Univerzita Pavla Jozefa Šafárika v Košiciach

Prírodovedecká fakulta

ANALÝZA PHISHINGU Z POHĽADU HONEYPOTOV

BAKALÁRSKA PRÁCA

Študijný odbor: Informatika
Školiace pracovisko: Ústav informatiky
Vedúci záverečnej práce: RNDr. JUDr. Pavol Sokol, PhD.

Košice 2016 **Martin Glova**

Poďakovanie

Na tomto mieste by som chcel poďakovať dr. Pavlovi Sokolovi, môjmu vedúcemu práce, ktorý mi s celou tvorbou pomáhal, vybavoval veci, ak som niečo potreboval a aj v neskoré nočné hodiny odpovedal na moje e-maily a dohliadal na to, aby práca mala čo najväčšiu hodnotu.

Tiež by som sa chcel poďakovať rodine, kamarátom a známym, ktorí prejavili záujem vypočuť si o čom je moja práca a často ma povzbudili k tvorbe práve svojím úprimným záujmom a nadšením pre túto tému.

Univerzita P. J. Šafárika v Košiciach
Prírodovedecká fakulta

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Martin Glova
Študijný program: Informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: Bakalárska práca
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Analýza phishingu z pohľadu honeypotov

Názov EN: Analysis of phishing from the perspective of the honeypots

Cieľ:
(1) Analyzovať možnosti využitia honeypotov pri výskume phishingu.
(2) Porovnať aktuálne prístupy k analýze phishingu pomocou honeypotov.
(3) Navrhnúť a implementovať honeypot na analýzu phishingu.

Literatúra:
[1] Hadnagy, Ch.: Phishing Dark Waters The Offensive and Defensive Sides of Malicious Emails, Wiley, 2015
[2] Joshi, R. C., and Anjali Sardana. Honeypots: A New Paradigm to Information Security. Science Publishers, 2011.
[3] ENISA: Proactive Detection of Security Incidents, 2012,

Vedúci: RNDr. JUDr. Pavol Sokol, PhD.
Ústav : ÚINF - Ústav informatiky
Riaditeľ ústavu: prof. RNDr. Viliam Geffert, DrSc.

Dátum schválenia: 05.05.2016

prof. RNDr. Viliam Geffert, DrSc.
riaditeľ ústavu

Univerzita Pavla Jozefa Šafárika v Košiciach
Prírodovedecká fakulta
Ústav informatiky

Abstrakt

Sociálne inžinierstvo využíva najzraniteľnejšie miesto bezpečnosti - ľudský faktor. Veľmi častým prejavom sociálneho inžinierstva sú phishingové správy. Ich účelom je od používateľa získať prístupové údaje k e-mailovému účtu, bankovému účtu, resp. k iným systémom, ktoré využíva. V rámci práce sa venujeme phishingu. K tomuto účelu využívame tzv. honeypoty (pasce na útočníkov). Účelom honeypotov je prilákať útočníkov a zistiť o nich informácie.

Súčasťou práce je implementácia spamového honeypotu, ktorý bude spracovávať phishingové emaily a komunikovať s útočníkom. Medzi hlavné implementačné problémy, ktorým je nutné sa v práci venovať, môžeme zaradiť prenos phishingových e-mailov od používateľov, komunikácia s útočníkom (napr. vyplnenie formulára), zabezpečenie neodhaliteľnosti honeypotu a korelácia získaných údajov.

Kľúčové slová: phishing, honeypot, honeynet, spam, e-mail.

Abstract

Social engineering uses the most vulnerable part of information security - the human factor. Very common appearances of social engineering are phishing messages. Their purpose is to obtain access data to the e-mail account, bank account, respectively to other systems user uses. Within the paper we are devoted to phishing. For this purpose we use so called honeypots (traps the attacker). The purpose of the honeypot is to attract attackers and get information about them.

Part of the paper is to implement a spam honeypot which will handle phishing e-mails and communicating with an attacker. The main implementation issue that must be devoted in paper we can include the transfer of phishing e-mails from users, communication with an attacker (eg. filling out forms...), providing not-detect of honeypots and correlation of the data obtained.

Keywords: phishing, honeypot, honeynet, spam, e-mail.

Obsah

Zoznam skratiek a značiek

URL - Uniform Resource Locator

IP - Internet Protocol

DNS - Domain Name System

WWW - World Wide Web

SMTP - Simple Mail Transfer Protocol

API - Application Programming Interface

JSON - JavaScript Object Notation

Úvod

Elektronická pošta sa stala neodmysliteľnou súčasťou života pre väčšinu ľudí po celej zemi. Obsah e-mailu môže byť rôznorodý. V tomto množstve e-mailov sa však veľmi často objavujú aj e-maily, ktoré nie sú vyžiadané - tzv. spam. Spam sa vo väčšine prípadoch podarí odfiltrovať a je presmerovaný do špeciálneho adresára, ale v menšej miere môže prekonať spamový filter a dostať sa do adresára pre doručenú poštu. Väčšina ľudí si s tým poradí a veľmi rýchlo vie filtrovať, či ide o vyžiadanú alebo nevyžiadanú poštu do (spamového) koša. Takto nie je pre mnohých táto skutočnosť nijako nebezpečná.

Často sa už hovorí aj o internetovej bezpečnosti, opatrnosti pri klikaní na náhodné odkazy, nenavštevovaní neznámych webových stránok, kontrola protokolu HTTPS pri surfovaní, kde nám záleží, aby sa komunikácia nedostala k tretej strane atď. Napriek tomu sa nájde nemalé množstvo ľudí, ktorí tieto skutočnosti vedia podceniť. Niekedy stačí jednoduchá forma útoku a môže byť pre útočníka úspešný. Niekdy je potrebný viac cielený útok, aby útočník zaznamenal vyššiu úspešnosť útoku. Dôležité je ale povedať to, že aj jeden uniknutý e-mailový účet v organizácii môže znamenať závažné bezpečnostné riziko pre celú organizáciu.

Informovať ľudí o informačnej bezpečnosti je veľmi dôležité. Nie sú to len starí ľudia, ktorí mnohokrát "neovládajú" prostriedky informačnej bezpečnosti, ale s informačnou bezpečnosťou majú problém aj mladšie generácie používateľov. Je to však rôzne, no všetci ľudia, ktorým záleží na tom, aby ich práca s internetom bola zabezpečená, by mali mať práve dostatočnú informovanosť aj o formách sociálneho inžinierstva (napr. o phishingu).

Podľa prieskumu, ktorý sme vykonali a ktorého výsledky uvádzame v práci je vidieť, že na neznámy odkaz v e-maile môže kliknúť až necelých 6% používateľov a phishingový formulár následne vyše 54% z nich vyplniť (to je z pôvodného počtu vyše 3%). Reálne sa tak dá získať nemalý počet prístupov bez vynaloženia veľkej námahy.

V tejto práci sa venujeme analýze phishingu z pohľadu honeypotov. V druhej ka-

pitole sú zhrnuté rôzne známe prístupy a výskumy v tejto oblasti. Rozanalyzovali sme niekoľko odborných a vedeckých článkov, kde boli skúmané podobné problémy. Súčasne zhrnieme v čom je náš prístup iný. Kapitola obsahuje aj porovnanie jednotlivých prístupov.

V poslednej časti tejto práce je popísaný návrh a postup implementácie e-mailového honeypotu, ktorý bude spracovávať phishingové e-maily, ktoré mu do databázy môže poskytnúť ktokoľvek prostredníctvom doplnku do e-mailového klienta, ďalej vytvorí e-mailové konto, poskytne údaje tohto vytvoreného e-mailu v phishingovom formulári a bude čakať na prípadný útok. E-mailový server bude zabezpečený proti zneužitiu, pôjde o honeypot, ktorý bude simulovať reálne bežiaci e-mailový server, ktorý ale nikde žiadne správy nepošle, bude iba zaznamenávať všetku aktivitu, ktorá sa na ňom vykonáva prostredníctvom používateľov (v tomto prípade prostredníctvom útočníkov). V kapitole sú vysvetlené jednotlivé stavy systému, vysvetlená celá schéma systému a presnejšie implementačné informácie. Súčasne sme analyzovali problémové oblasti, ktoré by sa ešte dali vylepšiť, respektíve doplniť.

Kapitola 1

Spam a phishing

Termín **spam** je používaný na pomenovanie procesu posielania nevyžiadanej pošty bez konkrétneho cieleného adresáta vo veľkom množstve za komerčným účelom [12].

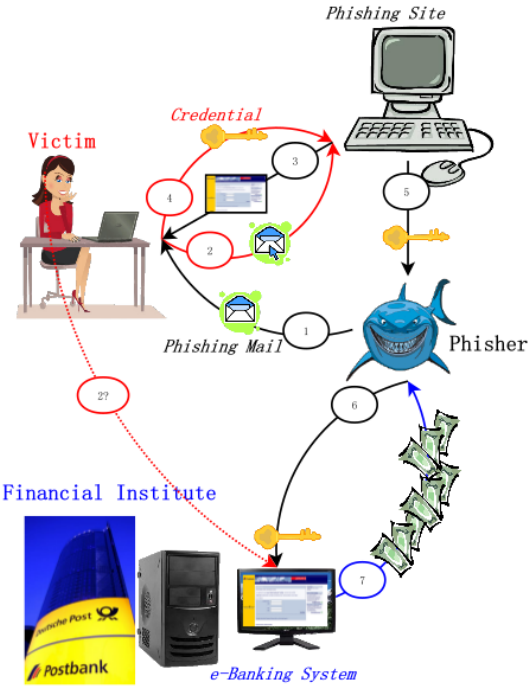
Spam je jedna z aktivít na internete, ktorá sa rozmohla v posledných rokoch. Dnes je zodpovedný za významnú časť e-mailov v obehu na internete. Prevažná časť spamu nie je pre používateľa nebezpečná, ale spam sa tiež používa ako prostriedok na odoslanie phishingu a šírenie škodlivého kódu [12], ktoré už pre používateľa nebezpečné určite sú.

Phishing je definovaný ako forma útoku sociálneho inžinierstva, v ktorom phisheri, t.j. útočníci oklamú obeť, aby získali súkromné informácie [4]. Phishingové útoky boli spočiatku zamerané všeobecne na spotrebiteľa. Ich cieľom bolo ukradnúť identitu a informácie o kreditných kartách. V súčasnosti sa phishing zameral na vysokoprofilové ciele, kde je cieľom ukradnúť výsledky tvorivej duševnej činnosti, firemné tajomstvá a citlivé informácie týkajúce sa národnej bezpečnosti [16]. Phishingové útoky zahrňujú tri hlavné fázy:

1. potenciálna obeť prijme tzv. **phish**,
2. obeť vykoná akcie navrhované v správe (zvyčajne ide na podvodnú webovú stránku, ale môže obsahovať aj inštalovanie malvéru alebo je potrebné odpovedať s uvedením citlivých informácií),
3. speňaženie ukradnutých informácií [16].

