VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

SOFTWARE PRO ZACHYTÁVÁNÍ A INTELIGENTNÍ ZPRACOVÁNÍ SPAMU

SOFTWARE FOR CAPTURING AND INTELLIGENT PARSING OF SPAM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Bc. SILVIE CHLUPOVÁ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. LUKÁŠ ZOBAL**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2019/2020

Zadání bakalářské práce

Studentka: **Chlupová Silvie, Bc.**
Program: Informační technologie
Název: **Software pro zachytávání a inteligentní zpracování spamu**
Software for Capturing and Intelligent Parsing of Spam
Kategorie: Bezpečnost

Zadání:

1. Nastudujte problematiku honeypotů se zaměřením na SMTP honeypoty a seznamte se s existujícími nástroji.
2. Po konzultaci s vedoucím vybraný nástroj nasaďte na veřejně dostupný server, vyhodnoťte jeho použitelnost a navrhněte vylepšení, která povedou k efektivnějšímu sběru škodlivých vzorků.
3. Návrh implementujte a otestujte.
4. Vyhodnoťte nasbíraná data a dopad implementovaných vylepšení a uveďte výhled na další možný vývoj.

Literatura:

- Nawrocki, Marcin, et al. "A Survey on Honeypot Software and Data Analysis." arXiv preprint arXiv:1608.06249 (2016).
- Dinh, Son, et al. "Spam Campaign Detection, Analysis, and Investigation." Digital Investigation 12 (2015): S12-S21.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2, částečně bod 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Zobal Lukáš, Ing.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 22. října 2019

Abstrakt

Tato práce se zabývá tvorbou SMTP honeypotu, který bude připraven pro rychlé nasazení a bude podporovat pokročilé funkce. V práci je popsána teorie SMTP protokolu, POP3 protokolu a IMAP protokolu. Dále je v práci rozebrána problematika nevyžádaných e-mailů a boj proti nim. V práci jsou představeny různé druhy honeypotů a také existující řešení e-mailových honeypotů. Jedno z těchto řešení tato práce používá jako vzor. Nový honeypot podporuje autentizaci, ukládá e-maily do adresáře, odkud jsou postupně odebírány a analyzovány. Na základě analýzy jsou některé e-maily příjemcům přeposílány. Dále je možné honeypot jedním kliknutím nainstalovat a spustit. Honeypot také podporuje ničení obsahu e-mailů za účelem ochrany uživatelů.

Abstract

This work deals with the creation of an SMTP honeypot, which will be ready for rapid deployment and will support advanced features. The thesis describes the theory of SMTP protocol, POP3 protocol and IMAP protocol. Furthermore, the work discusses the issue of unsolicited e-mails and the fight against them. The work presents various types of honeypots as well as existing solutions for e-mail honeypots. One of these solutions uses this work as a model. The new honeypot supports authentication, stores e-mails in a directory, from where they are gradually removed and analyzed. Based on the analysis, some e-mails are forwarded to the recipients. It is also possible to install and run the honeypot with one click. Honeypot also supports the destruction of email content to protect users.

Klíčová slova

Honeypot, bezpečnost, SMTP honeypot, spam

Keywords

Honeypot, security, SMTP honeypot, spam

Citace

CHLUPOVÁ, Silvie. *Software pro zachytávání a inteligentní zpracování spamu.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Lukáš Zobal

Software pro zachytávání a inteligentní zpracování spamu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana Ing. Lukáše Zobala. Uvedla jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpala.

.....
Silvie Chlupová
28. května 2020

Poděkování

Tímto bych ráda poděkovala Ing. Lukáši Zobalovi za odbornou pomoc a cenné postřehy, které mi při psaní této bakalářské práce poskytl. Dále bych chtěla poděkovat doc. PhDr. Evě Minářové, CSc., za jazykovou korekturu práce.

Obsah

Kapitola 1

Úvod

S rozmachem internetu v 90. letech minulého století a především pak v novém tisíciletí si lidé po celém světě oblíbili elektronickou poštu jako snadný způsob komunikace.

E-mailová schránka dnes patří k naprostým samozřejmostem. Bez e-mailové komunikace by asi neexistovala většina dnešních firem a obchodů. S rozvojem e-mailové komunikace také docházelo ke stále častějšímu rozesílání zpráv obsahujících reklamu, které svoje tak dobře známé označení spam získaly až koncem 90. let [11]. Spam je nevyžádaná zpráva, která propaguje různé produkty a služby. Stane-li se vaše e-mailová schránka obětí spammerů, začnou se v ní pravidelně objevovat zprávy od osob, které se vydávají za vaše kamarády, rodinné příslušníky nebo známé firmy. Spammer je osoba zasílající lidem nevyžádanou poštu e-mailem.

S rozvojem informačních technologií se stala kybernetická bezpečnost velmi důležitým tématem. Přestože spam může být ve spoustě případů pouze neškodný e-mail, o který příjemce nemá zájem, může se objevovat i mnohem nebezpečnější forma spamu, a to phishingové e-maily nebo e-maily obsahující různé druhy malwaru. Phishing je snaha počítačových podvodníků získat vaše citlivé osobní informace. Malware, což znamená malicious software (z anglického překladu škodlivý software), je typ obtěžujícího nebo škodlivého softwaru, který má za úkol ovládnout vaše zařízení, nějakým způsobem ho poškodit, odcizit data nebo sledovat uživatele.

Pro efektivní obranu proti útočníkům je nutné znát jejich postupy a pro získání co nejvíce informací o útočnících je možné jako jeden ze způsobů použít honeypot. V překladu do češtiny se jedná o medový hrnec. Honeypot je výpočetní zdroj, která simuluje nějaký prvek nebo službu v síti (např. server, router) s částí jeho vlastností nebo i se všemi jeho vlastnostmi tak, aby útočník měl co nejmenší šanci přijít na to, že se jedná o honeypot. Druh honeypotu se určuje podle toho, jaký druh útočníků se snažíme přilákat a identifikovat. Obecně existují dva druhy útočníků. První druh jsou ti, kteří útočí se záměrem napadnout co nejvíce systémů bez ohledu na to, kdo je vlastní, a to s co nejmenším úsilím. To má i za následek, že se soustředí na cíle, které je snadné napadnout a u kterých je menší předpoklad, že budou schopni se bránit. Druhý druh útočníků má jasnější cíl, snaží se napadnout specifický systém nebo systém s vysokou hodnotou, např. databázový server nějaké firmy. Tento druh útočníků je mnohem zkušenější. Za jejich útoky bývá často skryt motiv nějaké finanční odměny nebo uznání [24] [16].

Tato práce se zabývá problémem e-mailových honeypotů a jejich podrobnější popis obsahuje část 3.3. V kapitole 2 je popsán e-mail, problematika spamu a jakým způsobem dnešní e-mailové servery proti spamu bojují. Kapitola 3 se zabývá honeypoty a podrobněji pak e-mailovými honepoty. Cílem práce je vytvořit efektivní nástroj na rozpoznávání a pře-

posílání spamu na základě toho, co e-mail obsahuje. Návrh řešení je popsán v kapitole 4. Vytvořený nástroj bude umět analyzovat obdržené e-maily pomocí nových modulů, které jsou popsány v kapitole 5, která se zabývá implementací honeypotu. V poslední kapitole 6 je pak popsáno testování a analýza zachycených dat.

Kapitola 2

E-mail

Slovo e-mail je zkratka ze slov electronic mail, což v překladu do češtiny znamená elektronická pošta. Jedná se o informaci uchovávanou na nějakém počítači, která může být následně vyměněna mezi dvěma uživateli přes internet. Na té nejvyšší úrovni se jedná o sérii znaků. Tyto znaky mohou být v případě různého kodóvání v různém rozmezí hodnot, např. v případě kódování US-ASCII (americký standardní kód pro výměnu informací) se jedná o znaky s hodnotami v rozmezí 1-127, kdy hovoříme o tzv. sedmibitovém kódu. Každá kombinace 7 bitů reprezentuje nějaký znak v kódové tabulce znaků.

Zpráva obsahuje hlavičky, za kterými následuje nepovinné tělo zprávy. Způsob, jakým se do zprávy hlavičky vkládají, je standardizovaný. Oproti tomu tělo zprávy je pouze sekvence znaků oddělených od hlavičkové části pomocí prázdného řádku. Jednotlivé řádky nesmí být delší než 998 znaků, je však dobrým zvykem nedělat řádky delší než 78 znaků pro lepší přehlednost e-mailu.

Hlavičky začínají názvem hlavičky, následuje dvojtečka a za ní tělo hlavičky. Ukázkou mohou být hlavičky specifikující odesílatele e-mailu (název hlavičky je From) a adresu příjemce (název hlavičky je To):

```
From: odesílatel
To: příjemce
```

Příjemců může být více, v takovém případě je možné uvést hlavičku:

```
Cc: seznam adres
```

Hlavička s názvem Cc určuje příjemce e-mailu, kteří obdrží kopii zprávy, ale nejedná se o hlavní příjemce e-mailu. Adres v této části může být více a jsou oddělené čárkou.

Doručovací adresa se skládá ze dvou částí:

1. Jméno příjemce, které není povinné uvádět.
2. Adresa příjemce v závorkách < a >.

Adresa příjemce začíná specifickým identifikátorem, který obsahuje jméno příjemce, jeho přezdívku, jméno firmy nebo oddělení a tak podobně. Identifikátor nesmí být delší jak 64 znaků. Dále následuje povinný znak `@`, který se smí objevit pouze jednou. Adresa končí internetovou doménou, ke které uživatel patří. Tato část nesmí mít více než 254 znaků.

Jako další nepovinná hlavička se může v e-mailu objevit předmět (anglicky Subject). Předmět zprávy nese informaci o zasílané zprávě a obsahuje text, který popisuje hlavní téma obsažené v těle zprávy.

Jednoduchá ukázka e-mailové zprávy, která má pouze hlavičky specifikující jednoho autora Pavla Nováka, jednoho příjemce Martina Nového a předmět. Za hlavičkami následuje tělo zprávy [21].

```
From: Pavel Novak <pavel@test.cz>
To: Martin Novy <martin@test.cz>
Subject: Testovaci email

Telo emailu.
```

2.1 Protokol SMTP

Základní protokol, který se používá pro zasílání elektronických zpráv, je SMTP, což znamená Simple Mail Transfer Protocol (v překladu jednoduchý protokol pro přenos e-mailu). Protokol zde znamená standard určující, jakým způsobem se budou elektronické zprávy zasílat. Hlavním cílem protokolu SMTP je spolehlivý přenos e-mailů.

Když má klient zprávu, kterou chce odeslat, tak naváže dvoucestné spojení s SMTP serverem. Odpovědností klienta je přenos e-mailu jednomu nebo více serverům, popř. oznámit chybu. SMTP server může být konečný cíl zasílané zprávy nebo jen prostředník mezi klientem a skutečným cílem zprávy. Klient zašle příkaz v textové podobě na server, ten zasílá odpověď zpět klientovi. Klient vždy čeká na odpověď serveru. Server odpovídá trojciferným číslem, např. `250 Message accepted`. Podle odpovědi je možné určit, že klientův požadavek byl úspěšný, server požaduje další informace, nebo že nastal nějaký problém. Text obsažený v odpovědi je obvykle pouze informativní. Číslo obsažené v odpovědi určuje její typ. Nejdůležitější je první číslice, protože ta určuje, co bude klient muset dále provést, aby byl e-mail odeslán [13]. V tabulce 2.1 je možné vidět, jak se liší význam odpovědí v návaznosti na první číslici v odpovědi.

Tabulka 2.1: SMTP kódy odpovědí od serveru

Kód	Význam
2xx	Požadavek byl úspěšný
3xx	Další informace jsou požadovány
4xx	Požadavek byl dočasně neúspěšný
5xx	Požadavek byl trvale neúspěšný

Odpovědi mohou obsahovat několik řádků textu. Všechny, až na poslední řádek, obsahují kód odpovědi oddělený pomlčkou od textu. Například:

```
250-Prvni radek
250-Druhy radek
250 Posledni radek
```

Když se klient připojí k serveru, tak musí počkat na odpověď, že příkaz byl úspěšný. Klient zahajuje komunikaci se serverem zasláním příkazu `EHLO`, za kterým následuje jméno klienta. Například:

```
EHLO client.test.cz
```

Server na tento příkaz odpovídá příslušným kódem, za kterým následuje jméno serveru. Dále může následovat nepovinný text a na dalších řádcích seznam rozšiřujících SMTP příkazů, které server podporuje. Odpověď může vypadat například takto:

```
250-server.test.cz Hello client.test.cz
250 SIZE 105868840
```

Část `SIZE 105868840` znamená, že server podporuje možnost SIZE s maximální možnou velikostí zprávy 105868840 znaků. Jakmile byl příkaz `EHLO` přijat a klient dostane odpověď začínající číslem 2, může začít zasílat zprávy příjemci. Taková zpráva začíná příkazem `MAIL FROM`, kde je popsáno, kdo se snaží zprávu zaslat a komu. Například:

```
MAIL FROM: <client@test.cz>
RCPT TO: <recipient@test.cz>
```

Server některé příjemce může odmítnout i přijmout. Odmítnutí může být trvalé, nebo dočasné, které může být vyřešeno. Naopak trvalé odmítnutí způsobí, že klient se musí pokusit o znovuodeslání e-mailu až při novém navázání spojení se serverem.

Potom, co klient zadá všechny příjemce a poslední z příjemců byl serverem přijat, klient pošle příkaz `DATA` a server odpovídá kódem 354 a s dalšími zaslanými řádky od klienta nakládá jako se samotným tělem e-mailu. Klient odešle zprávu bez dalšího čekání na odpověď serveru. Tato data musí být následně ukončena tečkou na samostatném řádku. Tečka indikuje konec zasílané zprávy. Server pak následně zpracuje uloženou informaci o e-mailové transakci. Server bere plnou zodpovědnost za doručení zprávy. Buď je zpráva celá přijata serverem, nebo je odmítnuta. Není možné jen částečné potvrzení. V případě přijetí zprávy odpovídá server `250 OK`. Jakmile klient odešle všechny zprávy, ukončuje spojení příkazem `QUIT` [12] [9].

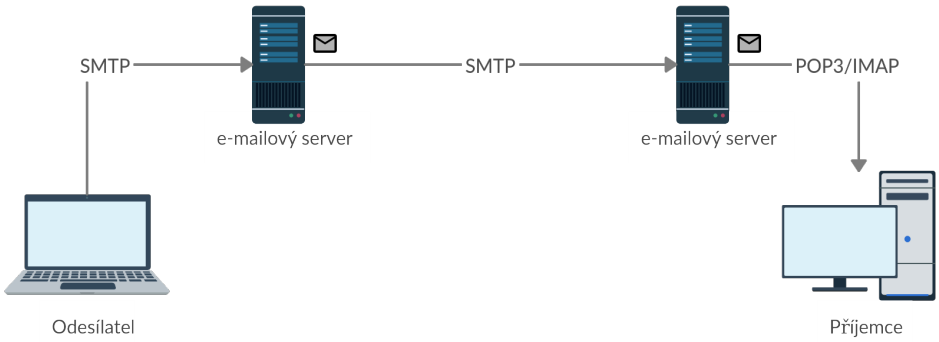
2.2 Protokol POP3

Jakmile je e-mail odeslán pomocí protokolu SMTP na e-mailový server příjemce, je zde uložen, dokud si ho příjemce nevyzvedne k přečtení. K získání e-mailů ze serveru se nejčastěji používají protokoly POP3 a IMAP. Na obrázku 2.1 je možné vidět použití e-mailových protokolů při zaslání e-mailu.

Protokol POP3 (Post Office Protocol, v překladu poštovní protokol) je velmi jednoduchý protokol zajišťující přístup k e-mailům. Vzhledem k tomu, že je tento protokol tak jednoduchý, je také jeho funkcionalita poměrně omezená. Na začátku uživatel spustí POP3 službu, která otevře TCP spojení na portu 110 s jeho e-mailovým serverem, kde doposud čekají e-maily.

Příkazy tohoto protokolu tvoří klíčové slovo, které je case-insensitive (nezáleží tedy na velikosti písmen). Některé příkazy také mohou být následovány argumenty. Každý argument může mít až 40 znaků. Oproti tomu POP3 odpovědi tvoří návratový kód, který udává úspěch či neúspěch příkazu, a klíčové slovo. Dále můžou následovat i dodatečné informace. Návratový kód určující úspěch je `+OK` a neúspěch `-ERR` [13].

POP3 postupuje během jednoho navázaného spojení několika stavy. Po navázání TCP spojení server odpovídá pozdravem `+OK POP3 server ready` a je možné postoupit do další fáze, kde probíhá autorizace, tj. ověření uživatele, který si chce přečíst e-maily. Ověření je



Obrázek 2.1: Základní model e-mailové komunikace

možné pomocí příkazů `USER` a `PASS` nebo pomocí příkazu `APOP`. Po autentizaci je uživateli podle poskytnutých údajů umožněn přístup do jeho e-mailové schránky a přechází se do stavu transakce. Po otevření schránky je každé zprávě přiřazeno číslo počínaje jedničkou. V této fázi je možné, aby uživatel zadal některý z příkazů `STAT`, `LIST [zpráva]`, `RETR zpráva`, `DELE zpráva`, `RSET`. Zatímco příkazy `STAT` a `RSET` podávají všeobecné informace o e-mailové schránce, zbylé příkazy pracují přímo s e-maily. Například `LIST [zpráva]` zobrazí informace o dané zprávě, `RETR zpráva` vrátí celou zprávu, `DELE zpráva` smaže danou zprávu. Tato fáze končí tak, že uživatel zadá příkaz `QUIT` a přechází se do fáze aktualizace. V této fázi dojde k odstranění všech zpráv označených k odstranění z e-mailové schránky serveru. V případě, kdy dojde k ukončení spojení se serverem jiným způsobem než zadáním příkazu `QUIT`, není možné se do této fáze dostat a je nemožné jakékoliv e-maily odstranit. Server musí odpovědět v případě obdržení neznámého příkazu odpovědí `-ERR`.

