VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

ANALÝZA SÍŤOVÝCH BEZPEČNOSTNÍCH HLÁŠENÍ

ANALYSIS OF NETWORK SECURITY ALERTS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — ERIK DOBEŠ
AUTHOR

VEDOUCÍ PRÁCE — JAN WRONA, Ing.
SUPERVISOR

BRNO 2019

Ústav počítačových systémů (UPSY) Akademický rok 2018/2019

Zadání bakalářské práce

21766

Student: **Dobeš Erik**
Program: Informační technologie
Název: **Analýza síťových bezpečnostních hlášení**
Analysis of Network Security Alerts
Kategorie: Počítačové sítě
Zadání:

1. Nastudujte problematiku síťových bezpečnostních hlášení. Zaměřte se na systém Warden.
2. Analyzujte vhodné nástroje pro zpracování velkého množství bezpečnostních hlášení.
3. Navrhněte metodu, která dokáže v datech odhalit skupinu entit (např. IP adres), které vykazují opakovanou časovou korelaci.
4. Navržené řešení implementujte.
5. Funkčnost implementace ověřte na datech ze systému Warden.
6. Zhodnoťte dosažené výsledky a navrhněte možná rozšíření.

Literatura:

- Dle pokynů vedoucího práce.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Wrona Jan, Ing.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 26. října 2018

Abstrakt

Cílem této práce je nalézt v síťových bezpečnostních hlášeních skupiny IP adres, které byly detekovány ve stejných či velmi podobných, časových úsecích. Práce představuje algoritmus, jenž data z bezpečnostních hlášení převede do časových řad. Mezi jednotlivými časovými řadami se následně vyhledávají podobné dvojice vektorů. Poté, uspěje-li algoritmus při hledání dvojic, se v nalezených dvojicích hledají podobné trojice, v nich pak podobné čtveřice atd. Vytvořené řešení úspěšně nalezlo v množině analyzovaných dat 208 podobných skupin, přičemž největší z nich obsahují 11 podobných IP adres. Na základě zjištěných údajů je možné v síťových bezpečnostních hlášeních odhalit stroje, které jsou součástí tzv. botnetu.

Abstract

The goal of this work is to find groups of IP addresses in network security reports, which were detected in the same, or very similar, time interval. The work introduces an algorithm, which transforms data from security reports into time series. Between all the time series, similar pairs are searched. Subsequently, in the found pairs, we are looking for similar threesomes, in which we try to find similar foursomes, etc. The created solution successfully found 208 similar groups in the set of analyzed data, the largest of which contains 11 similar IP addresses. Based on the data found it is possible to detect machines that are part of the so-called botnet in network security reports.

Klíčová slova

analýza, sítě, bezpečnostní hlášení, korelace, vzdálenost vektorů, Hammingova vzdálenost, časové řady, Warden

Keywords

analysis, network, security report, correlation, vector distance, Hamming distance, time series, Warden

Citace

DOBEŠ, Erik. *Analýza síťových bezpečnostních hlášení.* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Jan Wrona, Ing.

Analýza síťových bezpečnostních hlášení

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jana Wrony. Další informace mi poskytl Ing. Martin Žádník, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

. . . . . .
Erik Dobeš
14. května 2019

Poděkování

Touto cestou bych rád poděkoval mému vedoucímu práce, kterým byl Ing. Jan Wrona, za odborné a velmi nápomocné vedení při tvorbě této práce.

Obsah

Kapitola 1

Úvod

Internet, jakožto celosvětově rozšířený systém propojených počítačových sítí („síť sítí“) [2], si po celou dobu své existence udržuje jednu základní vlastnost. Touto vlastností je, že se neustále rozšiřuje. Rozšiřování se týká jak počtu připojených zařízení, tak i počtu lidí, kteří Internet používají. Podíváme-li se např. na počet uživatelů Internetu v roce 2008 a 2018, tedy desetiletý vývoj, zjistíme, že nárůst činí neuvěřitelných 2,103 miliard [3].