Bežný scenár útoku (obr. 1) začína poslaním phishingového e-mailu obeti. Adresát si myslí, že e-mail prišiel z legitímnej e-mailovej adresy, napríklad z banky alebo univerzity, ale pýta od neho prihlasovacie meno a chce aktualizovať niektoré osobné

údaje (napríklad heslo prislúchajúce prihlasovaciemu menu). Typicky je k e-mailu pridaná varovná správa, ktorá má presvedčiť prijímateľa emailu (obeť), aby vyplnenie údajov neodkladal a odpovedal okamžite. Správa typicky obsahuje webovú adresu, ktorá sa javí ako legitímna, ale presmeruje obeť na podvodnú webovú stránku, ktorá vyzerá identicky s originálnou. Ak obeť vyplní údaje na webovej stránke, napríklad meno a heslo, údaje sa pošlú útočníkovi a obeť je presmerovaná na originálnu webovú stránku, aby náhodou neodhalila útok.



Obr. 1: Štandardný scenár phishingového útoku [11]

Prvé phishingové útoky sú známe od roku 1996, ale v posledných rokoch ich počet narástol. Útočníci ich bežne používajú na cielené útoky na banky, elektronické obchodovanie alebo sociálne siete, aby ukradli tisícky, ak nie, milióny eur. Útoky sa stále zlepšujú a útočníci hľadajú nové spôsoby, aby oklamali aj problému znalých ľudí [4].

Phishing sa rozšíril mimo e-mailu aj do iných oblastí zahŕňajúc VOIP, SMS, okamžité správy, sociálne siete a dokonca masívne aj v hrách pre viac hráčov [13, 14, 15].

1.1 Spôsoby phishingových útokov

V posledných rokoch bol zaznamenaný nárast phishingových útokov kvôli zvýšenému počtu používania online služieb (napr. internetové transakcie alebo webové sídla s

elektronickým obchodovaním). Nové spôsoby phishingu, ako napríklad spread phishing alebo pharming (vysvetlené sú nižšie), sú neustále vo vývoji, za účelom nachytania aj bezpečnosti-znalých používateľov. V tejto sekcii zosumarizujeme efektívne spôsoby phishingových útokov [4].

1.1.1 Sady nástrojov na tvorbu phishingu

Obeťou phishingu sa môže stať aj ten, kto sa rozhodne využiť nejaký z online nástroj na jeho tvorbu [4]. Mali by pomôcť pri tvorbe a nastavovaní phishingovej webstránky, ale štúdie ukazujú [6], že takéto nástroje v skutočnosti neexistujú. Ukazuje sa, že práve tieto nástroje na tvorbu phishingu v sebe obsahujú zadné vrátka od tvorcov (tzv. backdoor), ktoré im preposielajú údaje, ktoré útočník nelegálne vyzbieral [4]. Útočník je tak mimo priameho kontaktu s obeťou a okrem toho sa zvyšuje aj efektivita zberu ilegálnych dát.

1.1.2 Podobné doménové mená

Je bežné, že rôzne organizácie si pred začatím svojho podnikania rezervujú doménové mená pre ich hlavné webové stránky. Na druhej strane útočníci si rezervujú doménové mená, ktoré sú podobné týmto legitímnym [4]. Ideou je hostovať rovnako vyzerajúcu webovú stránku, ako je tá legitímna s použitím podobnej domény. Bežne si užívatelia nekontrolujú všetky znaky v názve domény, a tak ľahko môžu podľahnúť tomuto útoku. Ako príklad môže poslúžiť podvodné doménové meno www.citiibank.com pre originálnu webovú stránku www.citibank.com.

V práci Dhamija et al., ktorá hovorí o tom, prečo phishing funguje [5], bolo dokázané, ako môže malá, ale chytrá zmena doménového mena spôsobiť veľké množstvo nachytaných obetí phishingu. V práci bolo doménové meno www.bankofthewest.com zmenené na www.bankofthevvest.com. Jedinou zmenou teda je zmenené jedno písmeno “w” na dve písmená “v”. Až 91% respondentov sa nechalo nachytať a nevšimli si túto zmenu.

1.1.3 Vizuálne klamanie

Dobre navrhnuté podvodné webové stránky môžu zaujať viacero obetí. Vyššie spomenutá práca [5] na 22 účastníkoch s rôznym zázemím ukázala, že dobre phishingové stránky môžu nachytať aj používateľov s dobrým bezpečnostným zázemím [5]. Phis-

hingové stránky, ktoré vytvorili, nachytali 90% respondentov. Respondenti sa nachytali, lebo videli, že webová stránka jednoducho vyzerá dobre. Niekedy dali ako dôvod, prečo tejto webovej stránke uverili aj to, že podvodné webové stránky nikdy nemôžu byť také dobré alebo pretože by museli (útočníci) vynaložiť pri ich kopírovaní veľa úsilia.

1.1.4 Pharming

Pharming alebo **DNS poisoning** je považovaný za jeden z najťažšie detekovateľných phishingových útokov. Útok zahŕňa posielanie podvodných záznamov do DNS servera, ktorý potom poskytne klientom podvodnú IP adresu. Prehliadače ale budú zobrazovať originálne doménové meno. Preto dokonca aj užívatelia, ktorí si dobre skontrolujú doménové meno webovej stránky, ktorú navštevujú, nezbadajú žiaden rozdiel [4].

1.1.5 Anti-Phishing Filter Evasion

Bezpečnostné filtre sa spoliehajú výlučne na význam textu, aby našli podozrivé reťazce a vzory. Niektorí útočníci preto zvyknú poslať namiesto textu obrázky obsahujúce text, aby prešli cez filtre. Keď proti-phishingové filtre nedetegujú podvodný e-mail alebo webovú stránku, je to už len na používateľovi, či prípadný podvod odhalí alebo nie [4].

1.1.6 Spear Phishing

Spear phishing je typ phishingového útoku, ktorý je zameraný na špecifickú vzorku používateľov. Útočníci si vyberú nejakú cieľovú skupinu obetí a skôr ako masové phishingové e-maily posielajú e-maily iba tejto skupine používateľov [4]. Útočníci sa neponáhľajú s prieskumom o istej skupine ľudí a vytvoria správu, ktorá je priamo adresovaná obeti a nejako s ňou súvisiaca [1]. Väčšinou ide o používateľov, ktorí majú menšie povedomie o takomto spôsobe podvodov (alebo sú vyberané podľa iných kritérií na základe nejakého kľúča - napríklad len používatelia jedného konkrétneho produktu). Dôvod, prečo vznikol spear phishing je ten, že takto vytipovaná cieľová skupina obetí môže viesť ešte k väčšej efektivite a neskoršiemu odhaleniu podvodnej činnosti, avšak ak už je správa raz odhalená, tak väčšinou sa vyvinie o to väčší tlak, aby sa hrozba odstránila. Pri necielenom útoku ľudia správu, ktorú ako útok

identifikovali, väčšinou odignorujú a predpokladajú, že tak sa k tomu postaví každý. V prípade cieleného útoku užívatelia po odhalení predpokladajú, že mohlo danému útoku podľahnúť väčšie množstvo používateľov.

1.2 Ochrana proti phishingu

Bolo vyvinutých niekoľko nástrojov, ktorých cieľom je bojovať proti phishingu. Tieto nástroje môžu byť rozdelené do týchto kategórii:

- bezpečnostné panely nástrojov v prehliadačoch,
- proti-phishingové filtre,
- proti-phishingové honeypoty.

Webové prehliadače používajú **bezpečnostné panely nástrojov**, ktoré sú navrhnuté na to, aby ukázali informácie týkajúce sa bezpečnosti o práve navštívenej webovej stránke. Ak panel nástrojov deteguje chýbajúci bezpečnostný certifikát, tak zobrazí varovnú správu, že navštívená webová stránka môže byť podvodná [7]. Niektoré štúdie hodnotia proti-phishingové panely nástrojov, aby testovali ich efektivitu v boji proti phishingu a zistili, že sú neefektívne v ochrane používateľov proti phishingovým útokov. Tieto varovné správy sú ignorované niektorými používateľmi z dôvodu ich neinformovanosti o phishingových útokoch [8, 9].

Druhou kategóriou nástrojov sú **phishingové filtre** typicky pracujú na princípe kontroly doménových mien navštevovaných webových stránok pomocou databázy nahlásených phishingových domén. Ak je nájdená zhoda, tak zablokujú prístup na webovú stránku alebo zobrazia varovanie. Tieto filtre však môžu byť obídené pomocou **pharmingového útoku** [10].

Poslednou skupinou nástrojov sú **honeypoty**, ktoré sú vo veľkej miere nasadzované, aby detegovali phishingové e-maily a identifikovali phishingové stránky [11]. Princípy ich fungovania sa snažíme využiť.

1.3 Phishingový test

Aby sme získali lepšiu predstavu o tom, čím sa útočník musí zaoberať pri posielaní phishingu, uskutočnili sme phishingový test. Metodiku a výsledky tohto testu uvádzame v tejto kapitole. Celý prieskum sme rozdelili na tri fázy a jeho jednotlivé výsledky sú

uvedené v nasledujúcich podsekciách. Jednotlivé fázy testu sa zamerali na špecifické skupiny používateľov (študenti, zamestnanci).

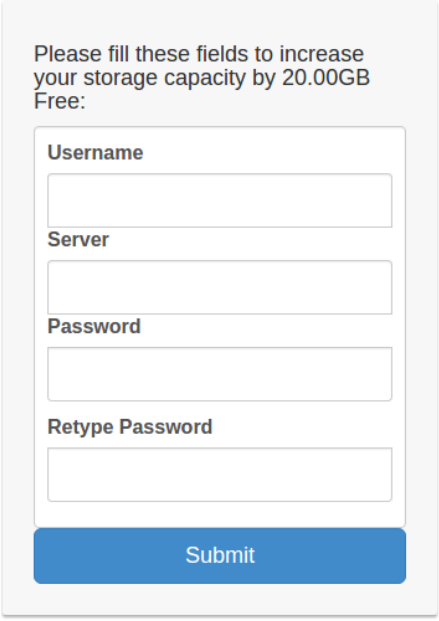
Dôležité spomenúť, že vo všetkých fázach testu bola použitá tá istá vzorka ľudí. Táto skutočnosť mohla spôsobiť ovplyvnenie vzorky skrz jednotlivé fázy phishingového testu. Tiež niektoré e-mailové kontá nemusia byť funkčné alebo používané.