2.3 Protokol IMAP

Druhou volbou, jak si uživatel může vyzvednout došlou poštu, je protokol IMAP. Oproti protokolu POP3 čeká na spojení na portu 143 a má mnohem více užitečných vlastností, díky kterým je také složitější. Tento protokol je vhodný pro uživatele, kteří chtějí mít možnost si nějakým způsobem spravovat e-maily v adresářích. IMAP server přiřadí každému emailu adresář, odkud může být přesunut do jiného adresáře, který si uživatel vytvořil. Tento protokol také umožňuje prohledávat adresáře na základě nějakého kritéria. Protokol také umožňuje současnou práci více uživatelů se schránkami na vzdáleném serveru. Další důležitou vlastností IMAP je, že nenačítá celou zprávu, ale pouze její části, například pouze hlavičky. Až jakmile chce uživatel přistoupit ke zprávě, je načteno celé tělo zprávy. Zmenšují se tak potřebné datové přenosy [13].

Protokol začíná navázáním spojení klienta a serveru a úvodním pozdravem serveru. Následují příkazy klienta a odpovědi serveru. Každý příkaz klienta začíná různým identifikátorem, který obvykle obsahuje alfanumerické znaky jako `A0001`. Data přenesená serverem jako odpověď klientovi, která neindikují dokončení, musí začínat hvězdičkou `"*"`. Odpověď serveru při dokončení může značit úspěch, či neúspěch. Tato odpověď je označena stejným identifikátorem jako klientův příkaz, který začal komunikaci se serverem. To také znamená, že v případě, kdy je právě zpracováváno více příkazů, je možné určit, kterému patří která

odpověď. Server odpovídá `OK` v případě úspěchu, `NO` v případě neúspěchu a `BAD`, což značí obdržení neznámého příkazu nebo příkazu obsahujícího syntaktickou chybu [4].

Stejně jako tomu bylo u protokolu POP3, i zde se během spojení prochází několika fázemi. Po navázání spojení se dostává klient do autentizovaného stavu, kde může zadávat příkazy pro manipulaci se schránkou. Před začátkem práce si musí klient příkazy `SELECT` nebo `EXAMINE` vybrat schránku, se kterou bude pracovat. Po ukončení práce s touto schránkou příkazem `CLOSE` přechází zpět do autentizovaného stavu. V tomto stavu může vytvářet, přejmenovávat či rušit schránky (`CREATE`, `RENAME`, `DELETE`), označit schránku pro sledování (`SUBSCRIBE`, `UNSUBSCRIBE`) apod. Ve stavu Selected lze provádět operace nad zprávami dané schránky – můžeme je načítat (`FETCH`), prohledávat (`SEARCH`), rušit (`EXPUNGE`) apod. Po ukončení práce se schránkami se klient příkazem `LOGOUT` odhlásí a uzavře spojení [15].

2.4 Spam

E-mail je levnou formou komunikace, ale je i náchylný k různým bezpečnostním hrozbám. Jednou z nejčastějších hrozeb je spam. V podstatě pokaždé, kdy uživatel obdrží nevyžádaný e-mail s nějakým reklamním sdělením, tak se jedná o spam. Se vzrůstajícím počtem lidí s přístupem k internetu vzrůstá také množství spamu odeslaného každý den. Přes 80% odeslaných e-mailů tvoří spam. Některé z těchto e-mailů jsou neškodné a pouze slibují, že vyděláte spoustu peněz nebo rychle zhubnete. Některé však mohou být i nebezpečné a lákají vás na kliknutí na neznámý odkaz, stažení přílohy nebo zaslání osobních informací. Přestože firmy dnes používají filtry na spam, může se stát, že i tak se spam objeví v běžné obdržené poště. Filtry totiž většinou fungují na základě textu, který se objevuje v e-mailu, ať už v hlavičce, v předmětu nebo oslovení příjemce. Spammeři však přišli s ještě lepší taktikou, a to s vkládáním textu do obrázků a je tak nutné opět vyvinout lepší technologie pro rozeznání spamu od běžné pošty, kterou uživatel chce číst [6].

I běžný člověk může nevědomě zapříčinit, že množství obdrženého spamu se najednou zvýší. Spammeři často lákají ve svých e-mailech k tomu, aby lidé klikli na nějaký odkaz, aby se odhlásili z odebírání konkrétní nevyžádané pošty. Jedná se však pouze o trik, kdy uživatel pouze potvrdí, že e-mailová adresa skutečně existuje a někdo ji používá. Dnes jsou totiž běžně využívané programy, které generují náhodné e-mailové adresy, na které je následně odesílaná nevyžádaná pošta. Další variantou, která mohla zapříčit nárůst obdrženého spamu, je únik dat nějaké firmy. Pokud vaše osobní informace, jako např. heslo a e-mail, nikdy neunikly, je to pouze otázka času. Existují i webové stránky[1], kde je možné si ověřit, jestli už k takovému úniku vašich dat došlo. Často se také stává, že člověk zapomene odznačit v zaškrtávacím poli, že nemá zájem o reklamní sdělení [14].

Spammeři jsou velmi chytří a dokáží také využívat různých příležitostí, jako je den svatého Valentýna nebo v období Vánoc je snadné nalákat lidi na kliknutí na nějaký odkaz, aby si vyzvedli svůj dárek, který na ně už čeká. Asi nejnebezpečnější formou jsou v tom případě phishingové e-maily. Phishing je anglické slovo týkající se útoku na bezpečnost či soukromí, kde hlavní roli hraje to, že někdo „nahodí udičku a oběť na ni ochotně naskočí“. Původcem je fishing, chytání ryb a také proto se v češtině setkáte se slovem rhybaření. Phishing patří pod sociální inženýrství [7]. Sociální inženýrství je nástroj, kterým někoho zmanipulujete tak, aby udělal něco, co za normálních okolností neudělá. Cílem je zpravidla získat informace, ale také to, že cíl sociálního inženýrství vykoná nějakou aktivitu, kterou by normálně neudělal. Poměrně zásadním prvkem sociálního inženýrství v počítačovém

[1] Jednou z takových webových stránek je https://www.avast.com/hackcheck.

prostředí je to, že většina uživatelů nemá dostatečně znalosti k tomu, aby dokázala rozeznat nebezpečí [8]. Podvodníci se snaží lidi nalákat k tomu, aby jim sdělili osobní informace, jako jsou číslo bankovního účtu nebo heslo. Může se jednat i o varování, že se na vašem účtu objevila nečekaná aktivita a byla provedena nějaká platba v zahraničí a pro potvrzení této platby je nutné potvrdit číslo vaší karty. Jako varovné signály, že se jedná o phishing, může sloužit následující:

- E-mail od banky žádající o potvrzení vašich osobních údajů.
- E-mail není adresován vám nebo se v něm nachází chyba ve vašem jménu.
- Webová adresa vypadá podezřele a žádá o potvrzení informací o vás nebo kliknutí na nějaký odkaz.

Stále častějším druhem phishingu v posledních letech je tzv. fake CEO fraud, jedná se o označení pro podvody založené na cíleném poslání e-mailů s pokynem k platbě. Pokyn k platbě může být falešný nebo využívá existující fakturu, na které bylo změněno číslo účtu. Škody způsobené takovým typem podvodu pak šplhají do milionů [18].

Dalším typem nebezpečné nevyžádané pošty je malware. Existují různé druhy malwaru, jako např. spyware, ransomware, různé druhy počítačových virů. Malware často vyvíjí hackeři, kteří ho pak sami využívají nebo dále prodávají. Motivace jeho tvorby je tak nejčastěji finanční odměna. To, že se v zařízení nachází škodlivý software, trvá odhalit a je těžké se bránit. Nejlepší prevence je nikomu na internetu nedůvěřovat a neklikat na různé odkazy, nejdříve si zkontrolovat důvěryhodnost stránek, ze kterých stahujeme nějaký program, a stáhnout si některý z programů zamezujících otevírání vyskakovacích oken ve webovém prohlížeči [20].

Zasílání nevyžádané pošty může být v České republice i trestná činnost [27]. Úřad pro ochranu osobních údajů dohlíží na šíření obchodních sdělení podle zákona č. 480/2004 Sb., o některých službách informační společnosti a o změně některých zákonů (zákon o některých službách informační společnosti). Tento zákon zakazuje zasílání elektronické pošty za účelem šíření obchodního sdělení pokud:

- E-mail není zřetelně a jasně označen jako obchodní sdělení.
- Skrývá nebo utajuje totožnost odesílatele, jehož jménem se komunikace uskutečňuje.
- Je zaslána bez platné adresy, na kterou by mohl adresát přímo a účinně zaslat informaci o tom, že si nepřeje, aby mu byly obchodní informace odesílatelem nadále zasílány.

Nicméně spammeři po druhé uložené pokutě, tedy až na výjimky, s rozesíláním nevyžádaných obchodních sdělení přestanou, neboť počet nově získaných zákazníků je oproti uložené pokutě zanedbatelný [26].

Ověření odesílatele e-mailu

Čím dál běžnější praxí u velkých poštovních služeb, jako jsou například Google, Yahoo, Outlook, je dnes ověřování odesílatele e-mailu pomocí technologie DMARC, což je zkratka pro Domain-based Message Authentication, Reporting, and Conformance (v překladu ověřování zpráv, hlášení a shoda s doménami). Spamměři totiž ve spamu podvrhují pole `From`

(odesílatel zprávy) tak, aby to vypadalo, že zpráva byla odeslána od uživatele v nějaké známé doméně. Tato technologie může způsobit, že i když honeypot poštu příjemci skutečně odesílá, poštovní server příjemce ji odmítne. DMARC rozšiřuje SPF a DKIM. SPF je zkratka pro Sender Policy Framework (v překladu aplikační rámec politiky odesílatele), DKIM je zkratka pro DomainKeys Identified Mail (v překladu pošta identifikovaná doménovým klíčem).

SPF je jedno z opatření, jak zvýšit důvěryhodnost e-mailové komunikace pomocí speciálních autorizačních DNS záznamů. Principem SPF je dát možnost vlastníkovi DNS domény deklarovat, které servery jsou oprávněny odesílat e-mailové zprávy jménem domény. Příjemce pošty si pak může ověřit, jestli poštu od dané domény dostává z autorizovaného serveru, a podle toho se rozhodnout, co dále provést se zprávou.

Majitel domény `example.com` může například specifikovat politiku, že poštu z adres `@example.com` mohou předávat pouze servery z adresního rozsahu 192.0.2.0/24 takto:

```
1 example.com   IN  TXT  "v=spf1 ip4:192.0.2.0/24 -all"
```

Obsah TXT[2] záznamu obsahuje vždy povinné záhlaví `v=spf1` následované mezerou oddělenými slovy, která definují povolené, nebo naopak zakázané rozsahy adres.

Příjemce pošty, který SPF záznam ověřuje, tak činí ideálně ještě v průběhu SMTP komunikace. Nejprve by měl pomocí SPF ověřit jméno hostitele, kterým se představí klient v příkazu `HELO` nebo `EHLO`, následně ověří adresu domény, uvedenou v příkazu `MAIL FROM`. Pokud je na základě vyhodnocení SPF rozhodnuto o nepřijetí zprávy, měla by být odmítnuta již během SMTP komunikace.

SPF nijak nezkoumá ani hlavičky, ani vlastní obsah e-mailové zprávy. Proti šíření nejrůznějších podvržených zpráv, sloužících nejčastěji phishingu, které mají v pořádku obálkovou adresu odesílatele, žádným způsobem nepomáhá. Naopak k autorizaci obsahu zprávy zcela bez ohledu na to, jakým způsobem byla zpráva doručena, je zde DKIM [3].

DKIM je systém zabezpečení obsahu e-mailových zpráv před změnou prostřednictvím elektronického podpisu. Od zavedených systémů PGP[3] a S/MIME[4] se liší především tím, že podepisování může provádět kterýkoli článek e-mailové infrastruktury, takže není nezbytně nutné vyžadovat používání této technologie po uživatelích.

Samotný standard DKIM také na rozdíl od SPF nepředepisuje žádnou politiku, která by stanovovala, co se má stát se zprávami, jejichž elektronický podpis nevyhoví. Dokonce jeden z cílů DKIMu vyžaduje, aby zprávy s nevalidním DKIM podpisem byly hodnoceny stejně jako zprávy bez jakéhokoli podpisu. Standard DKIM byl primárně vyvinut jako vylepšení stávajícího systému s důrazem na možnost postupného nasazení bez rizika poškození funkčnosti stávající e-mailové infrastruktury.

Aby mohl DKIM opravdu pomoci proti šíření podvržených zpráv, definuje RFC 5617 politiku Author Domain Signing Practices (v překladu postupy při podepisování domén autora). Jedná se o další DNS záznam typu TXT na pevně dané subdoméně [2].

DMARC přidává možnosti pro hlášení výsledků při ověřování. Jeho klíčové vlastnosti jsou:

- Vychází z adresy odesílatele uvedené v hlavičce `From` dopisu a snaží se ověřit, jestli dopis byl skutečně odeslán z uvedené domény.

[2] DNS záznam, do kterého lze zadat libovolný textový řetězec.

[3] PGP je jednou z technik zajištění bezpečnosti e-mailových zpráv, která umožňuje šifrovat zprávy, autentizovat odesílatele a ověřit integritu e-mailu.

[4] S/MIME protokol je určen pro elektronický podpis a podepisování dokumentů.

- Využívá SPF a DKIM - pokud bylo splněno SPF odesílatelovy domény nebo dopis nese její ověřený DKIM podpis, považuje ji za důvěryhodnou.
- Definuje politiku pro neúspěšné dopisy - jak se zachovat k těm, které DMARC ověřením neprojdou.
- DMARC zveřejňuje držitel odesílající domény a ověřuje příjemce dopisu.

Pomocí SPF a DKIM si příjemce ověří, že e-mail vyhovuje pravidlům pro odesílání z domény podle `MAIL FROM` z SMTP komunikace. Dále si ověří, ze kterých domén má platné podpisy. Tyto informace se předají DMARC modulu, jenž zkoumá, zda domény ověřené v SPF nebo DKIM mají něco společného s doménou v hlavičce `From`. Pokud alespoň jedna uspěje, dá se podle DMARC věřit tomu, že dopis byl skutečně odeslán z uvedené domény [23].

Kapitola 3

Honeypoty

Honeypot v doslovném překladu z angličtiny znamená hrnec medu. Honeypot je bezpečností prvek, jehož největší hodnota spočívá v tom, že jej někdo prozkoumá a napadne. Je tedy důležité útočníky nejdříve nalákat na něco, co se podobá skutečnému prvku, jako je např. server nebo router. Pro honeypot je totiž významné, aby byl napadnut, protože jinak nemá žádnou hodnotu. Honeypoty nejsou limitovány v řešení pouze jednoho zadaného problému, ale mohou se přizpůsobovat podle situace, ve které mají být použity. Jedním z mnoha případů použití mohou být http honeypoty sloužící k odhalení útoků na webovou aplikaci. Dále databázové honeypoty, protože databáze bývají často terčem útoků, nebo např. honeypoty zachytávající spam.

Honeypoty mají určité výhody, které ovlivňují jejich hodnotu. Jsou to především:

- Získaná data,
- zdroje,
- jednoduchost.

Jednou z mnoha výhod jsou data získaná pomocí honeypotu. Organizace každý den nasbírá velké množství dat a je těžké z nich získat užitečné hodnoty. Oproti tomu honeypoty sbírají menší množství dat s mnohem vyšší informační hodnotou. Místo toho, aby honeypot sbíral velký objem informací každý den, sbírá pouze užitečná data nasbíraná během útoku na honeypot. Další z mnoha výhod je ušetření zdrojů. Mezi výhody honeypotů lze dále zařadit jejich jednoduchost a relativní finanční návratnost. Není nutné použití složitých programů. Čím jednodušší honeypot je, tím spolehlivěji funguje. Honeypot je snadné nasadit a čekat na útočníky, o kterých je pak snadné zjistit užitečné informace [24].

Nevýhody honeypotů:

- Úzké zaměření,
- odhalitelnost honeypotu,
- risk.

Jednou z největších nevýhod je jejich úzké zaměření. Zachytávají pouza aktivitu, která je mířená proti nim. V případě, kdy se útočník dostane do sítě organizace a napadne různé prvky v síti, honeypot nemá šanci o napadení nic zjistit. Navíc v případě, kdy si je útočník honeypotu vědom, může se mu při napadení vyhnout. Další nevýhodou je tzv. fingerprinting

(z anglického překladu otisk prstu). Jedná se o problém především komerčních honeypotů, kdy pro útočníka je snadné zjistit, že se jedná pouze o honeypot a ne např. o skutečný webový server. Stačí, aby odpověď serveru byla špatně napsaná a je snadné určit, že se jedná pouze o emulaci webového serveru. Třetí významnou nevýhodou honeypotů je risk, který některé z nich představují. V případě honeypotů s mnoha funkcemi se může jednat o riziko toho, že se útočník honeypotu zmocní a využije jeho vlastnosti k dalším útokům [24].

Existují dva základní typy honeypotů podle jejich účelu, a to produkční honeypoty a výzkumné honeypoty.