Rozmach Internetu s sebou však naneštěstí nese i negativa. Ta, která jsou pro tuto práci relevantní, mohou být shrnuta pod pojmem kyberkriminalita. Pojem kyberkriminalita je odvozován od pojmu kybernetický prostor, případně zkráceně kyberprostor. Kyberprostorem se rozumí virtuální prostředí, které nemá začátek ani konec, nezná hranice národních států a nelze určit jeho velikost. Kybernetická kriminalita, dříve také označována jako informační kriminalita, je Policií ČR uváděna jako trestná činnost, která je páchána v prostředí informačních a komunikačních technologií včetně počítačových sítí. Samotná oblast informačních a komunikačních technologií je buď předmětem útoku, nebo je páchána trestná činnost za výrazného využití informačních a komunikačních technologií jakožto významného prostředku k jejímu páchání [21]. Množství kyberkriminální aktivity roste společně s Internetem a dosavadní tendence ukazují, že tomu tak bude i nadále. Proto je nutné tyto aktivity nejen dokázat detekovat a uložit, ale také analyzovat.

V České republice je jednou z organizací, která se zabývá sběrem výše popsaných dat, sdružení vysokých škol a Akademie věd České republiky – CESNET [8]. CESNET dal vzniknout systému Warden [9], který představuje efektivní způsob sdílení informací o detekovaných událostech (hrozbách). Existence systému Warden a fakt, že počet kyberkriminálních aktivit neustále vzrůstá, byli motivací pro vznik této práce, jelikož Warden sám detekované hrozby neanalyzuje, pouze je umožňuje sdílet.

Způsobů, jak analyzovat detekované hrozby, je mnoho. Zde se zaměřím na analýzu v čase. To znamená, že budu hledat hrozby, které vykazují opakovanou časovou korelaci. Pomocí takové analýzy mohu ku příkladu nalézt v datech tzv. botnet. Botnet je síť napadených počítačů, které mohou být na dálku člověkem zneužity k provedení škodlivých aktivit. Těmito aktivitami mohou být např. DDoS útoky, rozesílání spamu apod. [13], a právě tyto činnosti jsou hlášeny do systému Warden.

Ve 2. kapitole bude popsána metoda, která se zabývá ziskem dat pomocí analýz, a matematický teoretický základ potřebný k její realizaci ve spojitosti s touto prací. Nicméně jelikož cílem práce není popis a definování matematického pojmosloví, tak se jedná pouze o uvedení teoretického základu pro účely cíle práce. Poté v kapitole 3 budou popsána vstupní data této práce, načež kapitola 4 doplní hlavní detektory, kterými jsou daná data získávána. Následně v kapitole 5 bude uvedeno, čeho se analyzovaná data týkají a příklady

aktivit, jenž zapříčiňují jejich vznik. Dále bude následovat praktická část, počínaje kapitolou 6, ve které bude názorně ukázáno, jak probíhá analýza dat. Nakonec je praktická část i zakončena kapitolou 7 o experimentech.

Kapitola 2

Data mining

Množství dat, ať statistických či z bezpečnostních hlášení, dosahuje v dnešní digitální době enormních rozměrů a stejně tak i rychlost, jakou jsou generována nová data. Při zkoumání těchto dat nemusí být na první pohled patrné skutečnosti z nich vyplývající, které však mohou být velmi důležité. Zároveň přestává být, či spíše již není, časově únosné, aby všechna data byla zkoumána lidmi samotnými.

Data mining je jedním z oborů, který se těmito problémy zabývá. Data mining lze přeložit jako dolování z dat, což velmi výstižně popisuje cíl oboru. Data mining si klade za cíl pomocí různých analýz nalézt v datech netriviální, skryté a potenciálně užitečné informace [1]. Z pohledu statistiky se pak jedná o hledání korelací, což znamená určení vzájemných vztahů či vzorů v datech. Smyslem je tedy pomocí různých analýz nalézt nějaký vztah mezi daty, který mezi nimi může, avšak také nemusí být.

