatelské údaje z databáze. Hesla v naší

databázi budou zahashována, tudíž tím by si příliš nepomohl, ale musí se tato možnost samozřejmě ošetřit. Prepared statements v mysqli tomuto právě zabraňují tak, že rozdělují co by normálně byl jeden SQL dotaz na dvě fáze: přípravu a exekuci. Rozdělením na tyto fáze jsme ochráneni proti SQL injekcím (pokud je ovšem použijeme všude, kde můžeme). [50] Společně se základním připojením k naší databázi jsem v tomto souboru definoval dvě podpurné funkce, `checkAdminRights()` a `checkCampaignsForReview()`. První nám vrátí true v případě, že přihlášený uživatel má administrátorská práva (používané k zajištění, že se normální uživatelé nedostanou do souborů, které jsou pouze pro administrátory) a druhou používám k zobrazení, zda jsou potřebné nějaké akce s kampaněmi v hlavním menu.

5.2.4 login.php

Další potřebnou záležitostí v tvorbě PhisherFrienda je přihlašovací forma. Jedná se vizuálně o jednu z nejméně zajímavých částí PhisherFrienda, ovšem z hlediska backendu obsahuje mnoho věcí, které musíme zajistit. Vytvořil jsem ji jako login.php, schválně ne jako index.php. Nutnost přihlásit se uděláme později v samotném index.php. Na začátku tohoto a všech ostatních souborů potřebujeme zavolat podpurné soubory, jako například db.php, abychom měli přístup k přihlášení se do databáze. Toto provedeme za pomoci příkazů `require` nebo `include`. Tyto příkazy obalíme v bufferu, za pomocí příkazů `ob_start()` a `ob_end_clean()`, které nám zaručí, že se output těchto podpurných souborů nevypíše do zbytku login.php.

```php
<?php
// bufferem načteme konfiguraci databáze
ob_start();
require_once 'config/db.php';
include 'config/php-csrf.php';
ob_end_clean();
// začneme session
session_start();
// inicializace instance tokenu proti CSRF
$csrf = new CSRF();
```

Obrázek 10 Začátek většiny souborů naší platformy

Pro přihlášení stačí dvě textové input pole, jedno pro uživatelské jméno, druhé pro heslo. Textová pole jsou následovaná elementem `<span class="invalid-feedback">`, který nám zajišťuje zobrazení chybové hlášky v případě, že uživatel zadal špatné

údaje/nezadal údaje. Tyto dvě pole jsou zaobaleny ve formě s metodou POST, čímž se po odkliknutí přihlášení údaje z polí aktivně pošlou do samotné logiky přihlášení, která je dělána v PHP. Pokud si toto odeslání údajů chytneme například příkazem `if($_SERVER["REQUEST_METHOD"] == "POST")`, jsme schopni doprogramovat zbytek logiky přihlášení. Nejdříve jsem udělal dvě if else podmínky, které zkoumají, zda jsou údaje vůbec vyplněné, a pokud ne, nahradí se proměnná uložená v již zmíněném spanu na chybovou hlášku. Následuje další if podmínka, zkoumající zda obě proměnné pro chybové hlášky jsou prázdné. Pokud ano nastupují databázové operace: nejdříve si do proměnné uložíme žádaný SQL dotaz, v našem případě vyber všechny uživatelská jména, hesla, a jejich ID, kde uživatelské jméno je stejné, co jméno zadané do formy. Za pomoci prepared statements si daný dotaz připravíme a napojíme na něj nutné parametry (zadané uživatelské jméno).

```php
// pokusíme se provést připravený dotaz
if(mysqli_stmt_execute($stmt)){
    // uložíme výsledek
    mysqli_stmt_store_result($stmt);
    // pokud je uživatelské jméno nalezeno, zjistíme zda heslo sedí
    if(mysqli_stmt_num_rows($stmt) == 1){
        // navážeme výsledné proměnné
        mysqli_stmt_bind_result($stmt, &...vars: $id, $username, $hashed_password);
        if(mysqli_stmt_fetch($stmt)){
            if(password_verify($password, $hashed_password)){

                // uložíme data do session proměnných
                $_SESSION["logged_in"] = true;
                $_SESSION["id"] = $id;
                $_SESSION["username"] = $username;

                // přesměrujeme uživatele na index.php
                header( header: "location: index.php");
            } else{
                // error pokud heslo nesedí
                $login_err = "Neplatné uživatelské jméno nebo heslo.";
            }
        }
    } else{
        // error pokud uživatelské jméno neexistuje v databázi
        $login_err = "Neplatné uživatelské jméno nebo heslo.";
    }
} else{
    // generický error, něco špatně s databází
    echo "Něco se pokazilo, zkuste prosím znova později.";
}
```

Obrázek 11 Logika přihlášení za pomocí Mysqli prepared statements

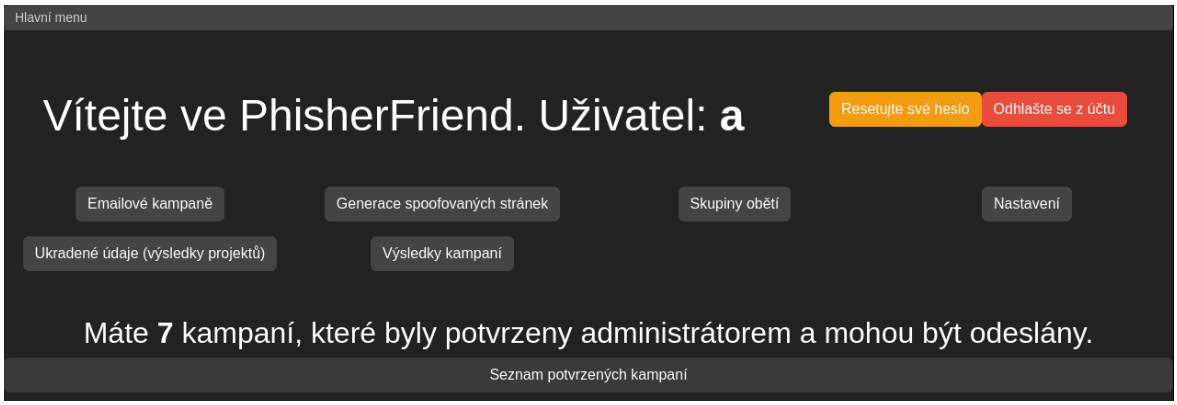
Po exekuci tohoto dotazu uložíme výsledek do další proměnné a zkontrolujeme, zda počet výsledných políček nalezených v databázi naším dotazem se rovná jedné. Pokud ano, uživatel je registrován a pokračuje se k heslu. Poté můžeme do proměnných vepsat výsledky SELECT dotazu, které použijeme k získání informací uživatele, který se přihlašuje. Poté použijeme zabudovanou MariaDB funkci `password_verify($zadané_heslo,$hashované_heslo_z_databáze)` v if podmínce, čímž zjistíme zda zadané heslo patří k uloženému uživateli v databázi. Pokud ano, uložíme do superglobální proměnné (funkcionalita PHP, kde jisté proměnné jsou použitelné v celém rozsahu projektu) $_SESSION id uživatele, jeho uživatelské jméno a boolean logged_in nastavíme jako true. Session je způsob, kterým PHP lze přenášet

informace mezi jednotlivými stránkami, a tyto proměnné se budou hodit v mnoha databázových operacích později. V poslední řadě pošleme přihlášeného uživatele do index.php, hlavního menu PhisherFrienda.

5.2.5 index.php

Index.php je v rámci PhisherFrienda pouhé hlavní menu. Začátek souboru index.php bude obsahovat podobné potřebné přízaky, jako tomu bylo u login.php: zavoláme potřebné externí soubory/knihovny za pomocí include/require, obalené v bufferu, aby se uživateli nevypsal kód těchto externích souborů uvnitř hlavního menu. Novinkou oproti login.php, je kód, který nám zajišťuje, že nepřihlášený uživatel nedostane přístup ke zbytku souboru. Toto jsem zajistil jednoduchou `if` funkcí, která zkoumá, zda není nastavená session superproměnná, za pomocí `!isset($_SESSION["logged_in"])`. Pokud tato proměnná není nastavena (nastavuje se při úspěšném přihlášení), tak je uživatel ihned poslán zpět do login.php. Index.php obsahuje pozdrav pro přihlášeného uživatele společně s výpisem, zda se přihlásil normální uživatel či administrátor a výpis jeho uživatelského jména. S tímto "titulkem" (ne HTML <title>, spíše první věc, které si přihlášený uživatel všimne) hlavního menu jsem přidal dvě tlačítka na stejném řádku. První by uživatele přeneslo na formu resetování hesla: tato forma je zjednodušenou verzí přihlašovací formy, kde po uživateli chceme pouze aby své chtěné nové heslo napsal dvakrát. Po postu této formy se zkontroluje, zda obě zadaná hesla jsou stejná, a v případě, že projdou touto kontrolou, zahashujeme nové heslo a aktualizujeme jím uživatele v databázi (za pomocí SQL příkazu UPDATE). Poté schválně zničíme aktivní session (za pomocí PHP funkce `session_destroy()`) a pošleme uživatele zpět na index.php. Ovšem protože jsme zničili aktivní session, uživatel bude odchytnut naším kódem, který kontroluje přihlášené uživele, a bude poslán zpět na login.php. Druhým tlačítkem je tlačítko pro odhlášení se z PhisherFriend, referencí na soubor logout.php. Logout.php je nejjednodušším souborem celé platformy, jeho funkcionalita spočívá v přemazání celé `$_SESSION` proměnné, zničení aktivní session, a poslání uživatele zpět na login.php. Hlavní funkcionalitou index.php jsou tlačítka přenášející uživatele do různých dalších částí PhisherFriend podle chtěné funkcionality. Předpřipravil jsem tlačítka pro menu emailových kampaní, menu spoofovaných přihlašovacích stránek, menu seznamů obětí pro spearphishingové kampaně, menu uživatelkého nastavení, výsledků

spoofovaných stránek/projektů (ukradené údaje) a výsledků kampaní (kdo otevřel naše kampaňové maily).



Obrázek 12 Hlavní menu platformy PhisherFriend

Pokud je přihlášen administrátor, získá dodatečné tlačítko, které vede do administrátorské sekce/nastavení. V poslední je v rámci index.php také textový element, který podle toho, zda je přihlášen uživatel, nebo administrátor, ukáže podle dat z databáze zda má uživatel nějaké administrátorem potvrzené kampaně, které může odeslat, nebo v případě přihlášeného administrátora ukáže, zda existují nějaké nové emailové kampaně, které potřebují potvrzení (více k tomuto cyklu uživatel => administrátor => uživatel v podkapitole o programování kampaňových forem).

5.2.6 Rozložení složek v root adresáři PhisherFriend

Kvůli přehlednosti a bezpečnosti je potřeba vytvořit několik složek, do kterých budeme dále přidávat další soubory/funkcionality platformy. Toto děláme abychom neměli veškeré soubory naší platformy v jedné složce, která je také nastavená jako NGINX root (/var/www/html/). Z estetického hlediska, je jasné, že pokud bychom všechny soubory nechali pouze v jedné složce, platforma by pro programování byla velice nepřehledná. Z bezpečnostního hlediska ovšem, můžeme za pomocí nastavení samotné NGINX stránky pro PhisherFriend i zakazovat vnější přístup do některých složek, což bude velice vhodné později při manipulaci s NGINX server bloky a SSL certifikáty. Funkcionalita všech těchto složek bude vysvětlena v průběhu této programovací kapitoly.

Výsledně stav složek v adresáři PhisherFriend vypadá takto:

I. **Active** - Obsahuje další složky podle uživatele, do kterých se ukládají hotové spoofované stránky vygenerované příslušnou formou, čímž se stávají root složkou jiné domény, v případě že uživatel chce, aby jeho stránka byla hostovaná na stejném NGINX webserveru, jako PhisherFriend.

II. **Blocks** – obsahuje konfigurační NGINX soubory pro další domény, které si uživatel může za pomocí příslušného PhisherFriend menu navázat na své vytvořené spoofované stránky (jedná se o ekvivalent /etc/nginx/sites-enabled, proč nepoužíváme tento defaultní NGINX adresář bude detailněji vysvětleno později).

III. **Campaigns –** Zde se nachází veškeré soubory, které zajišťují funkcionalitu emailových kampaní, společně se soubory knihovny PHPMailer a soubory WYSIWYG (What you see is what you get) editoru TinyMCE. Hlavními soubory jsou formy zajišťující kampaňový cyklus mezi uživatelem a administrátorem a formy pro tvorbu seznamů obětí pro spearphishingové kampaně. Navíc ještě obsahuje formu k zobrazení výsledků našich kampaní (kdo kliknul na naše kampaňové emaily).

IV. **Config –** Složka config obsahuje veškeré soubory, které nějakým způsobem mění nastavení PhisherFriend. Obsahuje uživatelské i administrátorské nastavení, již zmíněný db.php, který nám zajišťuje komunikaci s databází, php-csrf.php knihovnu pro zabezpečení vůči CSRF útokům, a také obsahuje veškeré soubory knihovny PNServer, která nám zajišťuje funkcionalitu Web Push notifikací.

V. **Images –** zde se ukládají obrázky, které uživatel vloží do emailové kampaně pomocí editoru TinyMCE.

VI. **Recieve –** Složka receive obsahuje pouze jeden soubor index.php, který nám zajišťuje příjem ukradených údajů ze spoofovaných stránek, zajišťuje redirect oběti na uživatelem nastavenou stránku a odesílá Slack notifikace o úspěšném phishingu.

VII. **Results –** Results obsahuje formy k zobrazení výsledků spoofovaných stránek/projektů, neboli úspěšně ukradené údaje od oběti.

VIII. **Spoofsites –** Spoofsites obsahuje formy k uploadování nových, uživatelsky vytvořených templátů, tvorbu spoofovaných stránek za pomoci uploadovaných templátů a tvorbu NGINX server bloků, kde si uživatel navede svoji zakoupenou doménu na jím vytvořenou spoofovanou stránku. Obsahuje také další složku “templates”, do které se ukládají uploadované templáty.

IX. **SSL –** Obsahuje dvě podsložky: “cert” a “key”. Do těchto složek se při tvorbě NGINX server blocků ukládá veřejný SSL certifikát a privátní klíč. Je důležité, aby tyto složky měli správně nastavené oprávnění, což bude probrané později v kapitole.

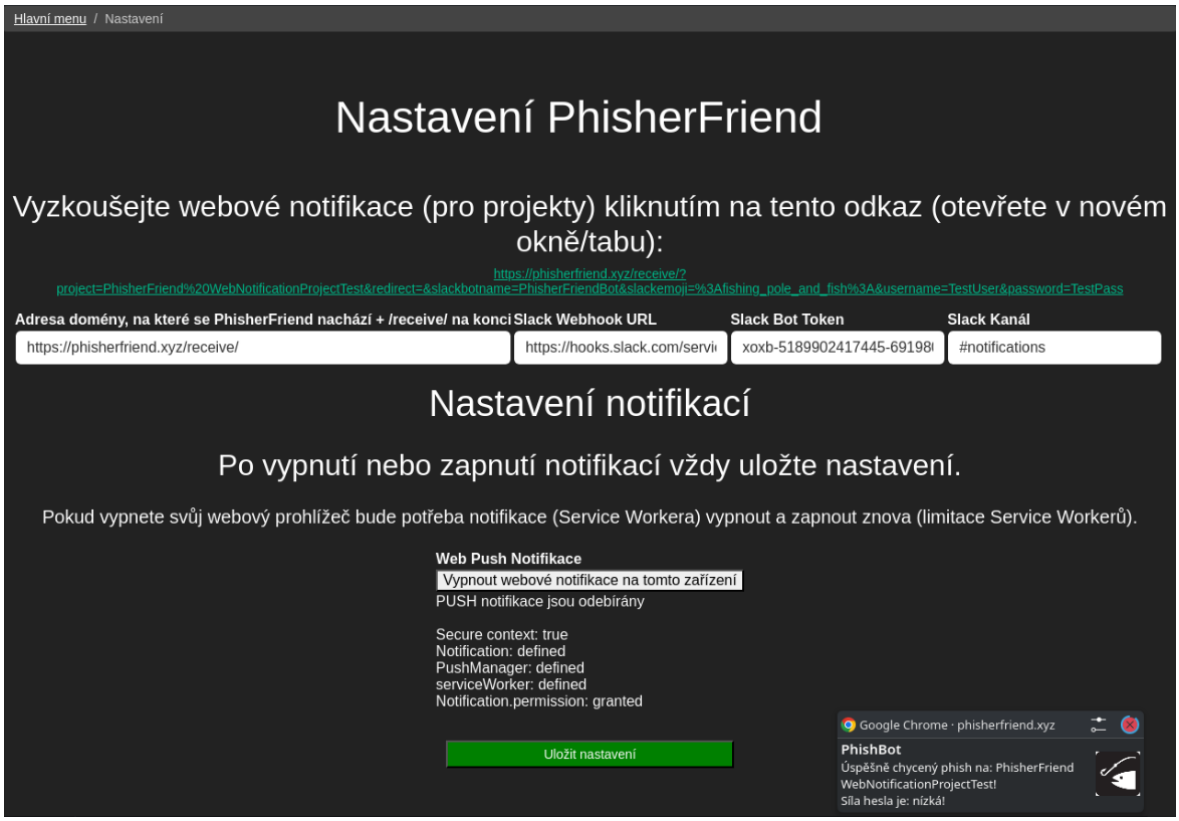
X. **Temp –** Poslední vytvořenou složkou je “temp”, do které se dočasně ukládají uploadované soubory, například při uploadování nových templátů nebo SSL certifikátů. Po použití PHP funkcí k manipulaci s těmito soubory se tyto soubory ze složky “temp” vymažou.