Produkční honeypoty

Využití produkčních honeypotů je obvyklé ve firmách pro zvýšení ochrany a snížení rizik. Firmy často nasazují honeypoty pro detekci útoků. Tento druh honeypotu je snadnější na nasazení a údržbu než výzkumný honeypot, protože nepotřebují tolik funkcionality. To však znamená, že nám produkční honeypot bude poskytovat méně informací o útocích a útočnících. Je možné zjistit, z jaké operačního systému byl útok proveden, co používají při útocích, ale není možné zjistit, jak komunikují mezi sebou nebo jak probíhá vývoj jejich nástrojů [24].

Výzkumné honeypoty

Oproti tomu výzkumné honeypoty se vyvíjí právě za účelem zjistit o útočnících co nejvíce informací. Jejich hlavním účelem je prouzkoumat rizika, kterým mohou organizace čelit, kdo jsou útočníci, jaké nástroje používají k napadnutí dalších systémů. Informace z tohoto druhu honeypotu nám umožňuje lépe pochopit, co jsou útočníci zač a jaké jsou jejich postupy. Získáním těchto znalostí je pak možné se v budoucnu proti nim lépe bránit. Tyto honeypoty bývají často složitější, protože je potřeba simulovat skutečný operační systém a aplikace, se kterými může útočník interagovat. Jen tak lze zjistit o útočnících podrobnější informace. Množství informací získaných pomocí výzkumného honeypotu je však vykoupeno tím, že jsou složitější, představují potencionálně větší riziko a vyžadují více času na svou správu [24].

Přestože je spíše běžný přístup, kdy honeypot pasivně vyčkává na útočníka, je možné se také setkat s klientskými honeypoty. To znamená, že je dále možné dělení honeypotů na:

- Server honeypoty,
- klientské honeypoty.

Klientské honeypoty procházejí internet a hledají a identifikují servery, které zneužívají zranitelnosti na straně klienta. Tradičně jsou tyto servery identifikovány změnami stavů klientského honeypotu během monitorování, které jsou výsledkem interakce se serverem [19].

Další možná klasifikace záleží na formě honeypotu. Z tohoto úhlu pohledu můžeme honeypoty dělit na:

- Fyzické honeypoty,
- virtuální honeypoty.

Fyzický honeypot znamená, že honeypot běží na fyzickém počítači. Fyzické honeypoty bývají často s vysokou mírou interakce. Instalace a údržba jsou obvykle drahé. Pro velké adresní prostory je nepraktické nebo nemožné nasadit fyzický honeypot pro každou IP adresu. V takovém případě musíme nasadit virtuální honeypoty.

Hlavními důvody použití virtuálních honeypotů je jejich škálovatelnost a snadná údržba. Na jednom stroji můžeme mít tisíce honeypotů. Jsou nenákladné na nasazení a jsou přístupné téměř každému. Ve srovnání s fyzickými honeypoty je tento přístup lehčí. Namísto nasazení fyzického počítačového systému, který funguje jako honeypot, můžeme také nasadit jeden fyzický počítač, který hostí několik virtuálních strojů, které fungují jako honeypoty. To vede k snadnější údržbě a nižším fyzickým požadavkům.

3.1 Honeypoty podle úrovně interakce

Honeypoty je možné dělit podle míry jejich interakce. Čím více toho honeypot nabízí útočníkovi, tím více je možné toho o útočníkovi zjistit. Každý z přístupů při vytváření honeypotu má však své pro a proti, tyto specifické rysy jsou uvedeny v tabulce 3.1 [1].

Tabulka 3.1: Vlastnosti honeypotů podle míry jejich interakce

	nízká míra interakce	vysoká míra interakce
služba	emulovaná	skutečná
nastavení	snadné	složité
údržba	snadná	náročná
rychlost	vysoká	nízká

Honeypoty s nízkou mírou interakce

Honeypoty s nízkou mírou interakce bývá často snadné nasadit a udržovat, protože mají jednoduchou implementaci. Tyto honeypoty často emulují řadu služeb. Příkladem může být Unix server, na kterém běží několik služeb jako FTP nebo Telnet. Nízkointeraktivní honeypoty obvykle simulují i síťové prvky nebo celé počítačové systémy. Z pohledu útočníka se jedná o běžnou službu, která bývá často pouze emulovaná. Útočník se může na server dostat například hádáním přihlašovacích hesel. Honeypot pak může zaznamenávat nepovedené pokusy, ve skutečnosti zde ale není žádný operační systém. Vše, co může útočník zkusit je, přihlásit se [17].

Hlavním cílem těchto honeypotů je detekovat neautorizovaný přístup. Vzhledem k tomu, že jsou takto jednoduché, představují nejmenší míru rizika, že bude honeypot využit útočníkem k další nezákonné činnosti. Je zde také velké riziko toho, že útočník snadno odhalí, že se jedná pouze o honeypot. Zároveň i množství získaných dat je omezené především na čas a datum útoku, zdrojovou IP adresu a port [24].

Honeypoty s vysokou mírou interakce

Narozdíl od předešlého typu honeypotu jsou honeypoty s vysokou mírou interakce mnohem náročnější na implementaci, nasazení a údržbu. Mohou představovat i bezpečnostní riziko pro organizace, které se rozhodnou jej nasadit bez použití dalších bezpečnostních opatření jako např. firewall[1]. Informace získané z tohoto honeypotu jsou pak ovšem mnohonásobně

[1]Firewall je zařízení, které filtruje příchozí i odchozí komunikaci do vnitřní sítě.

důležitější a cennější. Na rozdíl od předcházející kategorie tento typ honeypotů využívá reálné systémy (například kompletní virtuální stroje). I vzhledem k větší náročnosti na čas a zdroje je tento druh honeypotů dražší a je zde zapojeno mnoho dalších technologií. Důležité je také zmínit, že tato kategorie honeypotů nemá žádnou cenu, pokud nejdojde k útoku, protože jen tak je možné objevit nová bezpečnostní rizika a zjistit, jak mezi sebou navzájem útočníci komunikují. Tyto vlastnosti dělají z vysokointerakčních honeypotů silnou zbraň [24] [17].

3.2 Honeytoken

Honeypot nemusí být pouze kus hardwaru nebo program, ale jedná se o zdroj, který má být zneužit. Toto zahrnuje i digitální zdroje jako čísla kreditních karet nebo soubory. Jako honeytoken lze použít i e-mailové adresy. Tyto adresy je možné rozmístit na různá místa na internetu a následně sledovat jejich využití. Stejně jako u honeypotu nesmí útočník vědět, že nepracuje s reálnými daty, ale s honeytokenem. Obdobně pak jakákoliv operace s honeytokenem je podezřelá, protože tato data nemají být běžně používána, ale slouží jako návnada [17].

Klasickým příkladem může být lékařský záznam o neexistujícím pacientovi. Nemocnice jsou nuceny chránit soukromí svých pacientů a pouze zaměstnanci, jako jsou například lékaři a sestry, mají přístup k lékařským záznamům. Do databáze má však přístup tolik lidí, že může být těžké určit, jestli někdo neoprávněně nevyhledával informace o pacientech. Pro tyto účely lze vytvořit falešného pacienta, který není skutečný a nikdo k němu nemá autorizovaný přístup. Pokud tedy nějaký zaměstnanec prohledává informace o pacientech, narazí na tohoto pacienta a pokusí se zobrazit podrobnější data, je jasné, že porušuje soukromí pacienta.

Existuje mnoho způsobů, jak spammeři mohou nasbírat tisíce e-mailových adres na různých místech na internetu. Jedním z dalších možných způsobů jsou špatně nastavené seznamy e-mailů, které tak volně zpřístupňují seznam jejich odběratelů. Dalším ze způsobů jsou programy, které brázdí webové stránky na internetu, a v každé `HTML` stránce, kterou najdou, se snaží získat pole `mailto`, které obsahuje e-mailovou adresu.

3.3 E-mailové honeypoty

E-mailové honeypoty simulují SMTP server, který se stará o posílání e-mailů. Ve většině případů na adrese SMTP serveru poslouchá MTA (Mail Transfer Agent), který se, jak z překladu vyplývá, stará o přenášení e-mailů od odesílatele k příjemci. Případu, kdy MTA bez omezení doručí každý e-mail, se říká open-relay, což v překladu znamená otevřený přenos. V tomto případě pak nastává riziko toho, že honeypot bude uveden na tzv. blacklistu, tj. seznamu, na kterém jsou uvedené servery rozesílající nevyžádanou poštu. Nastane situace, kdy ostatní přestanou od honeypotu e-maily přijímat, a odhalení toho, že se jedná o honeypot, bude pro útočníky potom snadnější. Útočníci se velmi často snaží ujistit, že se nejedná o honeypot tím, že než přes něj začnou odesílat velkou kampaň obsahující stovky e-mailů, zašlou na zkoušku pár e-mailů sami sobě na známé e-mailové adresy na ověření, že server funguje.

Kromě honeypotu se může jednat i o špatně nastavený MTA nějakého dodavatele internetového připojení. MTA tak umožňuje spammerům využívat zdroje a síť nějaké organizace.

Taková situace může naštvat interní uživatele, protože pak nemůžou nadále používat jejich e-mailové účty. Může se tak stát, že organizace přijde o své zákazníky [10].

Honeypot může dále implementovat autorizaci pomocí jména a hesla. Následně tyto přihlašovací údaje mohou být šířeny internetem za účelem nalákat co nejvíce útočníků na náš honeypot. Honeypot řešený v této bakalářské práci může mít více takových uživatelských jmen a hesel.

Podle účelu, kterému e-mailový honeypot slouží, je možné jej rozšířit i o další vlastnosti, např. možnost e-maily uchovávat v databázi a následně z nich systematicky získávat data. Velmi zajímavé mohou být informace o útočnících, jako je jejich IP adresa, jak často odesílají velké kampaně obsahující stovky e-mailů, jaké škodlivé přílohy rozesílají atd.

Přihlašovací jméno pro honeypot

Pokud organizace vlastní nějakou doménu, pak přihlašovací jméno k honeypotu může být cokoliv z této domény. Např. `h10n5@example.com` nebo `david@example.com`. Je však dobré držet se pár doporučení při tvorbě jména:

- Není dobré používat adresu, který bývala kdysi používána.
- Neměla by se používat jména, která odpovídají formátu, který je organizací běžně používán, jako například `jmeno.prijmeni@example.com`.
- Je vhodné použít jméno, které nebylo nikde použito veřejně.
- Běžná jména jako `info@`, `jobs@`, `sales@`, `accounting@` nebo `support@` jsou jednou z nejlepších návnad pro spammery.
- Je příhodné vytvořit více jmen, protože spam je obvykle poslán na několik e-mailových adres.

Existující e-mailové honeypoty

Existujících e-mailových honeypotů je celá řada; zde jsou uvedeny 4 e-mailové honeypoty, jejichž kód je volně dostupný.

- SHIVA[2] - honeypot napsaný v programovacím jazyce `Python`. SHIVA podporuje sběr a analýzu e-mailů. Podrobnější popis tohoto honeypotu je v části 4.
- Honeymail[3] - honeypot napsaný v programovacím jazyce `Go` a je možné jej šířit pod licencí `GPLv3`. Podporuje odpovědi SMTP protokolu, které lze dále konfigurovat. Dále má podporu TLS a ukládá e-maily do databáze `BoltDB` po dnech. Má implementováno API, díky kterému je možné číst e-maily podle jejich specifického identifikačního čísla. API dále umožňuje číst e-maily, které přišly v různé dny. Automaticky zjišťuje informace z e-mailu, například odkazy, zdrojová IP adresa e-mailu, země odkud e-mail přišel, přílohy, text z těla e-mailu. Počítá sha256 hash z částí e-mailu. Honeypot je stále ve vývoji a má mít další vlastnosti, nicméně poslední změna v kódu proběhla už před několika lety.

[2] https://github.com/shiva-spampot/shiva
[3] https://github.com/sec51/honeymail

- Mailoney[4] - honeypot napsaný v programovacím jazyce `Python`. Je možné ho spustit ve třech různých módech. Honeypot může běžet jako tzv. open-relay, kdy se do souboru s logy zaznamená celý text e-mailu, který se spammer pokoušel přes Mailoney přeposlat. Další možností je spuštění s přepínačem `postfix_creds`. V tomto případě modul pouze zapisuje do logu přihlašovací údaje, které spammer použil při pokusu přihlásit se k honeypotu. Poslední možnost je spuštění honeypotu s přepínačem `schizo_open_relay`. V tomto režimu je zaznamenávána veškerá aktivita do logu. Velkou nevýhodou honeypotu je, že vlastně nic nakonec nepřepošle a možnost jeho odhalení je tak celkem vysoká. Stejně jako Honeymail jsou pro tento honeypot naplánovány další vlastnosti, ale poslední změna v kódu proběhla už před pár roky.

- Spamhat[5] - honeypot napsaný v programovacím jazyce `Perl`. Ze zmíněných e-mailů je bez dalšího vývoje nejdéle. Instalace závisí na starých verzích Debian 6 a Debian 7, které byly již nahrazeny novějšími verzemi. Honeypot funguje jako open-relay. Spamhat využívá `MySQL`, kterou je nutné dopředu vytvořit, než se honeypot spustí.

E-mailové adresy

Důležitým krokem pro spammery, a tedy i proto, aby honeypot někdo využíval, je získání e-mailových adres. Technika, kterou využívají spamměři k nalezení existujících e-mailových adres v nějaké doméně, se nazývá directory harvest attack (v překladu útok na adresář). Při pokusu o nalezení těchto adres na poštovním serveru využívá directory harvest attack strategii pokusů a chyb nazvanou útok hrubé síly. Vyzkouší všechny možné alfanumerické kombinace používané pro běžné uživatelské jméno.

Další přístup při directory harvest attack je odesílání e-mailů na různé e-mailové adresy pomocí slovníku běžných křestních jmen a příjmení nebo jejich počátečních písmen. Adresy, na kterých jsou e-maily přijímány, jsou považovány za platné a tyto adresy jsou připojeny na seznam platných e-mailových adres, který spammeři vytváří. Organizace používající emailové adresy se standardizovaným formátem křestního jména a příjmení před doménou `@domena` jsou proto často oběťmi útoků spammerů.

Konkrétní e-mailová zpráva při directory harvest attack často používá krátkou náhodnou frázi, jako například „ahoj“, aby zpráva unikla filtru pro spam. Skutečný obsah určený pro reklamu bude v pozdější kampani odeslán pouze na platné e-mailové adresy, které neodpoví zprávou o selhání. Existují poštovní servery a dodavatelé zabezpečení, kteří nabízejí funkce pro minimalizaci těchto útoků. Tyto poštovní servery obvykle sledují statistiky chybně zadaných e-mailů. Když neplatné e-maily přijaté poštovním serverem překročí určitou prahovou hodnotu, zprávy (popř. odesílatelé) jsou odmítnuty nebo odloženy na určité časové období. Tyto poštovní servery se pokoušejí zajistit, aby legitimní e-maily nebyly označeny jako pokus o directory harvest attack [25].

[4] https://github.com/awhitehatter/mailoney
[5] https://github.com/miguelraulb/spamhat

Kapitola 4

Návrh

Tato kapitola popisuje návrh řešení SMTP honeypotu. V kapitole je rozebráno existující řešení a návrh úprav pro efektivní sběr informací o spammerech.

Předlohou pro náš honeypot je honeypot SHIVA, který je implementován za použití e-mailového serveru Lamson. Lamson vznikl v roce 2009 a je napsaný v programovacím jazyce Python verze 2, jehož podpora skončila na začátku roku 2020[1] a linuxové distribuce se snaží odstranit závislosti na balíčcích napsaných v této verzi Pythonu. Jedná se tedy o jednu z motivací přepsat honeypot do modernější podoby, která bude podporovat Python 3. Kód tohoto honeypotu je volně dostupný pod licencí GPLv3. Jedná se o honeypot s vysokou mírou interakce, jehož cílem je také sběr nevyžádané pošty. Aby bylo pro spammery těžší poznat, že se jedná o honeypot, část e-mailů je příjemcům skutečně doručena. Zároveň má SHIVA další vlastnosti, které pro naši implementaci nejsou důležité jako sdílení analyzovaných dat z obdržených e-mailů pomocí hpfeeds nebo možnost zcela zakázat přeposílání e-mailů.

SHIVA po instalaci vytváří dvě virtuální prostředí. Virtuální prostředí je oddělené prostředí pro Python, kam se dají instalovat jednotlivé knihovny, které jsou potom aktivní jen uvnitř. Použití tohoto přístupu má dvě hlavní výhody:

- Je-li prostředí aktivované, příkaz `python` spustí verzi Pythonu, se kterou bylo prostředí nainstalováno.
- Instalace knihoven nezasahuje do systémového nastavení ani do jiných virtuálních prostředí. Je tak možné oddělit jednotlivé projekty a v každém prostředí můžou být nainstalované jiné verze knihoven. A když se něco pokazí, adresář s virtuálním prostředím je možné prostě smazat a vytvořit znovu.