1.3.1 1. fáza

V tejto fáze išlo o poslanie anglického e-mailu vzorke niečo vyše 10 tisíc ľudí. Vzhľad phishingovej stránky z 1. fázy je znázornený na obr. 2. V prvej fáze sme sa museli vysporiadať s niekoľkými problémami (napr. obmedzenia počtu poslaných e-mailov). Pôvodne sme mali v úmysle poslať všetky e-maily z jedného e-mailového účtu tak, aby v priebehu jednej noci mali všetci adresáti phishingové e-maily vo svojich schránkach. E-mailový server **azet.sk** má obmedzenie na poslanie e-mailov (iba 50 e-mailov na deň). To je veľmi silné obmedzenie na posielanie niekoľkých tisícov e-mailov. V dôsledku toho sme pristúpili k otestovaniu iných e-mailových serverov. **centrum.sk** bol problémový pri vypĺňaní. Nie je totiž možný prístup na tento server pomocou protokolu SMTP, iba cez webové rozhranie. Automatizátory vypĺňania formulárov boli však v tomto prípade nepoužiteľné a mali sme tušenie, že asi aj keby sa podarilo automatizovane zaslať e-mail, tak objem 10 tisíc e-mailov by bol pre tento server neúnosný.

Keďže sme sa chceli zamerať na to, aby e-maily naozaj používateľom prišli, vytvorili sme si e-mailový účet v službe **gmail.com**. Tu sa dá pristúpiť k serveru aj cez protokol SMTP. Podarilo sa tak poslať ďalších okolo 1000 e-mailov. Aj keď má **gmail.com** oficiálne obmedzenie zaslať len 500 e-mailov za deň. Postupne sme teda vytvorili ďalšie 4 e-mailové účty v službe **gmail.com** a podarilo sa teda poslať ďalších pár tisíc e-mailov. Nakoniec v priebehu 3 dní sa podarilo poslať väčšinu e-mailov, ale keďže sme chceli phishing rozoslať v čo najkratšom možnom čase, bolo nutné nájsť iné riešenie.

Posledné e-maily sme teda poslali z vlastného e-mailového serveru. Tento server sme používali aj vo všetkých ostatných fázach. V databáze sa ukladal každý prístup na odkaz (zaznamenala sa IP adresa, čas). Po vyplnení všetkých položiek formulára sa uložila IP adresa a čas vyplnenia spolu so všetkými položkami okrem hesla.



Obr. 2: Vzhľad webovej stránky, na ktorú smeroval odkaz z e-mailu pri 1. fáze

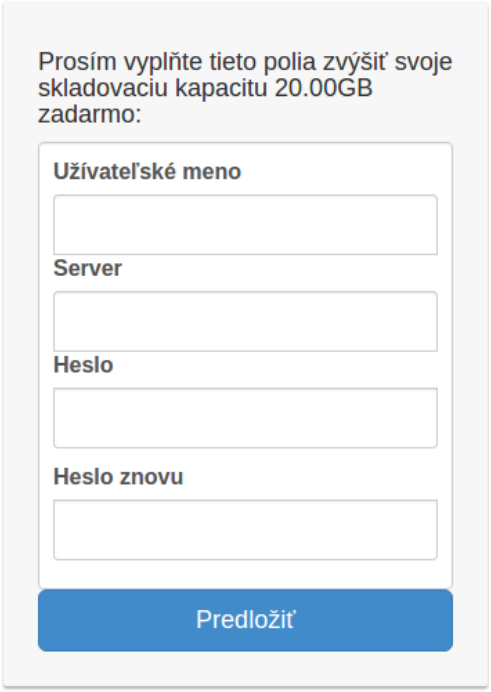
1.3.2 2. fáza

V tejto fáze sme zmenili jazyk phishingových e-mailov a phishingovej webovej stránky na slovenčinu (preložené cez **google translator** bez nejakej dodatočnej špeciálnej úpravy). Všetky e-maily už boli poslané cez náš e-mailový server. Ďalšou zmenou bol originálny odkaz pre každý e-mail. Dôvodom bola tvorba lepších štatistík (aká skupina ľudí klikla a vyplnila formulár atď.). Vzhľad phishingovej stránky z tejto fázy je znázornený na obr. 3.

V databáze sa ukladal každý prístup na odkaz (zaznamenala sa IP adresa, čas a jednoznačný identifikátor odkazu). Po vyplnení všetkých položiek formulára sa uložila IP adresa, čas vyplnenia a jednoznačný identifikátor odkazu spolu so všetkými položkami okrem hesla.

1.3.3 3. fáza

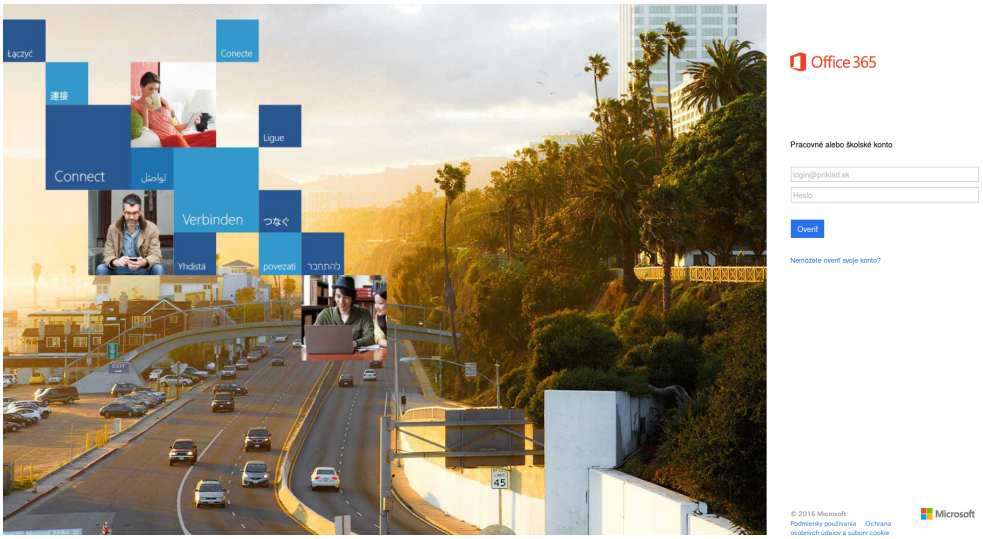
V poslednej fáze sme zmenili jazyk phishingovej webstránky a phishingového e-mailu na slovenčinu. Vzhľad phishingovej stránky bol rovnaký ako používa spoločnosť **Microsoft** pre svoju službu **Office365** (obr. 4). Phishingový e-mail v svojom texte upozorňoval na zvýšenú frekvenciu phishingu v poslednom období a vyžadoval spustenie



Obr. 3: Vzhľad webovej stránky, na ktorú smeroval odkaz z e-mailu pri 2. fáze

overovacieho modulu za účelom overenia hesla. Po vyplnení e-mailu a hesla bola obeť presmerovaná do online služby **Office365**.

V tejto fáze sme po kliknutí zaznamenávali opäť jednoznačný identifikátor e-mailu, IP adresou, časovú pečiatku a v prípade vyplnenia formulára aj prihlasovacie meno, aké bolo zadané.



Obr. 4: Vzhľad webovej stránky, na ktorú smeroval odkaz z e-mailu pri 3. fáze

Kapitola 2

Využitie honeypotov pri analýze phishingu

Spojenie phishingu a honeypotov môže byť rôzne. V našom prípade pôjde o zaslanie prihlasovacích údajov útočníkovi a umožnenie prístupu do e-mailového účtu, ktorý ale reálne nič mimo server odosielať nebude. Pohľad však môže byť aj iný. Napríklad na **Masarykovej univerzite v Brne** pracujú na e-mailovom honeypote, kde očakávajú phishingové e-maily a iné e-maily na svojom e-mailovom honeypote a potom na základe týchto incidentov sa snažia ochrániť bežné e-mailové účty pred touto hrozbou (napríklad zákazom určitých webových stránok) [17].

V tejto kapitole budeme analyzovať dostupné riešenia, ktoré sa zaoberajú podobnou oblasťou problémov ako táto práca. Budú rozanalyzované rôzne prístupy k spamu a phishingu, kde určitým spôsobom boli využité honeypoty.

2.1 Honeypoty, honeynety

Honeypot je nástroj, ktorý má vzhľad nejakej atraktívnej služby, série služieb, celého operačného systému alebo tiež celej siete. V skutočnosti ide o dôkladne uzavretý systém navrhnutý na to, aby nalákal útočníka [2].

V honeypote beží emulovaný operačný systém a služby, ktoré lákajú útočníka, aby ohrozil systém. V skutočnosti honeypoty zaznamenávajú všetko, čo útočník na systéme vykonal, aby do systému prenikol a tiež aj čo so systémom následne vykonal [3]. Sofistikovanejšie riešenie pre subjekty, ktoré chcú honeypoty využívať je použiť **honeynetu**, ktorý pozostáva z väčšieho počtu honeypotov distribuovaných po sieti daného subjektu. Honeynet je teda sieť honeypotov [2].

Honeypoty môžeme zaradiť do niekoľkých kategórií na základe niekoľkých kritérií. Podľa delenia na základe použitia môžeme honeypoty rozdeliť na produkčné a výskumné. **Produkčné honeypoty** sa používajú na ochranu určitej organizácie, kde zasielajú administrátorom upozornenia pri ohrození [37]. **Výskumné honeypoty** slúžia na skúmanie hrozieb a ochrany proti týmto hrozbám. Tieto výskumné honeypoty sa skôr zameriavajú na nejakú konkrétnu oblasť, ktorú skúmajú. Nechcú primárne slúžiť na ochranu nejakého subjektu [2].

Podľa úrovne interakcie poznáme honeypoty s **vysokou**, **strednou** a **nízkou interakciou**. Honeypoty s **nízkou interakciou** zvyčajne simulujú nejakú konkrétnu službu alebo protokol a sú menej náchylné na prienik do systému a ľahšie sa spravujú [3] (napr. Dionaea [38], Glastopf [39], Kippo [40] atď.). Často sú použité aj ako produkčné honeypoty najmä vďaka nízkej možnosti ohrozenia systému. Honeypoty s **vysokou interakciou** zahrňujú do útoku celý operačný systém a poskytujú tak väčšiu možnosť zneužitia (napr. Argos [41]). Na druhej strane sú schopné získať väčšie množstvo informácií. Pri získaní kontroly nad honeypotom môžeme totiž ohroziť celú lokálnu počítačovú sieť, lebo útočník je tak schopný ovládať počítač priamo v nejakej sieti. Honeypoty so **strednou interakciou** sú medzikrokom a snažia sa využiť výhody honeypotov s nízkou aj vysokou interakciou [2].

Honeypoty môžu byť fyzické alebo virtuálne. Pri **fyzických hoenypotoch** ide o jeden fyzický stroj s operačným systémom a službami, kde je honeypot pripojený do siete. **Virtuálne honeypoty** väčšinou využívajú jeden fyzický stroj, ktorý je hostiteľom viacerých virtuálnych honeypotov [3].