5.2.7 Složka Config

Ve složce config máme již náš vytvořený db.php soubor, zajišťující propojení s naší MariaDB databází. Toto ovšem není zdaleka jediný nutný soubor a je potřeba přidělat několik dalších souborů.

Začneme souborem index.php, který ve složce config bude fungovat jako menu pro uživatelské nastavení. V rámci uživatelského nastavení představujeme uživateli menu s několika zapisovacími <input> elementy pro nutné informace, které potřebujeme aby uživatelský účet měl. Specificky po uživateli chceme doménu, na které běží samotná PhisherFriend platforma společně s přídavkem “/receive/” na konci a nutná data ohledně uživatelského Slack bot nastavení. Část s doménou je nutná především k překontrolování správného infa, samotná doména je automaticky zjištěna za pomocí PHP superproměnné `$_SERVER[HTTP_HOST]`, a “/receive/” na konci domény navede potřebné funkce do index.php ve složce receive, která zajišťuje příjem údajů z úspěšného phishingu a Slack notifikace. Pokud chce uživatel získávat notifikace o všech událostech v rámci platformy, musí si založit Slack účet a vytvořit Slack bota. Jedná se o překvapivě jednoduchou záležitost, pokud je schopen uživatel ovládat PhisherFriend, tak by neměl mít žádné problémy s nastavením Slacku. Po vytvoření Slack bota získá jeho informace, a pro nás jsou důležité především Slack Webhook (URL, kterým se zpráva z PhisherFriend pošle do Slacku), Slack Bot Token (identifikační kód specifického Slack bota) a název kanálu, do kterého Slack bot pošle zprávu. V rámci nastavování Slack bota si uživatel může přímo nastavit, do kterého Slack Workspace (skupina Slack kanálů) uživatel chce, aby Slack bot posílal zprávy, proto pro nás je pouze důležité nastavit, do kterého kanálu ve Workspace má bot posílat notifikace. V rámci uživatelského nastavení máme i druhou možnost, jak nastavit uživatelské notifikace: Web Push notifikace. Uživatelské nastavení obsahuje tabulku s možností spustit, či vypnout Web Push notifikace. Tyto Web Push notifikace jsou zařízeny za pomocí knihovny PNServer. Funkcionálně se jedná o tvorbu tzv. service workera, neboli služba, která pracuje na pozadí v prohlížeči, naslouchá (fetch) příkazům, a reaguje (push). V našem případě registrovaný (spuštěný) service worker čeká, dokud nepřijde

požadavek/ukradené údaje na https://phisherfriend.xyz/receive/, kde se po projetí získaných informací zavolá soubor PNPush.php, který sestaví "payload" (obsah notifikace) a donutí service workera aby notifikaci pushnul do prohlížeče.

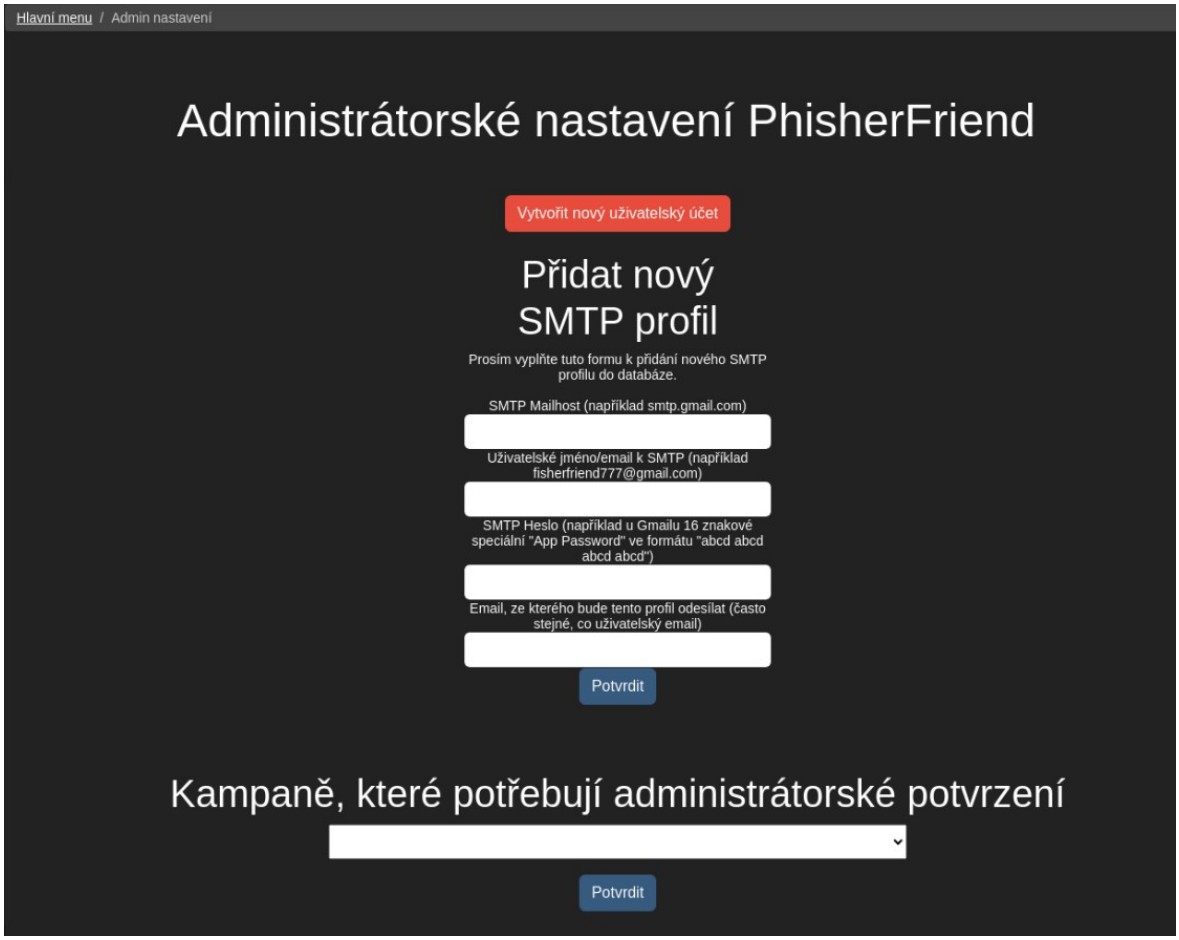


Obrázek 13 Nastavení PhisherFriend společně s Web Push notifikací

Service worker jako koncept má ovšem mnoho limitací, a z tohoto důvodu jsem ho použil pouze pro notifikace úspěšných phishů. Hlavním problémem je fakt, že service worker neběží nepřetržitě. Jak dlouho je schopen service worker fungovat záleží na prohlížeči, který uživatel používá, a po nějaké době je service worker ukončen. Stejně tak v případě vypnutí a zapnutí celého prohlížeče má prohlížeč tendenci service workera vypnout. Vypnutý service worker, který je stále zaregistrován v prohlížeči se sice dá opět zapnout za pomocí interních development menu v prohlížečích, ovšem toto by regulerní uživatel neměl být nucen dělat. Jednoduchým způsobem, jak opět zapnout service workera je jej odregistrovat a opět zaregistrovat, ovšem i toto je otravné. Kvůli těmto limitacím jsem se spíše soustředil na notifikace pro Slack. Poslední částí uživatelského nastavení je automaticky vygenerovaný odkaz, který si uživatel může otevřít (nejlépe v novém okně, či tabu), aby si zkontroloval, zda notifikace (ať již Web Push nebo Slack) fungují.

V rámci složky config máme také administrátorské nastavení, v souboru adminsmtp.php. Administrátorská forma obsahuje kromě zjištění, zda je uživatel přihlášen, také zjištění, zda

přihlášený uživatel má administrátorská práva. Administrátorské nastavení obsahuje těžkou většinu administrativních funkcí, které pouze účet s administrátorskými povoleními je schopen dělat. První takovou funkcí je tvorba uživatelských účtů. Podle požadavků je administrátor jediný, kdo je schopen vytvářet nové účty. Tlačítko pro tvorbu nových účtů administrátora přenese do souboru registrationform.php, který obsahuje formu ke tvorbě nového účtu. Funkcionálně je registrationform.php podobná přihlašovací formě a formě k resetování hesla. Administrátor zadá uživatelské jméno, dvakrát heslo (pro ověření) a emailovou adresu uživatele (v případě, že by administrátor chtěl uživatele v budoucnu kontaktovat, a nevěděl by o koho se jedná pouze z uživatelského jména). Možnost dát novému účtu administrátorská práva jsem přímo do platformy nedával, více bezpečnější bude, když si administrátor vytvoří nový účet, a poté v samotné databázi přehodí boolean permission (respektive TINYINT, který MariaDB používá namísto booleanu). Další funkcionalitou administrátorského nastavení je možnost přidávat nové SMTP profily, protože podle zadání veškeré phishingové kampaně budou odeslané přes SMTP. Forma k přidání SMTP profilu je velice podobná formě ke tvorbě nového uživatele, administrátor musí zadat SMTP mailhost server (například smtp.gmail.com), uživatelské přístupové jméno či email a heslo k účtu u SMTP služby a samotný email, ze kterého se budou kampaně odesílat (často stejné, co přístupový email). Tímto zajistíme, že email, ze kterého se odesílají kampaně, se nedá jednoduše změnit při tvorbě samotné kampaně. Hlavním důvodem k tomuto rozhodnutí, je fakt, že v dnešní době spoofování odesílací emailové adresy by akorát vedlo k reputačním/filtrovým problémům, a my se snažíme co nejvíce, aby naše phishingové kampaně došly ke koncové oběti. Poslední částí administrátorského nastavení, je výběr (za pomoci elementu <select>) nových, uživateli vytvořených emailových kampaní, které potřebují administrátorské potvrzení, předtím než mohou být finálně odeslány. Z tohoto výběru si administrátor vybere novou kampaň, a po kliknutí tlačítka je přemístěn na výpis nové kampaně. Toto je ovšem záležitost složky campaign, tudíž si ji vysvětlíme později.



Obrázek 14 Administrátorské nastavení PhisherFriend

Posledním souborem ve složce config, který by měl být zmíněn, je php-csrf.php. [51] Jedná se knihovnu usnadňující protekci proti CSRF (Cross-site request forgery) útokům, což jsou kybernetické útoky na internetové aplikace, kdy se hacker snaží využít faktu, že jsme přihlášeni do aplikace a snaží se využít této naší autentifikace k vykonání akce, která jistým způsobem změní aktuální stav. Příkladem může být, pokud bychom byly přihlášeni například do internetového bankovního, hacker by nám mohl poslat odkaz s kódem, který využije naší autentifikace, a převedl by peníze aniž bychom si toho v momentě všimli. Obzvlášť špatné by bylo, kdyby tento CSRF útok byl úspěšně vykonán na osobu přihlášenou účtem, který má administrátorská práva, v tomto případě by byl schopen vykonání jakékoliv akce v platformě. Nejčastějším způsobem, jak se proti CSRF útokům bránit, je implementace takzvaných CSRF tokenů. CSRF token je tajná hodnota, generovaná serverem a sdílena s uživatelem. Tato hodnota se validuje u odeslání každé formy, do které CSRF token vložíme, a CSRF útoky značně ztěžuje. Knihovna php-csrf.php nám značně usnadňuje implementaci CSRF tokenů a jejich validaci. Stačí v každém souboru, kde CSRF token potřebujeme, vyvolat

novou instanci CSRF třídy, do formy vložit generaci CSRF tokenu a do logiky, která probíhá po odeslání formy, začít validací tohoto tokenu.

5.2.8 Složka Spoofsites

Složka spoofsites obsahuje všechny soubory, potřebné k tvorbě spoofovaných stránek, neboli stránek, do kterých chceme, aby oběť zapsala své přihlašovací informace a odeslala je nám, čímž úspěšně ukončíme phishing. Tvorba těchto stránek probíhá v několika bodech, ve kterých se prochází několika soubory.

Prvním z těchto souborů je index.php, který funguje jako menu celého procesu tvorby spoofované stránky. Máme zde mnoho různých možností, použitelných podle toho, v jakém bodě tvorby spoofované stránky se nacházíme. První možností je přidání nového templátu. Toto tlačítko nás převede do souboru newtemplate.php, kde je uživatel schopen uploadovat nové templáty. Další možností je výběr z již uploadovaných templátů k tvorbě samotné spoofované stránky. Tato možnost nás přenese do souboru spooffactory.php. Následuje možnost vybrat si z již hotových spoofovaných stránek, které jsme označili, že chceme hostovat přímo na stejném webserveru co PhisherFriend. Toto nás přenese do souboru nginxbuilder.php. Tyto poslední dvě možnosti odesílají uživatele na nutné souboru pomocí PHP funkce `header`, ve které jsme schopni odkazovat se na jiné soubory společně s odesláním proměnných. Tyto proměnné se poté dají chytit v následujícím souboru za pomoci PHP superproměnné `$_REQUEST`.

```
header( header: "location: nginxbuilder.php?spoofproject=".$_POST['choosespoof']);
exit;
```

Obrázek 15 Příklad odeslání uživatele do jiného souboru společně s proměnnou

Poslední dvě možnosti v index.php mají opačnou funkcionalitu oproti předchozím v tom, že na místo tvorby něčeho nového, stávající záznamy mažou. První z těchto možností je výběr spoofovaných stránek, které uživatel hostuje na stejném webserveru co PhisherFriend. Vybráním takovéto stránky a potvrzením se vymažou hostovací údaje spoofované stránky z databáze, ze složky blocks se vymaže příslušný NGINX server block, a ze složky ssl se vymaže jak certifikát, tak klíč. Uživatel je poté schopen spoofovanou stránku opět vybrat v předchozích možnostech na hostování pod jinou doménou. Poslední možností je výběr spoofované stránky a její následné kompletní vymazání z databáze.

Po výběru tlačítka k přidání nového templátu, jsme odesláni do souboru newtemplate.php, který slouží k uploadování templátu do PhisherFrienda a jeho zapsání do databáze. Forma v tomto souboru chce po uživateli název nového templátu (nejlépe název stránky, kterou spoofuje), krátký popis templátu a file <input> element pro upload zazipovaného templátu. Po vyplnění a odeslání formy se nejdříve zkontroluje, zda byly zadány všechny údaje, a poté za pomoci PHP funkce `explode(".", $_FILES["attachment"]["name"])` zjistí, zda uploadovaný soubor má příponu zip. Důvodem k nutnosti uploadovat zip soubory, je využití PHP třídy ZipArchive, pomocí které jsme schopni jednoduše extrahovat templát na správné místo (v našem případě se ve složce templates vytvoří složka s názvem templátu, a do ní se extrahuje samotný templát). Posledním bodem je zapsání názvu, popisu a celé cesty k templátu do databáze. Soubor newtemplate.php také obsahuje návod, za pomocí kterého si uživatel může vytvořit svůj vlastní templát.