Okruhů, které data mining využívají pro dolovaní informací ze svých dat, je velké množství, což z data miningu dělá velmi obecný obor. Z tohoto důvodu nelze konkrétně definovat, jakými metodami analyzovat data a ani jaké všechny informace jsou v datech podstatné. Pro každý okruh je nutné odpovědi na tyto otázky hledat individuálně.

Data, která jsou v této práci použita, a vymezení jejich podstatných částí jsou popsána v kapitole 3. Při hledání vhodné metody, kterou bych ona data analyzoval, jsem zjistil, že volně dostupné práce, které se zabývaly obdobnou analýzou dříve, téměř nelze nalézt. Přesto jsem nalezl inspiraci v práci [19]. Tato práce se zaměřuje pouze na hledání podobných dvojic IP adres, nicméně mi poskytla představu o tom, jak hledat podobnosti v bezpečnostních hlášeních.

2.1 Metrický prostor

Ke správnému výběru metody analýzy dat je na začátku nutné určit prostor, nad kterým jsou data definována. Obecně je tento prostor nazýván metrický prostor a je definován v definici 1. Příkladem metrického prostoru je například *eukleidovský prostor* či *diskrétní metrický prostor*.

Přínosem výběru metrického prostoru je, že nad každým metrickým prostorem lze zavést tzv. metriku, jejíž definice je popsána v definici 2, díky níž lze vypočítat vzdálenost mezi objekty, které jsou definovány v daném metrickém prostoru. Pomocí vzdálenosti pak již můžeme určit podobnost, jelikož tyto dvě hodnoty jsou k sobě duální. To znamená, že čím více jsou si dva objekty podobné, tím menší je vzdálenost mezi nimi [15]. Určování vzdálenosti je tedy důležitým krokem data miningu této práce.

Definice 1. *Nechť $\mathbb{R}$ označuje množinu reálných čísel a $\mathbb{R}_+$ množinu nezáporných reálných čísel.*
Metrickým prostorem nazýváme dvojici (P, ρ), kde P je libovolná neprázdná množina a zobrazení $\rho : P \times P \longrightarrow \mathbb{R}_+$ splňuje pro každé $x, y, z \in P$ následující tři axiomy:

(M1) $\rho(x, y) = 0$ *právě když* $x = y$ *(axiom totožnosti)*

(M2) $\rho(x, y) = \rho(y, x)$ *(axiom symetrie)*

(M3) $\rho(x, y) + \rho(y, z) \geq \rho(x, z)$ *(trojúhelníková nerovnost)*

Zobrazení ρ nazýváme metrikou na P, prvky množiny P obvykle nazýváme body prostoru (P, ρ), číslo $\rho(x, y)$ nazýváme vzdáleností bodů x, y v prostoru (P, ρ). [12]

Definice 2. *Metrikou D na χ nazýváme takovou funkci D: $\chi \times \chi \longrightarrow R$, kde χ je n-rozměrný prostor a R je množina reálných čísel, splňující následující předpoklady:*

- $\exists D_0 \in R : -\infty < D_0 \leq D(x, y) < +\infty, \forall x, y \in \chi$
- $D(x, x) = D_0, \forall x \in \chi$

a má následující vlastnosti:

- $D(x, y) = D(y, x), \forall x, y \in \chi$ *(symetrie)*
- $D(x, y) = D_0$ *když a jen když* $x = y$ *(totožnost)* [15]

2.2 Eukleidovský prostor

Eukleidovský prostor je metrický prostor, který se nejvíce přibližuje představě člověka o světě kolem něj. Formálně je definován v definici 3.