Poslednou klasifikáciou honeypotov je delenie na klientske a serverovské. **Klientske honeypoty** sú tie, ktoré interakciu s útočníkom vyhľadávajú a napodobňujú používateľa, ktorý spustil aplikáciu napríklad na serveri. **Serverové honeypoty** zase pasívne čakajú na interakciu od útočníka [2].

2.2 Identifikácia phishingu

V predchádzajúcej podkapitole sme sa venovali honeypotom a honeynetom. V tejto časti rozoberáme rôzne techniky na identifikáciu phishingu. Existujú rôzne programové spôsoby, ako je možné identifikovať, že ide o phishing. Medzi základné patria:

- phihsingové blacklisty,
- heuristiky,

- vizuálna podobnosť,
- strojové učenie [18].

Phihsingový blacklist je zoznam webových stránok, kde sa potenciálne nachádza phishing. Pri tejto metóde sa detegujú odkazy v e-maily a porovnávajú, či sa nachádza daný odkaz na webovú stránku v databáze. Ako príklad môže poslúžiť Google Safe Browsing API [19], ktoré umožňuje porovnávať blacklist aktualizovaný Googlom.

Pri **phishingových heuristikách** skúmame rôzne vlastnosti e-mailu a podľa vybraných vlastnosti a prítomnosti určitej vlastnosti v e-maile môžeme určiť, či o phishing ide alebo nie. Tieto phishingové heuristiky majú tú výhodu, že môžu odhaliť aj zero-day útoky. **Zero-day útoky** sú útoky na zraniteľnosť softvéru, ktorá zatiaľ neznáma pre poskytovateľa softvéru [26]. Typy zvolených vlastností pre rozlíšenie spamu alebo phishingu je možné nájsť v článku [18].

Pri **vizuálnej podobnosti** ide o výpočtovo náročnejšiu úlohu, lebo sa webové stránky v e-maile vizuálne porovnajú s webovými stránkami, ktoré sa najčastejšie používajú na phishing (tzv. **whitelist**). Ak teda daná webová stránka má veľmi veľkú podobu a nie je to webová stránka z whitelistu, zrejme pôjde práve o phishing.

Pri **strojovom učení** je potrebné e-maily rozdeliť do určitých kategórii a následne systému poskytnúť dostatočne dobrú dátovú vzorku, aby sa na základe nej naučil ďalej triediť nový e-mail [18].

2.3 Prístupy k analýze spamu a phishingu pomocou honeypotov

Je niekoľko možností, aké kroky vykonať po identifikácií spamu. Najjednoduchšia, ale z hľadiska výskumu nezaujímavá možnosť, je spam vymazať. Jednoducho po identifikovaní spamu (či už phishingu alebo iného druhu spamu) danú správu vymažeme.

Ďalšia možnosť je **spam analyzovať**. Tu sa ponúka viacero možností. Môžeme napríklad zaznamenať rôzne informácie spamu, a to ponechať na ďalšiu analýzu. Medzi takéto aspekty patrí samotný text správy, čo obsahuje, prípadne odkazy na webové stránky, prílohy spamu, atď. Po zozbieraní takýchto údajov možno robiť ďalšie analýzy.

Môžeme napríklad pracovať s jedným spamom samostatne a analyzovať, čo sa v ňom nachádza. Tomuto sa venujú napríklad v článku [20], kde zachytia parametre

spamu do databázy a niektoré rovno analyzujú v automatizovaných testoch. V danom článku používajú honeynet na to, aby rozanalyzovali prílohy spamu a identifikovali prípadný škodlivý kód (malvér) v uzavretom virtuálnom systéme. Systém sa špeciálne phishingu nevenuje, skôr kladie dôraz na spam a škodlivý kód v e-mailoch.

V článkoch [11, 22] sa autori zameriavajú na implementáciu e-banking systému, ktorý automaticky deteguje útočníkov. V článku [11] napríklad systém poskytne útočníkovi falošné údaje a na základe nich vie identifikovať, kto je útočník. Každý kto použije tieto falošné údaje, je automaticky považovaný za útočníka a obeť, ktorá tiež údaje poskytla je informovaná bankovým systémom, že došlo k zneužitiu účtu. Pre správne fungovanie systému sa predpokladá to, že útočník získa množinu prístupov a medzi nimi bude aj podstrčený **honeytoken** (je to špeciálny druh honeypotu - elektronická informácia [11]). Takto je považovaný za útočníka. Systém využíva aj fakt, že transakcie neprebiehajú hneď, a teda pri skorom identifikovaní podvodu sa dá zabrániť finančným únikom.

Honeypoty sa tiež dajú využiť na simuláciu prehľadávania internetu reálnym používateľom. V článku [21] je popísaný model klientskeho honeypotu. Model napodobňuje používateľa surfujúceho po internete. Zdrojom webových stránok, ktoré pritom využíva môžu byť napríklad aj webové stránky zo spamových e-mailov. Systém simuluje bežné série krokov, ktoré užívatelia robia - s cieľom identifikovať webové stránky s malvérom, ktoré sa zameriavajú na zraniteľnosť klientskej aplikácie a chcú prevziať nad klientskym počítačom kontrolu. Tento model je celkom užitočný aj pre nás, pretože tiež pôjde o určité napodobnenie správania sa používateľa, ktorý vyhľadáva a vypĺňa phishingové formuláre.

Článok [23] konštatuje, že vypĺňanie falošných údajov v phishingových formulároch zvyšuje cenu útoku. Databáza útočníka je tak naplnená falošnými údajmi. Zvyšuje sa tak náklad pri získaní týchto údajov. Tiež používajú falošné webové stránky s falošnými údajmi, ktoré útočníkovi vyplnili a potom si zaznamenávajú jeho aktivitu.

Enisa (European Network and Information Security Agency) vo svojej štúdii [3] popisuje rôzne druhy honeypotom a medzi nimi je aj honeypot Monkey-Spider, ktorý používa webový prehľadávač Heritrix [30] na ukladanie webových stránok a následné skenovanie pomocou Clam AntiVirus [31].

2.4 Zmysel v posielaní spamu a phishingu

Na základe vedeckých článkov [11, 21, 24] je možné dospieť k názoru, že spam a phishing predstavujú jednu zo závažných bezpečnostných hrozieb.

Dôvod, prečo takéto správy pre niekoho má zmysel posielať, je ten, že najslabším článkom v bezpečnosti akokoľvek dobre zabezpečeného systému je **ľudský faktor**. Dôkazom toho je aj mediálne známy prípad, kde sa podarilo útočníkom dostať do **Twiter** účtu len na základe niekoľkých telefonátov do ústrední rôznych spoločností a po sérii obnovy hesiel na rôznych účtoch ako **Amazon**, **Apple**, **Google** [25]. Je to dôkaz toho, že aj zdanlivo jednoduchá, nevinne vyzerajúca požiadavka na niekoho, môže viesť k vydarenému bezpečnostnému incidentu.

Phishing je teda cestou a spôsobom, ako zaútočiť na najslabší článok zabezpečenia systémov - používateľov. Konkrétne môže ísť o jednoduché priamočiare vyžiadanie prihlasovacích alebo iných citlivých údajov pod zámienkou pomôcť používateľovi. To ale vedie k poskytnutiu týchto údajov tretej strane. Prípadne aj nepozornosť a kontrola známych bezpečnostných prvkov (napríklad ignorovanie neplatnosti certifikátov alebo nekontrolovanie tohto faktu atď.) môže viesť k phishingovému útoku.

2.4.1 Ochrana proti phishingu

Je nutné spomenúť aj to, ako je možné sa proti phishingu brániť. Jednou možnosťou je dôkladne kontrolovať bezpečnosť každej komunikácie, ktorá je sprostredkovaná cez informačno-komunikačné technológie. Konkrétne môže ísť o zabezpečenie počítača proti škodlivému kódu a prienikom a tiež aj proti každej nechcenej manipulácii týmito technológiami (napríklad aj v rámci fyzickej bezpečnosti). Ďalším spôsobom ochrany proti phishingu je následná kontrola zabezpečenia a neignorovanie bezpečnostných upozornení (napríklad o neplatnosti certifikátu). V neposlednom rade je dôležitá neustála kontrola pri komunikácii na internete, kde obom stranám záleží, aby sa obsah ich komunikácie nedostal k tretej strane (napríklad komunikácia cez šifrovaný protokol).

Vyššie uvedené body je však často veľmi ťažké kontrolovať. Kontrola týchto bodov môže časovo zaťažiť používateľov systému. Z tohto dôvodu sa vymyslel aj systém na rôzne iné, rozšírené druhy autentifikácie. Môže ísť o **GRID karty** (ale aj iné prenositeľné veci - napríklad aj čipy), prípadne dvojitú autentifikáciu [11] (napríklad pomocou stále aktuálneho SMS kódu, atď). Všetky tieto rozšírené spôsoby autorizácie už pokrývajú nedostatky, ktoré boli spôsobené pôvodnou jednostupňovou autorizáciu.

2.5 Využitie honeypotov pri analýze spamu

Spôsobov ako využiť honeypoty a honeynety pri analýze spamu je niekoľko. V článku [20] využívajú tieto systémy pri detegovaní malvéru v prílohach spamových e-mailov. V článku [11] naopak poskytujú falošné prihlasovacie údaje pri phishingových e-mailoch útočníkovi z dôvodu následnej analýzy využitia týchto údajov zo strany útočníka.

Honeypoty sa tiež dajú využiť na simuláciu prehľadávania internetu reálnym používateľom. V článku [21] je popísané, ako sa snaží honeypot simulovať prácu používateľa na internete pri prehliadaní webových stránok. Zdrojom takýchto webových stránok môžu byť napríklad aj webové stránky zo spamových e-mailov.

V našom systéme budeme vypĺňať phishingové formuláre, čo bude niečo ako klientsky honeypot. Súčasne budeme mať serverový honeypot - e-mailový honeypot, ktorého údaje budeme vypĺňať a následne čakať na útočníka, aby sme mohli zaznamenať jeho aktivitu.

Kapitola 3

Návrh a implementácia e-mailového honeypotu

Táto časť práce sa venuje návrhu a implementácii e-mailového honeypotu. V rámci tejto kapitoly sa venujeme nasledujúcim problémom:

- získavanie phishingových e-mailov,
- vypĺňanie phishingových formulárov,

V našom systéme budeme vypĺňať phishingové formuláre, čo bude niečo ako klientsky honeypot. Súčasne budeme mať serverový honeypot - e-mailový honeypot, ktorého údaje budeme vypĺňať a následne čakať na útočníka, aby sme mohli zaznamenať jeho aktivitu.

3.1 Udalosť

Udalosť je znázornená na obr. 5. Za **udalosť** v našom systéme budeme považovať nastavenie systému do počiatočného stavu. To je vyvolané zaslaním požiadavky na známu webovú stránku, kde sa v POST požiadavke (je to požiadavka na server, ktorá v tele požiadavky môže serveru poslať nejaké údaje [27]) čaká pôvodný e-mail aj s hlavičkou. Implementácia je vytvorená pomocou Spring Frameworku 1.3.0 pre Javu [32].