Následuje soubor spooffactory.php, který zajišťuje vytvoření spoofované stránky za pomocí vybraného templátu. Jedná se o dlouhou formu, do které musí uživatel zapsat několik nutných údajů. Prvním je opět doména PhisherFriend s /receive/ na konci, ovšem toto URL je vypsáno automaticky z uživatelova nastavení, zde se dá měnit pouze v případě, že by si uživatel toto URL nenastavil předem. Následuje pojmenování projektu (spoofované stránky), které bude použito v následujícím hostování a tvorbě kampaně. Poté si uživatel pojmenuje “aktivní složku”, což bude složka, ve které se spoofovaná stránka bude nacházet v adresáři active (/active/”uživatelské jméno”/”aktivní složka”/index.html). Tímto způsobem si může uživatel hostovat více spoofovaných stránek na jednom uživatelském účtu. Následuje doména, kterou tímto templátem spoofujeme. Toto nám umožní oběť převést po získání jejích údajů na originální stránku, do které si myslí, že se přihlašuje. Poté si může uživatel přidat své vlastní logo a titulek spoofované stránky. Tyto údaje ovšem nejsou nutné, pokud je uživatel nevyplní, vezmou se z templátu. Následují dvě nastavení pro Slack: jméno bota a ikonka notifikace. Tyto hodnoty také lze vynechat, ovšem ničím se nenahradí. Posledním vyplňovatelným políčkem je CSRF: toto pole se nechá prázdné, v případě, že stránka, kterou spoofujeme, nemá CSRF token protekci. Pokud ji má implementovanou jednoduchým způsobem, kdy originální stránka obsahuje skrytý input, který drží hodnotu tokenu, zapsal by se název tohoto skrytého <input> elementu. Další možností je výběr multifaktorové autentifikace: můžeme si vybrat, zda chceme, aby se na spoofované stránce objevil input pro vypsání MFA tokenu, a zda ho po oběti požadujeme, a nenecháme ji odeslat údaje bez něj. Po vyplnění všech těchto informací, má uživatel možnost posledního výběru: zda

chce hostovat tuto spoofovanou stránku na stejném webserveru jako PhisherFriend, nebo pokud si ji chce stáhnout ke svým vlastním potřebám (například hostovat ji na svém vlastním webserveru). Možnost hostovat jej na PhisherFriend webserveru je preferovaná, nejen protože se jednalo o jeden z požadavků vedoucího práce Ing. Malaníka, ale také protože jsme schopni jednoduše "lokálně" hostovanou spoofovanou stránku navázat do zbytku funkcionality PhisherFrienda (zejména do kampaní). Na druhou stranu, stáhnutí spoofované stránky pro hosting jinde by v teorii snížilo šanci, že by se veřejná IP adresa webserveru PhisherFriend dostala do nějakého blacklistu. Výsledky úspěšného phishingu by se stejně odesílali zpět do PhisherFrienda, tudíž je jednodušší tyto spoofované stránky hostovat na stejném webserveru. V případě, že vybereme preferovanou možnost, všechny zadané hodnoty se vypíší do svých vlastních proměnných, a za pomocí bufferu načteme obsah templátu, ten uložíme do proměnné, a vytvoříme nový soubor index.html, do kterého načteme obsah templátu a přepíšeme hodnoty podle zadaných informací. Poté zkontrolujeme, zda uživatel má svoji vlastní složku v /active/ (pokud ne tak ji vytvoříme), uvnitř ní vytvoříme novou složku, pojmenovanou podle vybraného jména "aktivní složky" a do ní vložíme hotovou spoofovanou stránku ve formátu souboru index.html. V případě, že si uživatel vybere spoofovanou stránku stáhnout, index.html se vytvoří stejným způsobem, a poté se akorát zazipuje a uživateli se nabídne tlačítko, po kterém začne stahování.

```php
//  navádíme PF na složku, kde se nachází vybraný templát
$chosenTemplate = preg_replace( pattern: '/[^a-zA-Z0-9 ]/', replacement: '', $_REQUEST['chosenTemplate']);
$templatePath = "templates/" . $chosenTemplate . "/template.php";


//  za pomocí bufferu načteme obsah templátu
ob_start();
include_once($templatePath);
$html = ob_get_contents();
ob_end_clean();

//  vytvoříme nový index.html soubor, do kterého načteme obsah templátu a změníme proměnné
$my_file = "templates/" . $chosenTemplate . "/index.html";
$handle = fopen($my_file, mode: 'w') or die('Cannot open file:  ' . $my_file);
fwrite($handle, $html);
```

Obrázek 16 Tvorba spoofované stránky z templátu

Posledním souborem ve tvorbě spoofovaných stránek je nginxbuilder.php. Tento soubor obsahuje formu k vytvoření takzvaných NGINX server blocks. Jedná se o NGINX ekvivalent virtuálních hostů z Apache, neboli možnost hostovat další domény/subdomény na jednom webserveru. Aby tvorba tohoto server blocku fungovala, uživatel si musí opatřit svoji novou doménu, a SSL certifikát/klíč. Nginxbuilder.php obsahuje formu, kam uživatel zapíše svoji

novou doménu, vybere SSL certifikát a klíč k uploadování a vybere, zda jeho doména má v DNS záznam pro www. před samotnou doménou, či ne. Po vyplnění a odeslání formy se nejdříve zkontroluje, zda byly zadané všechny údaje, a poté jsou zkontrolovány oprávnění složek cert a key ve složce SSL. Oprávnění složek a souborů v Linuxu je občas poněkud chaotické, obzvlášť vzhledem k tomu, že nezáleží jen na samotných oprávněních, ale také jaký Linux účet vlastní daný soubor/složku (a navíc daný účet je také v určitých skupinách uživatelů). Celkově oprávnění mi dělali velký problém při tvorbě platformy, naštěstí jsem narazil na skvělý článek, který mi oprávnění společně s účty více osvětlil. [52] Podle instrukcí v tomto článku jsem vytvořil nový Linux účet phisherfriend, a novou skupinu phisherfriend, do které účet náležel. Tento účet je prakticky vlastník PHP-FPM, a je to účet, který PHP používá ke spouštění skriptů. Jedná se o "vlastníka stránky". Další účet, který ovšem již existoval, je uživatel www-data. Toto je účet používaný webserverem pro všechny stránky, které jsou na něm hostovány. Za pomocí terminálového příkazu `usermod -a -G phisherfriend www-data` přidáme účet www-data do skupiny phisherfriend, čímž získáme schopnost ovládat k čemu má NGINX přístup za pomocí phisherfriend účtu, díky Linuxovým skupinovým oprávěním. Poté za pomocí `chown` (změna vlastníka souboru/složky) a `chmod` (změna jednotlivých oprávnění souboru či složky) finálně nastavíme oprávnění tak, jak by měli být. Tímto zajistíme základní bezpečnostní opatření, co se týče oprávnění, pro celý náš projekt. Ovšem v případě SSL složek zde máme další potřebné změny. Obzvlášť u složky key, kde se budou nacházet privátní SSL klíče, potřebujeme silnější protekci za pomocí oprávnění. Když ovšem změníme oprávnění na nižší (například `chmod 600`) dostaneme se do situace, kdy PhisherFriend nebude schopen zkopírovat uploadované klíče do jejich složky. Z tohoto důvodu v rámci hlavní logiky nginxbuilder.php provádím dvě hlavní změny oprávnění za pomocí PHP kódu. Nejdříve zjistím, jaká oprávnění mají obě SSL složky. Pokud jsou jiná, než 755 (což je upřímně i více, nežli by bylo potřeba), změním za pomocí PHP verze chmod jejich oprávnění na 755. Tímto získáme otevřené složky, do kterých můžeme jednoduše zapisovat nové soubory. Zkopírujeme to jednotlivých složek uploadovaný SSL certifikát a klíč, a poté opět změním oprávnění zpět (710 pro certifikáty, a 600 pro klíče, protože klíče by měli mít dodatečnou protekci).

```php
$perms = substr(decoct(fileperms($rootcertfolderpath)), offset: 2);
$perms2 = substr(decoct(fileperms($rootkeyfolderpath)), offset: 2);

//nutné přidat mezi PHP příkazy, které mění vlastnictví/permise souborů/složek
clearstatcache();

if (intval($perms) !== 755) {
    chmod($rootcertfolderpath, permissions: 0755);
}
if (intval($perms2) !== 755) {
    chmod($rootkeyfolderpath, permissions: 0755);
}
```

Obrázek 17 Zjištění a změna oprávnění SSL složek

Tímto jsme vyřešili záležitost ohledně SSL certifikátů, a nastupuje samotná tvorba NGINX server bloku. Server block je tvořen připojováním (append, .=) potřebného textu/proměnných to jednoho velkého stringu. Vzhledem k tomu, že používáme HTTPS, server block bude tvořen ze dvou serverových bloků. První bude naslouchat HTTP portu 80 a zadané doméně. Ovšem pokud získá HTTP požadavek z této domény, vrátí kód 301 (Moved Permanently, permanentně přestěhováno). Následuje druhý block, který již naslouchá HTTPS portu 443 a vyžaduje SSL. Následují základní nutné informace, jako která složka je root této domény (v našem případě /active/"uživatelské jméno"/"aktivní složka"/) a které soubory má NGINX zkusit načíst (v našem případě index.html, neboli naše spoofovaná stránka). Následuje zápis cest k SSL certifikátu a klíče, společně se zapnutím podpory PHP, do kterého se musí zapsat cesta k PHP-FPM socketu (tato cesta je neměnná). Za pomocí direktivy location jsme taky schopni určit různé podmínky, jako například že tento server block má ignorovat soubory začínající na .ht (například .htaccess), nebo v jakém pořadí má NGINX zkoušet soubory v root složce. Tyto location direktivy jsem zadal podle defaultních NGINX nastavení. Po složení tohoto NGINX server block souboru zkontrolujeme, zda se v aktivní složce nenachází ten samý soubor (automaticky se pojmenovávají podle domény, kterou uživatel zadá), v případě, že by se tam nacházel, vymaže se a uloží se nový. Posledním bodem je zapsání potřebných informací do databáze: doména, cesta k SSL certifikátu a cesta k SSL klíči. Databáze u vybrané spoofované stránky se také aktualizuje názvem nové domény, cestou v SSL certifikátu a cestou ke klíči. V tomto bodě je na uživateli, aby si navázal DNS své domény na veřejnou IP adresu našeho NGINX webserveru (DNS zápis typu A, se jménem @ a hodnotou veřejné IP adresy našeho webserveru, případně druhý DNS zápis, téměř

identický, ale se jménem www, pokud chce uživatel doménu používat i s předponou www.), i když by se dalo očekávat, že toto již uživatel udělal, protože si musel pro svou doménu získat SSL certifikát. Ovšem kdybychom skončili logiku této formy v tomto bodě, brzy bychom zjistili, že naše zbrusu nová hostovaná stránka nefunguje. Toto je z důvodu, že NGINX potřebuje nějaký signál, že se nějak změnil status, a má překontrolovat nastavení, jaké změny proběhli. Nejjednodušším, ale nepraktickým způsobem je celý NGINX resetovat (v debianu za pomoci `sudo systemctl restart nginx`), toto nám ovšem webserver na chvíli vypne, což by bylo velice nepraktické v případě, že by do PhisherFrienda byli připojení ostatní uživatelé, kteří by dělali své záležitosti. Naštěstí pro nás, existuje způsob jak NGINX donutit k tomu, aby přezkoumal nastavení i bez toho, aniž bychom ztráceli uptime. Pomocí příkazu `sudo /usr/sbin/nginx -s reload` jsme schopni tuto akci přinutit. Problém ovšem, v obou těchto možnostech, je fakt, že tyto příkazy musí být poslány s administrátorskými právy za pomocí sudo. Tímto se dostáváme do situace, kde kdykoliv by si uživatel vytvořil nový NGINX server block, musel by otravovat administrátora s přístupem k sudoers/root účtu, aby poslal tento reload signál, což je velice nepraktické pro uživatele, a extrémně otravné pro administrátora. Naštěstí pro nás, existuje způsob, jak vyvolat sudo příkazy bez nutnosti zapisovat administrátorské heslo, a tudíž jej můžeme použít uvnitř PHP kódu. Tímto způsobem je tvorba a editace nového souboru ve složce /etc/sudoers nebo /etc/sudoers.d/. Za pomocí příkazu `#sudo visudo -f /etc/sudoers.d/phisherfriend` jsem vytvořil nový soubor phisherfriend v daném adresáři, jehož editací (za pomoci visudo, což je speciální příkaz umožňující bezpečnou editaci těchto sudoers souborů, protože kontroluje syntaxi před zapsáním editovaného souboru) jsme schopni uživateli phisherfriend dát sudo příkazy, u kterých se mu uleví od nutnosti zapisovat heslo.

```
 GNU nano 7.2                                  /etc/sudoers.d/phisherfriend.tmp
phisherfriend ALL=(ALL) NOPASSWD: /usr/sbin/nginx -s reload,/bin/systemctl restart nginx
```

Obrázek 18 Uživateli phisherfriend dáváme schopnost používat dané příkazy bez nutnosti zapisovat heslo

Po důkladném uložení tohoto souboru (a restartu) jsme schopni signalizovat reload NGINXu, aniž by po nás systém požadoval administrátorské heslo. A protože náš účet phisherfriend je také vlastník souborů v root destinaci PhisherFrienda, jsme schopni tento příkaz použít přímo v kódu. Tímto způsobem, jsem na úplném konci logiky nginxbuilder.php (v případě, že předchozí akce proběhly správně, jako například tvorba samotného server blocku

a jeho přesun) přidal tento signalizační příkaz a spustil ho za pomocí PHP funkce `exec()`. Tímto, ihned poté co uživatel vytvoří svůj server block, NGINX již hledá, co se v jeho nastavení změnilo/co bylo přidáno, a v případě, že je nová doména navázána správně ze strany uživatele, tak bude funkční v několika vteřinách.

5.2.9 Složka Receive

Abychom mohli vyzkoušet funkcionalitu našich vytvořených spoofovaných stránek, potřebujeme způsob, kterým se k nám ukradené údaje dostanou. Původně jsem se pokoušel pro tuto funkcionalitu použít AJAX (Asynchronous JavaScript And XML) s nápadem, že pokud bych tento příjem zařídil tímto způsobem, dalo by se teoreticky zvláštním redirectům na naší PhisherFriend doménu, které by proběhli v případě, že by oběť odeslala své údaje. Ovšem samotná práce s AJAXem mě od tohoto způsobu odradila, a vzhledem k domluvě s vedoucím práce Ing. Malaníkem, kdy jsme v platformě chtěli co nejméně JavaScriptu, jsem raději přešel na striktně PHP řešení tohoto zpětného odběru ukradených dat.

Složka receive obsahuje pouze jeden soubor, index.php. Za začátku jsem začat vytvářet tuto PHP funkcionalitu odběru dat přímo v hlavním menu celého PhisherFriend, protože mi z hlediska UX přišlo, že by pro uživatele bylo více příjemné vypisovat pouhou doménu PhisherFriend, v místech, kde je to potřeba (například při tvorbě spoofované stránky z templátu). Ovšem zde jsem narazil na problém jakmile jsem začal pracovat na multi-user přístupu do platformy, protože přijímaná data se zarazila v hlavním menu u bloku kódu, který nám zajišťuje, že nepřihlášený uživatel nemá do hlavního menu přístup. Z tohoto důvodu jsem oddělil kód na odchytávání ukradených dat do své vlastní složky, od čeho se odvíjí nutnost zapisovat doménu s /receive/ na konci. Naštěstí, pokud je potřeba získat samotnou doménu bez /receive/ na konci, jsme schopni tohoto dosáhnout za pomoci již zmíněné PHP superproměnné `$_SERVER[HTTP_HOST]`, kterou můžeme pro naše účely uložit do proměnných.

```
<form action="<?php echo $APIURL; ?>" method="POST">
        <label for="username">Username:</label>
        <input type="text" id="username" name="username">
        <label for="password">Password:</label>
        <input type="password" id="password" name="password">

        <?php if($MFA == "on"){ ?>
        <label for="token">Token<br></label>
        <input type="text" name="token" id="token" <?php if($MFArequired == "on"){ ?>required<?php } ?>>
        <?php } ?>

          <input type="hidden" name="project" value="<?php echo $Project; ?>">
           <input type="hidden" name="redirect" value="<?php echo $Redirect; ?>">
           <input type="hidden" name="slackbotname" value="<?php echo $SlackBotName; ?>">
           <input type="hidden" name="slackemoji" value="<?php echo $SlackEmoji; ?>">

        <div id="lower">
           <input type="checkbox"><label class="check" for="checkbox">Keep me logged in</label>
           <input type="submit" value="Login">
```