První virtuální prostředí je určené pro `receiver`, druhé virtuální prostředí tvoří `analyzer`. Obě tato prostředí jsou vlastně Lamson projekt upravený pro potřeby honeypotu. Část `analyzer` využívá moduly, které byly přidány do honeypotu za účelem zjišťování informací o e-mailech, které přes honeypot projdou, a následné uložení těchto informací do databáze `MySQL`. Tato část není pro naše potřeby důležitá, protože samotná analýza e-mailů, které přes honeypot projdou, probíhá na jiném serveru s jinou databází. Proto bude změněna na část `relay`, jelikož její hlavní funkcí je inteligentní přeposílání e-mailů na základě jejich obsahu spíše než jejich analýza a ukládání do databáze, která navíc může zabírat zbytečně moc místa na serveru. Přitom účelem práce je vytvořit nenáročný honeypot, který nebude

[1] https://www.python.org/doc/sunset-python-2/

potřebovat příliš paměti. Nevýhodou honeypotu SHIVA je právě to, že stačí, aby přes něj prošlo pár e-mailových kampaní za týden, což může být tisíce e-mailů, a databáze bude zabírat prakticky veškeré místo na disku. Uživatelé tohoto honeypotu tuto skutečnost často právě řeší tím, že databáze je od honeypotu zcela oddělena nebo jsou některé e-mailové adresy, ze kterých chodí velké kampaně, zakázány. Pak může docházet ke ztrátě informací z e-mailů, kterým je cíleně zabráněno přes honeypot projít. Oproti tomuto existujícímu řešení, které pouze dovoluje zakázat nebo povolit přeposílání e-mailů, bude náš honeypot umožňovat inteligentní přeposílání nevyžádané pošty příjemci tak, aby honeypot působil co nejvěrohodněji pro spammery, kteří jej používají pro rozesílání velkého množství nevyžádané pošty.

4.1 Architektura návrhu

Základem našeho honeypotu je e-mailový server Salmon[2] napsaný v programovacím jazyce Python 3. Tento server je navržený pro vytváření robustních a složitých e-mailových aplikací, umožňuje tak běh aplikace v mnoha kontextech pro zpracování pošty pomocí nejlepší dostupné technologie.

Implementace serveru umožňuje:

- Zpracovat e-maily v libovolném kódování,
- přeposílat nedotčené e-maily příjemcům,
- běh v odděleném virtuálním prostředí a spouštění s právy správce i bez těchto práv,
- flexibilní směrování e-mailů,
- nastavení dalších možných systémů, jako např. databáze,
- použití `Jinja2` nebo `Mako` vzorů pro tvorbu e-mailů,
- snadné vytvoření nového projektu pomocí příkazu `salmon gen`.

Implementace tohoto serveru byla pro naše účely upravena, jelikož server běžně přeposílá všechny e-maily, zároveň je neukládá do žádného adresáře a stejně tak bere v potaz pouze jednoho příjemce e-mailu, i když jich může být víc.

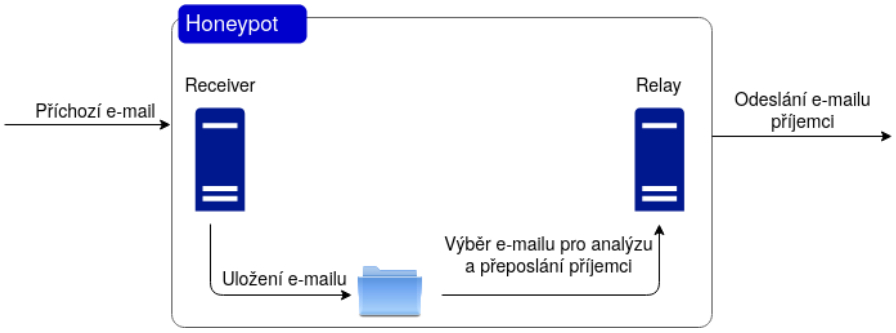
Náš honeypot je rozdělen do dvou částí:

- `receiver`,
- `relay`.

První část (`receiver`) je zodpovědný za příjem e-mailů. Všechny e-maily jsou ukládány do adresáře, odkud je pak následně druhá část honeypotu (`relay`) vyzvedne a postará se o jejich přeposlání příjemci v případě, kdy je přeposílání povoleno, a zpracuje jednotlivé části e-mailu k možné další analýze.

Na obrázku 4.1 je možné vidět, že pro odesílatele nevyžádané pošty je obtížné odhalit, že se nejedná pouze o SMTP server, ale o honeypot, který před odesláním příchozí pošty email zpracuje a zjistí si všechny důležité informace o odesílateli. Cílem implementace našeho honeypotu je toto krytí zajistit co nejefektivnějším způsobem. V následující části návrhu bude popsán `receiver` a `relay` podrobněji.

[2] https://github.com/moggers87/salmon



Obrázek 4.1: Schéma zpracování e-mailu honeypotem

Receiver

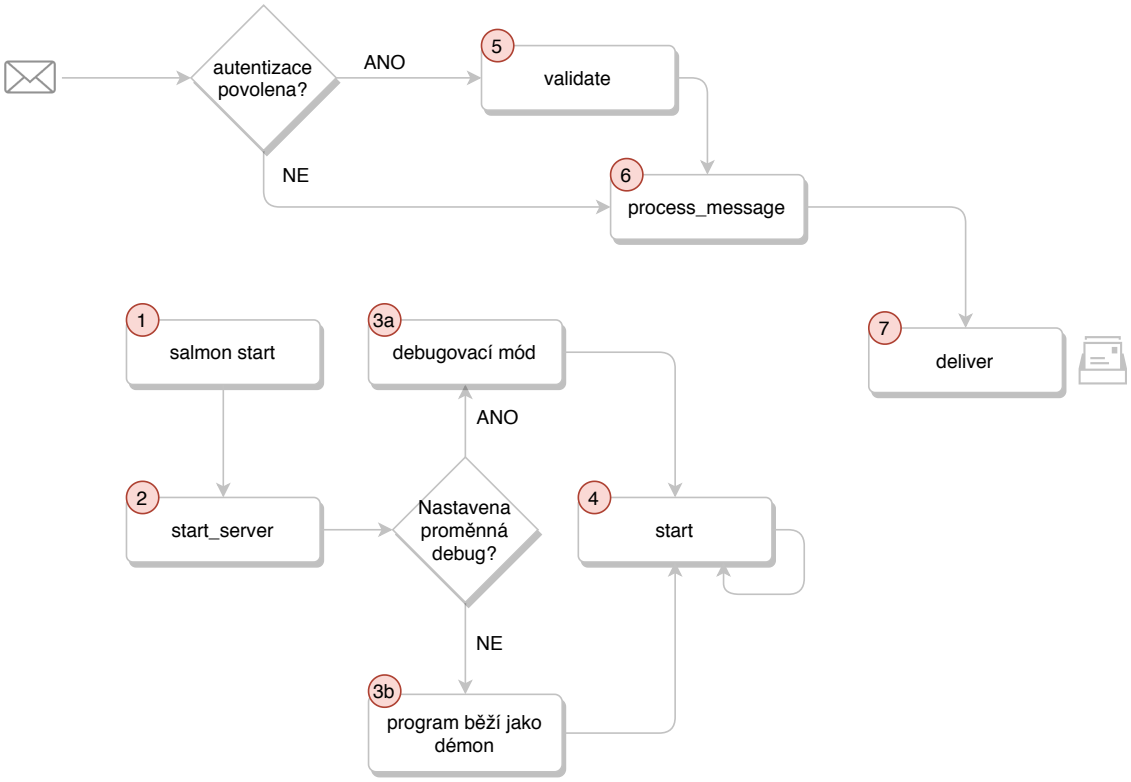
Jak již bylo řečeno, `receiver` je část honeypotu zodpovědná za ukládání e-mailů do adresáře, který je nastavený v souboru `salmon.yaml`. Tato část funguje tak, že `receiver` začne naslouchat na stanovené IP adrese a portu. Jedná se o upravený Salmon projekt sloužící jako SMTP server. V případě, kdy obdrží e-mail, uloží jej do předem nastaveného adresáře. E-maily jsou v nastaveném adresáři ukládány pod jménem, které musí být dodrženo.

```
identifikator_emailu-IP_adresa_odesilatele-salmon-uzivatelske_jmeno
```

Jméno se skládá z identifikátoru e-mailu, IP adresy odesílatele, jména honeypotu, které lze také nastavit v konfiguračním souboru (v naší ukázce `salmon`), a uživatelského jména, které bylo použito při autentizaci společně s heslem k SMTP serveru. Na obrázku 4.2 lze vidět, jak tato část honeypotu funguje.

Při spouštění části `receiver` je nutné mít aktivované virtuální prostředí nazvané `salmon-receiver`, jelikož v něm je nainstalovaný projekt salmon. V adresáři `myproject` se použije příkaz `salmon start`. Samotný server salmon zná více příkazů, pomocí kterých je možné ho ovládat, ale pro potřeby honeypotu je kromě příkazu `salmon start` důležitý už pouze příkaz `salmon stop`, který tuto část honeypotu vypne, a `salmon status`, pomocí kterého je možné zjistit, jestli tato část honeypotu běží.

Následně se zavolá funkce `start_server` v modulu `utils`, která se postará o to, že tato část honeypotu poběží jako démon nebo v debugovacím módu, kdy `receiver` neběží jako démon, ale v nekonečné smyčce a je možné s ním interagovat. Nakonec se zavolá metoda `start` z třídy `SMTPReceiver` a tato část honeypotu začne naslouchat na zadaném rozhraní a portu, které je možné nastavit v konfiguračním souboru `salmon.yaml`. Právě třída `SMTPReceiver` nedědí jako původní `Salmon` projekt z `smtpd` ze standardní knihovny, ale z `smtpd`, které bylo do projektu přidáno a rozšířeno tak, aby byl podporován příkaz `AUTH`, tzn. aby server v případě potřeby podporoval ověření uživatele pomocí jména a hesla. Když přijde nový e-mail, nejprve se ověří, že autentizace je povolena, a zavolá se metoda `validate` třídy `CredentialValidator`, o kterou bylo `smtpd` rozšířeno, a ověří se jméno a heslo. Když je autentizace úspěšná, zavolá se metoda `process_message`. V opačném případě je část, kde se volá `validate`, přeskočena. Metoda `deliver` pak se zprávou projde registrované smyčky a zpráva se vloží do fronty, odkud je potom vyjmuta a uložena do adresáře, ve kterém ji očekává druhá část honeypotu.



Obrázek 4.2: Schéma receiveru

Relay

Tato část honeypotu opět vychází ze Salmon projektu, ale už zde není server, který by naslouchal na IP adrese a portu, ale `relay` čeká na e-maily, které mu `receiver` odloží do předem nastaveného adresáře v souboru `salmon.yaml`. V případě, kdy se v adresáři objeví e-mail, `relay` ho vezme a zpracuje pomocí několika modulů.

Modul `relay` v nekonečném cyklu kontroluje adresář a pokud se v něm objeví nějaké e-maily, tak postupuje následovně:

- Vyzvedne e-maily z adresáře.
- Do souboru obsahující log zapíše informaci o tom, že vyzvedl zprávy s daným identifikátorem.
- Zprávy předá modulu `salmonmailparser`, který se stará o zpracování jednotlivých částí e-mailu.
- Zprávy předá modulu `salmonconclude`, který vypočítá výsledné hodnocení e-mailu.
- Zprávy předá modulu `salmonrelay`, který se podle výsledného hodnocení rozhodne, jestli bude e-mail přeposlán.
- Postupně odstraňuje zprávy z adresáře.
- Uspí se na pár sekund.

Jednotlivé kroky se neustále opakují.

Honeypot dále používá modul `salmonscheduler`, který se stará o kontrolu časového intervalu mezi znovu vynulováním proměnné, která se stará o počítání množství přeposlaných e-mailů. Délku intervalu i počet e-mailů je možné nastavit v konfiguračním souboru. Toto nastavení má velký vliv především v případech, kdy je přes honeypot zasíláno velké množství e-mailů v krátkém časovém intervalu.

To, co dělá tuto část honeypotu zajímavou, jsou právě jednotlivé moduly, o které byl původní SMTP server rozšířen a které anaylizují e-mail před tím, než je přeposlán příjemci v případě, že je vyhodnocen jako testovací e-mail. V opačném případě je pouze uložen do adresáře, který je nastaven v `salmon.yaml`. Moduly budou podrobněji popsány v následující kapitole.

MQTT

V modulu `salmonrelay` se využívá protokolu `MQTT`. `MQTT` je jednoduchý a nenáročný protokol pro předávání zpráv mezi klienty prostřednictvím centrálního bodu – brokeru. U protokolu `MQTT` probíhá přenos pomocí TCP a používá návrhový vzor publisher – subscriber. Tedy existuje jeden centrální bod (`MQTT broker`), který se stará o výměnu zpráv. Zprávy jsou tříděny do tzv. témat (topic). Náš honeypot publikuje v daném tématu (publish), to znamená, že posílá data brokeru, který je ukládá. V konfiguračním souboru lze nastavit, jestli má být `MQTT` využíváno, dále URL adresu `MQTT` serveru a cílovou IP adresu `MQTT`. Tento modul pak předává zprávu dále k přeposlání příjemci.

Cíl návrhu

Cílem je navrhnout SMTP honeypot, který bude možné rychlé nasadit a zároveň bude podporovat následující rozšíření:

- Přijímání e-mailů na portu 25 a ukládání e-mailů do adresáře nebo publikování na `MQTT`,
- podporova autentizace,
- jednoduché nasazení jedním kliknutím,
- inteligentní přeposílání e-mailů,
- nemožné odhalit, že se jedná o honeypot,
- možné poškození škodlivého obsahu e-mailu za účelem ochrany uživatelů.

Tato bakalářská práce si klade za cíl vytvořit honeypot, který bude nevyžádanou poštu sbírat co nejefektivněji. Zároveň nesmí být možné odhalit, že se jedná o honeypot. Honeypot bude během určitého časového intervalu sbírat informace o e-mailech a ukládat je do databáze, odkud budou informace mazány po určitém časovém úseku. Jeho účelem není sběr všech informací o spammerech za nějaký dlouhý časový úsek, jako je například rok, ale v řádu hodin nasbírat informace, které se opakují v e-mailech, a na jejich základě rozhodnout, které e-maily je vhodné dále přeposlat.

Implementace honeypotu bude tedy rozšířena o databázi `SQLite`. Jedná se o databázi vhodnou pro malé aplikace, které nevyžadují náročnou správu více uživateli, a umožňuje tak ukládat informace v jednom souboru, který se v případě potřeby snadno kopíruje, stejně tak

velikost je pouze okolo stovek KB. Tato databáze poslouží pro ukládání základních informací o e-mailech přeposlaných přes honeypot. Je totiž neefektivní a náchylené k odhalení, když honeypot ve stanovém intervalu odešle pouze určité množství e-mailů příjemci. Spammeři totiž obvykle před tím, než odešlou stovky e-mailů v rámci nějaké reklamní kampaně, otestují SMTP server, jestli skutečně funguje například tak, že si pošlou pár zkušebních e-mailů na svoje vlastní známé e-mailové adresy.

Zároveň pokud má honeypot povolenou autentizaci (v opačném případě se jedná o open-relay), tak zná několik uživatelských jmen a hesel. Může pak nastat scénář, kdy jeden spammer odešle např. 3 testovací e-maily, jestli SMTP server funguje, a pak třeba tisíc e-mailů. Takto se vyplýtvá množství e-mailů, které bylo možné odeslat za nějaký časový okamžik, a úplně jiný spammer, který může použít i jiné uživatelské jméno a heslo, nebude mít možnost odeslat ani testovací e-mail a server se tak bude jevit nefunkční. Z nasbíraných informací z obdržených e-mailů bude snažší se rozhodnout, který e-mail má smysl přeposlat a který ne, protože se zrovna jedná pouze o součást reklamní kampaně.

Fuzzy hashing

Při rozhodování, které e-maily je třeba uložit do databáze, se využívá fuzzy hash (v překladu do češtině haš a jedná se o výstup hašovací funkce) vypočítaný ze zprávy. Hašovací funkce je matematická funkce pro převod vstupních dat do relativně malého čísla. Fuzzy hashing využívá rolling hash a piecewise hash. Rolling hash je hašovací funkce, která generuje části haš řetězce vytvářením pseudo-náhodné hodnoty založené na kontextu vstupu. Piecewise hash generuje finální součet pro celý dokument, nejdříve ale soubor rozdělí na několik částí a až potom generuje haš pro každou z těchto částí. Jednotlivé hodnoty získané pro části dokumentu pak tvoří konečný haš. Fuzzy hash dokáže porovnat dva soubory, které mohou být téměř identické. Jejich podobnost je vyhodnocena na základě procentuální shody, která se pohybuje od 0 do 100%, kde 0% znamená malou podobnost a 100% velkou podobnost. Pro porovnávání souborů pomocí fuzzy hashingu bude použit nástroj ssdeep[3] [22].

Výstup hašovací funkce se označuje jako otisk či hash. Pomocí otisku budeme schopni snadno porovnat dva e-maily. Je-li přes honeypot přeposlán e-mail, který je označen jako testovací, ale podobný se už v databázi nachází, není potřeba ho znovu ukládat.