V eukleidovském prostoru je definováno mnoho metrik pro výpočet vzdálenosti. Vzdálenost jako geometrický pojem zavedl v roce 1932 George D. Birkhoff. Je definovaná mezi body A a B, zapisujeme $|AB|$, a je vyjádřena nezáporným reálným číslem [18]. Vzdálenost lze měřit i mezi vektory. Přesný postup záleží na zvolené metrice, kterou je např. *Euklidovská*, *Manhattanská*, *Čebyševova* atd.

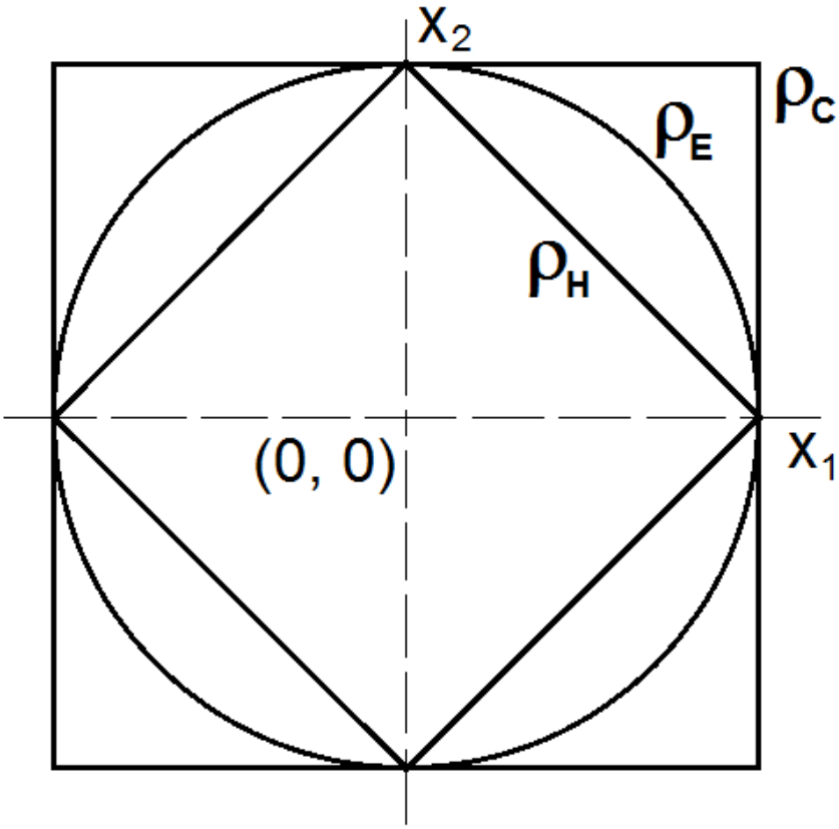
Pro tuto práci jsou zásadní 2 z nich – *Euklidova metrika* a *Hammingova metrika*. Následné zasazení analyzovaných dat do eukleidovského prostoru a výběr metriky budou ukázány v kapitole 6.

Definice 3. *Eukleidovský prostor E_n je afinní prostor (X, V) dimenze n, přitom V je lineární prostor se skalárním součinem. Z tohoto součinu je odvozena norma a metrika V. Metrika na X je definována takto: vzdálenost bodů P, Q je rovna velikosti vektoru $P - Q$.* [20]

2.2.1 Euklidova metrika

Informace v této sekci jsou převzaty ze zdroje [15]. Euklidova metrika je vyjádřena vztahem v rovnici 2.1. Jedná se o metriku s zřejmě nejnázornější geometrickou interpretací. V této metrice je objektem, který má ve dvourozměrném prostoru vzdálenost všech svých bodů od jednoho bodu (středu) stejnou, kruh, ve třírozměrném prostoru se jedná o kouli. V jiných metrikách tento pro člověka přirozený fakt nemusí platit (viz obrázek 2.1).