Obrázek 19 Příklad templátu pro spoofované stránky

K pochopení, jak bude fungovat náš kód pro získání ukradených údajů, je dobré zvýraznit, jak funguje spoofovaná stránka. Jak vidíme na obrázku (Obr. 19), najdeme formu, která se stará o odesílání údajů, které chceme ukradnout, a políčko action nahradíme proměnnou, která se nám při tvorbě samotné spoofované stránky přehodí na doménu PhisherFriend + /receive/. Toto nám zajistí, že po odeslání údajů nám hodnoty z <input> elementů ve formě přichází do index.php ve složce receive. Důležité je také v templátu (a poté ve spoofované stránce) pojmenování jednotlivých <input> elementů, protože receive sbírá údaje právě z těchto pojmenovaných elementů. Po přijetí postované formy naším index.php ve složce receive se začnou ukládat hodnoty z jednotlivých <input> elementů do proměnných, včetně IP adresy, ze které požadavek přišel (za pomocí `$_SERVER['REMOTE_ADDR']`). Speciální případ je CSRF token, který pokud má <input> element v templátu, jehož jméno je uživatelem zadáno při tvorbě spoofované stránky, použijeme cURL (umožňuje transfer dat za pomocí protokolů, dále bude používán pro Slack notifikace) příkaz k odeslání GET requestu na stránku, kterou spoofujeme, a na kterou chceme oběť poslat po získání jejich údajů. Za pomoci tohoto cURL příkazu a pár funkcí ke zjištění polohy části value ve skrytém CSRF inputu jsme schopni získat hodnotu CSRF tokenu. Následuje část ohledně redirectu oběti na uživatelem zadanou stránku (nejlépe ta samá, kterou spoofujeme). Zde za pomocí shell příkazu `curl -s "'.$redirect.'" -I | grep -Fi X-Frame-Options` zjistíme, zda webserver spoofované domény obsahuje konfiguraci X-Frame-Options. X-Frame-Options určuje, zda je možné webovou stránku zobrazit použitím HTML elementů <frame>, <iframe> nebo <object>. Nás zajímá především <iframe>, který umožňuje vkládat (embed) nezávislý HTML dokument, do aktuálního. Pokud dostaneme jakoukoliv odpověď

z onoho cURL/grep příkazu, znamená to, že webserver dané domény obsahuje toto nastavení, a ani nás příliš nezajímá, jakého typu. Spíše to pro nás znamená, že použití <iframe> by bylo příliš riskantní, a tudíž redirect uděláme jinak. V tomto případě vytvoříme prázdnou kostru HTML dokumentu, od <html> přes <head> a <body> do </html>, a uvnitř hlavičky vložíme <meta> element, který zařídí, že jakmile se dostaneme k tomuto kódu, stránka se instantně refreshne a pošle oběť na stránku zadanou jako redirect. V případě, že nedostaneme odpověď z našeho shell příkazu, uděláme obdobnou kostru, ovšem namísto <meta> elementu v hlavičce dáme do těla element <iframe> s hodnotou src (source) jako naše redirect stránka. Oba tyto způsoby zajistí, že jakmile načteme potřebné ukradené údaje, oběti bude zobrazena zadaná redirect stránka (v případě <iframe> bez dalšího refreshe, který by mohl oběť vyděsit). Následuje zapsání ukradených údajů do databáze a tvorba notifikací. Pro tuto notifikaci jsem použil i několik podpůrných funkcí a cURL příkazů, které interagují s HaveIBeenPwned databází, k získání hodnot, které by pro uživatele mohli být zajímavé. První je za použití PHP funkce preg_match (PHP regex) zjištění komplexity hesla, podle počtu charakterů, velkých/malých písmen a čísel použitých. Další je kontrola hesla oproti HIBP, ze které získáme hodnotu kolikrát bylo dané heslo použito v účtech, které se objevily v nějakém leaku, a tudíž jsou obsažené v HIBP databázi. Tuto hodnotu také zapíšeme do databáze, abychom ji mohli později zobrazit v auditu. Poslední částí je složení a push notifikace. Zpráva notifikace je dělán připojováním částí zprávy (podobně jako u nginxbuilder.php), a obsahuje i hodnoty, které jsme phishingem sebrali. Pokud jsme získali heslo oběti, zobrazí se jeho síla v notifikace, stejně tak hodnota, kolikrát bylo toto stejné heslo použité jinde, díky HIBP databázi a v případě, že jsme získali token multifaktorové autentifikace, přidáme ho tam také. Posílají se notifikace dvě, jedna Web Push a druhá na Slack. Web Push notifikace spočívá v pouhém spuštění souboru PNPush.php ve složce config (jedná se o část PNServeru, lehce předělaná, aby fungovala s PhisherFriendem) společně s proměnnou obsahující notifikační zprávu. Slack notifikace je složitější, u ní je potřeba vytvořit cURL array, obsahující zprávu a hodnoty, které získáme z nastavení uživatele, jako Slack webhook URL, nebo Slack bot token. Správně nastavený takovýto cURL array je postován na Slack webhook, a pokud zbytek hodnot je nastaven správně, odešleme Slack notifikaci.

```php
//inicializace cURL k odeslání Slack notifikace
$curl = curl_init();
curl_setopt_array($curl, [
    CURLOPT_URL => $SlackWebhookURL,
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_CUSTOMREQUEST => 'POST',
    CURLOPT_POSTFIELDS => json_encode([
        "channel" => $SlackChannel,
        "username" => $slackbotname,
        "text" => $message,
        "icon_emoji" => $slackemoji,
    ]),
    CURLOPT_HTTPHEADER => [
        "Authorization: Bearer {$SlackBotToken}",
        'Content-Type: application/json; charset=utf-8',
    ],
]);
curl_exec($curl);
curl_close($curl);
```

Obrázek 20 Odesílání Slack notifikací pomocí cURL a dat z databáze

5.2.10 Složka Results

Složka results obsahuje veškeré soubory, které zajišťují zobrazování výsledků úspěšných projektů (spoofovaných stránek/ukradených údajů). Obsahuje soubory index.php, kde se děje většina funkcionality této složky, a soubor audit.php, který získaná data z HIBP databáze převede na hezčí formu.

Index.php obsahuje <select> element, navedený na databázi, jehož možnosti jsou projekty/spoofované stránky, které zrovna přihlášený uživatel založil, a které mají nějaké výsledky. Po vybrání projektu dostaneme list výsledků, s různými údaji složenými do sloupců a seřazený podle nejnovějších výsledků. Zobrazené hodnoty jsou uživatelské jméno, heslo, přesný čas kdy PhisherFriend získal údaje, IP adresa, ze které údaje přišly, název projektu, do kterého tyto údaje náleží, a pokud byl zadán, multifaktorový autentifikační klíč. Tento list je doprovázen dvěma tlačítky, jedno je pro smazání všech výsledků vybraného projektu (původně jsem přemýšlel o jednotlivých tlačítkách u každého výsledku, ovšem k tomu by byl potřeba javascript, a také mi to upřímně nedávalo smysl, proč by uživatel chtěl

vymazávat pouze některé výsledky), které všechny vybrané výsledky vymaže z databáze, a tlačítko pro audit získaných hesel, který uživatele vezme do druhého souboru složky results.

Audit.php je soubor obsahující přehlednější zobrazení hesel z výsledků vybraného projektu, kde se dává pozornost především na různé elementy těchto hesel. Využívá k tomu data z databáze, které jsme získali při úspěšném odchytnutí uživatelských údajů ve složce /receive/. První částí je přehled délek hesel v daném projektu, a počet hesel v projektu s danou délkou. Následuje přehled opakujících se hesel pouze v rámci projektu, a kolikrát se dané heslo v projektu opakovalo. Nejzajímavější částí auditu je následující přehled, který využívá dat, které jsme získali z HIBP databáze: u každého hesla uvidíme hodnotu, která značí, kolik leaknutých účtů (které jsou v HIBP databázi) používalo to samé heslo, které jsme my získali úspěšným phishingem. Poslední částí je zobrazení kolik phishingem získaných hesel se již objevilo v nějakém leaku, a tudíž jsou kompromitovaná (ve formátu například 10/10). Chtěl jsem zde také přidat relativní sílu získaných hesel, za pomocí MySQL funkce `VALIDATE_PASSWORD_STRENGTH`, ovšem tato funkce byla odebrána v nejnovějších verzích MariaDB. Celkově je audit.php akorát přehlednější a uživatelsky přívětivější způsob, jak se uživatel může podívat na data ohledně získaných hesel, což by se dalo brát jako ta nejdůležitější věc, kterou se phishingem snažíme získat.

5.2.11 Složka Campaigns

Do teď veškeré složky a soubory měli co dočinění s jednou polovinou funkcionality PhisherFrienda, kterou je tvorba spoofovaných stránek, jejich hostování a získávání ukradených údajů. Hlavním důvodem k probírání této části PhisherFriend jako první, je protože se jedná o logický postup, kterým by nový uživatel PhisherFriend postupoval. Funkcionalita spoofovaných stránek komplementuje a určitým způsobem i umožňuje správnou a plnou funkcionalitu druhé půlky funkcionality platformy. Tímto tedy přestupujeme k druhé polovině PhisherFriend funkcionality, kterou je tvorba, potvrzení a odeslání phishingových a spearphishingových kampaní. Složka campaigns je domovem této funkcionality, jedná se o velice obsáhlou složku (složka config byla také velice obsáhlá, ovšem to bylo především kvůli souborům PNServeru) s mnoha soubory, některými zařizujícími funkcionalitu samotných kampaní, některé zařizují podpůrnou funkcionalitu spearphishingu pomocí seznamů obětí, a samozřejmě také zobrazení výsledků.

Uživatel je schopen založit si emailovou phishingovou, či spearphishingovou kampaň. Podle tohoto výběru se liší způsob, jakým se samotná kampaň tvoří. Nejdříve bych chtěl začít částí

spearphishingovou, zejména z důvodu, že potřebuje dodatečný input od uživatele předtím, než by byl schopen takovou kampaň založit: potřebuje seznam obětí. Victimlist.php je soubor obsahující menu funkcionality seznamů obětí. Vizuelně victimlist.php připomíná předchozí soubory jako například výsledky projektů. Uživatel má zde buď možnost vytvořit nový seznam obětí, čímž bude přenesen do souboru newvictimlist.php, který zajišťuje tvorbu zbrusu nových seznamů obětí, nebo si může vybrat jím už vytvořený seznam, zkontrolovat oběti v něm zapsané, případně jednotlivé oběti (nebo celý seznam) vymazat z databáze. Newvictimlist.php obsahuje formu s <input> elementem pro název celého nového seznamu obětí, a pro zapsání jména a příjmení oběti společně s jejich emailovou adresou. Po odeslání formy se zkontroluje, zda byl napsán název celého seznamu kód projde všechny políčka emailových adres, a zjistí, zda formát emailových adres je správný (toto je implementováno pomocí PHP funkce `filter_val($email, FILTER_VALIDATE_EMAIL)`). Pokud ano, zapíšeme nejdříve nový seznam do databáze, a poté jednotlivé oběti s odkazem na název seznamu, abychom mohli tyto tabulky spojit. Newvictimlist.php obsahuje dodatečnou JavaScript funkcionalitu kvůli lepšímu UX, při tvorbě nového seznamu obětí si může uživatel dynamicky přidat více políček pro více obětí, nebo případně určité políčko vymazat. Toto nebylo potřeba při mazání jednotlivých obětí z již hotového seznamu, toto mazání jsem zařídil skrytým <input> elementem s hodnotou přesného času, kdy daná oběť byla přidána do databáze (toto uzavřené uvnitř while cyklu, který nám zaplňuje <select> element daty z databáze), což nám umožňuje jednoduše navést smazání na specifickou oběť. Dostal jsem se tímto rozhodnutím do humorné situace, kdy mi kliknutí na smazání specifické oběti smazalo oběti všechny z určitého seznamu. Byl jsem lehce mystifikován, předtím než jsem si uvědomil, že kód v newvictimlist.php probíhá tak rychle, že se mi klidně všechny oběti uloží do databáze v tu samou vteřinu. Z tohoto důvodu jsem přidal jedno vteřinové čekání (PHP funkce `sleep()`), čímž se všechny oběti uloží v jiný čas. Kvůli tomuto jsem musel v nastavení NGINX a PHP-FPM změnit hodnotu timeout, která určuje kolik času webserver dává PHP k projetí dané funkce, aby uživatel tvořící velice dlouhý seznam obětí nedostal error kód 504 Gateway Timeout.

Vytvořením nového seznamu obětí jsme si umožnili funkcionalitu tvorby spearphishingových kampaní. Jak již víme z teoretické části, spearphishingové kampaně/útoky jsou mnohem více personalizované, nežli phishingové. Se stejnou filozofií jsem vytvářel formy na tvorbu potvrzování a odesílání naších kampaní. Uživatel si v souboru campaignlist.php složky campaigns (který funguje jako menu kampaňové části pro uživatele, či

administrátora) může vybrat, zda chce vytvořit phishingovou, či spearphishingovou kampaň, a nebo si může vybrat administrátorem již potvrzené kampaně, které může odeslat (více o tomto později). Ať už si uživatel vybere phishingovou, či spearphishingovou kampaň, v obou případech bude poslán do souboru emailform.php. Emailform.php zařizuje prvotní tvorbu emailové kampaně. Obsahuje ohromnou formu, která zajišťuje zápis veškerých dat, které potřebujeme k úspěšnému odeslání SMTP emailu. Začíná <select> elementem pro SMTP profily, který je dynamicky naplněn uloženými daty databáze, hlavně SMTP mailhosty společně s emaily, ze kterých bude finální kampaň odeslána. Tyto profily mohou být zapisovány pouze administrátory v jejich nastavení, a jsou poté sdíleny pro všechny uživatele k tvorbě kampaní. Následuje jméno kampaně, kde si uživatel jednoduše pojmenuje, jak chce aby se jeho kampaň jmenovala. Další <input> elementy jsou informace nutné k samotnému odeslání kampaně: jméno odesílatele, které se ukáže příjemci při přijetí emailu, reply-to, které se normálně používá k zadání emailové adresy, která bude použitá jako příjemcova adresa, v případě, že by oběť použila funkci emailových klientů reply (toto pole jsem nastavil jako nepotřebné, protože zaprvé neočekáváme emailovou odpověď od oběti, a zadruhé, poněkud překvapivě, kampaně, které jsem poslal se zadaným reply-to měli horší reputační skóre jak emaily bez zadaného reply-to, ve službách jako SpamAssassin), a titulek (subject) emailů v rámci kampaně. Dalším elementem je file input pro nahrání přílohy. Kampaňová forma podporuje jednu přílohu, více jsem nebyl schopen přes PHPMailer rozchodit, ovšem si myslím, že jedna bude více než stačit, vzhledem k tomu, že se spíše snažíme přinutit oběť kliknout na náš odkaz. Následuje další <select> element, který je naplněn spoofovanými stránkami, které uživatel tvořící kampaň úspěšně začal hostovat na našem NGINX webserveru, za pomoci tvorby server bloků. Uživatel následuje textové instrukce ve formě, které mu říkají, že si má vybrat projekt/spoofovanou stránku, kterou chce použít v této emailové kampani ze <select> elementu, a poté samozřejmě musí stejnou stránku zapsat do samotného těla emailu. Zde nastupuje editor těla emailu, WYSIWYG editor TinyMCE. Původně, v rámci redukce JavaScriptu v platformě jsem měl základní textarea element, ovšem přišlo mi, že k tvorbě emailů, kterými se snažíme za pomocí sociálního inženýrství manipulovat oběti, byla tato možnost příliš primitivní. Proto jsem se rozhodl jej nahradit již zmíněným TinyMCE, což je velice dobrý WYSIWYG textový editor, podporující mnoho pluginů, ze kterých nám některé značně usnadní způsoby jak udělat hezky vypadající, a věrohodný email. I sám o sobě, bez dodatečných pluginů je TinyMCE velice uživatelsky příjemný editor, obsahující stejnou funkcionalitu, kterou by uživatel očekával v jiných emailových

klientech, jako například Outlook. Lze v něm bez dodatečných pluginů měnit písmo na kurzívu a tučné písmo, zarovnat text doleva, doprava, centrálně a na celou šířku, a i stylisticky měnit text do nadpisů (veškerá tato funkcionalita spočívá v přidávání HTML tagů do daného textu, jak lze zkontrolovat uvnitř editoru za pomocí pluginu "code") TinyMCE se musí inicializovat jako script v HTML hlavičce souboru, kde se také dá samotný TinyMCE velice přizpůsobovat. Mezi nutnými parametry je selector daného HTML elementu, který chceme nahradit TinyMCE editorem, licenční klíč (který je potřeba v případě používání TinyMCE ke komerčním účelům, pokud bychom chtěli prémiové pluginy, či v případě, že bychom chtěli aby TinyMCE byl spuštěn z cloudu, tudíž my jej nepotřebujeme a stačí jej nastavit jako gpl, podle licence GPL verze 2) a pluginy, které chceme v naší verzi TinyMCE použít. Pro nás jsou nejdůležitější pluginy "link" (který nám dá submenu, do kterého dáme URL naší spoofované stránky, a můžeme toto URL nahradit jakýmkoliv textem, který bude fungovat jako hyperlink na naší spoofovanou stránku, například nahradit URL textem "KLIKNĚTE ZDE A PŘIHLAŠTE SE"), "code" (umožňuje nám zobrazit source kód emailu, případně změnit elementy za pomocí HTML kódu), "preview" (není úplně potřeba, ale umožní nám zobrazení celého emailu v přehledném okně) a "image" (umožňuje nám vložení/embed obrázku přímo do těla emailu, společně s editací velikosti obrázku). Vzhledem k použití pluginu "image" potřebujeme více kódu uvnitř a mimo inicializační script TinyMCE, který nám zajistí úplnou chtěnou funkcionalitu. Mezi hlavní patří automatic_uploads direktiva, která zajistí, že se embedované obrázky automaticky uloží na webserver, aby mohly být odeslány ve finální kampaňové fázi, a images_upload_url direktiva, kterou navážeme na externí soubor imagehandler.php, který nám umožní zadat cílovou cestu uploadování souboru. Imagehandler.php je velice jednoduchý soubor, ve kterém je zadaná cesta, do které se má vložený obrázek uploadovat (v našem případě /images/, a za pomocí PHP kódu se zrovna nahraný obrázek (který se podle mého nastavení PHP v php.ini uloží do složky /temp/ v rootu PhisherFriend) zkopíruje do zadané složky a dočasný upload se automaticky vymaže. Poslední částí nastavení TinyMCE je file_picker_callback, kterým se nastaví část UI uvnitř TinyMCE, kterou použijeme k uploadování obrázků, a její funkcionalita. Pro tento účel jsem použil část ukázkového kódu z dokumentace TinyMCE, obsahovala vše, co jsem od tohoto pluginu chtěl. [53] Hned vedle editoru těla emailové zprávy jsem za pomoci Bootstrap CSS mřížek (za pomocí col-lg-X, neboli column large X, kde X je hodnota od jedné do dvanácti, jsme schopni jednoduše seřadit HTML elementy vedle sebe horizontálně tím, že hodnotou X určíme procentuelně kolik horizontálního místa element bude