4.2 Konfigurace

Součástí honeypotu je konfigurační soubor `salmon.yaml`, pomocí kterého lze snadno nastavit na jednom místě důležité parametry pro obě části honeypotu. Po instalaci obsahuje honeypot kromě `receiver` a `relay` i část `configuration`. V části `configuration` lze nastavit, na které IP adrese a portu bude server naslouchat, jestli má server požadovat po uživatelích uživatelské jméno a heslo a zároveň je zde možné uvést tyto autentizační údaje. Pro server je možné nastavit více uživatelů a jejich jména. V části `relay` je pak možné nastavit kolik e-mailů a za kolik minut se má přeposlat příjemcům, na jaké IP adrese a portu naslouchá MTA (mail transfer agent) a jestli se mají e-maily vůbec dále přeposílat. Dále je zde možné povolit ukládání e-mailů ve formátu EML[4] a ukládání příloh e-mailů. V konfiguračním souboru je také možné nastavit, jestli část `relay` má ničit některé části e-mailu. Na výběr je ničení linků obsažených v e-mailu, ničení příloh e-mailu a ničení pole reply-to, tj. e-mailové adresy, na kterou může příjemce e-mailu odeslat svoji odpověď. Vzhledem

[3] https://ssdeep-project.github.io/ssdeep/

[4] Specifikace formátu souboru EML jsou k dispozici podle standardního formátu RFC 822 [5].

k tomu, že e-maily, které prošly přes honeypot, jsou nevyžádané e-maily a mohou tedy být i nebezpečné, není žádoucí, aby tato e-mailová adresa byla validní. Stejně tak linky vedoucí na různé internetové stránky je dobré zneškodnit, aby příjemce neměl šanci se dostat na podezřelé webové stránky. Přílohy e-mailu jsou nenávratně ničeny tak, aby po otevření např. PDF souboru bylo uživateli sděleno, že soubor je poškozen. Konfigurační soubor dále obsahuje možnost povolit použití souboru `rules.json`, v němž je možné specifikovat, co má obsahovat e-mail, který má být přeposlán příjemci. Kromě toho je zde možnost zapnutí analýzy textu a použití MQTT. Proto, aby vše fungovalo dohromady tak, jak je naznačeno na obrázku 4.1, je zde uveden adresář, kam se ukládají e-maily, které byly přes honeypot poslány. Dalším důležitým adresářem, který je zde nastaven, je adresář, kam se ukládají e-maily, které nebylo možné doručit, a také adresář, kam jsou ukládány přílohy e-mailů před tím, než jsou zničeny v případě, kdy je tato možnost v konfiguračním souboru povolena. Obsah konfiguračního souboru je při zapnutí honeypotu načten v souboru `settings.py` do proměnné data a pomocí modulu `utils`, který se v Salmonu již nachází, je k němu možné přistoupit v jakékoliv části honeypotu.

4.3 Instalace

Honeypot je možné nainstalovat dvěma způsoby. První ze způsobů je použití skriptu napsaného v shellu, který je možné využít v případě, kdy uživatel má složku s kódem honeypotu na počítači, kde chce honeypot nainstalovat. Druhý ze způsobů je použití `playbook` napsaného v Ansible. V tomto případě stačí mít pouze složku s playbookem a konfiguračními soubory Ansiblu.

Shell

Náš honeypot je možné nasadit pomocí shellového skriptu. Skript si nejdříve na systému, kde má honeypot běžet, zjistí, které balíčky chybí, a doinstaluje je. Dále se nastaví MTA tak, aby naslouchal na určeném portu. Nastavují se také adresáře, kam `receiver` ukládá e-maily. Nainstalují se dvě virtuální prostředí pro `receiver` a `relay`, kam se nainstalují potřebné knihovny. Skript také ukládá do adresáře `bin` kód, který spouští a vypíná honeypot. Na tento kód je vytvořen symbolický odkaz v `/usr/bin` tak, aby se dal spustit odkudkoliv.

Ansible

Honeypot je také možné nainstalovat pomocí Ansible[5]. Ansible je jednoduchý nástroj určený k automatizaci. Dokáže konfigurovat různé systémy, nasadit software a řídit složitější IT úlohy, jako jsou složitá nasazení nových systémů nebo aktualizace. Důležité je snadné použití a bezpečnost za použití technologie `OpenSSH`[6] k přenosu souborů a vzdálená připojení na server. Jedná se tedy o jednoduchý automatizační jazyk, který dokáže popsat infrastrukturu aplikace pomocí kusu kódu uloženého v souboru, kterému se říká `playbook`. Tento kus kódu pak spouští jednotlivé úlohy popořadě a v angličtině je jeho název `play` (v překladu hra). Kus kódu v `playbook` je napsaný ve formátu `YAML`[7].

[5] https://www.ansible.com/

[6] Soubor počítačových programů poskytujících zašifrovaná komunikační sezení, která se používají pomocí ssh protokolu přes počítačovou síť.

[7] Formát pro serializaci strukturovaných dat.

Typický postup:

1. Ansible se připojí k určenému zařízení pomocí `OpenSSH`.
2. Spustí se jednotlivé úkoly pomocí tzv. `modulů`, což jsou malé kousky programů, které se posílají na určené zařízení.
3. `Moduly` uvedou zařízení do požadovaného stavu.
4. `Moduly` jsou ze zařízení odstraněny, jakmile Ansible skončí se svými úkoly.

Když se v nějaké úloze spustí `modul` (obvykle napsaný v Pythonu, PowerShellu nebo nějakém jiném jazyce), tak se tento `modul` ujistí, že se nějaká specifická část zařízení nachází v požadovaném stavu. Pokud se zařízení nenachází v požadovaném stavu, `modul` se postará o to, aby se v něm nacházel. Pokud se v něm ovšem už nachází, nic se nezmění. V případě, kdy má například `modul` vytvořit soubor, ale ten již existuje, nebude vytvářet další. Toto chování se označuje jako idempotentní.

V architektuře Ansible existují dva typy zařízení:

- Kontrolní uzly,
- spravovaná zařízení.

Náš `playbook` je spouštěn z kontrolního uzlu a tento uzel má kopie Ansible souborů. Běžně se tedy jedná o počítač správce systému nebo nějaký server. Toto zařízení může využívat jakýkoliv UNIX systém nebo Linux. Jediným požadavkem je nainstalovaný programovací jazyk Python a pak přímo Ansible.

Spravovaná zařízení jsou uvedena v tzv. inventáři, který organizuje tato zařízení do skupin pro snažší správu více počítačů. Inventář může být definován dynamicky pomocí informací z externích zdrojů, nicméně v našem případě postačí druhá možnost, tj. statický soubor, ve kterém jsou uvedeny IP adresy nebo názvy zařízení, kde chceme, aby byl honeypot nainstalován. Je tak tedy možné během jednoho spuštění nasadit více honeypotů.

Obsah inventáře pak tedy může vypadat následovně v případě, kdy chceme nainstalovat například 2 honeypoty, jejichž IP adresy jsou uvedeny v `/etc/hosts`:

```
1 [honeypots]
2 salmon-honeypot-1
3 salmon-honeypot-2
```

Kapitola 5

Implementace

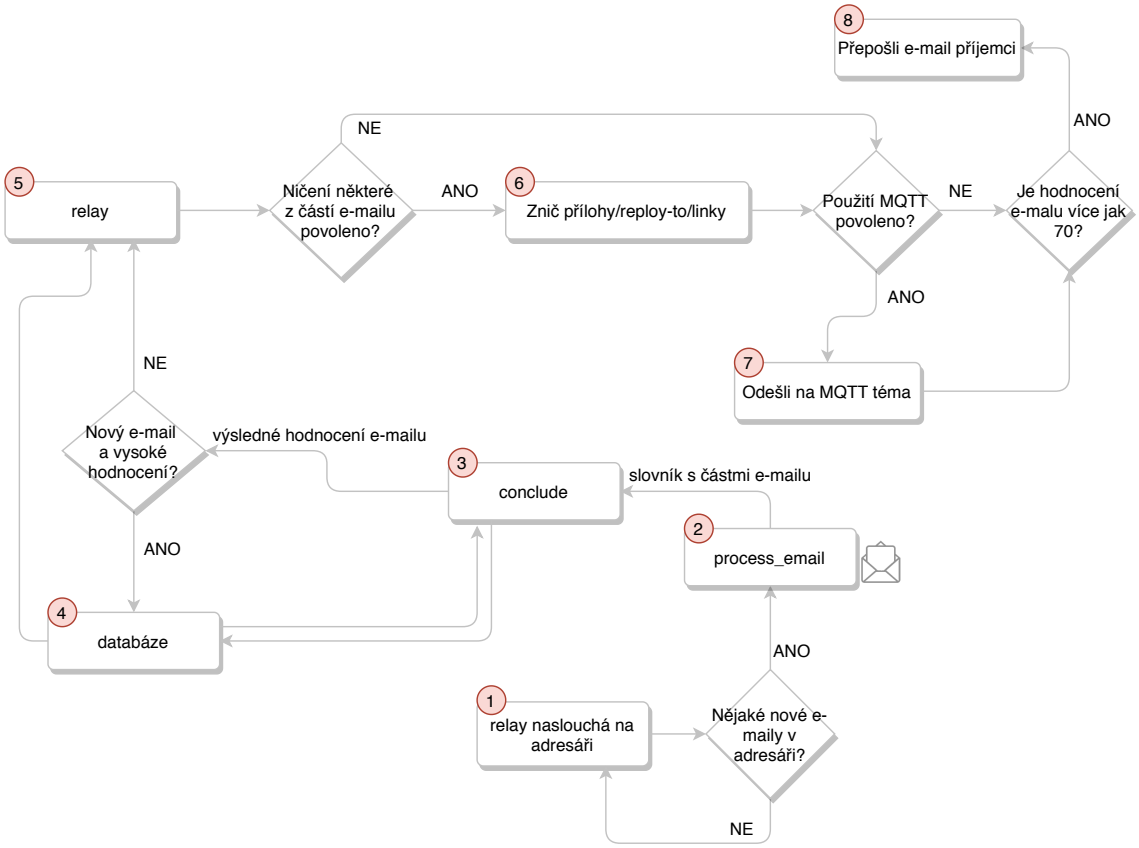
V této kapitole bude popsána implementace modulů, které byly do SMTP serveru přidány tak, aby mohl sloužit jako honeypot a přitom se navenek stále jevil jako SMTP server. Původně byl Salmon určený k tomu, aby byl využit s tradičním e-mailovým serverem. Lze jej tedy použít například jako MDA (zprávy doručovací agent). Má mnoho vlastností stejně jako webové aplikace a dokáže tak spolupracovat s dalšími knihovnami. Tyto vlastnosti jsou našemu serveru ponechány, jako například použití směrování, které by našlo spíše své využití při použití s knihovnou na vývoj webové aplikace. Server však bylo nutné rozšířit o:

- Modul, který bude zjišťovat informace uložené v e-mailech.
- Databázi, která ty nejdůležitější informace z e-mailů bude uchovávat.
- Modul, který umí ničit některé části e-mailů a který bude rozhodovat, které e-maily se mají dále přeposlat.
- Modul, který obsahuje třídu, která podle obsahu e-mailu dokáže určit hodnocení emailu.

V původní verzi přeposílá Salmon vše, co přes něj projde tak, jak by se od SMTP serveru očekávalo.

5.1 Úprava existujících modulů

Na začátku bylo nutné některé moduly Salmonu upravit, aby část `relay` fungovala podle požadavků na tuto část honeypotu. Modul `boot`, který se využívá při startu serveru, obsahuje instanciaci třídy `QueueReceiver` namísto původní třídy `LMTPReceiver`, která by se použila v případě, kdyby Salmon projekt sloužil jinak než SMTP server. LMTP je protokol, který vychází z rozšířeného SMTP protokolu. Za určitých okolností, mimo výměny e-mailové pošty mezi dvěma hosty, je žádoucí implementace systému, kde příjemce pošty neimplementuje frontu, což je například MDA. Třída `QueueReceiver` dostane do svého konstruktoru cestu k adresáři, na kterém má poslouchat a kde se bude tvořit fronta nově příchozích e-mailů. Dále bylo nutné upravit třídu `RoutingBase` v modulu `routing`, protože generátor, který prochází všechny registrované směrovací smyčky, očekává e-mailovou adresu ve formátu `jmeno@ukazka.cz`, ale e-maily odesílané přes náš honeypot můžou mít adresy příjemců ve formátu `"Jmeno Prijmeni"<jmeno@ukazka.cz>`. Dále bylo nutné upravit modul `sample` tak, aby obsahoval funkci `FORWARD`, která se stará o doručení e-mailu ke



Obrázek 5.1: Schéma relay

třídě `Relay`, která stojí na úplném konci zpracování přijatého e-mailu a stará se o doručení příjemci. Modul `server` má pak upravenou třídu `QueueReceiver` tak, že v cyklu kontroluje, jestli se v předem nastaveném adresáři objevily nové e-maily. Pokud se objeví nové e-maily, vyzvedne je z adresáře a předá je modulu `salmonmailparser`, konkrétně funkci `process_email`. Na obrázku 5.1 se jedná o body 1 a 2.

5.2 Modul salmonmailparser

Tento modul byl do honeypotu přidám jako první krok na cestě e-mailu z adresáře, kam byl částí `receiver` odložen, až ke třídě `Relay`. Jako první funkce je v tomto modulu zavolána funkce `process_email`, která očekává dva argumenty při zavolání. Jako první argument přijímá název souboru, který byl vytvořen částí `receiver` při ukládání nového e-mailu do souboru. Jako druhý argument potom přijímá proměnnou `mail_request`, což je instance třídy `MailRequest` z modulu `mail`, který se v Salmonu již nachází. Právě instance této třídy nese proměnnou `Data`, kde je uložený e-mail, který byl přes honeypot poslán. Pomocí knihovny `email` jsou tato data zpracována do proměnné `msg`, která je pak postupně předávána společně s datovou strukturou `mail_fields` všem pomocným funkcím v tomto modulu. Pomocné funkce postupně naplní `mail_fields` částmi e-mailu, jako jsou například e-mailová adresa odesílatele, e-mailová adresa příjemce, předmět e-mailu, text z těla e-mailu, přílohy e-mailu, různé odkazy, které se v e-mailu vyskytly, hash e-mailu. Jednou takovou pomocnou funkcí v tomto modulu je `get_fuzzy_hash`, která počítá hash

z těla e-mailu nebo předmětu. V případě, kdy je tělo e-mailu příliš krátké (kratší než 120 znaků), vypočítá se hash z těla e-mailu, předmětu a textu, který se k těmto částem emailu přidá, aby se hash nepočítal z malého počtu znaků. Další zajímavou funkcí v tomto modulu je `process_email_parts_recursively`, která se stará o zpracování jednotlivých částí e-mailu v případě, kdy je obsah e-mailu ve standardu `MIME`. Pokud je zpráva tvořena více typy obsahu, použije se speciální typ `Multipart`, který rozdělí zprávu na samostatně kódované části. Z těchto částí pak funkce `process_email_parts_recursively` zjišťuje například přílohy e-mailu. Pokud nějakou důležitou část e-mailu není možné zpracovat, funkce `move_to_undeliverable` se postará o její přesunutí do adresáře pro nedoručitelné e-maily. Na konci se slovník `mail_fields` pošle do dalšího modulu `salmonconclude`.

5.3 Modul salmonconclude

Na obrázku 5.1 se jedná o bod 3. Modul počítá konečné hodnocení e-mailu pomocí dalších funkcí, které se nachází v tomto modulu, a dále pomocí metod, které obsahuje třída `Spam`, která se nachází v modulu `salmonspam` a která je popsána v odstavci 5.4. Tento modul obsahuje funkci `conclude`, kde se nejdříve vytvoří instance třídy `Spam` z datové struktury `mail_fields`, kterou tento modul získal od modulu `salmonmailparser`. Dále se vytvoří instance tříd `MailFields`, `Sender` a seznam instancí tříd `Recipient`. Tyto třídy reprezentují tabulky v databázi a jsou lépe popsány v části 5.5. Funkce postupně prochází jednotlivé části e-mailu a snaží se vypočítat konečně hodnocení. To, jakým způsobem jsou jednotlivé kontrolní body ohodnoceny, vychází z dat získaných na jiném SMTP honeypotu.

Jednotlivé kontrolní kroky ověřují tyto body v e-mailu:

1. Heslo k honeypotu,
2. přihlašovací jméno k honeypotu,
3. odkazy,
4. přílohy,
5. slovo test nebo testing,
6. příjemce už byl jednou použit v testovacím e-mailu,
7. čas, kdy byl e-mail obdržen,
8. IP adresa honeypotu,
9. podobný e-mail už se v databázi nachází,
10. nebyl překročen nastavený limit e-mailů, které se mohou přeposlat za určitý čas.

Jako první bod se ověřuje, jestli se heslo k honeypotu, které se používá při autentizaci, neobjevilo v e-mailu. V tomto případě se jedná určitě o testovací e-mail, jelikož heslo je důvěrné a nemělo by se takto v e-mailech šířit. U přihlašovacího jména k honeypotu je o trochu těžší určit, jestli se jedná skutečně o testovací e-mail, jelikož e-maily, které přihlašovací jméno obsahují, jsou často netestovací. Také různé odkazy a přílohy testovací e-mail obvykle neobsahuje. Naopak malé procento testovacích e-mailů obsahuje slovo test nebo testing. Často se stává, že spammeři využívají k zasílání testovacích e-mailů jistou e-mailovou

adresu. Zjistí-li se tedy, že na danou e-mailovou adresu už jednou testovací e-mail byl zaslán, všechny e-maily zaslané na tu samou adresu je možné prohlásit za testovací. Nejčastěji jsou testovací e-maily zasílány mezi 12:00 a 18:00, jedná se tedy o bod, který konečné hodnocení e-mailu také může ovlivnit. IP adresa honeypotu by stejně jako heslo neměla být veřejně zasílána, e-maily obsahující tuto IP adresu jsou tedy taktéž testovací. Pokud se stane, že nějakou náhodou spammer zašle během nastavené doby více než povolený počet `n` e-mailů, které budou vyhodnoceny jako testovací, přepošle se jich jen `n`.

Kromě výše zmíněných bodů je ještě možné v konfiguračním souboru `salmon.yaml` zapnout i analýzu textu, který se nachází v těle e-mailu. Tento kontrolní bod je založen na předpokladu, že testovací e-maily bývají krátké a obvykle neobsahují souvislý text. Například e-maily, které lze ve většině případů prohlásit jako netestovací, obsahují nějakou marketingovou nabídku nebo se snaží příjemce e-mailu na něco nalákat. Pro tyto účely je nutné využívat slova běžné řeči, která zahrnují podstatná jména a slovesa. Tyto e-maily i často přímo o nějakém tématu hovoří, což je také poměrně neobvyklé pro testovací e-maily.