Takáto implementácia ponúka možnosť doimplementovať napríklad doplnok do e-mailového klienta, ktorý používateľovi poskytne možnosť, jednoduchou interakciou s prostredím, zaslať phishingový e-mail do nášho systému. Analyzovali sme rôzne

možnosti tvorby doplnkov ako napríklad doplnok do Thunderbirdu alebo do Outlooku a nakoniec sme sa rozhodli vytvoriť doplnok pre Outlook.

3.2 Prípad

Každá udalosť sa môže stať **prípadom** (je to časť obr. 5, ktorá má tmavšie stavy). To nastane vtedy, ak systém po analýze predpokladá, že na webovej stránke je formulár s phishingovým obsahom. V takom prípade sa ho snaží vyplniť. Práve tieto prípady sú jednotlivé riadky v administračnom systéme, kde je zaznamenaný aj jeden z ich stavov:

- "Spracovanie...",
- "Nie je link na vyplnenie",
- "Neidentifikovana polozka",
- "Logovanie",
- "Ukoncene".

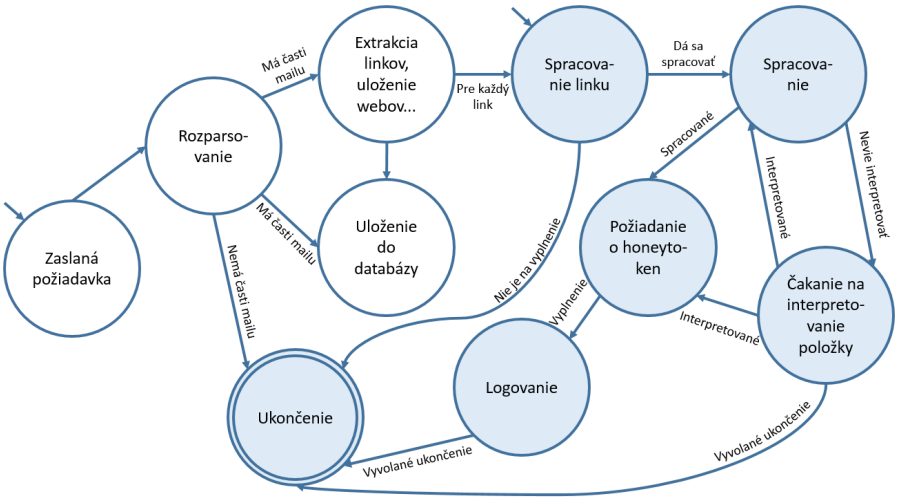
V prípade, že niekto zašle priamo odkaz do systému alebo sa extrahovaním z tela e-mailu tento odkaz získa a systém označil odkaz za odkaz s formulárom, objaví sa v databáze označený ako **"Spracovanie..."**. Práve výsledkom tejto fázy je rozhodnutie o vyplnení/nevyplnení odkazov. Systém sa pokúša vyplniť každý taký odkaz, kde nájde aspoň jednu položku na vyplnenie.

Ak formulár na odkaze vedel systém vyplniť, požiada e-mailový honeypot o honeytoken na prihlásenie do e-mailového honeypotu a prípad prejde do stavu **"Logovanie"**. E-mailový honeypot potom vie zaslať informácie o účte pre administračný mód, ktorý ich následne vie zobraziť.

V prípade, že formulár, na ktorý odkazuje odkaz z phishingového e-mailu, systém nevedel vyplniť, prípad sa dostane do stavu **"Naidentifikovana polozka"**. V administračnom móde potom uvidíme konkrétne položky prípadu, ktoré sa ako podarilo/nepodarilo vyplniť spolu z informáciami o týchto položkách, aby sme sa vedeli rozhodnúť o akú položku ide. Systém umožňuje vykonanie troch činností. Odkaz môžeme označiť **"Nie je link na vyplnenie"**, čím sa celý proces ukončí. To urobíme v prípade, ak uznáme, že webová stránka nie je phishingová a položky, na vyplnenie ktoré na nej sú, nie sú pre položky phishingového formulára. Môžeme tiež systém

naučiť interpretovať dané položky a nabudúce už takáto istá situácia nevznikne alebo iba pre tento prípad označíme systému, ako má položku vyplniť. V oboch posledných prípadoch sa prípad prepne do stavu **“Logovanie”**.

Prípad je označený za **“Ukoncene”**, ak sa takto rozhodol administrátor pre daný prípad. Takto sa ukončí zaznamenávanie a znefunkční sa prístup na daný účet v e-mailovom honeypote. Do tohto stavu sa vieme dostať len zo stavu “Logovanie”.



Obr. 5: Schéma zobrazujúca udalosť (modré stavy zobrazujú stavy prípadu)

3.3 Databázový model systému

V pozadí celého systému sa nachádza databáza, kde sa uchovávajú jednotlivé údaje pre samotný systém a aj pre administračné rozhranie. V systéme je použitá MySQL databáza (Ver 14.14 Distrib 5.5.47 [28]).

Databáza pozostáva z 8 tabuliek (obr. 6). Prvou tabuľkou je **mail**. Tu sú informácie o e-maily zaslanom do systému. Uchová sa jeho originálny tvar, ako bol zaslaný, obzvlášť aj text e-mailu (kde očakávame phishingové odkazy) a predmet e-mailu. Tiež si pridávame k e-mailu aj číselný jednoznačný identifikátor.

Tabuľka **honeyuser** pozostáva z číselného identifikátora, mena, servera a hesla. Tieto údaje sú použité pri vypĺňaní formulárov a viažu sa k jednotlivým prípadom (preto je tam väzba s prípadom 1:1 [33]).

Tabuľka **stav** obsahuje identifikátor stavu a jeho názov. Stavy môžu byť “Spracovanie...”, “Nie je link na vyplnenie”, “Naidentifikovana polozka”, “Logovanie”, “Ukoncene”, viažu sa na prípad a podrobnejšie boli popísané vyššie.

Ďalšou tabuľkou je tabuľka **prípad**, kde je popísaný jeden prípad. Pozostáva zo stĺpcov id (jednoznačný číselný identifikátor), honeyuserId, stav, mailId a link. HoneyuserId smeruje na tabuľku s honeyuser-mi, aby bolo jasné, na ktorého používateľa je daný prípad viazaný. Je povolená hodnota NULL (nedefinovaná hodnota), lebo na začiatku ešte nevieme, či daný prípad bude potrebné vyplniť (napr. ak zistíme, že neide o phishing). Stav smeruje na tabuľku stavov (spomínaná vyššie). MailId smeruje na tabuľku mail spomínanú vyššie, aby bolo jasné z akého e-mailu vznikol tento prípad a odkaz, kde je samotná adresa odkazu. MailId môže byť aj NULL, ak bol odkaz poslaný priamo. Stĺpec link už obsahuje samotný odkaz prípadu.

V tabuľke **polozky** je iba číselný identifikátor a položky prihlasovacie meno ("login"), "heslo", "server", "e-mail" alebo "none" (ak položku netreba interpretovať, lebo nie je podstatná).

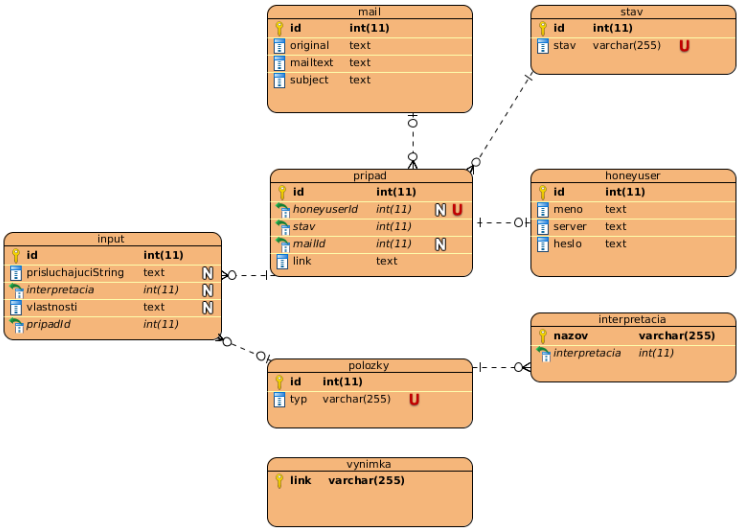
Tabuľka **interpretácie**, kde je názov položky ako jednoznačný identifikátor (jednu položku možno interpretovať iba jedným spôsobom) a interpretácia smerujúca na id z predchádzajúcej tabuľky, aby bolo jasné, ako daný názov položky má byť interpretovaný. Význam tejto tabuľky je ďalej popísaný v časti, ktorá je venovaná automatickému vypĺňaniu formulárov.

Predposledná tabuľka **input** predstavuje pamäť systému ak sa nepodarilo vyplniť formulár (dostal sa do stavu "Naidentifikovana polozka"). Obsahuje informácie o položkách na vyplnenie pre jednotlivé prípady. Sú to informácie ako prisluchajuciString, interpretacia, vlastnosti a pripadId. PripadId smeruje na konkrétny prípad, na ktorý sa záznam viaže. Interpretácia obsahuje hodnotu, ak sa ju už podarilo systémom interpretovať (alebo bola dodaná cez administračné rozhranie). PrisluchajuciString obsahuje časť webovej stránky (bez tagov), ktorá bola pred položkou na vyplnenie ale za predchádzajúcou (ak také existuje, teda dané položka nie je prvá).

Tabuľka **vynimka** obahuje iba jeden stĺpec a to sú odkazy, ktoré majú byť ignorované. Môžeme tam pridať známe odkazy, o ktorých vieme, že neobsahujú phishing a ušetriť tak systému čas analýzy.

3.4 Rozparsovanie e-mailu

Zo stavového automatu (obr. 5) je vidieť, že po zaslaní požiadavky sa e-mail rozparsuje a uložia sa základné údaje. V tomto kroku prebehne rozparsovanie e-mailu pomocou Python skriptu (upravený kód dostupný na [36]), ktorý hneď vygeneruje údaje vo formáte JSON ("JavaScript Object Notation" - syntax na ukladanie a výmenu dát



Obr. 6: Databázový model systému

[29]) a pošle ho do ďalšej metódy (uloženie do databázy a extrakcia odkazov). Tieto údaje obsahujú originál e-mailu (kompletný e-mail aj s hlavičkou), obsah e-mailu a predmet e-mailu. Obsah sa zaznamenáva pre ďalšiu analýzu odkazov a následné pokračovanie algoritmu. Predmet e-mailu sa zaznamenáva z dôvodu prehľadnosti v administračnom rozhraní.