zabírat) přidal legendu s proměnnými, které uživatel může použít v těle emailové kampaně. Tyto proměnné mohou uživateli pomoci při tvorbě co nejlepší manipulační zprávy pro jejich oběť. Mezi tyto použitelné proměnné patří například aktuální datum a čas (ve formátu den/měsíc/rok hodina/minuta/vteřina), buď celý, nebo jednotlivé části odesílacího emailu vybraného SMTP profilu (části před a po zavináči), stejně tak části nebo celý email naší oběti, náhodně vygenerovaná písmena, čísla, kombinace obou, a MD5. Kombinace těchto proměnných v těle phishingového emailu by mohla uživateli značně zvýšit šanci úspěšného phishingu (například vystrašením oběti za pomocí přesného času, kdy jim byl účet zablokován). Tyto proměnné jsou ústřední částí tvorby spearphishingových emailových kampaní, rozdíly vysvětleny později. Následující částí je text area pro zápis emailových adres příjemců/obětí. Do tohoto elementu se dá zapisovat mnoho emailových adres, oddělených řádky. Protože vypisování velkého listu emailových adres řádek po řádku by bylo poněkud otravné, přidal jsem tlačítko, které uživateli otevře v novém prohlížečovém tabu filtr/extraktor emailů. Podle názvu je tato forma rozdělená na dvě části, obě požadují buď zkopírování textu obsahujícího chtěné emailové adresy do příslušné textarea, a nebo upload textového souboru obsahujícího text s emailovými adresami. Poté si uživatel může vybrat ze dvou funkcí, nebo je použít dohromady: extraktor z textu extrahuje všechny emailové adresy (za pomocí regexu klasického formátu emailové adresy) a buď je vypíše do další textarea, nebo je zapíše do textového souboru, který si uživatel může stáhnout, záleží na volbě uživatele. Filtr emailových adres přečte seznam mailů a podle uživatelem zadaných domén (například pouze emailové adresy s doménou gmail.com, nebo pouze emailové adresy končící na .cz) vyfiltruje ze zapsaných emailů ty, které odpovídají zadaným parametrům. Výsledek, stejně jako u extraktoru, si uživatel může nechat vypsat do textarea, nebo stáhnout jako textový dokument. Uživatel samozřejmě tyto dvě funkce může kombinovat, například nahrát textový dokument obsahující text společně s emailovými adresami, ty si nechá extrahovat z textu, a extrahované emailové adresy může hned vyfiltrovat podle své libosti. Výsledky filtrování a extrakce jsou vygenerovaný řádek po řádku, stejně jak je potřeba zapisovat emailové adresy do kampaně, a tudíž uživatel může své výsledky přímo zkopírovat zpět do formy tvorby kampaně. Mezi poslední možnosti patřil výběr typu zprávy, za pomocí radio elementu, pro výběr mezi HTML zprávou a plain text zprávou. Vzhledem k využití editoru TinyMCE, který používá HTML tagy k tvorbě těla zprávy, a hlavně kvůli kampaňovým notifikacím, ke kterým se dostaneme později, jsem se rozhodl tento výběr odebrat, a nechat ve formě pouze jeden zaškrtlý radio button jako připomínku, že zpráva se bude odesílat jako HTML. Dalšími

možnostmi jsou dva <select> elementy, jeden pro výběr character setu (buď UTF-8, nebo ISO-8859-1) a encodingu (zakódování) zprávy, které může být 8 bit, 7 bit, binární, base64 nebo quoted-printable. Toto jsou všechny možnosti podporované knihovnou PHPMailer, ovšem pokud uživatel přesně neví, co dělá, doporučil bych tyto možnosti nechat na základním nastavení, tudíž UTF-8 a 8 bit. Následuje poslední zapisovací textový <input> element, do kterého může uživatel zapsat poznámku pro administrátora. Zde by bylo dobré, si načrtnout jak bude probíhat životní cyklus takovéto kampaně. Uživatel, který si založí novou kampaň, ať již phishingovou či spearphishingovou, ji nemůže ihned odeslat. Po "odeslání" jsou veškerá data, která zrovna uživatel zapisoval do této formy, validována a uložena v databázi (s mírnými rozdíly mezi phishingovou a spearphishingovou kampaní). Po úspěšném uložení do databáze, si kód z databáze zjistí administrátorské účty (nebo pouze jeden, záleží na administrátorovi, či chce být jediný a mít pouze svůj vlastní účet, či pokud chce tuto pravomoc dát ostatním) a z jejich uživatelského nastavení načte data, potřebná k odeslání Slack notifikace administrátorům. Tato Slack notifikace administrátora upozorní, že uživatel X vytvořil novou kampaň o jméně Y, případně s onou poznámkou od uživatele. V tomto bodě přechází kampaň z bodu vytvoření do bodu čekání na revizi. K administrátorské revizi se dostaneme velice brzo, proto si nejdříve povíme o poslední funkcionalitě formy na tvorbu kampaně. Jedná se o odkaz, který uživateli otevře v novém okně/tabu kontrolu blacklistů. Kontrola blacklistů vyvolá veřejnou IP adresu, na které se nachází náš webserver (za pomocí služby ipecho.net) [54] a zkontroluje zda blacklisty vypsané v seznamu obsahují DNS zápis naší veřejné IP adresy nebo ne. Seznam blacklistů je vepsán přímo v těle funkce, jednoduše se z něj dá vždy odebrat, nebo naopak do něj přidat. Pokud získáme odpověď, že se v určitém blacklistu vyskytuje náš DNS zápis, je u daného blacklistu jasně znázorněno, že jsme jím blacklistováni, naopak v případě, že se tento zápis u blacklistu neobjevil, označíme, že (ještě) nejsme zde blacklistováni. Tímto jsme probrali funkcionalitu formy na tvorbu kampaně phishingové, ovšem pokud uživatel před tvorbou vybere, že chce vytvořit kampaň spearphishingovou, forma bude mít jisté změny. Hlavní změnou je, že uživatel nedostane možnost vypisovat emailové adresy přímo do formy. Tato textarea je nahrazena elementem <select>, který z databáze do možností vypíše seznamy obětí, které daný přihlášený uživatel vytvořil.

```php
//SQL prodecura pro získání seznamů obětí specifického uživatele
$victimlistquery = "CALL GetUsersVictimLists('$usersID')";
//PHP mysqli prepared statement (proti SQL injection)
if($stmt = mysqli_prepare($DBConnect, $victimlistquery))
{
    mysqli_stmt_execute($stmt);
}
$result = $stmt->get_result();
//prázdná defaultní možnost
echo '<option value="" selected></option>';
//pokud již seznam byl vybrán, zobrazit ho
if(isset($_POST['victimlist'])&& $_POST['victimlist'] !== "")
{
    $isset = $result->fetch_assoc();
    echo '<option value="'.$_POST['victimlist'].'" selected>' . $_POST['victimlist'] . '</option>';
}
//pokud ne, vypsat všechny seznamy z prodecury jako možnosti <select> elementu
while ($row = $result->fetch_assoc()) {
    echo '<option value="'.$row['name'].'">' . $row['name'] . '</option>';
}
//uvolnit připojení databáze pro další SQL dotazy
mysqli_next_result($DBConnect);
```

Obrázek 21 Tvorba možností select elementu výpisem dat z databáze

Po vybrání a odkliknutí příslušného tlačítka, se mu pod tímto elementem vygenerují dvě textarea vedle sebe, jedna s vypsanými emailovými adresami obětí a druhá s jejich jmény. Pro jistotu jsou tyto elementy zamklé jako readonly, aby uživatel nechtěně neudělal někdo chybu v již vytvořeném seznamu (pro tyto účely máme victimlist.php a newvictimlist.php). Hlavním rozdílem v samotné funkcionalitě tvorby formy, krom tohoto výběru seznamu obětí, je přidání několika dalších proměnných, použitelných v těle emailové zprávy. Vzhledem k zvýšené personifikaci spearphishingových zpráv oproti zprávám phishingovým, mají všechny nové proměnné co dočinění se jmény našich obětí: lze přidávat jméno, příjmení, či celé jméno do těla emailu, která se po finálním odeslání zprávy dynamicky upraví na jméno oběti, podle emailu, který se zrovna odesílá. Tímto jsme schopni zacílit co nejvíce personifikovaných emailů, podobných jako známe z praxe, kdy často podobné maily používají alespoň příjmení na začátku zprávy, čímž se snažíme oběť ujistit, že vzhledem k tomu, že víme a používáme její jméno, že jsme dostatečně věrohodní. Zbytek spearphishingové verze této formy obsahuje identickou funkcionalitu co phishingová verze, obě jsou společně v jednom souboru (emailform.php). Po odeslání spearphishingové kampaně se narozdíl od phishingové do databáze zapíše i jméno seznamu obětí, který byl vybrán pro tuto kampaň (v následujících formách z tohoto získáváme samotné oběti, kterým emailovou kampaň odesíláme).

Tímto jsme dokončili první část ze tří potřebných k úspěšnému odeslání naší kampaně. V tomto bodě je potřeba administrátora, který dostal Slack notifikaci o existenci nové kampaně, která potřebuje jeho potvrzení. Administrátor po přihlášení se do PhisherFrienda hned v hlavním menu uvidí, zda existují nějaké tyto nové kampaně, a pokud ano, tak i jejich počet. Kliknutím na tlačítko u této informace (nebo vybráním tlačítka pro administrátorské nastavení) je přemístěn do souboru adminsmtp.php, nebo administrátorské nastavení. Zde Nastupuje poslední možnost tohoto souboru, kterou jsme si lehce načrtli ve vysvětlení tohoto nastavení. Jedná se o <select> element, ve kterém jsou možnosti vygenerovány za pomocí informací o kampaních z databáze. Administrátorovi se v možnostech zobrazí, zda se jedná o phishingovou či spearphishingovou kampaň, její název a jméno uživatele, který ji vytvořit. Po vybrání některé z těchto kampaní a potvrzení je přenesen do admincampaign.php zpět ve složce campaigns. Jedná se o formu, jejíž funkcionalitou je přehled nové kampaně a všech jejich částí. Vizuelně jsem schválně dělal všechny tři formy, které se starají o samotnou kampaň co nejvíce podobné, aby uživatele či administrátora nematli rozdílné rozložení stránek. Vytvořil jsem je ovšem ve svých třech souborech, kvůli oddělení funkcionality a přehlednosti kódu. Administrátorská forma pro potvrzení kampaní má stejně, jako ostatní soubory, do kterých má mít přístup pouze administrátor (nastavení adminsmtp.php, registrace registrationform.php a tato admincampaign.php) má kód ověřující oprávnění přihlášeného uživatele, zda se jedná o administrátora. Pokud ne, místo obsahu souboru získá pouze zprávu, že k této části platformy nemá přístup. Forma pro potvrzení kampaní má několik změn oproti formě k její tvorbě. Na vrcholku stránky jsem před vybraný SMTP profil navíc přidal informace o uživateli, který kampaň založil, především uživatelské jméno a email. Zbytek je graficky identický formě k tvorbě kampaně, ovšem všechny elementy jsou zamčené jako readonly, protože mi nepřišlo etické, že by administrátor mohl měnit části kampaně založené někým jiným (pokud má administrátor přístup k databázi, tak by samozřejmě mohl měnit data tam, ovšem toto alespoň odrazuje náhodné překlepy při kontrole). U konce formy má výpis poznámky, kterou zadal uživatel (pokud ji zadal, aby nemusel přehazovat mezi PhisherFriend a Slack notifikací, aby viděl danou zprávu), a místo pro zapsání své administrátorské poznámky. Forma končí dvěma tlačítky, buď kampaň potvrdit, nebo odmítnout. Potvrzení v tomto případě změní v databázi u dané kampaně pole reviewdone z 0 na 1, čímž se umožní uživateli, který tuto kampaň vytvořil, ji vybrat v následné formě k odeslání kampaně. Navíc se vyhledají z databáze jeho Slack údaje, a pošle se mu notifikace, že jeho kampaň byla potvrzená, společně se zprávou od administrátora, pokud nějaká byla.

Odmítnutí kampaně danou kampaň z databáze kompletně smaže, a uživateli pošle notifikaci o tomto faktu, společně s případnou zprávou od administrátora.

V případě potvrzení kampaně se dostáváme do třetího, finálního kroku tvorby kampaně: její odeslání. Toto je na uživateli, který kampaň založil, a dostal notifikaci, že je potvrzena. Poté co se přihlásí do PhisherFrienda, uvidí v hlavním menu, zda má potvrzené kampaně, případně kolik jich má. Může kliknout na přidělené tlačítko, či na tlačítko "Emailové kampaně", obě možnosti ho přesunou do menu kampaní v campaignlist.php. Zde si ze <select> elementu může vybrat svoji potvrzenou kampaň a potvrdit svůj výběr. Tato akce ho přenese do poslední kampaňové formy: sendcampaign.php. Graficky téměř identická formě samotné tvorby kampaně, ovšem navíc obsahuje zprávu od administrátora (opět aby nemusel přehazovat mezi platformou a Slackem). Veškeré elementy jsou opět zamklé jako readonly, nedávalo by smysl uživateli dávat možnost změnit něco, co již administrátor potvrdil. Uživatel má v této formě dvě možnosti: kampaň odeslat, nebo smazat. Smazání je zde v případě, že by si všiml nějaké kritické chyby (nebo si kampaň rozmyslel), a tato volba vymaže danou kampaň z databáze, stejně jako v případě, kdyby kampaň odmítl administrátor. Uživatel pouze získá zprávu, že kampaň byla smazána, a že má opustit tuto stránku. V případě, že uživatel vybere odeslání kampaně, začne probíhat hlavní logika této formy, samotné odesílání emailových zpráv za pomoci PHPMailer. Po CSRF token validaci se rozdělí výpis koncových emailových adres na jednotlivé adresy podle řádku (za pomoci PHP funkce `explode("\r\n", $sendto)`) a do proměnné si uložíme počet těchto emailových adres za pomocí funkce `count()`. Nastavíme si proměnnou $x na hodnotu 1, k tvorbě přehledného statusu odeslaných mailů společně s proměnnou držící hodnotu emailových adres. Předtím, než se dostaneme do cyklu odesílajícího samotný email, uložím si originální znění těla zprávy do proměnné, pro použití s embed notifikací. Většina zbytku logiky odesílání je ve foreach cyklu, díky kterému můžeme odesílat jeden mail za druhým z našeho listu emailových adres. První částí je vypsání čísla zrovna odesílaného mailu z hodnoty všech adres (ve formě například [1/4]), následované výpisem samotné adresy, na kterou zrovna odesíláme. Následuje rychlá kontrola emailové adresy, zda je ve správném formátu. Pokud ano, přechází se na nutná nastavení PHPMailer. Vyvoláme novou instanci třídy PHPMailer, do které budeme ukládat všechny nutné informace: zapneme odesílání přes SMTP, zapneme SMTP autentifikaci, předáme informace o SMTP mailhostu, přihlašovací jméno a heslo, zapneme TLS enkrypci, a nastavíme TCP protokol jako 587 (většinu těchto dat, jako SMTP profil, seznam obětí či použitá spoofovaná stránka máme již načtené z databáze).