Jako další volitelný kontrolní bod lze nastavit určité pravidlo v souboru `rules.json`, který se nachází v části `configuration`. Pravidlo může vypadat následovně:

```
{
   "name": "priklad",
   "AND": [
       {
           "field": "subject",
           "pattern": "ukazkovy predmet"
       },
       {
           "field": "attachment",
           "pattern": false
       }
   ]
 }
```

Pravidlo má vždy svůj název (`name`) a dále očekávanou strukturu e-mailu. V naší ukázce se tedy očekává, že e-mail, který bude danému pravidlu odpovídat, bude obsahovat v předmětu (`subject`) větu `ukazkovy predmet` a zároveň nebude tento e-mail mít přílohu, jelikož `attachment` je nastavený na false.

Konečné hodnocení e-mailu

Tabulka 5.1: Vyhodnocovací tabulka

hodnocení	závěr	databázová tabulka
0 - 49	není testovací	-
50 - 69	možná testovací	`maybe_test_emails`
70 - 100	určitě testovací	`test_emails`

Na základě konečného hodnocení je pak vyvozen závěr, jestli bude e-mail přeposlán příjemci a uložen do databáze. Rozdělení hodnocení je možné vidět v tabulce 5.1. V případě, kdy e-mail získal jen velmi malé hodnocení, můžeme jej označit za netestovací. Jedná se obvykle o e-maily, které něco nabízejí nebo o něčem mluví, dále mají velmi často nějakou přílohu nebo odkaz na nebezpečnou webovou stránku, na kterou příjemce lákají v e-mailu.

Možná testovací e-maily jsou takové, že nespadají svým hodnocením do kategorie netestovací, ale ještě není možné úplně prohlásit, že jsou určitě testovací. Jsou to e-maily, které získaly hodnocení alespoň 50, což jsou velmi často e-maily, co obsahují například přihlašovací jméno k honeypotu, byly obdrženy v čas obvyklý pro testovací e-maily a neobsahují žádné přílohy nebo odkazy. V případě, kdy je konečné hodnocení e-mailu 70 a více, tak je prohlášen za testovací a bude v dalších krocích postupně dopraven k odeslání. Tyto testovací e-maily velmi často obsahují nějakou informaci o honeypotu. Jedním z příkladů může být tento řádek.

```
127.0.0.1:25 | no auth | SSL: False | Hostname:
```

Informace pravděpodobně vypovídá o honeypotu (tj. SMTP serveru), že běží na IP adrese 127.0.0.1 (adresa byla pro účely ukázky změněna) a naslouchá na portu 25. Nepodporuje autentizaci (v případě, že je tedy na honeypotu zakázána) a nepodporuje SSL protokoly.

Modul `salmonconclude` dále obsahuje třídu `DBUpdater`, která rozhoduje, jestli právě zpracovávaný e-mail nevypadá více jako testovací e-mail než nějaký podobný e-mail, který se již v databázi nachází. Tato třída pracuje pouze s e-maily, které byly označeny jako možná testovací, tj. ten druh e-mailů, které sice jako testovací vypadají, ale není to možné určit s jistotou. Třída má své vlastní interní hodnocení, které když přesáhne určitý práh, tak je aktuální e-mail uložen do databáze a podobný e-mail, který se v databázi již nachází, je z ní odstraněn. Toto se může stát, když například aktuální e-mail nemá žádné přílohy nebo odkazy oproti e-mailu, který se v databázi nachází, popř. obsahuje v některé své části přihlašovací jméno k honeypotu.

E-maily se ze stavu, kdy byly označeny jako možná testovací, mohou dostat tak, že přes honeypot je zaslán podobný e-mail, ale hodnotící systém mu z nějakého důvodu přiřadí vyšší hodnocení. Nižší hodnocení honeypot neřeší. Pokud byl e-mail jednou označen jako možná testovací, tak je v této kategorii i ponechán. Dále se pak může jednat o situaci, kdy přes honeypot projde podobný e-mail, ale adresa příjemce byla již jednou viděna v e-mailu, který byl označen jako skutečně testovací. E-mail je pak přesunut do tabulky, která drží informaci o známých testovacích e-mailech.

5.4 Modul salmonspam

Tento modul obsahuje třídu `Spam`, která po celou dobu, kdy modul `salmonconclude` vyhodnocuje e-mail, udržuje aktuální hodnocení a stejně tak ho může měnit. Jednotlivé atributy této třídy odpovídají částem e-mailu, které byly získány pomocí modulu `salmonmailparser`. Jednotlivé metody této třídy slouží především k prozkoumání e-mailu podle jednotlivých kontrolních bodů, které `salmonconclude` zná. Potom, co je vypočteno konečné hodnocení e-mailu, je zavolán destruktor této třídy, který konečné hodnocení e-mailu zapíše do souboru `salmon.log`.

Kromě této třídy pak modul ještě obsahuje funkci `update_statistics`, která mění statistiky sbírané honeypotem během vyhodnocování e-mailu a ukládá je do databáze popsané v části 5.5.

5.5 Salmon databáze

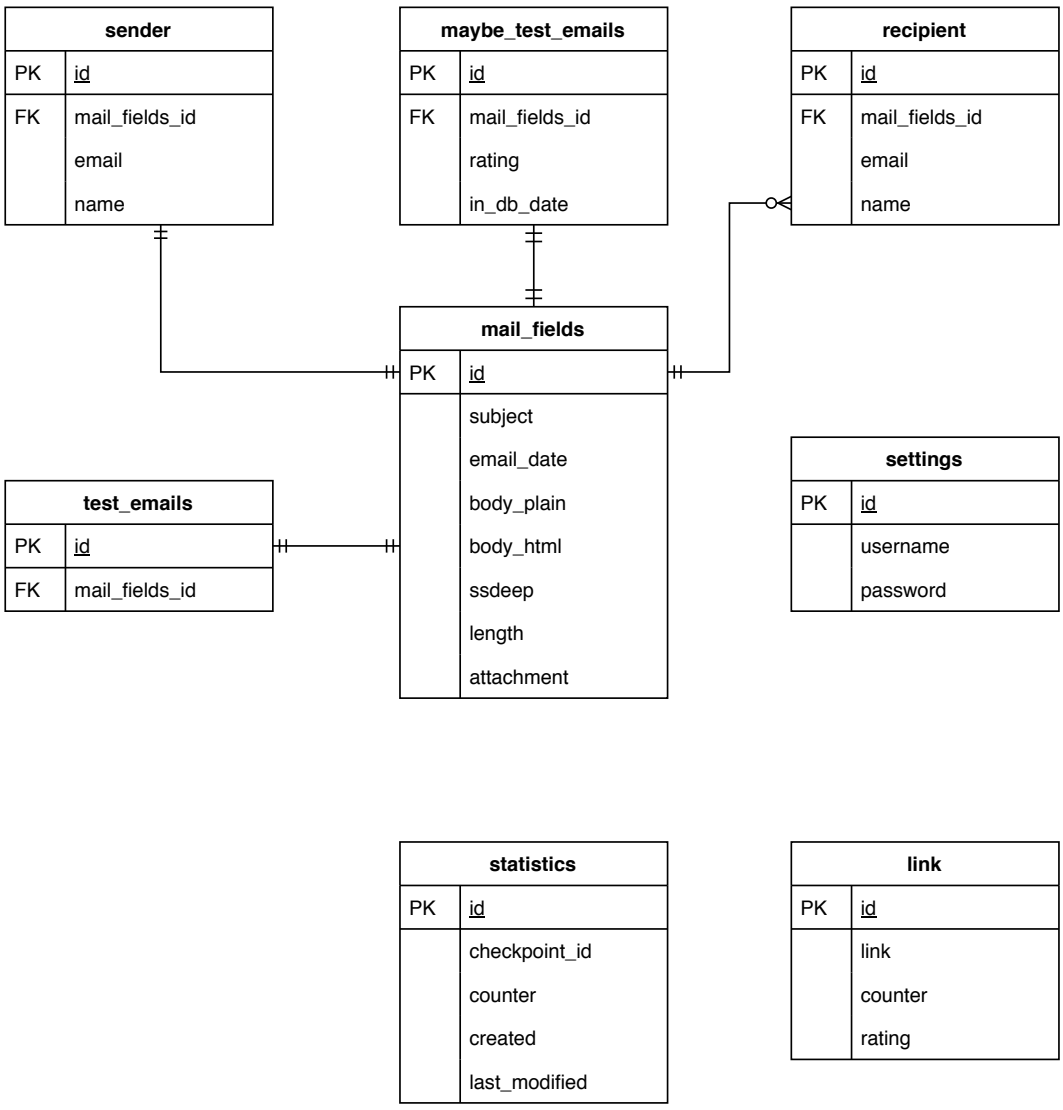
Na obrázku 5.1 se jedná o bod 4. Honeypot využívá `SQLite` databázi společně s `SQLAlchemy`. `SQLAlchemy` je sada nástrojů `Python SQL` a Object Relational Mapper[1] (v překladu mapovač objektových a relačních dat), která umožňuje použití `SQL`. Díky tomuto mapovači nemusíme používat přímo příkazy jazyka `SQL`, ale syntaxi jazyka `Python`. Je tedy nutné vytvořit pouze třídy, které odpovídají tabulkám v databázi. S tabulkami je pak možné pracovat pomocí běžných metod a funkcí napsaných v jazyce `Python`. Databáze `SQLite` byla vybrána pro svoji jednoduchost. Má velmi málo závilostí na dalších knihovnách. Na rozdíl od ostatních databází neběží jako separátní server proces. Dalším výhodou je to, že databázi není potřeba před spuštěním honeypotu nějak nastavovat, není potřeba předem vytvořit databázi společně s uživateli, kteří by museli mít nějaká práva. `SQLite` čte a zapisuje do souboru, který je možné kopírovat mezi systémy a různými architekturami. Informace o spammerech je tak možné přesunout, například k nově nainstalovanému honeypotu, pouze kopírováním souboru `salmon.db`, kde je uložena databáze našeho honeypotu. Když se snažíme přistoupit k `SQLite` databázi na místě, kde databáze ještě nebyla vytvořena, tak se vytvoří nový soubor s databází. Kvůli tomuto je možné spouštět honeypot pouze z jednoho místa. Tento problém ale řeší zmíněné symbolické odkazy na skripty `salmon-receiver.sh` a `salmon-relay.sh`, které se do požadovaného místa dostanou a spustí požadovanou část honeypotu.

Na obrázku 5.2 je možné vidět schéma databáze, se kterou komunikuje honeypot. Jednotlivé tabulky uchovávají následující informace:

- Tabulka `mail_fields` uchovává informace o e-mailu, které honeypot získá z modulu `salmonmailparser`.
- Tabulka `recipient` uchovává e-mailové adresy a jména příjemců e-mailu, e-mail může mít více příjemců.
- Tabulka `maybe_test_emails` nese informaci o tom, který e-mail byl označen jako možná testovací. Jako cizí klíč zde figuruje primární klíč z tabulky `mail_fields`, přes kterou je možné pak tento e-mail spojit například s příjemci e-mailu. Dále je zde hodnocení e-mailu a to pro případ, kdy přes honeypot bude poslán e-mail, jehož konečné hodnocení bude větší než hodnocení e-mailu, který se v databázi už nachází a zároveň bude stále spadat do kategorie možná testovacích e-mailů podle tabulky 5.1. V takovém případě se totiž nový e-mail bude jevit více jako testovací a ten, který se v databázi už nachází, bude odstraněn. Další sloupec v této tabulce je `in_db_date`, který reprezentuje čas, kdy se daný e-mail označil jako možná testovací a byl uložen do databáze. Tento čas je důležitý, protože po uplynutí jednoho měsíce bude záznam z databáze smazán. Stane se tak, protože během této doby nebyl přes honeypot odeslán žádný e-mail, který by se jevil více jako testovací, a tak by byl aktuální záznam přepsán nebo přesunut do tabulky `test_emails`.
- Tabulka `sender` drží informaci o odesílatelích e-mailů. Konkrétně e-mailové adresy a jména.
- Tabulka `test_emails` uchovává informaci o skutečně testovacích e-mailech, které byly poslány přes honeypot.

[1] programovací technika pro převod dat mezi různými systémy pomocí objektově orientovaných programovacích jazyků.

- Tabulka `settings` je zde pro uložení informace o heslech a přihlašovacích jménech, které kdy byly pro honeypot nastaveny v konfiguračním souboru `salmon.yaml`.
- Tabulka `link` ukládá informaci o odkazech, které se v e-mailech vyskytují a které tak můžou měnit hodnocení e-mailu. Sloupec `link` představuje samotný odkaz z emailu, sloupec `counter` uchovává informaci, kolikrát byl odkaz v e-mailech použit, a sloupec `rating` představuje hodnocení e-mailu, ve kterém se odkaz nacházel. Lze tak zjistit, jestli se jednalo o odkaz v testovacím e-mailu, nebo v možná testovacím e-mailu. Odkazy z e-mailů s hodnocením nižším než 50 nejsou do databáze ukládány.
- Tabulka `statistics` reprezentuje informaci o jednotlivých kontrolních bodech, které mění hodnocení e-mailu. Tabulka zná id kontrolního bodu, kolikrát byl během rozhodování už použit, kdy byl poprvé použit a kdy byl naposledy použit.



Obrázek 5.2: Schéma databáze

5.6 Modul salmonrelay

Na obrázku 5.1 se jedná o bod 5. Nejdůležitější funkcí, kterou si zavolá `salmonconclude`, je funkce `relay`. Jako jedním ze 4 parametrů přijme tato funkce konečné hodnocení emailu, název souboru, který byl vytvořen částí `receiver` při ukládání nového e-mailu do souboru, proměnnou `mail_request`, což je instance třídy `MailRequest` z modulu `mail`, a proměnnou `mail_fields`, což je slovník obsahující jednotlivé části e-mailu. Všechny tyto parametry jsou důležité pro další kroky, které tato funkce podnikne.

1. Zjistí si celou cestu k adresáři, kde je e-mail jako EML soubor skutečně uložen.
2. Ověří, jestli je přeposílání příjemci povoleno.
3. Pokud je ukládání e-mailů do předem stanoveného adresáře povoleno, přesune e-mail tam. Tento adresář je nastaven v konfiguračním souboru `salmon.yaml`.
4. Pokud je ukládání příloh e-mailů do předem stanoveného adresáře povoleno, uloží přílohy. Tento adresář je opět nastaven v konfiguračním souboru `salmon.yaml`.
5. Pokud je povoleno použití `MQTT`, tak odešle zprávu na `topic` (v překladu téma). Toto téma je možné nastavit v konfiguračním souboru `salmon.yaml` společně s přihlašovacími údaji k `MQTT`, které jsou povinné.
6. Ověří si, že přeposílání je povoleno a nebyl přesažen limit počtu přeposlaných e-mailů za určitý nastavený čas. Pokud je výsledné hodnocení e-mailu 70 a více, je e-mail postupně dopraven ke třídě `Relay`, která ho přepošle příjemci.

Ničení příloh probíhá tak, že se většina bytů uprostřed přílohy zamíchá, takže v případě, že se jednalo o soubor ve formátu PDF, stále bude možné soubor uložit do počítače příjemce, ale po jeho otevření mu bude prohlížečem PDF sděleno, že soubor je poškozen. Jedná se o méně nápadnou variantu, jak zacházet s nebezpečnými přílohami, než kdybychom přílohu vůbec nepřeposílali.

V případě odkazů, které se nachazí v e-mailu, se postupuje tak, že se v odkazu najde poslední tečka a jedno písmeno je v odkazu nahrazeno náhodným jiným písmenem. Nebude tak možné si všimnout, že odkaz byl zničen, ale zároveň nevezme příjemce e-mail na existující webovou stránku.

Obdobně se postupuje v případě pole `reply-to`, kde je uvedena adresa, na kterou může příjemce odpovědět. Pokud toto pole obsahuje například adresu `test@test.com`, tak se najde znak `"@"`a poslední tečka, tj. v tomto případě `test` a první písmeno `t` bude nahrazeno za `f`. Vznikne tak nová adresa `test@fest.com`, která vůbec nemusí existovat. Tento postup je nutné dodržovat, jelikož nemáme zájem měnit písmena v části před znakem zavináče. Adresa by stále mohla fungovat, pouze by vedla k jinému uživateli.

Kapitola 6

Testování a analýza dat

V této kapitole je popsáno testování našeho honeypotu. Dále je zde popsána analýza emailů, které byly nasbírány na našem honeypotu nasazeném na veřejném serveru, a také porovnání s existujícím e-mailovým honeypotem.

Za účelem otestování našeho honeypotu byly vytvořeny jednotkové testy, které jsou popsány v části 6.1. Tyto testy lze spustit odděleně od běžícího honeypotu, protože používají vlastní databázi. Dále také zapisování do logu se děje odděleně, protože tyto testy používají vlastní nastavení zápisu do logu v souboru `test_logging_relay.conf`. Testy používají svůj vlastní soubor obsahující pravidla `testing_rules.json`, kde je možné specifikovat co smí obsahovat e-mail, který má být označen jako testovací. Tento soubor odpovídá souboru `rules.json`, který je popsaný v části 5.3. Tyto testy také používají svůj konfigurační soubor `testing_salmon.yaml` tak, aby běh testů nemohl ovlivnit spuštěný honeypot, který naslouchá na nějaké IP adrese a portu. Tento soubor odpovídá souboru `salmon.yaml`, který je popsaný v části 4.2.