3.4.1 Uloženie e-mailu do databázy

Na vstup prídu údaje vo formáte JSON vygenerované v predchádzajúcom kroku a daný záznam sa už môže uložiť do databázy. V databázovom modele je tabuľka označená ako **mail**, do ktorej sa záznam uloží. E-mail sa tam uloží iba ak sa identifikujú jeho základné časti: predmet a samotný text e-mailu. Text e-mailu systém ďalej analyzuje.

3.4.2 Analýza odkazov v e-maily

Pomocou regulárnych výrazov systém vyhľadá odkazy v e-maily. Tieto odkazy následne vyskúša, či smerujú na nejakú adresu, ktorá je funkčná. Ak áno, zálohuje si webovú stránku a daný odkaz považuje za phishing. V danej chvíli sa otvára nový prípad, ktorý už bude aj označený v administračnom rozhraní.

Tento stav vieme zavolať aj osobitným spôsobom, a to, ak do systému zašleme len odkaz na webovú stránku. Odkaz sa spracuje a systém zistí, či ide o odkaz s formulárom, ktorý by mohol byť vyplnený. Ak ho systém deteguje ako neplatný odkaz s formulárom, hneď končíme a neuloží sa o tom ani informácia v databáze. Ak ho však

systém označí ako odkaz s formulárom, vytvorí v tabuľke s prípadmi nový prípad, ktorý sa nastaví do stavu "Spracovanie...". Nasledujúci postup je bližšie špecifikovaný v predchádzajúcich častiach.

Implementácia tejto časti je urobená v Jave, pričom na uloženie webovej stránky sme vyskúšali niekoľko možností. Webovú stránku je možné stiahnuť pomocou niekoľkých nástrojov. Momentálne používame linuxový program **wget** s niekoľkými prepínačmi [34]. Syntax príkazu vyzerá následovne, pričom "link" je URL adresa k webovej stránke, ktorú chceme uložiť a "adresa" je cesta, kde chceme odkaz uložiť:

```
wget -E -H -k -K -p -U Mozilla link -P adresa
```

Názov adresára, kde daný odkaz ukladáme, môžeme zvoliť podľa ID, ktoré pridelí databáza pri tvorbe nového prípadu. Ukladáme totiž iba webové stránky, pre ktoré sa robí aj záznam v databáze.

3.5 Vytvorenie a logovanie HoneyUsera

Ak je systém schopný vyplniť formulár a vie ako interpretovať jednotlivé položky formulára, môže požiadať e-mailový honeypot o vytvorenie **HoneyUsera** a poskytnutie honeytokenov na prihlásenie. Ďalšou nevyhnutnou udalosťou je nainicializovanie určitej e-mailovej komunikácie a súčasne pridanie phishingového e-mailu, na ktorý honeypot reagoval.

Samotné zaznamenávanie spočíva v zisťovaní informácií na e-mailovom honeypote. Ide o spôsob prihlásenia útočníka, čas, odkiaľ atď. Naše riešenia prinesie plnú automatizáciu získavania a vypĺňania phishingových formulárov. V súčasnej dobe sa tieto činnosti vykonávajú manuálne.

3.6 Vypĺňanie phishingových formulárov

Problém správneho vyplnenia formulára sa ukázal ako jeden z najkomplikovanejších problémov tejto záverečnej práce. Nami zvolený algoritmus sa skladá z niekoľkých častí. Aby sme formulár považovali za vyplnený, je najdôležitejšie vyplniť prihlasovacie údaje - meno a heslo (prípadne meno, názov servera a heslo). Po identifikácii týchto položiek je už prakticky phishingový formulár vyplnený. Ak sa v rámci formulára nachádzajú aj iné položky, nie sú také podstatné, a ich účelom je zakrytie odhalenia.

Navrhnutý algoritmus funguje vo viacerých fázach. Časť tohto algoritmu je popísaná presnejšie aj s implementáciou v prílohe. Najprv si získame html kód webovej stránky, ktorú je potrebné vyplniť a identifikuje na nej položky formulára a ich jednotlivé atribúty (id, name, typ atď.). Identifikovanie položky, kde je potrebné vyplniť heslo/e-mail môže uľahčiť to, že má svoj tip označený ako "password"/"email". Nie vždy je to pravidlo. Najprv systém vyskúša na základe známych názvov identifikovať jednotlivé položky. Na to poslúži databáza známych názvov položiek, kde napr. názvy ako "pass", "password", "confirm_password" sú pre náš účel ekvivalentné názvy. Takto sa môže podariť identifikovať všetky potrebné položky. Môže nastať prípad, že heslo nebude typu "password" a aj samotné vstupy formulára neprinesú veľmi veľa informácií (napr. id, name a iné položky sú len náhodné čísla alebo reťazce).

V tomto prípade sa berú do úvahy aj reťazce, ktoré sú medzi vstupmi formulára. Vychádza sa z faktu, aby modla obeť vyplniť jednotlivé položky formulára správne, malo by sa niekde nachádzať niektoré kľúčové slovo na identifikáciu položiek.

Systém sa niekedy môže spoľahnúť na to, že útočník spracuje aj zle vyplnené údaje. Napriek tomu, že sa systém vyskúša na akejkoľvek veľkej množine údajov, môžu sa vyskytnúť také údaje, kde to nebude fungovať.

Samotné vyplnenie potom prebieha zaslaním POST požiadavky na server s interpretovanými údajmi. Analyzovali sme aj doplnok **Selenium**, ktorým sa dajú tiež vyplniť webové formuláre, ale generovanie POST požiadavky je rýchlejšie a pre náš účel praktickejšie, lebo sa vyhne rôznym blokovaniam JavaScriptu, ak by sme napríklad niektorú položku nevyplnili. Vieme, že údaje sa na server takto dostanú a je vysoký predpoklad, že budú spracované systémom útočníka bez špeciálnej analýzy ako boli na server zaslané (väčšina útokov prebieha automatizovane).

3.7 Možné stavy ukončenia udalosti, prípadu

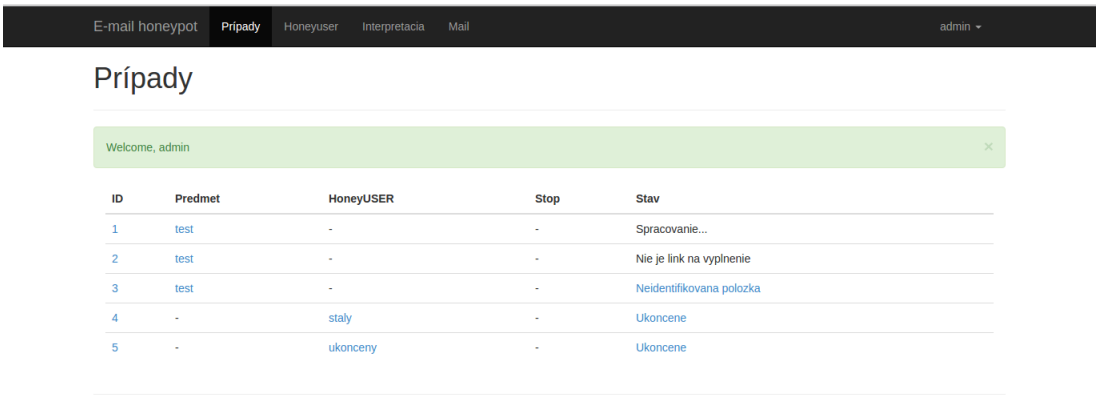
V rámci systému môžu nastať štyri prípady pre ukončenie udalosti. Prípad sa môže ukončiť v dvoch situáciách. Prípad sa môže ukončiť buď ak je v priebehu zaznamenávania alebo ešte pred zaznamenávaním. Pred zaznamenávaním to je, ak sa nepodarí interpretovať všetky položky a zistíme na základe analýzy odkazu, že v skutočnosti ani neide o phishing.

Udalosť ešte môžeme ukončiť v ďalších dvoch prípadoch. Prvým prípadom je stav, keď aj po rozparsovaní nemajú jednotlivé časti zhodné časti s e-mailovým originálom. Druhý prípad môže nastať, ak udalosť skončí hneď po analýze linku, lebo na ňom nie

sú žiadne položky na vyplnenie.

3.8 Administračné rozhranie systému

Administračné rozhranie je naprogramované v jazyku PHP a je použitý framework FuelPHP [35]. Administračné rozhranie spočíva v uľahčení práce s jednotlivými prípadmi a zobrazením informácií o nich. Skladá sa z jednoduchého menu, kde sú jednak všetky prípady, honeyuseri, interpretácie (kvôli prípadnej úprave) a e-maily, ktoré sú v systéme zaznamenané. Webové rozhranie je dostupné na adrese `https://support.eu.sk/emailhoneypot/admin` a náhľad po prihlásení a tabuľku s prípadmi je znázornený na obr. 7.



Obr. 7: Vzhľad administračného rozhrania po prihlásení

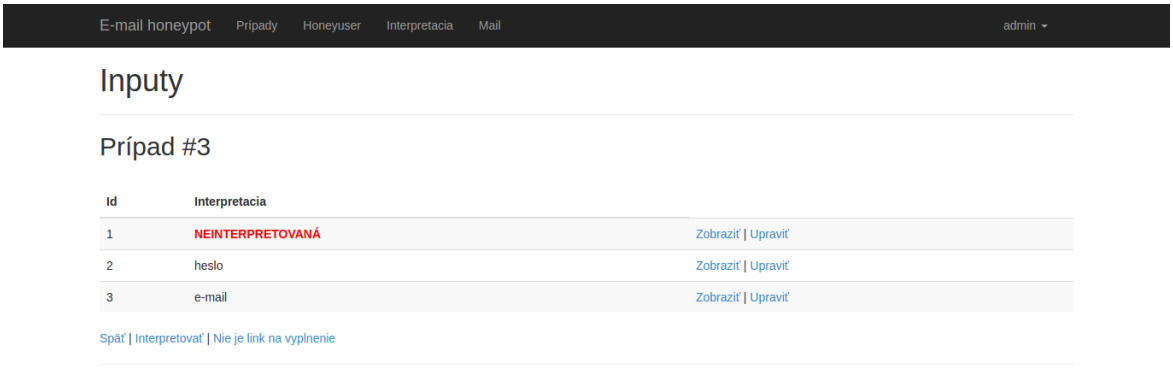
V prípade úvodnej obrazovky teda ide o tabuľku s piatimi stĺpcami. Po kliknutí na ID sa zobrazia jednotlivé parametre daného prípadu. Po kliknutí na predmet e-mailu sa zobrazí celý e-mail, z ktorého prípad vznikol. Môže sa stať aj to, že prípad bol pridaný iba na základe linku. Potom je v tomto stĺpci "-". Ak je už k prípadu pridelený HoneyUser (alebo bol) po kliknutí na jeho meno sa o ňom zobrazia informácie. Položka "STOP" je dostupná iba ak je daný prípad v stave zaznamenávania. Po kliknutí na "STOP" sa manuálne ukončí prípad.