Následuje načtení správné oběti ze správného indexu (v případě, že se jedná o spearphishingovou kampaň). Následuje zápis všech částí vytvořené kampaně do naší instance PHPMailer, především odesílací email, jméno odesílatele, email na který zrovna posíláme, že je email v HTML formě, přílohu (pokud existuje), titulek, charset a šifrování. Samotné tělo zprávy je trochu zajímavější. Do nové proměnné uvnitř cyklu uložím hodnotu originálního těla zprávy, a poté k ní připojím obrázkový element, 1x1 pixelů velký, kde zdroj obrázku je odkaz na naší doménu, složku campaigns, a se zapsanou emailovou adresou, na který zrovna odesíláme, společně s názvem kampaně. Toto připojené k originálnímu tělu emailové zprávy a nastaveno jako tělo mailu v PHPMailer nám společně s kódem v souboru index.php složky campaigns (ke které jsme se ještě nedostali) zajišťuje notifikaci, v případě, že oběť otevře náš email.

```php
//originální znění těla emailové zprávy
$mailcontent = $ogmailcontent;
//neviditelný obrázek, který nám pošle odpověď po otevření emailu obětí
$embed = " <IMG SRC=\"$Domain/campaigns?email=$email&campaignname=$campaignname\" height=1 width=1> ";
//připojení notifikačního obrázku ke zbytku těla emailu
$mailcontent .= $embed;
//zapsat tělo emailu do PHPMailer instance
$mail->Body = ClearFields($mailcontent, $email);
```

Obrázek 22 Embed obrázku pro získání notifikace po otevření emailu

Je zde nutné vždy vytvořit novou proměnnou, obsahující originální znění těla emailu, bez tohoto by se akorát v následujících emailech přidávali další a další vložené notifikátory. Následuje samotné odeslání emailu, které pokud je úspěšné, zobrazí nám zprávu o úspěchu a nastaví v databázi u dané kampaně hodnotu active na 1, čímž se znemožní její opětovné vybrání z menu kampaní. Tento cyklus se opakuje pro všechny ostatní zadané emaily/oběti. Po odeslání všech emailů v kampani lze přehledně vidět ve spodku formy všechny adresy, na které byl odeslán email, a zda se odeslání povedlo. Tímto jsme úspěšně ukončili cyklus tvorby a odeslání phishingové či spearphishingové kampaně, a můžeme čekat na naše výsledky.

Abychom z naších kampaní získali nějakou odezvu (krom potenciálního získání ukradených údajů z vložené spoofované stránky), potřebujeme kód, který odchytne náš vložený neviditelný obrázek a data spojená s ním. Toto se děje v souboru index.php složky campaigns. Vzhledem k tomu, že jsme ve zdroji našeho notifikačního obrázku poslali i název kampaně a email naší oběti, tak v případě, že by tato oběť klikla na náš email, získali bychom zpět odpověď se dvěma `$_REQUEST` superproměnnými, jednou pro email, druhou pro název kampaně. Toto je ošetřeno v if podmínce, pokud získáme obě tyto superproměnné, jsme si

jisti, že se jedná o oběť, která na náš email klikla. Nejdříve si uložíme, odkud byl tento soubor navštíven za pomocí `$_SERVER['HTTP_REFERER']`. Nejedná se o 100% jistotu, protože HTTP_REFERER může být spoofován [55], nám ovšem jde hlavně o to, aby se nejednalo o přístup zevnitř PhisherFriend. Proto za pomocí `!strpos($referer, "campaigns")` funkce zjistíme, zda se v URL nachází campaigns, název naší složky. Pokud ne, dá se předpokládat, že se jedná o legitimní otevření emailu obětí. V tom případě načteme data, která jsme poslala v notifikačním obrázku (emailová adresa, na který byla zpráva poslána a název kampaně) a externí data díky dalším parametrům superproměnné `$_SERVER` jako například REMOTE_ADDR, který nám dá IP adresu oběti a HTTP_USER_AGENT, který nám dá jejich user agent. Tyto informace uložíme do databáze do výsledků kampaní a příslušnému uživateli (kterému patří kampaň) pošle Slack notifikaci.

Poslední částí složky campaign je přehled výsledků kampaní v souboru resultslist.php. Tento soubor je velice podobný již probranému index.php ve složce results. Obsahuje <select> element s možnostmi kampaní, které přihlášený uživatel založil a poslal, a obsahují nějaké výsledky. Po vybrání takovéto kampaně získáme výčet výsledků kampaně, ve formě: přesný čas kdy oběť na email kliknula, IP adresa oběti, email oběti, kampaň do které výsledek patří a získaný user agent. Jako v případě výsledků projektů/spoofovaných stránek tento přehled obsahuje tlačítko k vymazání všech výsledků dané kampaně. Nevytvářel jsem způsob, jak vymazat celou kampaň (kromě zamítnutí adminem, či uživatelem v odesílací formě), která již byla aktivní, protože jsem v tom neviděl příliš pointu. V případě aktivní kampaně jsou emailové zprávy již odeslané, to už nezastavíme a výsledky by nám chodili i kdybychom vymazali kampaň z databáze, tudíž jsem nechal tabulku campaigns také jako jistou historii odeslaných kampaní. Navíc jsem vytvořil ve složce campaigns podsložku results se souborem index.php. Tento soubor je vyvolán uživatelem při kliknutí na odkaz ve Slack notifikaci, představený IP adresou oběti. Pokud na ni uživatel klikne, je převeden do tohoto souboru, který funguje jako výčet výsledků ze všech kampaní pro onen specifický email. Toto je jediná interaktivní součást Slack notifikací, kterou jsem tvořil, protože většina ostatních souborů má protekci vůči nepřihlášeným uživatelům, což by pro uživatele bylo velice otravné, narozdíl od tohoto souboru, který můžeme nechat odemknutý, protože je potřeba společně s odkazem na něj poslat danou IP adresu, ze které se zobrazí výsledky.

6 IMPLEMENTACE A TESTOVÁNÍ PLATFORMY

6.1 Implementace platformy

Vzhledem k tomu, že velkou většinu platformy jsem programoval přímo na virtualizovaném Debianu, dalo by se říct, že implementace platformy na testovacím prostředím byla úspěšně provedena. Je ovšem několik věcí, které musíme zařídit a zkontrolovat pro správný chod platformy v našem připraveném testovacím prostředí.

6.1.1 Potřebná nastavení testovacího prostředí

První konfigurační soubor, který je potřeba zkontrolovat, je php.ini v adresáři /etc/php/fpm/8.3/. Hlavní změna, kterou jsem provedl v tomto souboru, je nastavení uploadování souborů, především "upload_tmp_dir" direktivu, kterou jsem změnil na cestu /var/www/html/temp/, neboli složka temp v rootu PhisherFriend. Stejně tak jsem změnil maximální velikost uploadovaného souboru na 25MB, což je standartní maximální velikost příloh ve službách jako Gmail. Normálně by v tomto souboru také bylo nutné odkomentovat (a tím umožnit spuštění) pluginy, které potřebujeme ke správné funkcionalitě platformy. Těmito pluginy jsou curl, gmp, mbstring, mysqli a openssl, stejné které jsme instalovali po instalaci Debianu. Ovšem já jsem je v případě naší platformy musel opět zakomentovat, protože všechny tyto pluginy již byly nějakým jiným souborem zapnuté, PHP dávalo errory typu "tento plugin je již zapnutý v souboru Unknown (neznámý)" a ať jsem prošel veškeré PHP soubory, které jsou v těchto složkách a podobných, nenašel jsem instanci, kde by tyto pluginy byly odkomentované. Je možné, že byly automaticky zapnuté nějakým způsobem po instalaci těchto pluginů. Mimochoden i s těmito chybami PHP a platforma fungovali bez problémů, pouze bylo otravné dostávat toto upozornění při každém spuštění PHP v nějaké formě. Změnil jsem také "max_execution_time" neboli nejdéle, co může PHP script být spuštěn (v případě, že by například uživatel vytvářel velice dlouhý seznam obětí). Zbytek tohoto konfiguračního souboru se dá nechat s defaultními hodnotami, nutná další konfigurace je nutná pouze v případě, že byste chtěli nakonfigurovat debugger, jako například Xdebug. Ten jsem při programování používal, proto jsem na konec php.ini přidal několik parametrů, které vybrali, jaký debugger použít (pro PHP existuje Xdebug a Zend Debugger), v jakém módu má debugger fungovat (v našem případě debug), vnitřní IP adresa naší Debian instalace, a navedení Xdebugu na PhpStorm. Toto ovšem není nutné v případě pouhého používání platformy, a po ukončení programování se tyto hodnoty dají odebrat.

Druhým PHP souborem, který musíme zkontrolovat, je www.conf uvnitř /etc/php/8.3/pool.d. Zde je nutné přepsat vlastníka a skupinu vlastníků na náš účet phisherfriend a skupinu phisherfriend, a to jak u "user" a "group", tak u "listen.owner" a "listen.group". Tímto zajistíme správné spouštění PHP skriptů a oprávnění za použití námi vytvořeného účtu a skupiny.

Dalším souborem ke zkontrolování je nginx.conf uvnitř /etc/nginx/. Jedná se o centrální nastavení celého NGINX serveru. Hlavní změnou zde je opět přepsání "user" na phisherfriend phisherfriend, neboli účet a skupina. Další důležitou změnou, je zapsání naší cesty k novým NGINX server blokům za pomocí příkazu "include /var/www/html/blocks/*;". Toto jsou jediné změny, které v tomto souboru jsou potřeba.

Tímto jsme v teorii nastavili vše potřebné k tomu, aby platforma byla funkční. Problémem je, že v tomto momentě nám platforma běží v rámci localhostu. Tudíž potřebujeme doménu. Podle recenzí jsem si vybral hostovací službu dreamhost.com [56], na které jsem si vybral levnou doménu phisherfriend.xyz. Po koupi domény jsem za pomocí jejich nastavovacího interface nastavil DNS záznam typu A (IPv4 adresa), aby odkazoval na moji veřejnou IP adresu (vzhledem k tomu, že síť virtualizovaného Debianu máme v módu Bridge, čímž virtualizovaný Debian používá síťový adaptér hostového počítače). Zatim co jsem čekal, než se provede propagace DNS záznamů na internet, vytvořil jsem soubor "default" uvnitř /etc/nginx/sites-available/, který funguje jako NGINX server block PhisherFriend. Po tvorbě základního server bloku, jsem byl schopen připojit se k platformě přes http://phisherfriend.xyz. Toto je ovšem HTTP, a my chceme (a v jistých částech platformy potřebujeme) HTTPS. K vyřešení SSL certifikátu pro samotnou platformu PhisherFriend jsem využil certbot. Certbot je terminálový program, který automatizuje získání SSL certifikátu od služby LetsEncrypt, což je služba od neziskové organizace Internet Security Research Group (ISRG), která poskytuje volné SSL certifikáty zadarmo. Za pomocí apt příkazu `sudo apt install python3-certbot-nginx`, stáhneme verzi certbotu přímo pro NGINX, což nám ještě více zautomatizuje instalaci certifikátu. Spuštěním `certbot --nginx` začneme instalaci. Ta po nás bude chtít email (pouze zařizuje, že při vypršení certifikátu by nám poslali email jako připomínku), potvrdit podmínky a případě povolit sdílení vloženého emailu s Electronic Frontier Foundation (což z bezpečnostního hlediska nedoporučuji). Certbot se nás zeptá, pro jakou doménu chceme certifikát, načež začne kontrolovat, zda daná doména existuje a je aktivní (již běží na webserveru na portu 80, HTTP). Jako poslední část se nás certbot zeptá, zda chceme povolit i HTTP připojení,

nebo pokud chceme HTTP připojení redirectovat na HTTPS, z čehož jsem vybral druhou volbu. Tento proces nám nejen vygeneruje a nainstaluje potřebné certifikáty, změní i námi vytvořený server block pro PhisherFriend aby obsahoval dané certifikáty, poslouchal na portu 443, a v případě HTTP připojení je přehodí na HTTPS. Je zde ovšem problém, že certifikáty od certbotu fungují pouze 3 měsíce. Naštěstí v Linuxu jsme schopni i obnovu těchto certifikátů, za pomocí `sudo crontab -e`. Tento příkaz nám otevře soubor, do kterého můžete zapsat příkazy, které budou fungovat takzvané cronjoby. Cronjob zajišťuje automatizaci jakýchkoliv shell příkazů, které do tohoto souboru zapíšete. Proto do tohoto souboru zapíšeme cronjob, který každý měsíc zkusí obnovit certifikát naší hlavní domény: `0 0 1 * * certbot --nginx renew`. Tímto jsme zajistili téměř neomezenou HTTPS doménu pro naši platformu.

Stejně tak potřebujeme druhou doménu, za pomocí které můžeme otestovat tvorbu spoofovaných stránek a jejich hostování v rámci PhisherFriend. Díky Github Education nabídkám jsem byl schopen získat druhou doménu zdarma od společnosti namecheap.com. [57] Díky tomu jsem získat druhou doménu pftest.me, na zkoušení spoofovaných stránek. Ovšem hlavním důvodem k výběru této společnosti pro druhou doménu byla druhá nabídka od stejné společnosti, a tou je SSL certifikát od společnosti Sectigo, který by normálně stál 15 amerických dolarů ročně, v tomto případě zadarmo. Proč jsem byl obzvlášť rád, že tato nabídka existovala je, protože jsem potřeboval reálný SSL certifikát mimo automatizovaný certbot k otestování, zda jiné, nežli certbot certifikáty fungují, a také jestli naše funkcionalita tvorby NGINX server bloků s certifikáty funguje, jak má. Získání samotného certifikátu bylo překvapivě obsáhlá a zdlouhavá záležitost, ovšem je možné, že mi to tak přišlo pouze, protože jsem to dělal poprvé. Nejdříve na virtualizovaném Debianu jsem za pomoci openssl a RSA algoritmu vygeneroval CSR kód, neboli Certificate Signing Request. Při generaci těchto souborů po nás openssl chce několik informací, jako například lokalitu, název organizace, jméno domény a podobně, naštěstí většina z toho se dá přeskočit. Toto vygenerovalo nejen .csr soubor, nutný v dalším kroku k získání SSL certifikátu, ale také .key soubor, což je privátní klíč, který potřebujeme pro server bloky podporující HTTPS, tudíž je dobré tyto soubory zálohovat. Obsah .csr souboru se poté zkopíruje do InstantSSL (služba od namecheap) formy a vybere se způsob, jakým daný SSL certifikát validovat. Nejjednodušším je validace za pomocí CNAME záznamu v DNS, kdy po výběru této možnosti stačí přidat speciální vygenerovaný CNAME záznam do DNS, a počkat na propagaci. Jakmile se záznam propaguje, přijde email od Sectigo s přílohou se zazipovanými

soubory s příponama .ca-bundle a .crt. Tyto soubory se zkombinují (například za pomocí Linuxového příkazu `cat`) do jednoho souboru, a tento soubor je naším SSL certifikátem, společně s .key souborem z generace CSR kódu. [58]

Poslední záležitost, kterou potřebujeme vyřešit, je firewall. Pro tento účel jsem použit aplikaci ufw (Uncomplicated Firewall), jeden z nejjednodušších Linuxových firewallů k používání. Po jeho instalaci za pomocí apt jsme schopni jej plně nastavit pro naše účely za pomocí příkazů `ufw allow 80` a `ufw allow 443`. Tímto jsme nastavili firewall, aby povoloval připojení na HTTP a HTTPS porty, čímž budou lidi schopni připojit se na naši doménu. Ty samé porty se také musí otevřít v nastavení routerů, switchů a podobných zařízeních, které jsou mezi hostovým strojem, na kterém běží virtualizovaný Debian, a internetem. Toto je ovšem velice individuální, například já osobně s tímto měl větší problémy, protože k Vodafone (předtím UPC) internetu jsem připojen jejich modemem, který mám poté připojený k Asus routeru. S Asus routerem nebyly žádné problémy ohledně port forwardingu, ovšem Vodafone router urputně bojoval, a kvůli tomu jsem ho přehodil do bridge módu, ve kterém pouze přeposílá signál na další zařízení a nezajišťuje žádná nastavení sám o sobě. [59]

Tímto máme veškeré předpoklady vyřešené a platformu funkční na doméně https://phisher-friend.xyz. Díky tomu se můžeme přesunout na testování platformy.