Honeypot byl dále spuštěn na 3 veřejných serverech za účelem otestování při reálném nasazení. Instalace proběhla pomocí Ansible playbook `honeypot.yml`, který je popsaný v části 4.3. Jako kontrolní uzel využívá playbook počítač, ze kterého je spuštěn. Výhodou této metody instalace je, že není nutné mít u sebe na počítači veškerý kód honeypotu, ale pouze adresář `ansible`, který obsahuje inventář (soubor `inventory`) s IP adresami spravovaných serverů, playbook `honeypot.yml`, konfigurační soubor `ansible.cfg` a soubor využívající `Jinja2`[1] (soubor `salmon.j2`), ze kterého se během instalace vygeneruje konfigurační soubor `salmon.yaml`. Na konci instalace je honeypot spuštěn na stanoveném portu a IP adrese, které uživatel během instalace zadá. Playbook obsahuje modul `git`, který si kód honeypotu stáhne z veřejného repozitáře v průběhu instalace, proto je adresář `ansible` k instalaci honeypotu dostačující.

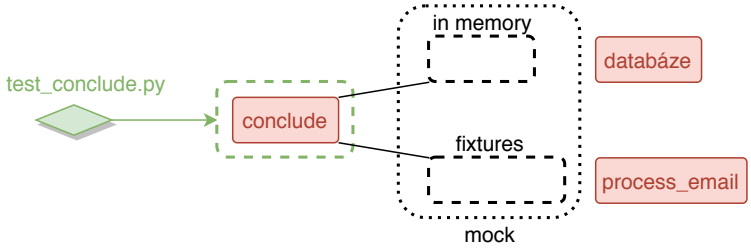
Na jednom ze serverů byl provoz monitorován po dobu 14 dní, na dalších 2 serverech byl provoz monitorován po dobu 22 dní. Všechny zachycené e-maily se ukládají proto, aby je pak bylo možné analyzovat. Analýza e-mailů je popsána v části 6.2. K analýze e-mailů pomáhá sbírání statistik do databázové tabulky `statistics` a zápisy z logů `salmon.log` z části `receiver` i `relay`. Tento sběr statistik funguje tak, že kdykoliv se během určování konečného hodnocení právě zpracovávaného e-mailu změní jeho hodnocení, tak je do tabulky `statistics` tato informace zapsána. Mechanismus tohoto ukládání změny hodnocení e-mailu funguje tak, že honeypot zjistí, jestli už podle daného pravidla (např. e-mail obsahuje přílohu) bylo někdy hodnocení nějakého e-mailu změněno, a pokud ano, pouze se

[1] Šablonovací jazyk pro programovací jazyk Python.

navýší počítadlo u tohoto pravidla. V opačném případě se vytvoří nový řádek v tabulce. Tabulka tak slouží k efektivnějšímu určení, podle jakých pravidel se hodnocení e-mailu mění nejčastěji. V budoucnu, pokud se zjistí, že honeypot téměř vůbec nemění hodnocení e-mailu podle nějakého definovaného pravidla, může být tak tato část kódu odstraněna a nahrazena jiným způsobem kontroly. Během analýzy jednoho e-mailu může být jeho hodnocení změněno vícekrát, to znamená, že pravidel může být použito více.

6.1 Jednotkové testy

Jednotkové testy testují zvolenou komponentu v izolaci. Obvykle se jedná o funkci, třídu nebo modul. V našem případě se jedná o testování nových modulů `salmonmailparser`, `salmonconclude` a modulu `salmondb`, který obsahuje třídy odpovídající databázovým tabulkám. Jak je vidět na obrázku 5.1, jednotlivé komponenty mezi sebou navzájem komunikují. Tuto komunikaci bylo nutné v našich testech simulovat, jelikož se jedná o jednotkové testy, nikoliv o integrační testy[2]. Náhražkám, které simulují nějakou část kódu, se v případě testování v angličtině říká různě, např. stub, mock nebo fake. V podstatě se vždy jedná o část kódu, která nahrazuje skutečný program nasazený v produkci, který ale není možné z nějakého důvodu přímo použít. Příkladem může být právě naše databáze, která je na začátku prázdná a postupně se plní daty, když jsou přes honeypot přeposílány e-maily.



Obrázek 6.1: Schéma použití mock při testování `salmonconclude`

Na obrázku 6.1 je možné vidět použití mock při našem testování, jedná se o část znázorněnou černě. Červenou barvou jsou znázorněny části kódu, které honeypot skutečně obsahuje, a zelená znázorňuje, že mají být spuštěny testy v souboru `test_conclude.py`, které testují funkci `conclude` v modulu `salmonconclude`. Místo využití reálné databáze se použije tzv. `in memory` databáze. To znamená, že při spuštění testů nikdy nevznikne skutečný soubor obsahující databázi, ale databáze je pouze v paměti, což znamená, že jakmile testy doběhnou, tak přestane existovat. Databázové tabulky jsou v tomto případě naplněny pouze testovacími daty, které byly vytvořeny podle toho, jak vypadají reálné e-maily, které jsou přes honeypot přeposílány. Druhou částí mock jsou `fixtures`, které jsou popsané v části Pytest a které zde nahrazují funkci `process_email`, která při běžném spuštění honeypotu volá funkci `conclude`.

Pytest

Pytest je framework (v překladu aplikační rámec) určený pro psaní testů v jazyce `Python`. Je určený pro použití na příkazové řádce. Dokáže automaticky najít testy a spustit je. Pytest

[2]Integrační testy slouží k otestování, že jednotlivé komponenty spolu dokáží spolupracovat.

očekává, že soubory obsahující testy budou pojmenovány tak, že jméno začíná `test_` nebo končí `_test.py`. Jak je vidět na obrázku 6.1, tato práce využívá první variantu pojmenování.

Dále naše testy používají `fixtures`. Jedná se o funkce, které dokážou jednotkovým testům vygenerovat data, připravit kontext, v němž se testy spouští, popř. připravit pomocné objekty použité v testech. Výhodou `fixtures` je jejich oddělenost od jednotlivých testů, částečná izolovanost a omezení použití globálních objektů. Všechny naše `fixtures` jsou tak oddělené od testů v souboru `salmon_test_case.py` ve třídě `SalmonTestCase`, ze které pak třídy testující jednotlivé moduly dědí. Také díky nim dochází k minimalizaci opakování kódu, např. když je před každým spuštěním několika testů potřeba inicializovat třídu. V našem příkladě na `test_conclude.py` se tyto pomocné funkce využijí tak, že místo skutečného volání funkce `process_email` z modulu `salmonmailparser` se datová struktura obsahující části e-mailu předpřipraví pomocí `fixtures`. K tomu, aby bylo možné poznat, že se jedná o `fixtures`, tak se k funkcím přidává dekorátor `@pytest.fixture` a testovací funkce pak názvy těchto pomocných funkcí přijímají jako svoje argumenty.

Testovací třídy

Celá cesta k adresáři obsahujícímu testy je `/relay/new/tests`, jelikož tyto testy slouží k otestování nových modulů v části `relay`. Názvy souborů obsahujících testovací třídy jsou `test_conclude.py` (obsahuje třídu `TestSalmonConclude`), `test_mailparser.py` (obsahuje třídu `TestSalmonMailParser`) a `test_models.py` (obsahuje třídu `TestSalmonModel`). Třídy mají následující účel:

- `SalmonTestCase` - třída obsahuje `fixtures` a ostatní výše zmíněné třídy z ní proto dědí. Obsahuje speciální třídní metodu `setup_class`, která před zavoláním prvního testu vygeneruje různé části e-mailu (např. e-mailové adresy nebo tělo e-mailu).

- `TestSalmonConclude` - třída využívá dvě speciální metody - `setup_method`, která se volá před každým spuštěním dalšího testu, a metodu `teardown_method`, která se volá po úspěšném dokončení každého testu. Testy v souboru `test_conclude.py` si nejdříve uloží potřebné informace do `in memory` databáze, následně si vytvoří datovou strukturu obsahující jednotlivé části e-mailu tak, jak by je modul `salmonmailparser` připravil modulu `salmonconclude` a zavolá se funkce `conclude`. Navrácené hodnocení e-mailu se pak porovná s očekávaným hodnocením.

- `TestSalmonMailParser` - třída využívá metody `setup_method` a `teardown_method`, které před každým zavoláním dalšího z testů nakopírují předem připravené soubory obsahující e-maily ve formátu EML, které se nachází v adresáři `rawspams`, do adresáře `queue/new`, kde je běžně očekává honeypot. Testy si vytvoří instanci třídy `MailRequest`, která je pak společně s názvem souboru obsahujícím e-mail předána funkci `process_email` modulu `salmonmailparser`. Testy pak následně kontrolují, jestli tato funkce dokázala jednotlivé části e-mailu správně zpracovat. Třídní metoda `setup_class` před spuštěním prvního testu připraví adresáře pro nové e-maily, nedoručitelné e-maily a přílohy. Metoda `teardown_class` se po doběhnutí posledního testu postará o jejich odstranění.

- `TestSalmonModel` - třída testuje databázové modely. Nejdříve se vytvoří záznam v určité databázové tabulce. Testy pak následně pomocí metod z modulu `salmondb` získají daný záznam z databáze a porovnají s daty, které byly na začátku do tabulky vloženy.

6.2 Analýza nasbíraných dat

Tato část kapitoly popisuje nasazení honeypotu na 3 veřejné servery poskytovatele v internetu. Provoz byl monitorován následovně:

- Server A - 22 dní (30.4 - 22.5.),
- server B - 22 dní (30.4 - 22.5.),
- server C - 14 dní (30.4 - 14.5.).

Instalace proběhla pomocí instalačního skriptu popsaného v části 4.3. K vyhodnocení byly použity statistiky, které honeypot sbírá a ukládá do tabulky `statistics` popsané v části 5.5. Všechny honeypoty fungovaly bez autentizace, byl otevřen port 25, byla povolena analýza textu v e-mailu a bylo povoleno přeposlání maximálně 12 e-mailů během jedné hodiny. Dále bylo povoleno přeposílání e-mailů příjemci, ukládání e-mailů a jejich příloh, stejně tak ničení částí e-mailu podle popisu v části 5.6. Během jednoho vyhodnocování emailu je možné použít více pravidel uvedených v části 5.3. Může se stát, že se nepoužije žádné pravidlo a konečné hodnocení e-mailu zůstane nulové. Taková situace může nastat v případě velkých kampaní, kdy je navíc možnost analýzy textu zakázána.

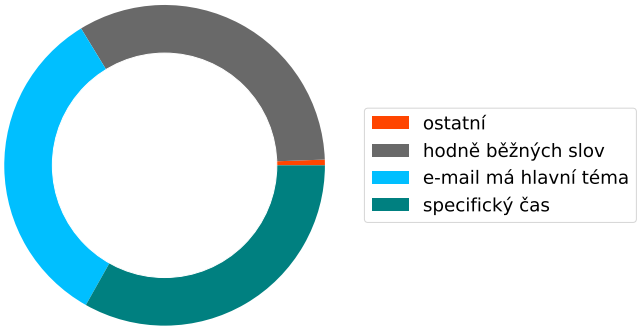
Server A nebyl 21 dní příliš využíván, bylo přes něj zasláno jen 60 e-mailů. Až 22. den byl honeypot využit k přeposlání velké kampaně. V logu `salmon.log` lze vidět, že e-maily byly odeslány ze stejné IP adresy a jsou všechny stejné. Jednalo se o e-maily, které se snaží své příjemce nalákat na to, že jim nějaká společnost má zaslat peníze a proto, aby se tak stalo, tak musí zaslat na uvedený e-mail své osobní údaje včetně kopie občanského průkazu. V této kampani bylo odesláno přes 8000 e-mailů. Všechny byly vyhodnoceny jako netestovací, protože obsahovaly příliš mnoho běžných slov jako jsou podstatná jména a slovesa. Dále bylo zjištěno, že text má nějaké hlavní téma, o kterém mluví a je příliš dlouhý na to, aby bylo testovací. Při zkoumání e-mailů je také možné si všimnout, že e-maily mají stejného odesílatele a příjemce. Další příjemci, pro které je e-mail nebezpečný, jsou uvedeni ve skryté kopii. Tento způsob zasílání zpráv je typický pro netestovací e-maily. E-maily, které byly označeny jako testovací, obsahovaly IP adresu honeypotu, byly krátké a neobsahují slova běžné řeči. Jedná se nejčastěji o tyto e-maily, kde IP adresa honeypotu byla nahrazena IP adresou 127.0.0.1 a e-maily příjemce a odesílatele e-mailem `ukazkovy@email.cz`.

```
From: ukazkovy@email.cz
Subject: 127.0.0.1
To: ukazkovy@email.cz
Date: Tue, 19 May 2020 13:46:55 +0200
X-Priority: 3
X-Library: Indy 8.0.25

t_Smtp.LocalIP
```

Celkově bylo přes honeypot zasláno 8300 e-mailů. Z tohoto počtu e-mailů bylo 37 vyhodnoceno jako testovací e-mail. Jedná se tedy o 0.5% z celkového počtu e-mailů, které byly přes honeypot poslány. Z grafu na obrázku 6.2 plyne, že nejčastěji se hodnocení e-mailu během jeho analýzy měnilo na základě toho, že obsahoval příliš mnoho běžných slov, měl nějaké téma, o kterém hovořil a přes honeypot byl zaslán v čas specifický pro testovací e-maily. Přesná data jsou uvedena v příloze B v tabulce B.1. Jako ostatní jsou v grafu označeny změny hodnocení e-mailu na základě toho, že e-mail obsahoval slovo test nebo

testing, dále obsahoval IP adresu honeypotu v těle e-mailu a předmětu, příjemce e-mailu byl použit v testovacím e-mailu již dříve, podobný testovací e-mail se v databázi už nacházel a také na základě toho, že e-mail obsahoval málo běžných slov.

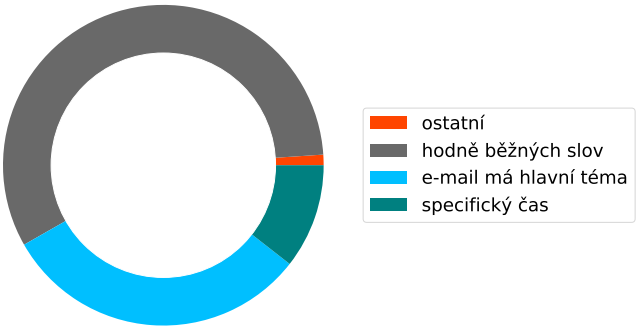


Obrázek 6.2: Graf nejčastěji použitých informací při změně hodnocení e-mailu (server A)

Server B byl spammery využíván nejvíce, protože port 25 byl otevřen už od listopadu 2019. Na server byl nasazen honeypot, jehož implementace se postupně upravovala. Spammeři tak měli šanci si ověřit, že přes server je možné e-maily přeposílat, a tak i za krátký čas přes něj bylo přeposláno velké množství e-mailů. Už první den byla přes honeypot zaslána velká kampaň, která své příjemce lákala k zaslání peněz a svých osobních údajů nějaké osobě, aby tak měli příjemci možnost dostat se k ještě větší částce. Na tomto serveru se často stalo, že e-mail byl označen jako nedoručitelný. Odesílatelé spamu si často zašlou email sami sobě a pouze jako skrytou kopii uvedou další příjemce. Občas se však stane, že zapomenou přidat pole `To` do e-mailu (přímého příjemce e-mailu) a uvedou pouze skryté příjemce. Toto pole je ale povinné, a tak e-mail není odeslán, i kdyby byl testovací. Celkově bylo přes honeypot zasláno 13517 e-mailů. Z tohoto počtu e-mailů bylo 55 vyhodnoceno jako testovací e-mail. Jedná se tedy o 0.4% z celkového počtu e-mailů, které byly přes honeypot poslány. Nejčastěji se jednalo o výše zmíněný vzor e-mailu, lze tedy předpokládat, že se jedná o stejného spammera. Na grafu na obrázku 6.3 je možné vidět, že se opět nejčastěji hodnocení e-mailu během jeho analýzy měnilo na základě toho, že obsahoval příliš mnoho běžných slov, měl nějaké téma, o kterém hovořil a přes honeypot byl zaslán v čas specifický pro testovací e-maily. Přesná data jsou uvedena v příloze B v tabulce B.2. Jako ostatní jsou v grafu označeny změny hodnocení e-mailu na základě toho, že e-mail obsahoval slovo test nebo testing, dále obsahoval IP adresu honeypotu v těle e-mailu a předmětu, příjemce e-mailu byl použit v testovacím e-mailu již dříve, podobný e-mail se v databázi už nacházel jako možná testovací a také na základě toho, že e-mail obsahoval málo běžných slov.

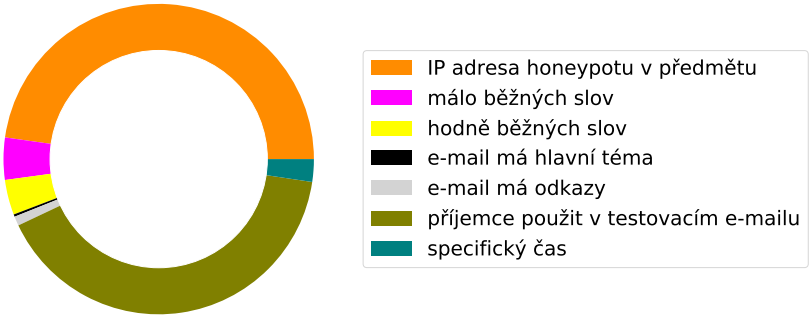
Server C nebyl příliš využívány, i když otevřený port 25 bylo možné vidět na stránce Shodan[3] do druhého dne od spuštění. Tento vyhledávač skenuje celou síť, prohledává otevřené TCP porty, na které „zaťuká“ a prohlédne si odpověď. Celkově bylo přes honeypot zasláno 78 e-mailů. Z tohoto počtu e-mailů bylo 26 vyhodnoceno jako testovací e-mail. Jedná se tedy o asi 33% z celkového počtu 78 e-mailů, které byly přes honeypot poslány. V 19% případů z celkového počtu 78 e-mailů má e-mail nějaké hlavní téma, o kterém se mluví. Často se jedná o nějakou osobu nebo instituci. Z toho plyne, že e-mail nebyl testovací, protože se snažil příjemce na něco nalákat. Stejně tak v případě e-mailů, které obsahovaly

[3] www.shodan.io



Obrázek 6.3: Graf nejčastěji použitých informací při změně hodnocení e-mailu (server B)

odkaz. Takový e-mail příjemce vyloženě láká na kliknutí na potencionálně škodlivý odkaz. Důvodem, proč bylo nakonec tolik e-mailů odesláno příjemci, je, že velmi často obsahovaly IP adresu honeypotu, takže byly vyhodnoceny jako testovací. Je možné předpokládat, že za delší časové období by se procento přeposlaných e-mailů snížilo, jelikož by spammeři začali honeypotu důvěřovat a začali by přes něj zasílat nějaké větší kampaně. Tento server je tak ukázkou toho, že obvykle zabere týdny než se přes honeypot začne zasílat spam ve velkém množství. Na grafu na obrázku 6.4 je možné vidět, že hodnocení e-mailu se během jeho analýzy měnilo nejčastěji na základě toho, že obsahoval málo běžných slov, IP adresu honeypotu v předmětu e-mailu a přes honeypot byl zaslán v čas specifický pro testovací e-maily. Dále je možné vidět, že oproti serveru A a B se příjemce testovacího e-mailu často opakuje. Přesná data jsou uvedena v příloze B v tabulce B.3.