Stav je na kliknutie dostupný v troch prípadoch:

- ak je stav "Ukoncene",
- ak je stav "Logovanie",

- ak je stav “Neidentifikovana polozka”.

V prvých dvoch prípadoch sa zobrazia záznamy z používania tohto účtu získané z e-mailového honeypotu. V poslednom prípade sa zobrazí webová stránka na pomoc pri interpretovaní položky (obr. 8).



Obr. 8: Vzhľad webovej stránky po kliknutí na “Neidentifikovana polozka”

Teraz môžeme pomôcť položku interpretovať (upraviť červeným zvýraznené interpretácie a uložiť) a potom kliknúť na “Interpretovať”, čo by malo systému opäť poslať rovnaký link na spracovanie spolu s vykonanými úpravami na interpretáciách. Môžeme aj usúdiť, že neide o phishing a klikneme na “Nie je link na vyplnenie”. Potom sa zmení aj stav daného prípadu na “Nie je link na vyplnenie”.

Môžeme systému upravovať aktuálne nastavené interpretácie položiek. Po kliknutí na položku “Interpretacia” v menu vieme pridávať novú interpretáciu alebo upravovať prípadne mazať staré. Takto vieme systém učiť interpretovať nové názvy položiek.

V položke menu “Honeyuser” sa zobrazí zoznam HoneyUserov a informácií o nich a v položke menu “Mail” sa zobrazí zoznam e-mailov a informácií o nich. Ide o všetkých HoneyUserov a všetky e-maily v systéme.

Záver

V práci sme vysvetlili základné pojmy ako honeypot alebo phishing a snažili sa vyzdvihnúť práve dôležitosť výskumu aj v tejto oblasti - phishing ako stále veľmi aktuálny problém a honeypot ako jeden zo spôsobov získavania informácií o útočníkoch. Ukázali sme, ako sa dá phishing kategorizovať a tiež aj to, že rôzne kategórie phishingu môžu dosahovať pre útočníka inú úspešnosť. Napríklad práve spear phishing, čo je istá forma cieleného phishingu na určitú kategóriu ľudí, môže zaznamenať aj niekoľkonásobne lepšiu úspešnosť ako štandardná forma phishingu.

V práci sme sa ďalej venovali porovnaniu a analýze aktuálnych prístupov v oblasti phishingu. Zaznamenali sme niekoľko prác, ktoré k analýze phishingu alebo spamu využívali honeypoty. Niektoré práce ponúkali ľahšie nasaditeľné riešenia, v iných prípadoch išlo skôr o práce, ktoré navrhovali zložitejšie nasaditeľné prostredie. Práve naše riešenie sme sa snažili navrhnúť tak, aby bolo čím ľahšie nasaditeľné, ale malo čím vyššiu pridanú hodnotu. Súčasne aby bolo schopné monitorovať útočníka v čo najväčšej miere.

V našom návrhu e-mailového honeypotu sme narazili na niekoľko problémov. Niektoré sa podarilo vyriešiť, niektoré sú v štádiu riešenia a niektoré sú iba v štádiu návrhu. E-mailový honeypot spočíva vo vypĺňaní phishingových formulárov honeytokenmi, ktorými je možné pristúpiť k e-mailovému honeypotu. Získať phishingové e-maily alebo priamo webové stránky sa dá od používateľov (u nás je to navrhnutý doplnok do e-mailového klienta, ktorý vie k nám e-mail preposlať) alebo z rôznych online zdrojov zhromažďujúcich práve tieto webové stránky. Po získaní phishingovej stránky sa ako jeden z najzávažnejších problémov ukázalo práve automatizované vypĺňanie formulároch na týchto webových stránkach, ktoré by malo fungovať čím viac autonómne.

Naše riešenie je popísané v tretej kapitole a otestované na sade phishingových stránok (u nás bola zdrojom takýchto webových stránok na testovanie webová stránka [42]). Až dlhodobá prevádzka ukáže jeho skutočnú funkčnosť. Každopádne aj čias-

točná automatizácia môže v našom výskume zefektívniť prácu a pomôcť dosahovať lepšie výsledky. Po vyplnení phishingových stránok už nasleduje práca e-mailového honeypotu, ktorého úlohou je logovať každú aktivitu na účte.

E-mailový honeypot je úplne navrhnutý a v určitých častiach aj implementovaný. Klientsky honeypot, ktorý vypĺňa phishingové stránky je naimplementovaný a otestovaný na množine reálnych phishingový stránok. Výzvou do budúcna je možno aj zlepšenie tohto prístupu a vypĺňanie webových stránok generovaných pomocou JavaScriptu, prípadne riešenie iných špeciálnych prípadov webových stránok (tie sú ale vo výrazne menšej miere bežné). Ďalšou výzvou do budúcna je analýza údajov za účelom získania presnejších informácií o útočníkoch.

Zoznam použitej literatúry

[1] HADNAGY, Christopher; FINCHER, Michele. Phishing Dark Waters: The Offensive and Defensive Sides of Malicious Emails. John Wiley & Sons, 2015.

[2] JOSHI, R. C.; SARDANA, Anjali (ed.). Honeypots: A New Paradigm to Information Security. CRC Press, 2011.

[3] ENISA: Proactive Detection of Security Incidents, 2012.

[4] DARWISH, Ali; ZARKA, A. E.; ALOUL, Fadi. Towards understanding phishing victims' profile. In: Computer Systems and Industrial Informatics (ICCSII), 2012 International Conference on. IEEE, 2012. p. 1-5.

[5] DHAMIJA, Rachna; TYGAR, J. Doug; HEARST, Marti. Why phishing works. In: Proceedings of the SIGCHI conference on Human Factors in computing systems. ACM, 2006. p. 581-590.

[6] COVA, Marco; KRUEGEL, Christopher; VIGNA, Giovanni. There Is No Free Phish: An Analysis of"Freeänd Live Phishing Kits. WOOT, 2008, 8: 1-8.

[7] CRANOR, Lorrie Faith, et al. Phinding Phish: An Evaluation of Anti-Phishing Toolbars. In: NDSS. 2007.

[8] WU, Min; MILLER, Robert C.; GARFINKEL, Simson L. Do security toolbars actually prevent phishing attacks?. In: Proceedings of the SIGCHI conference on Human Factors in computing systems. ACM, 2006. p. 601-610.

[9] EGELMAN, Serge; CRANOR, Lorrie Faith; HONG, Jason. You've been warned: an empirical study of the effectiveness of web browser phishing warnings. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, 2008. p. 1065-1074.

[10] OLLMANN, Gunter. The Pharming Guide: understanding & preventing DNS-related attacks by phishers. Next Generation Security Software, 2005.

[11] LI, Shujun; SCHMITZ, Roland. A novel anti-phishing framework based on honeypots. IEEE, 2009.

[12] STEDING-JESSEN, Klaus; MONTES, Antonio; VIJAYKUMAR, Nandamudi L. Design and Implementation of a Proxy Emulator to Capture Spam in Low Interaction Honeypots.

[13] ARTHUR, Charles. Facebook hit by phishing attack. The Guardian, 2009.

[14] CAVALLI, E. World of Warcraft phishing attempts on the rise. Wired (Apr. 29, 2009).

[15] VERISIGN; Fraud Alert: Phishing: The Latest Tactics and Potential Business Impact. White Paper, 2009. http://www.verisign.com/static/phishing-tactics.pdf

[16] HONG, Jason. The state of phishing attacks. Communications of the ACM, 2012, 55.1: 74-81.

[17] UNIVERZITA, Masarykova. PhiGARo [online]. 2013 [cit. 2015-03-17]. Dostupný z WWW: http://www.muni.cz/ics/services/csirt/tools/phigaro.

[18] KHONJI, Majid; IRAQI, Youssef; JONES, Andrew. Phishing detection: a literature survey. Communications Surveys & Tutorials, IEEE, 2013, 15.4: 2091-2121.

[19] GOOGLE, "Google safe browsing API," http://code.google.com/apis/safebrowsing/, accessed Oct 2011.

[20] RATHGEB, Erwin P.; HOFFSTADT, Dirk. The e-mail honeypot system concept, implementation and field test results. In: Digital Society, 2008 Second International Conference on the. IEEE, 2008. p. 1-6.

[21] QASSRAWI, Mahmoud T.; ZHANG, Hongli. Client honeypots: Approaches and challenges. In: New Trends in Information Science and Service Science (NISS), 2010 4th International Conference on. IEEE, 2010. p. 19-25.

[22] CHAUHAN, Shubhika; SHIWANI, Savita. A honeypots based anti-phishing framework. In: Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2014 International Conference on. IEEE, 2014. p. 618-625.

[23] GAJEK, Sebastian; SADEGHI, Ahmad-Reza. A forensic framework for tracing phishers. In: The Future of Identity in the Information Society. Springer US, 2007. p. 23-35.

[24] HUSÁK, Martin; CEGAN, Jakub. PhiGARo: Automatic Phishing Detection and Incident Response Framework. In: Availability, Reliability and Security (ARES), 2014 Ninth International Conference on. IEEE, 2014. p. 295-302.

[25] Kontá na Gmail či Me.com v ohrození. [online]. [cit. 2015-10-26]. http://www.zive.sk/clanok/59676/konta-na-gmail-ci-me-com-v-ohrozeni

[26] What is a Zero-Day Vulnerability?. [online]. [cit. 2016-01-26]. http://www.pctools.com/security-news/zero-day-vulnerability/

[27] Hypertext Transfer Protocol (HTTP/1.1). [online]. [cit. 2016-05-16]. http://tools.ietf.org/html/rfc7231#section-4.3.3

[28] Changes in MySQL 5.5.47. [online]. [cit. 2016-05-16]. http://dev.mysql.com/doc/relnotes/mysql/5.5/en/news-5-5-47.html

[29] JSON Tutorial. [online]. [cit. 2016-05-16]. http://www.w3schools.com/json/

[30] Heritrix. [online]. [cit. 2016-05-16]. https://webarchive.jira.com/wiki/display/Heritrix/Heritrix

[31] ClamAV. [online]. [cit. 2016-05-16]. http://www.clamav.net/

[32] Spring Boot 1.3.0 Release Notes. [online]. [cit. 2016-05-13]. https://github.com/spring-projects/spring-boot/wiki/spring-boot-1.3-release-notes

[33] One-to-one relationship. [online]. [cit. 2016-05-09]. http://www.databaseprimer.com/pages/relationship_1to1/

[34] GNU Wget 1.17.1 Manual. [online]. [cit. 2016-05-09]. https://www.gnu.org/software/wget/manual/wget.html

[35] FuelPHP. [online]. [cit. 2016-05-09]. http://fuelphp.com/

[36] MailToJson. [online]. [cit. 2016-03-09]. https://github.com/Newsman/MailToJson

[37] MOKUBE, Iyatiti; ADAMS, Michele. Honeypots: concepts, approaches, and challenges. In: Proceedings of the 45th annual southeast regional conference. ACM, 2007. p. 321-326.