6.2 Testování platformy

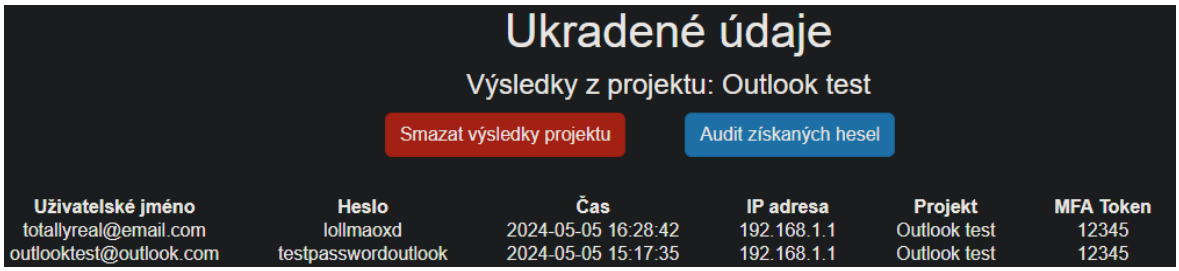
K testování funkcionality platformy jsem si předpřipravil dva účty: účet A je regulerní uživatel, a účet B je administrátorský účet. První zkouškou bylo otestování resetování hesla a odhlášení se z účtu. Obě funkce fungovali bez problému, nové heslo bylo zahashované, a odhlášení kompletně smazalo celý session. Za pomocí administrátorského účtu jsem do databáze uložil tři reálné funkční SMTP profily, jeden od Gmailu, dva od Mailtrap (jeden transakční, pro časově senzitivní maily jako reset hesla, a druhý hromadný, pro odesílání hromady emailových zpráv najednou). Přehlásil jsem se na uživatele A, abych platformu zejména testoval z pohledu normálního uživatele (ovšem veškerá funkcionalita PhisherFriend je umožněna i administrátorským účtům, například mohou si vytvořit svoji kampaň hned si ji potvrdit). Prvně otestuji funkcionalitu generování spoofované stránky. První krokem je upload nového templátu. Mám několik templátů na často používané stránky (twitter, outlook, facebook atp.) se starším layoutem než tyto stránky mají dnes (například přihlašování do twitteru narozdíl od nového X), ovšem vzhledem k tomu, že se platforma bude používat

na trénink znalostí o phishingu, říkám si že starším layoutem stránky můžeme lidem trochu pomoci zjistit, že se jedná o phishingový pokus. Uživatelé ovšem mohou vytvářet své vlastní templáty podle návodu, tudíž si kdokoliv může novější verze templátů. Vybral jsem například složku s templátem pro OWA (Outlook on the web), zazipoval jen ji a šel do menu nahrání nových templátu. Zde jsem podle instrukcí vyplnil jméno a popis templátu a uploadovat ZIP archív s templátem pojmenovaným template.php. Upload prošel úspěšně a já přešel ke generaci spoofované stránky z templátu. Zde jsem vypsal všechny potřebné informace, vybral si že chci a vyžaduji multifaktorovou autentifikace na spoofované stránce a že ji hodlám hostovat na našem webserveru. Spoofovaná stránka se úspěšně vygenerovala, je fyzicky ve správné složce /active/a/outlookontheweb/, kterou jsem pojmenoval při generaci stránky. Teď mám novou spoofovanou stránku Outlook test k výběru u stránek, které se dají hostovat na webserveru. Při tvorbě server bloku jsem zadal moji druhou doménu pftest.me, vybral k uploadu SSL certifikát a klíč a vybral, že chci stránku bez předpony www.. Úspěšně se přesunuly certifikát a klíč do svých složek v /ssl/, a uvnitř /blocks/ se vytvořil NGINX server block soubor pftest.me, ve kterém jsou správně zadána data, jako například root nové domény.

```
server {
    listen 80;
    server_name pftest.me;
return 301 https://pftest.me$request_uri;
}
server {
    server_name pftest.me;
    listen 443 ssl;
    index index.html index.php;
    root /var/www/html/active/a/outlookontheweb/;
    ssl_certificate /var/www/html/ssl/cert/pftest_me_chain.crt;
    ssl_certificate_key /var/www/html/ssl/key/pftest_me.key;

    location / {
    try_files $uri $uri/ =404;
    }

    location ~ \.php$ {
    include snippets/fastcgi-php.conf;
    fastcgi_pass unix:/var/run/php/php8.3-fpm.sock;
    }

    location ~ /\.ht {
    deny all;
    }
}
```

Obrázek 23 Nový vygenerovaný NGINX server blok soubor

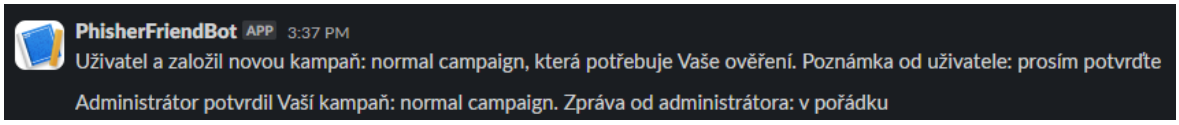
Vzhledem k tomu, že jsem použili NGINXový signál reload po vytvoření server bloku, nová doména pftest.me je funkční v pár vteřinách (pokud si uživatel již předpřipravil správné DNS nastavení). Po přihlášení získáme zapsaná data, získáme Web Push a Slack notifikaci a oběť, která zadala údaje je redirectována na stránku zadanou při tvorbě spoofované stránky (v našem případě reálná login forma do webového Outlooku. Pro oběť toto tudíž vypadá, jak kdyby se jim stránka pouze refreshovala (v případě, že spoofovaná a oficiální stránka vypadají identicky). My teď můžeme jít do výsledků našich projektů a po výběru daného projektu získáme výsledky úspěšného phishingu, včetně našeho nejnovějšího. Zde jsem také vyzkoušel funkcionalitu auditu hesel. Díky tomuto testu víme, že tvorba a hosting spoofovaných stránek, včetně odezvy ukradených údajů je funkční. Předtím, než daný projekt vyzkoušíme vypnout v hostingu a smazat projekt kompletně, využijeme jej v testování kampaňové části PhisherFrienda.



Obrázek 24 Výčet získaných údajů z úspěšného phishingu

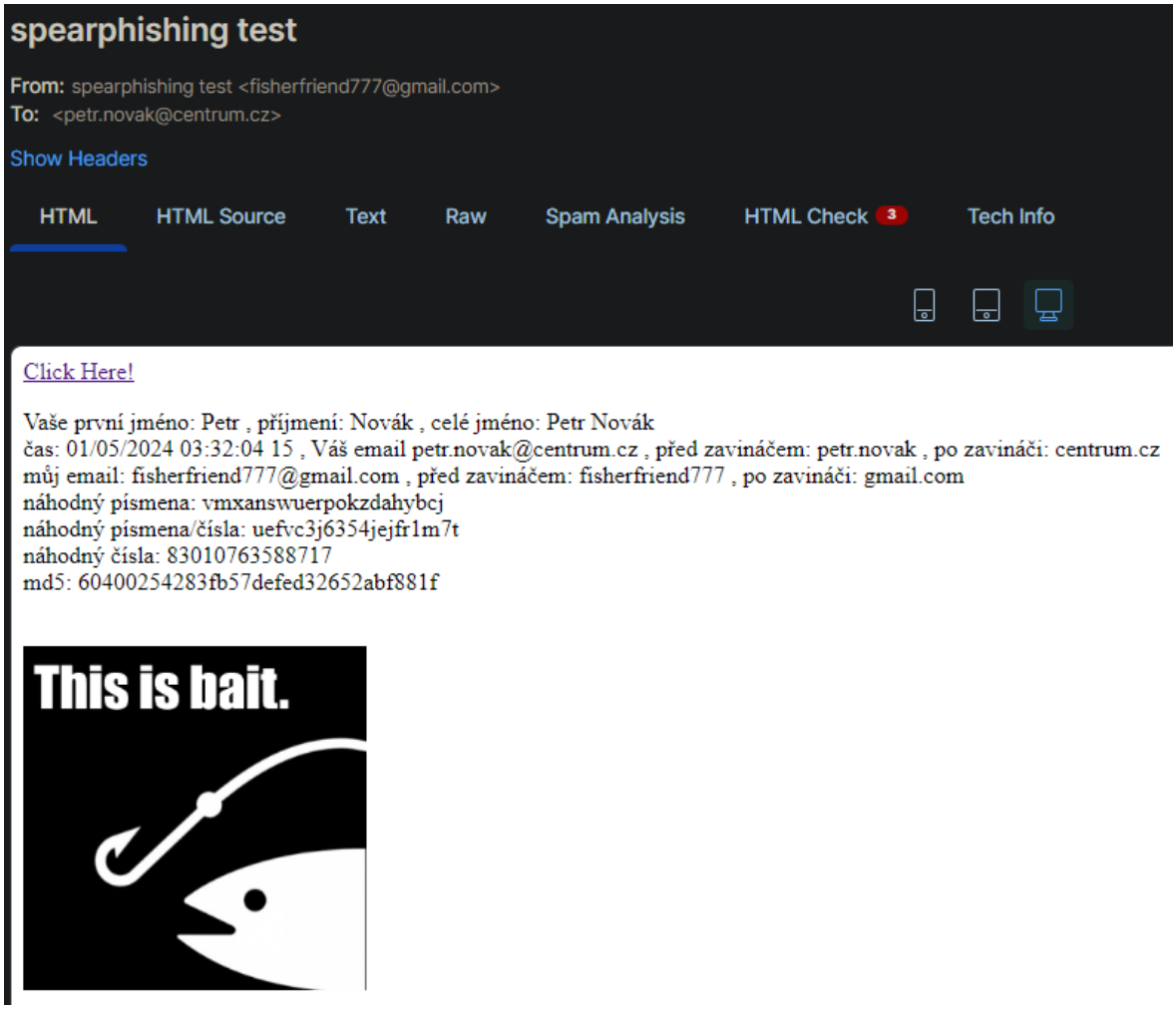
Vyzkoušíme oba typy kampaní, phishingovou a spearphishingovou. Vytvořil jsem si nový testovací seznam obětí obsahující, pro použití ve tvorbě spearphishingové kampaně. Po kontrole mazání jednotlivých obětí a celých seznamů, jsem dodatečně zařídil, aby při výběru prázdného listu byl daný list smazán. Nejdříve jsem vytvořil phishingovou kampaň, vyplněním všech potřebných elementů formy. Vybral jsem naší vytvořenou spoofovanou stránku, a její doménu za pomocí TinyMCE zamaskoval za text. Vypsal jsem do těla emailu většinu použitelných proměnných, a přidal obrázek za pomocí TinyMCE pluginu. Emailové adresy příjemců nejsou v tomto bodě tak důležité, protože kampaně testuji za pomocí email testing funkce společnosti MailTrap, jejíž údaje použijeme jako SMTP údaje pro PHPMailer a tudíž emaily z kampaně přijdou do virtuálního inboxu v MailTrap aniž bychom museli někoho reálně spamovat. Test s reálným SMTP profilem vyzkoušíme později. Po odeslání kampaně administrátorovi k potvrzení uděláme to samé s novou spearphishingovou kampaní, která obsahuje náš nový list obětí. Získali jsme z obou kampaní Slack notifikace míněné pro administrátora (kvůli testování jsem použil ty samé Slack údaje pro uživatele A a

administrátora B) a tudíž když se přehlásíme na administrátorský účet B, uvidíme v hlavním menu, že kampaně čekají na naši revizi. Vybereme kampaně uživatele A a úspěšně je potvrdíme. Toto nám dalo další Slack notifikace, míněné pro uživatele A.



Obrázek 25 Slack notifikace při založení a potvrzení kampaně

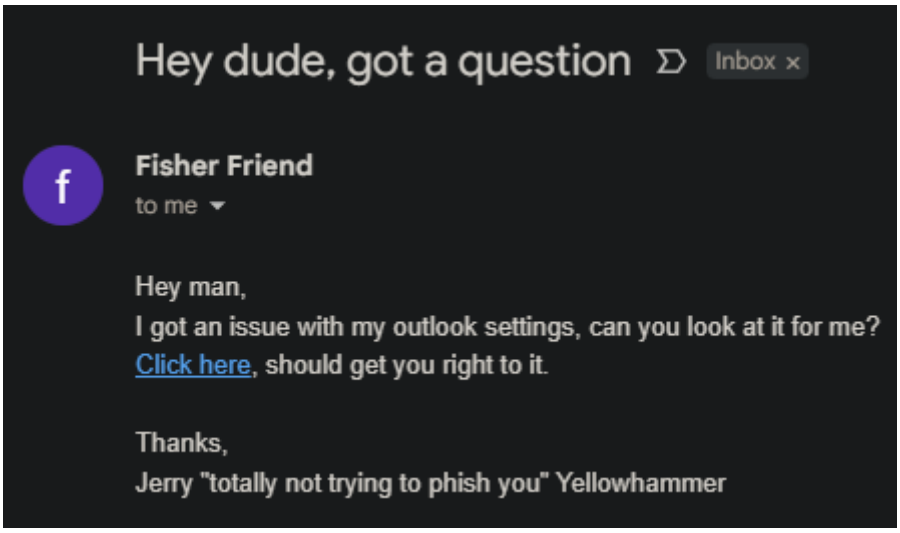
Po přehlášení zpět na uživatele A, vidíme naše potvrzené kampaně v tomu určeném <select> elementu. Po jejich vybrání máme poslední možnost zkontrolovat obsah naší kampaně a kampaň finálně odeslat. Hned po spuštění odelání vidím v MailTrap interface jak emaily naší kampaně přichází. Pokud vyberu jeden z těchto emailů, čímž ho otevřu, získám ihned Slack notifikaci, že email byl otevřen z dané emailové adresy, název kampaně, ze které daný otevřený email pochází, a kliknutelnou IP adresu oběti, která nás převede na přehled výsledků kampaní dané IP adresy, jak vysvětleno na konci kapitoly 5.2.11. Obsah emailové zprávy je správný, obsahuje hyperlink na spoofovanou doménu a proměnná byla změněna správně.



Obrázek 26 Přijatá emailová zpráva z kampaně

V MailTrap můžeme nahlédnout do source kódu emailu, a můžeme v něm vidět embedovaný neviditelný obrázek, který nám zaručuje naší notifikaci na otevřené emaily z kampaní. Můžeme také nahlédnout do “Spam Analysis”, což je projetí našeho emailu skrze Apache SpamAssassin (protispamová platforma od společnosti Apache), [60] která nám umožní nahlédnout do funkcionality antispamových filtrů, a různých částí emailů, které se těmto filtrům líbí a nelíbí. Po odeslání spearphishingové kampně výsledky proběhnout identicky. Největšími problémy, které SpamAssassin viděl v naších emailových zprávách, je fakt, že i když jsme odeslali email z Gmailového SMTP profilu, byla “odeslána” za pomocí MailTrap SMTP narozdíl od Gmailového, tudíž bere naší odesílací adresu jako spoofovanou. Toto není problém v případě, že bychom odeslali reálný email skrze celý nastavený SMTP profil. Dalšími zajímavými “problémy”, které SpamAssassin vidí, je posílání emailu, který obsahuje obrázek, když zbytek emailu obsahuje 0-400 bytů slov, či fakt, že naše odesílací adresa, fisherfriend777@gmail.com obsahuje čísla před doménou (smysl to dává, přidávání čísel na konci jména je nejjednodušší způsob, tak vytvořit adresu se jménem, které ještě nebylo

zabrané, ovšem bez tohoto přehledu by mě nenapadlo, že se v rámci filtrů jedná o špatnou věc). Všechny emaily ovšem i s těmito problémy nedostali tak špatné skóre, aby byly brány jako spam. Pokud se podíváme uvnitř PhisherFriend do výsledků kampaní, vidíme svoje kampaně k vybrání. Vybrání dané kampaně nám vypíše info všech emailů, které jsem otevřel. Dalším testem je poslání kampaně z reálného SMTP profilu na reálnou emailovou adresu. Po vytvoření a potvrzení nové kampaně jsem zakomentoval SMTP nastavení MailTrap, díky čemu bude SMTP server naší kampaně zadán informacemi z databáze, podle SMTP profilu, který vybereme (v našem případě Gmail SMTP). Odeslal jsem kampaň na svůj osobní emailovou adresu, a email z kampaně přišel téměř instantně, do normálního inboxu a ne spamu.



Obrázek 27 Přijatý email z kampaně odeslaný pomocí Gmail SMTP serveru

Otevření emailu mi opět získalo Slack notifikaci o otevřeném emailu, společně s moji reálnou IP adresou (předchozí IP výsledky byly localhost 192.168.1.1, protože jsem je otevíral uvnitř virtualizovaného Debianu). Stejně tak kliknutí na hyperlink mě přehodí do spoofované stránky na doméně https://pftest.me, a po jejím vyplnění získám zapsané údaje a notifikaci. Zajímavé je, že pokud je oběť již přihlášena do stránky, kterou máme zadanou jako redirect (například pomocí časté funkce přihlašovacích forem “pamatujte si mé přihlášení”), po získání údajů je oběť odeslána na tuto stránku již přihlášená, čímž se velice zvětšuje pravděpodobnost, že by oběť vůbec nevšimla, že zrovna na ní byl proveden úspěšný phishing. Poslední věc, kterou potřebujeme otestovat, je funkcionalita stáhnout hostovanou stránku. Vybrání spoofované stránky u které chceme vypnout hostování nám vymaže její NGINX server blok, SSL klíč a certifikát, a přepíše všechna data v databázi, které mají co společného s hostingem. Uživatel po refreshi stránky může teoreticky jít zpět do tvorby

server bloku a začít hostovat stejnou spoofovanou stránku na jiné doméně. Poslední částí je vymazání celého projektu, který kompletně vymaže záznam spoofované stránky z databáze.