Obrázek 6.4: Graf nejčastěji použitých informací při změně hodnocení e-mailu (server C)

Porovnání s honeypotem SHIVA

Proběhlo porovnání naší implementace s původním honeypotem SHIVA, jehož implementaci bylo za úkol vylepšit. Porovnání proběhlo pomocí dat, které byly v průběhu několika dní na honeypotu SHIVA nasbírána. Ze sdílené složky byly uložené e-maily ve formátu EML přesunuty na server, kde je nainstalovaný náš honeypot pomocí `playbook` popsaného

v části 4.3. Honeypot SHIVA běží už přes rok na serveru, je tedy spammery už hojně používaný a chodí přes něj i přes 2000 e-mailů za minutu. Na tomto honeypotu je povolena autentizace a bez přihlašovacích údajů ho tak není možné použít. Je zde nastaveno několik uživatelů a přihlašovacích hesel, které byly přidány také do našeho konfiguračního souboru `salmon.yaml` kvůli procházení jednotlivých kontrolních bodů z části 5.3. Na honeypotu SHIVA je nastaveno, že se smí přeposlat dvanáct e-mailů za jednu hodinu. Byly testovány e-maily, které byly zachyceny během 6 dnů. Celkově tak byla naše implementace otestována na 17141 e-mailech, které byly postupně vloženy do adresáře, ve kterém část `relay` očekává nově příchozí e-maily.

V tabulce 6.1 je shrnuto, kolik e-mailů daný den přes honeypot prošlo, kolik jich honeypot SHIVA přeposlal, kolik by nakonec přeposlal náš honeypot a kolik e-mailů bylo námi vyhodnoceno jako testovací postupným procházením e-mailů. Je možné vidět, že zatímco SHIVA přepošle přes sto e-mailů za den, náš honeypot jich přepošle pouze desítky. SHIVA navíc vždy přepošle pouze prvních dvanáct e-mailů, které přijdou danou hodinu, takže přeposílá zároveň testovací i netestovací e-maily. Pokud se však stane, že testovací e-mail je v řadě až třináctý, tak bylo úplně zbytečně přeposláno dvanáct netestovacích e-mailů a tím, že testovací e-mail nebyl odeslán, tak se honeypot může jevit jako nefunkční. Tato situace skutečně nastala v případě e-mailu, jehož tělo obsahovalo následující řádek:

```
1 UNIQ:smtp://m@ukaz.ka|smtp.ukaz.ka:25|m@ukaz.ka|honeypot_heslo
```

Přihlašovací jméno k honeypotu bylo nahrazeno `m@ukaz.ka` a heslo bylo nahrazeno za `honeypot_heslo`. Tento problém náš honeypot řeší tím, že počítá hodnocení e-mailu. V tomto případě by měl e-mail dostatečně vysoké hodnocení proto, aby byl odeslán. Jako testovací bylo naším honeypotem označeno 32 e-mailů z celkového počtu 17141 e-mailů, které bylo přes honeypot SHIVA přeposláno během 6 dnů. Nejčastěji se tak stalo díky heslu, které bylo obsaženo v těle e-mailu. Dále díku tomu, že příjemce e-mailu byl v naší databázi již uložen jako příjemce testovacích e-mailů a také kvůli tomu, že e-mail byl podobný emailu, který byl v naší databázi už uložen jako testovací. Honeypot SHIVA oproti tomu přeposlal 1212 e-mailů za 6 dnů a je velmi nepravděpodobné, že by spammeři během šesti dnů zaslali takové množství testovacích e-mailů.

Tabulka 6.1: Počet přeposlaných e-mailů

datum	počet e-mailů	přeposláno SHIVA	přeposláno salmon	testovací e-maily
3.5.2020	3658	264	0	0
4.5.2020	2357	180	0	2
5.5.2020	3106	228	3	8
6.5.2020	3396	216	21	21
7.5.2020	2409	180	6	6
8.5.2020	2215	144	2	2

Kapitola 7

Závěr

Tato bakalářská práce popisuje problematiku e-mailových honeypotů. Byla popsána teorie protokolů SMTP, IMAP a POP3 a to, jaký je účel jejich použití, pomocí jakých příkazů komunikují a co znamenají jednotlivé návratové kódy při komunikaci. Dále bylo popsáno, jakým způsobem musí být vytvořen e-mail, aby byl validní, a jak musí vypadat jednotlivé hlavičky a e-mailové adresy. Byla také rozebrána teorie spamu, jak taková pošta může vypadat a jak se mají uživatelé e-mailu chovat, aby se množství příchozí nevyžádané pošty nenavyšovalo. Také byl popsán mechanismus, kterým se dnes poskytovalé e-mailových serverů brání proti šíření spamu. Jako poslední téma je v teoretické části práce rozebrána teorie honeypotů. V práci je popsáno, co honeypot znamená a jak ho lze vytvořit. Jsou také popsány výhody a nevýhody honeypotů a jakým způsobem je možné je rozdělit do různých skupin podle jejich účelu využití a také podle toho, jak moc interagují s uživatelem.

V praktické části práce je popsáno existující řešení e-mailového honeypotu a jakým způsobem a proč bylo upraveno pro efektivnější využití honeypotu a také pro zmenšení možnosti jeho odhalení spammery. Architektura návrhu rozděluje nový honeypot na dvě části (`receiver` a `relay`), které se instalují do samostatných virtuálních prostředí tak, aby v případě potřeby bylo možné jednu z částí honeypotu vypnout nebo nahradit. Obě části jsou popsány pomocí schémat a to tak, aby bylo možné vidět, jak která část funguje a co bylo nutné implementovat, aby se snížila odhalitelnost honeypotu. V části návrhu je také popsáno využití protokolu MQTT a jakým způsobem je možné honeypot konfigurovat a instalovat. Honeypot je možné nainstalovat použitím shell skriptu nebo pomocí `playbook` napsaného v Ansible.

V implementační části je popis úpravy existujích a nových modulů, které byly do honeypotu přidány. Díky nové implementaci honeypot nejdříve rozdělí e-maily na části, které pak pomocí modulů a databáze dále zkoumá. Každému e-mailu je přiřazeno hodnocení, pomocí kterého se na konci rozhodne, jestli bude e-mail přeposlán příjemci. Honeypot také podporuje autentizaci, aby přes něj nebylo tak snadné e-maily přeposílat a nebyl tak příliš nápadný tím, že se bude chovat jako tzv. open-relay. Honeypot také dokáže ničit některé části e-mailu za účelem ochrany příjemců, což je jedna z vlastností, kterou je možné povolit nebo zákazat v konfiguračním souboru.

Po úspěšné implementaci bylo provedeno testování našeho kódu pomocí jednotkových testů. Honeypot byl nasazen na tři veřejné servery, ze kterých následně proběhla analýza nasbíraných dat pomocí statistik, které honeypot sbírá. Nakonec také proběhlo porovnání chování našeho honeypotu s již existujícím honeypotem SHIVA.

7.1 Výhled do budoucna

Budoucí možnosti vývoje vychází především z dlouhodobého nasazení honeypotu na veřejném serveru. Ze zachycených e-mailů by šlo provést analýzu, jaké e-maily se nejčastěji posílají, najít v nich nějaký vzor a pomoci tak lépe chránit uživatele před škodlivým obsahem a zároveň sbírat informace o útočnících. Honeypot by bylo také vhodné nasadit s použitím přihlašovacích údajů, aby nebyl identifikován a zablokován jako open-relay. Tento krok by šel realizovat pomocí honeytokenů popsaných v této práci. Šíření honeytokenů je však časově náročné a uplyne čas, než honeypot začne být využíván. Z použitých přihlašovacích údajů by pak šlo určit, kde se takové důvěrné informace šíří nejefektivněji. S použitím statistik honeypotu by také šlo po delší době vylepšit kontrolu některých částí e-mailu.

Literatura

[1] BARTL, V. *A client honeypot [online].* 2015 [cit. 2019-12-24]. Diplomová práce. Masarykova univerzita, Fakulta informatiky, Brno. Vedoucí práce NOVOTNÝ, M. Dostupné z: https://is.muni.cz/th/dtmhv/.

[2] CALETKA, O. *DKIM podpisy pro důveryhodnější e-mail* [online], 16. dubna 2015 [cit. 2020-05-17]. Dostupné z: https://www.root.cz/clanky/dkim-podpisy-pro-duveryhodnejsi-e-mail/.

[3] CALETKA, O. *SPF: proti spamu i přeposílání pošty* [online], 9. dubna 2015 [cit. 2020-05-17]. Dostupné z: https://www.root.cz/clanky/spf-proti-spamu-i-preposilani-posty.

[4] CRISPIN, M. *INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1* [Internet Requests for Comments]. RFC 3501. RFC Editor, 2003. http://www.rfc-editor.org/rfc/rfc3501.txt. Dostupné z: http://www.rfc-editor.org/rfc/rfc3501.txt.

[5] CROCKER, D. *STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES* [Internet Requests for Comments]. STD 11. RFC Editor, August 1982. http://www.rfc-editor.org/rfc/rfc822.txt. Dostupné z: http://www.rfc-editor.org/rfc/rfc822.txt.

[6] DHANARAJ, S. a KARTHIKEYANI, V. A study on e-mail image spam filtering techniques. In: *2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering.* Salem, India: Institute of Electrical and Electronics Engineers, únor 2013, s. 49–55. ISBN 978-1-4673-5845-3.

[7] DOČEKAL, D. B. *Co je to phishing a proč tomu říkáme rhybaření?* [online], 20. září 2016 [cit. 2019-12-22]. Dostupné z: https://365tipu.cz/2016/09/20/tip611-co-je-to-phishing-a-proc-tomu-rikame-rhybareni/.

[8] DOČEKAL, D. B. *Co je to CEO fraud?* [online], 18. ledna 2019 [cit. 2019-12-22]. Dostupné z: https://365tipu.cz/2019/01/18/tip1275-co-je-to-ceo-fraud/.

[9] HAZEL, P. *The Exim SMTP mail server: official guide for release 4.* 2. vyd. UIT Cambridge, 2007. ISBN 9780954452971.

[10] JOSHI, R. C. a SARDANA, A. *Honeypots: A New Paradigm to Information Security.* 1. vyd. Enfield, New Hampshire, USA: Science Publishers, 2011. ISBN 1578087082.

[11] KEJDUŠ, R. *Stručná historie emailu: už 40 let si posíláme počítačové dopisy* [online], 29. června 2012 [cit. 2019-12-10]. Dostupné z: https://www.cnews.cz/strucna-historie-emailu-uz-40-let-si-posilame-pocitacove-dopisy/.

[12] Klensin, J. *Simple Mail Transfer Protocol* [Internet Requests for Comments]. RFC 5321. RFC Editor, 2008. http://www.rfc-editor.org/rfc/rfc5321.txt. Dostupné z: http://www.rfc-editor.org/rfc/rfc5321.txt.

[13] Kurose, J. F. a Ross, K. W. *Computer networking: a top-down approach*. 7. vyd. Pearson, 2017. ISBN 9780133594140.

[14] Martin, M. *Everything About Spam Explained* [online], 26. září 2019 [cit. 2019-12-22]. Dostupné z: https://spadedesignlab.com/everything-about-spam-explained/.

[15] Matoušek, P. *Síťové aplikace a jejich architektura*. 1. vyd. Brno: VUTIUM, 2014. ISBN 978-80-214-3766-1.

[16] McAfee. *7 Types of Hacker Motivations* [online], 16. března 2011 [cit. 2019-12-10]. Dostupné z: https://www.mcafee.com/blogs/consumer/family-safety/7-types-of-hacker-motivations/.

[17] MERTA, M. *Detekce phishingu pomocí síťových pastí [online]*. 2016 [cit. 2019-12-24]. Diplomová práce. Masarykova univerzita, Fakulta informatiky, Brno. Vedoucí práce Vykopal, J. Dostupné z: https://is.muni.cz/th/zumf0/.

[18] News, C. *Mattel vs. Chinese cyberthieves: It's no game* [online], 24. dubna 2016 [cit. 2019-12-22]. Dostupné z: https://www.cbsnews.com/news/mattel-vs-chinese-cyberthieves-its-no-game/.

[19] Qassrawi, M. T. a Hongli Zhang. Client honeypots: Approaches and challenges. In: *4th International Conference on New Trends in Information Science and Service Science*. Gyeongju, South Korea: Institute of Electrical and Electronics Engineers, Květen 2010, s. 19–25. ISBN 978-89-88678-17-6.

[20] Regan, J. *What is Malware? How Malware Works and How to Remove It* [online], 11. července 2019 [cit. 2019-12-22]. Dostupné z: https://www.avg.com/en/signal/what-is-malware.

[21] Resnick, P. W. *Internet Message Format* [Internet Requests for Comments]. RFC 5322. RFC Editor, 2008. http://www.rfc-editor.org/rfc/rfc5322.txt. Dostupné z: http://www.rfc-editor.org/rfc/rfc5322.txt.

[22] Sarantinos, N., Benzaid, C., Arabiat, O. a Al Nemrat, A. Forensic Malware Analysis: The Value of Fuzzy Hashing Algorithms in Identifying Similarities. In: *2016 IEEE Trustcom/BigDataSE/ISPA*. Tianjin, China: Institute of Electrical and Electronics Engineers, Srpen 2016, s. 1782–1787. ISBN 978-1-5090-3205-1.

[23] Satrapa, P. *DMARC: ověření odesilatelovy domény* [online], 23. dubna 2015 [cit. 2020-05-17]. Dostupné z: https://www.root.cz/clanky/dmarc-overeni-odesilatelovy-domeny/.

[24] Spitzner, L. *Honeypots: Tracking Hackers*. 1. vyd. Addison-Wesley Professional, 2002. ISBN 0321108957.

[25] techopedia. *What is a Directory Harvest Attack (DHA)? - Definition from Techopedia* [online], 26. září 2012 [cit. 2020-05-10]. Dostupné z: https://www.techopedia.com/definition/1647/directory-harvest-attack-dha.

[26] ÚOOÚ. *Spam a nevyžádaná obchodní sdělení* [online], 13. prosince 2013 [cit. 2019-12-22]. Dostupné z: https://www.uoou.cz/spam-a-nevyzadana-obchodni-sdeleni/d-6170/p1=3109.

[27] Škorničková, E. *ÚOOÚ udělil rekordní pokutu za spam* [online], 21. května 2017 [cit. 2019-12-22]. Dostupné z: https://www.gdpr.cz/blog/uoou-udelil-rekordni-pokutu-za-spam/.

Příloha A

Obsah CD

Na přiloženém CD se nachází:

- tato práce ve formátu PDF
- zdrojové kódy honeypotu
- zdrojové kódy této práce v LaTeXu
- soubor README.md s návodem na instalaci a informacemi o vytvořených a upravených souborech

Příloha B

Tabulky použitých informací při detekci testovacích e-mailů

Tabulka B.1: Informace použité při detekci testovacích e-mailů (server A)

název pravidla	kolikrát použito
specifický testovací čas	8260
hodně běžných slov	8256
e-mail má nějaké hlavní téma, o kterém mluví	8246
IP adresa honeypotu v předmětu	61
málo běžných slov	34
příjemce byl použit v testovacím e-mailu	26
IP adresa honeypotu v těle e-mailu	10
slovo test v těle v e-mailu	8
e-mail má nějaké odkazy	2
podobný e-mail v tabulce s testovacími e-maily	1

Tabulka B.2: Informace použité při detekci testovacích e-mailů (server B)

název pravidla	kolikrát použito
specifický testovací čas	2178
hodně běžných slov	11803
e-mail má nějaké hlavní téma, o kterém mluví	6436
IP adresa honeypotu v předmětu	106
málo běžných slov	54
příjemce byl použit v testovacím e-mailu	54
IP adresa honeypotu v těle e-mailu	1
slovo test v těle v e-mailu	3
podobný e-mail v tabulce možná testovacích e-mailů	5171

Tabulka B.3: Informace použité při detekci testovacích e-mailů (server C)

název pravidla	kolikrát použito
specifický testovací čas	29
hodně běžných slov	15
e-mail má nějaké hlavní téma, o kterém mluví	15
IP adresa honeypotu v předmětu	42
málo běžných slov	27
příjemce byl použit v testovacím e-mailu	25
e-mail má nějaké odkazy	6