[38] dionaea. [online]. [cit. 2016-05-16]. https://github.com/rep/dionaea

[39] Glastopf. [online]. [cit. 2016-05-16]. https://github.com/mushorg/glastopf

[40] Kippo. [online]. [cit. 2016-05-16]. https://github.com/desaster/kippo

[41] Argos. [online]. [cit. 2016-05-16]. http://www.few.vu.nl/argos/

[42] PhishTank. [online]. [cit. 2016-04-10]. http://www.phishtank.com/

Príloha A

CD so:

- zdrojovými kódmi na vypĺňanie formulárov,
- zdrojovými kódmi administračného rozhrania.

Príloha B

Niektoré významnejšie časti kódu spolu s komentármi sú popísané nižšie.

Extrakcia odkazov z reťazca

Na vstupe metódy je reťazec, z ktorého ideme extrahovať odkazy a výnimky odkazov, ktoré majú byť preskakované.

```
public List<String> dajLinky(String retazec,
    Set<String> vynimky) {
```

Budeme pristupovať na webové stránky (aby sme zistili, či naozaj ide o odkaz), tak si nastavíme cookies.

```
CookieHandler.setDefault(
       new CookieManager(
              null,
              CookiePolicy.ACCEPT_ALL));
```

Nainicializujeme si výstupnú množinu odkazov, zo vstupu odstránime zlomy riadkov a rozdelíme reťazec podľa medzier. Ideme zistiť odkazy, ktoré sa nachádzajú v texte (napr. google.com).

```
Set<String> linky = new HashSet<>();
String[] casti = retazec
       .replaceAll("=\n", "")
       .replaceAll("\\\\n", "\n")
       .split("\\s+");
```

Prechádzame reťazce a skúšame sa dostať na odkaz.

```
for (String cast : casti) {
```

```
    try {
        preverLinkAPridaj(cast, linky, vynimky);
    } catch (Exception e) {
        // len idem dalej
    }
}
```

Prejdeme aj odkazy, ktoré sú v úvodzovkách, zátvorkách atď.

```
// zdroj: http://blog.houen.net/java-get-url-from-string/
String regex = "\\(?\\b(http://|www[.]|"
        + "https://)[-A-Za-z0-9+&amp;@#/%?=~_()|"
        + "!:,.;]*[-A-Za-z0-9+&amp;@#/%=~_()|]";
Pattern p = Pattern.compile(regex);
Matcher m = p.matcher(retazec);
while (m.find()) {
    String urlStr = m.group();
    if (urlStr.startsWith("(")
            && urlStr.endsWith(")")) {
        urlStr = urlStr
                .substring(1, urlStr.length() - 1);
    }
    try {
        preverLinkAPridaj(
                urlStr,
                linky,
                vynimky);
    } catch (Exception e) {
        // len idem dalej
    }
}
```

Všetky nájdené odkazy, ktoré sú funkčné, vraciame.

```
return new ArrayList<>(linky);
}
```

Nájdenie položiek na vyplnenie v zdrojovom kóde

Na vstup príde zdrojový kód webovej stránky.

```
public List<Input> inputyZoZdrojovehoKodu(
    String zdrojovyKod) {
List<Input> inputy = new ArrayList<>();
```

Zdrojový kód si rozdelíme podľa reťazca "<input ", lebo chceme dostať postupne jednotlivé "input" položky s časťami medzi nimi.

```
String[] casti = zdrojovyKod.split("<input ");
```

Ak tam "input" položka nie je, vraciame prázdny zoznam.

```
if (casti.length == 1) {
    return Collections.emptyList();
}
```

Ideme prejsť časti, ktoré sme získali v predchádzajúcom kroku.

```
for (int i = 1; i < casti.length; i++) {
    Input novy = new Input();
```

Zistíme, kde končí "input" tag.

```
    int indexZlomu = casti[i].indexOf(">");
```

Časť medzi jednotlivými "input"-mi si ukladám. Je to jeden z parametrov "input"-u.

```
    String inputovaCast = casti[i]
            .substring(0, indexZlomu);
```

Pomocou regulárnych výrazov budeme hľadať v časti vlastností "input"-u ich parametre. Zadefinovali sme si teda dva regulárne výrazy - na kľúče a na ich hodnoty.

```
    Matcher berieHodnoty
            = Pattern
            .compile("\\\"([^\"]+)\\\"|\\\"\\\"")
            .matcher(inputovaCast);
    Matcher berieKluce
            = Pattern
```

```
            .compile("([^=\"|^ |=|\"]+)\\=\"")
            .matcher(new String(inputovaCast));
```

Postupne berieme kľúče a hodnoty a ukladáme si k vlastnostiam "input"-u.

```
    while (berieHodnoty.find()
            && berieKluce.find()) {
        String kluc
                = berieKluce.group(1);
        String hodnota
                = berieHodnoty.group(1);
        novy.getVlastnosti()
                .put(kluc, hodnota);
    }

    casti[i] = casti[i]
            .substring(
                    indexZlomu,
                    casti[i].length());
    novy.setPrisluchajuciString(
            casti[i - 1]);
```

Ukladáme nový "input" a po prejdení všetkých častí, vrátime všetky nájdené "input"-y.

```
    inputy.add(novy);
}

return inputy;
}
```

Vytváranie post požiadavky a interpretovanie položiek na vyplnenie

Na vstup prídu "input"-y a odkaz, kde budeme generovať POST požiadavku.

```
public List<Input> posliPostPoziadavku(
        List<Input> inputy,
```

```
        String link)
        throws NepodarenaInterpretaciaPolozkyException {
```

Vytvoríme pole, kde si zaznačíme, ktoré položky už máme interpretované. Tieto položky je dôležité interpretovať.

```
    boolean[] suInterpretovaneLoginServerHeslo
            = {false, false, false};


    for (Input input : inputy) {
```

Ak ešte položka nie je interpretovaná...

```
        if (input.getInterpretacia() == null) {
```

...zistíme, či obsahuje typ a je skrytý alebo nemá "name" položku.

```
            if ((input.getVlastnosti()
                    .containsKey("type")
                    && input.getVlastnosti()
                    .get("type")
                    .trim().equals("hidden"))
                    || !input.getVlastnosti()
                    .containsKey("name")) {
```

Ak je niektorá podmienka splnená, tak položka nebude interpretovaná - nie je ju potrebné interpretovať.

```
                input.setInterpretacia(
                        Polozka.NONE);
                continue;
            }
```

Ak má položka typ "password"/"email", tak určite pôjde o heslo/e-mail.

```
            if (input.getVlastnosti()
                    .containsKey("type")
                    && input.getVlastnosti()
                    .get("type").trim()
                    .equals("password")) {
                input.setInterpretacia(
```

```
                Polozka.HESLO);
            suInterpretovaneLoginServerHeslo[2]
                = true;
            continue;
        }

        if (input.getVlastnosti()
            .containsKey("type")
            && input.getVlastnosti()
            .get("type").trim()
            .equals("email")) {
            input.setInterpretacia(
                Polozka.EMAIL);
            suInterpretovaneLoginServerHeslo[0]
                = true;
            suInterpretovaneLoginServerHeslo[1]
                = true;
            continue;
        }
```

Skúsime vyhľadať typ položky podľa známeho mena.

```
        Integer interpretacia
            = zistiInterpretaciuZName(
                input.getVlastnosti()
                .get("name").toLowerCase());
```

Ak sme interpretáciu neuhádli, tak ju ideme zistiť z id.

```
        if (interpretacia == null
            && input.getVlastnosti()
            .containsKey("id")) {
            interpretacia
                = zistiInterpretaciuZName(
                    input.getVlastnosti()
                    .get("id").toLowerCase());
        }
```

Ak sme interpretáciu neuhádli, tak ju ideme zistiť z reťazca pred "input"-om.

```
            if (interpretacia == null) {
                interpretacia
                        = zistiInterpretaciuZRetazca(
                                input
                                .getPrisluchajuciString());
            }
```

Nastavíme túto položku zistenou interpretáciou.

```
            input.setInterpretacia(interpretacia);
            if (interpretacia != null) {
                switch (interpretacia) {
                    case Polozka.SERVER:
                        suInterpretovaneLoginServerHeslo[1]
                                = true;
                        break;
                    case Polozka.LOGIN:
                        suInterpretovaneLoginServerHeslo[0]
                                = true;
                        break;
                    case Polozka.HESLO:
                        suInterpretovaneLoginServerHeslo[2]
                                = true;
                        break;
                    case Polozka.EMAIL:
                        suInterpretovaneLoginServerHeslo[0]
                                = true;
                        suInterpretovaneLoginServerHeslo[1]
                                = true;
                        break;
                    default:
                        break;
                }
            }
        }
    }
```

Ideme zistiť, či sa podarilo všetky kľúčové položky interpretovať.

```
if (suInterpretovaneLoginServerHeslo[0]
        && suInterpretovaneLoginServerHeslo[2]) {
```

Ak je interpretovaný "login", "heslo":

```
    if (suInterpretovaneLoginServerHeslo[1]) {
```

a) a "server" - tak zistíme údaje u honeyuserovi a vytvoríme POST požiadavku a pošleme ju na odkaz,

```
        // a) server - tak zistim udaje u honeyuserovi
        String[] honeyuserUdaje
                = pripravMenoServerAHeslo();
        try {
            pripravPostAPosli(
                    inputy,
                    honeyuserUdaje,
                    link);
        } catch (IOException ex) {
            System.err
                    .println(
                            "Nepodarilo"
                            + " sa poslat"
                            + " post"
                            + " poziadavku.");
        }
    } else {
```

b) a "server" nie je vyplnený, tak všetky "login"-y zmeníme za kompletný e-mail, zistíme údaje u honeyuserovi a vytvoríme POST požiadavku a pošleme ju na odkaz.

```
        for (Input input : inputy) {
            if (input.getInterpretacia()
                    != null
                    && input.getInterpretacia()
                    == Polozka.LOGIN) {
                input.setInterpretacia(
                        Polozka.EMAIL);
```

```
            }
        }
        String[] honeyuserUdaje
                = pripravMenoServerAHeslo();
        try {
            pripravPostAPosli(
                    inputy,
                    honeyuserUdaje,
                    link);
        } catch (IOException ex) {
            System.err.println(
                    "Nepodarilo "
                    + "sa poslat "
                    + "post "
                    + "poziadavku.");
        }
    }
} else {
```

Ak sa nepodarilo niektorú kľúčovú položku interpretovať a nevieme poslať POST požiadavku, tak končíme vyhodením výnimky.

```
    throw new NepodarenaInterpretaciaPolozkyException(
            inputy);
}
```

Inak vraciame kvôli testom upravené "input"-y.

```
    return inputy;
}
```