ZÁVĚR

Podle výsledků testováních bych osobně označil platformu za úspěšnou. Obsahuje veškeré zadané a domluvené části, umožňuje administrátorovi/administrátorům a uživatelům vytvářet spoofované stránky a zasílat je v kampaních za cílem etického testování znalostí o phishingu dobrovolníků. Snažil jsem se platformu vytvořit co nejvíce uživatelsky přívětivou a jednoduchou k používání za pomocí vysvětlivek k většině funkčních částí, ovšem uvědomuji si, že po grafické stránce je podprůměrná. V průběhu tvorby platformy jsem si zopakoval a zlepšil své znalosti práce s Linuxem na distribuci, kterou normálně nepoužívám, naučil jsem se poprvé pracovat s hypervizorem VMware Workstation, a především jsem se naučil mnoho nových věcí ohledně jazyka PHP a způsobů, jak phishing probíhá. Velkým problémem při tvorbě platformy byl fakt, že phishing je celkově braný jako špatná a ilegální záležitost (což je) a kvůli tomuto bylo hledání jakýchkoliv informací o tom jak probíhá phishing po technické stránce velice obtížné (i AI jako ChatGPT jednoduše odpověděli, že se jedná o ilegální téma, které odmítají probírat), zatímco všechny phishingové informace byly o způsobech, jak se takovým útokům vyvarovat. Jednalo se také o největší program, který jsem do teď programoval, tudíž ke konci tvorby jsem se v kódy platformy pomalu začal ztrácet. I přes tyto problémy jsem ovšem velice rád, že je platforma funkční, v rámci požadavků práce. Hlavní problém ve funkcionalitě platformy, který jsem vypozoroval, je způsob získávání ukradených údajů ze spoofované stránky a následný redirect oběti. Problém v tomto případě je krátký redirect na doménu PhisherFriend, k získání údajů. Toto by mohlo oběti odhalit, že na nich byl proveden phishing, ovšem pokud by si toto uvědomili, bylo by již pozdě. Tento problém by se teoreticky dal vyřešit použití AJAXu oproti PHP, ovšem vzhledem k mým limitovaným znalostem JavaScriptu toto nejsem schopen říct najisto. Ovšem pokud by se pokračovalo pracovat na platformě, doporučil bych tento postup k teoretickému zajištění téměř nezjistitelného phishingu. Další doporučení by bylo zlepšení UI, nejlépe někým graficky nadaným. Předpokládám, že za použití JavaScriptu a například Bootstrap frameworku by se dala platforma rozšířit nejen o příjemněji vypadající UI, ale také například o grafické zobrazení výsledků v grafech, jak jsem viděl u jiné platformy Gophish. Největší věc, kterou jsem se při tvorbě práce naučil, a která mě překvapila, ovšem i odpověděla na otázky, které jsem ohledně phishingu měl, byla jak jednoduché je v teorii začít provádět phishing na lidech. Při mém těžkém hledání informací ohledně jak phishing provádět, jsem narazil na několik programů, které extrémně zjednodušují možnost osoby provádět phishingové útoky. Nemusejí být tak obsáhlé jak PhisherFriend, ovšem i tak jsou schopny

dosáhnout finálního výsledku phishingu, krádeži údajů. Tento fakt společně s možností, že manipulace lidí bude jednodušší za pomocí AI, mě vede k závěru, že phishingové útoky budou čím dál častější a především čím dál úspěšnější. Budoucnost phishingu pro normální lidi vypadá velice temně a proto doufám, že se všechny skupiny potenciálních obětí, ať již firmy, či samotní jednotlivci, se co nejvíce soustředí na problematiku phishingu a fakt, že se mohou stát cílem tohoto typu útoku každý den. Jak jsem načrtnul v teoretické části, věřím, že nejlepším způsobem jak lidi připravit na možnost phishingového útoku, je pravidelná edukace a trénink, a po znalostech, které mám po tvorbě této práce tomuto věřím o to víc. Phishing nikdy nezmizí, a proto je jen na nás se na boj proti němu připravit.

SEZNAM POUŽITÉ LITERATURY

[1] RITCHIE, Hannah; MATHIEU, Edouard; ROSER, Max a ORTIZ-OSPINA, Esteban. Internet. Online. Https://ourworldindata.org. 2023. Dostupné z: https://ourworldindata.org/internet. [cit. 2024-05-01].

[2] DEMOPOULOS, Alaina. 'Scanners are complicated': why Gen Z faces workplace 'tech shame'. Online. The Guardian. 2023. Dostupné z: https://www.theguardian.com/technology/2023/feb/27/gen-z-tech-shame-office-technology-printers. [cit. 2024-05-01].

[3] NIELES, Michael; DEMPSEY, Kelley a PILLITTERI, Victoria Yan. An Introduction to Information Security. Online. National Institute of Standards and Technology. 2017. Dostupné z: https://doi.org/10.6028/NIST.SP.800-12r1. [cit. 2024-05-01].

[4] A guide to leetspeak. Online. IONOS. 2021. Dostupné z: https://www.ionos.com/digitalguide/online-marketing/social-media/what-is-leetspeak/. [cit. 2024-05-01].

[5] REKOUCHE, Koceilah. Early Phishing. Online. In: Cornell University. 2011. Dostupné z: https://arxiv.org/abs/1106.4692. [cit. 2024-05-01].

[6] The history of phishing. Online. CaniPhish. C2024. Dostupné z: https://caniphish.com/phishing-resources/history-of-phishing. [cit. 2024-05-01].

[7] Power grid cyberattack in Ukraine (2015). Online. Cyber Law Toolkit. 2023. Dostupné z: https://cyberlaw.ccdcoe.org/wiki/Power_grid_cyberattack_in_Ukraine_(2015). [cit. 2024-05-01].

[8] GREENBERG, Andy. The Untold Story of NotPetya, the Most Devastating Cyberattack in History. Online. Wired. 2018. Dostupné z: https://www.wired.com/story/notpetya-cyberattack-ukraine-russia-code-crashed-the-world/. [cit. 2024-05-01].

[9] PHISHING ACTIVITY TRENDS REPORTS. Online. APWG. C2023. Dostupné z: https://www.antiphishing.org/trendsreports/. [cit. 2024-05-01].

[10] What is whale phishing? Online. IBM. C2024. Dostupné z: https://www.ibm.com/topics/whale-phishing. [cit. 2024-05-01].

[11] CEO Fraud. Online. Česká Spořitelna. C2024. Dostupné z: https://www.csas.cz/cs/o-nas/bezpecnost-ochrana-dat/ceo-fraud. [cit. 2024-05-01].

[12] Vishing a spoofing. Online. Policie České Republiky. 2021. Dostupné z: https://www.policie.cz/clanek/vishing-a-spoofing.aspx. [cit. 2024-05-01].

[13] What is Smishing and How to Defend Against it. Online. Kaspersky. C2024. Dostupné z: https://www.kaspersky.com/resource-center/threats/what-is-smishing-and-how-to-defend-against-it. [cit. 2024-05-01].

[14] WALLEN, Jack, ORTIZ, Sabrina (ed.). Quishing is the new phishing: Why you need to think before you scan that QR code. Online. ZDNET. 2023. Dostupné z: https://www.zdnet.com/article/quishing-is-the-new-phishing-why-you-need-to-think-before-you-scan-that-qr-code/. [cit. 2024-05-01].

[15] IPhone calendar spam: What it is, and how to remove it. Online. Malware Bytes. 2023. Dostupné z: https://www.malwarebytes.com/blog/news/2023/02/iphone-calendar-spam-what-it-is-and-how-to-remove-it. [cit. 2024-05-01].

[16] BOSTOGANASHVILI, Ketevan. The Whys and The Hows of Email Spam Filters. Online. MailTrap. 2024. Dostupné z: https://mailtrap.io/blog/spam-filters/. [cit. 2024-05-01].

[17] YUNG, Zakhar. Email Sender Reputation Made Simple. Online. MailTrap. 2024. Dostupné z: https://mailtrap.io/blog/email-sender-reputation/. [cit. 2024-05-01].

[18] What DMARC Can (& Can't) Do for Domains. Online. Fraudmarc. 2022. Dostupné z: https://fraudmarc.com/post/what-dmarc-can-cant-do-for-domains. [cit. 2024-05-01].

[19] MALEK, Piotr. BIMI: the New Word in Email Authentication. Online. MailTrap. 2023. Dostupné z: https://mailtrap.io/blog/bimi-email/. [cit. 2024-05-01].

[20] What Is Endpoint Security? Online. Trellix. C2024. Dostupné z: https://www.trellix.com/security-awareness/endpoint/what-is-endpoint-security/. [cit. 2024-05-01].

[21] Protect against consent phishing. Online. Microsoft Learn. 2023. Dostupné z: https://learn.microsoft.com/en-us/entra/identity/enterprise-apps/protect-against-consent-phishing. [cit. 2024-05-01].

[22] MOLINARO, Domenic. What Is a SIM Swap Attack and How Can You Prevent It? Online. Avast. 2023. Dostupné z: https://www.avast.com/c-sim-swap-scam. [cit. 2024-05-01].

[23] Inženýrství sociální. Online. Sociologická encyklopedie. Dostupné z: https://encyklopedie.soc.cas.cz/w/Inženýrství_sociální. [cit. 2024-05-01].

[24] Social Engineering Defined. Online. Social Engineer. C2023. Dostupné z: https://www.social-engineer.org/framework/general-discussion/social-engineering-defined/. [cit. 2024-05-01].

[25] HASELTON, M. G.; NETTLE, D. a ANDREWS, P.W. In: BUSS, David M. (ed.). The Evolution of Cognitive Bias. John Wiley & Sons, 2005, s. 724-746. ISBN 0-471-26403-2.

[26] Cítíte se občas jako podvodník? Nejste v tom sami. Online. Malé velké myšlenky. 2023. Dostupné z: https://www.malevelkemyslenky.cz/blog/citite-se-obcas-jako-podvodnik-nejste-v-tom-sami/. [cit. 2024-05-01].

[27] Jak využít kognitivní zkreslení ke zvýšení konverzí. Online. Ilinčev. C2024. Dostupné z: https://www.ilincev.com/kognitivni-zkresleni. [cit. 2024-05-01].

[28] Cybersecurity Survivorship Bias and How to Avoid it. Online. Infosecurity Magazine. 2021. Dostupné z: https://www.infosecurity-magazine.com/blogs/cybersecurity-survivorship-bias/. [cit. 2024-05-01].

[29] LIT LIMITED, Security. Cognitive Biases: How Social Engineering Exploits Human Decision-Making. Online. Medium. 2023. Dostupné z: https://osint-team.blog/cognitive-biases-how-social-engineering-exploits-human-decision-making-9f7ca9a25bed. [cit. 2024-05-01].

[30] Y, Jon. Is the AI Boom Real? Online. The Asianometry Newsletter. 2024. Dostupné z: https://www.asianometry.com/p/is-the-ai-boom-real. [cit. 2024-05-01].

[31] ElevenLabs. Online. C2024. Dostupné z: https://elevenlabs.io/. [cit. 2024-05-02].

[32] Nonconsensual deepfake porn is an emergency that is ruining lives. Online. The Guardian. 2023. Dostupné z: https://www.theguardian.com/commentisfree/2023/apr/01/ai-deepfake-porn-fake-images. [cit. 2024-05-02].

[33] CHIN, Neo Chai. When tech support is a scam: How India's call-centre scams became big money. Online. Channel News Asia. 2022. Dostupné z: https://www.channelnewsasia.com/cna-insider/tech-support-scam-baiters-india-call-centre-big-money-2876366. [cit. 2024-05-02].

[34] SURVEILLANCE UNDER THE PATRIOT ACT. Online. ACLU. C2024. Dostupné z: https://www.aclu.org/issues/national-security/privacy-and-surveillance/surveillance-under-patriot-act. [cit. 2024-05-02].

[35] FUNG, Brian. Biden just signed a potential TikTok ban into law. Here's what happens next. Online. CNN Business. 2024. Dostupné z: https://edition.cnn.com/2024/04/23/tech/congress-tiktok-ban-what-next/index.html. [cit. 2024-05-02].

[36] HTML: HyperText Markup Language. Online. Mozilla Developer. C2024. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTML. [cit. 2024-05-02].

[37] PHP. Online. C2001-2024. Dostupné z: https://www.php.net/. [cit. 2024-05-02].

[38] CSS: Cascading Style Sheets. Online. Mozilla Developer. C2024. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS. [cit. 2024-05-02].

[39] VMware. Online. C2024. Dostupné z: https://www.vmware.com/. [cit. 2024-05-02].

[40] HARDING, Scharon. Broadcom execs say VMware price, subscription complaints are unwarranted. Online. ARS Technica. 2024. Dostupné z: https://arstechnica.com/information-technology/2024/04/broadcom-execs-say-vmware-price-subscription-complaints-are-unwarranted/. [cit. 2024-05-02].

[41] Debian. Online. C2024. Dostupné z: https://www.debian.org/. [cit. 2024-05-02].

[42] PhpStorm. Online. JetBrains. C2024. Dostupné z: https://www.jetbrains.com/phpstorm/. [cit. 2024-05-02].

[43] PHPMailer. Online. Github. C2024. Dostupné z: https://github.com/PHPMailer/PHPMailer. [cit. 2024-05-02].

[44] PNServer. Online. Github. C2024. Dostupné z: https://github.com/Stefanius67/PNServer. [cit. 2024-05-02].

[45] KIENTZLER, Stefan. How to Use a PHP Push Notifications Class on Your Web Site in 2023 with this Send Web Push Notifications Tutorial. Online. PHPclasses. 2023. Dostupné z: https://www.phpclasses.org/blog/package/11632/post/1-How-to-Use-PHP-to-Send-Web-Push-Notifications-for-Your-Web-Site-in-2020.html. [cit. 2024-05-02].

[46] JavaScript. Online. Mozilla Developer. C2024. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript. [cit. 2024-05-02].

[47] Debian Releases. Online. Debian. C2024. Dostupné z: https://www.debian.org/releases/. [cit. 2024-05-03].

[48] How to install or upgrade to PHP 8.3 on Ubuntu and Debian. Online. PHP Watch. 2023. Dostupné z: https://php.watch/articles/php-8.3-install-upgrade-on-debian-ubuntu. [cit. 2024-05-03].

[49] Apache vs Nginx. Online. Váš hosting. 2019. Dostupné z: https://www.vas-hosting.cz/blog-apache-vs-nginx. [cit. 2024-05-03].

[50] Prepared Statements. Online. PHP. C2001-2024. Dostupné z: https://www.php.net/manual/en/mysqli.quickstart.prepared-statements.php. [cit. 2024-05-03].

[51] VASILEIOS, Grammatopoulos Athanasios. Php-csrf. Online. Github. C2024. Dostupné z: https://github.com/GramThanos/php-csrf. [cit. 2024-05-03].

[52] VERSHININ, Danila. NGINX and PHP-FPM. What my permissions should be? Online. GetPageSpeed. 2020. Dostupné z: https://www.getpagespeed.com/server-setup/nginx-and-php-fpm-what-my-permissions-should-be. [cit. 2024-05-03].

[53] Interactive example. Online. Tiny Documentation. C2024. Dostupné z: https://www.tiny.cloud/docs/tinymce/latest/file-image-upload/#interactive-example. [cit. 2024-05-03].

[54] IPecho. Online. Dostupné z: http://ipecho.net/. [cit. 2024-05-03].

[55] Referer spoofing and defeating the XSS filter (Edge/IE). Online. Broken Browser. C2024. Dostupné z: https://www.brokenbrowser.com/referer-spoofing-defeating-xss-filter/. [cit. 2024-05-03].

[56] Dreamhost. Online. C2024. Dostupné z: https://www.dreamhost.com/. [cit. 2024-05-03].

[57] Namecheap. Online. C2000-2024. Dostupné z: https://www.namecheap.com/. [cit. 2024-05-03].

[58] Installing an SSL certificate on Nginx. Online. Namecheap. C2000-2024. Dostupné z: https://www.namecheap.com/support/knowledgebase/article.aspx/9419/33/installing-an-ssl-certificate-on-nginx/#cmbn. [cit. 2024-05-03].

[59] SMITH, Luke. Certbot and HTTPS. Online. Landchad. 2023. Dostupné z: https://landchad.net/basic/certbot/. [cit. 2024-05-03].

[60] Apache Spamassassin. Online. C2003-2024. Dostupné z: https://spamassassin.apache.org/. [cit. 2024-05-03].

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AOL	America online
OPSEC	Operational security
RAT	Remote access trojan
VoIP	Voice over IP
APWG	Anti-phishing working group
URL	Uniform resource locator
SPF	Sender policy framework
DKIM	DomainKeys Identified Mail
DMARC	Domain-based Message Authentification, Reporting & Conformance
BIMI	Brand Indicator Message Identification
OTP	One-time password
MFA	Multi-factor authentification
AI	Artificial intelligence
LLM	Large language model
NFT	Non-fungible token
MIT	Massachusetts Institute of Technology
SMTP	Simple Mail Transfer Protocol
VAPID	Voluntary Application Server Identity
UX	User Experience
SSL	Secure Sockets Layer
LVM	Logical Volume Management
GUI	Graphical User Interface
IDE	Integrated Development Environment
CSRF	Cross-Site Request Forgery

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

PŘÍLOHA P I: ARCHÍV S OBRAZEM VIRTUALIZOVANÉHO DEBIANU S PLATFORMOU A ZDROJOVÝM KÓDEM

Zazipovaný archív obsahující obraz pro VMware Workstation Pro s virtualizovaným Debian operačním systémem a funkční platformou, zdrojovým kódem, SQL dump platformové databáze a několik předpřipravených templátů.