$$D_E(X_1, X_2) = \sqrt{\sum_{i=1}^{n} (X_{1i} - X_{2i})^2} \tag{2.1}$$



Obrázek 2.1: Geometrická místa bodů se stejnou vzdáleností od souřadnicového počátku ve dvourozměrném metrickém prostoru: ρ_E – Euklidova metrika, ρ_C – Čebyševova metrika, ρ_H – Hammingova metrika [15]

Euklidova metrika klade výraznější důraz na větší rozdíly mezi souřadnicemi než v lineárním případě. Tento fakt vyplývá ze druhé mocniny rozdílů souřadnic. Existuje také upravená verze Eukleidovské metriky, tzv. *kvadratická Euklidova metrika*, u které se na rozdíl od vztahu v rovnici 2.1 výsledná suma nepodrobuje odmocnění. Jedná se bezpochybně o méně náročný výpočet, nicméně zde není dodržena trojúhelníková nerovnost. Na výsledky lze nahlížet jako na míry nepodobnosti, použitý výpočetní vztah však nesplňuje podmínky pravé metriky. Proto je možné *kvadratickou Eukleidovskou vzdálenost* aplikovat jen v případech, kdy je hlavní relativní porovnávání dvou hodnot.

2.2.2 Hammingova metrika

Informace v této sekci jsou převzaty ze zdroje [15]. Hammingova metrika, která je též v některých jiných literaturách nazývána *manhattanská metrika* popřípadě v angličtině *city-block* metrika čí *taxi driver* metrika, svým postupem výpočtu vzbuzuje představu vzdálenosti, jež automobil ujede z místa A do místa B v městských ulicích, které jsou navzájem propojeny pravoúhlými zatáčkami.

Po formální stránce je Hammingova metrika definována vztahem v rovnici 2.2, který vychází z Eukleidovské metriky, přičemž změna, kterou je nutné provést, se nazývá linearizace. Linearizace zde způsobuje, že rozdíly mezi dílčími souřadnicemi obou vektorů nabývající

vyšších hodnot jsou méně významné, než-li v případě Eukleidovské metriky, a také snížení výpočetní náročnosti oproti Eukleidovské metrice. Linearizace také vnesla do definičního vztahu absolutní hodnotu, bez které by jednotlivé rozdíly souřadnic mohly vycházet záporné, což je nežádoucí. Pro body, které mají stejnou Hammingovu vzdálenost od počátku v dvourozměrném prostoru je geometrickým místem čtverec, který se nachází uvnitř Eukleidovské kružnice (viz obrázek 2.1).

Hammingovu metriku lze uplatnit i při výpočtu vzdálenosti dvou binárních vektorů, čehož bude v této práci využito (viz kapitola 6).

$$D_H(X_1, X_2) = \sum_{i=1}^{n} |X_{1i} - X_{2i}| \tag{2.2}$$

Kapitola 3

Analyzovaná data

Informace v úvodní části této kapitoly a v sekci 3.1 jsou přebrány ze zdrojů [9, 10]. V současnosti existuje značné množství CERT/CSIRT[1] týmů, které se intenzivně zabývají provozem a vývojem užitečných nástrojů a systémů v oblasti monitorování síťového provozu, služeb a odhalování jejich anomálií. Jejich existence se stala nezbytností, jelikož počty kybernetických útoků neustále vzrůstají. Základ, na kterém jsou tyto nástroje ve většině případů založeny, tvoří systémy NetFlow, IDS, honeypoty, dále systémy, které analyzují logy atd. Jejich výstupy jsou poté CERT/CSIRT týmy používány k zajištění lepší bezpečnosti pro svou síť či další potřeby. Postupem času se ukázalo, že by bylo velmi výhodné, aby data a informace z těchto výstupů mohla být využita i dalšími bezpečnostními týmy působícími v jiných sítích. Přínosem takovéhoto sdílení může být například řešení situace, kdy jeden bezpečnostní tým detekuje na své síti nový druh hrozby, proti které ještě nebyla vytvořena účinná obranná strategie. Díky sdílení detekovaného průběhu hrozby mají ostatní bezpečnostní týmy možnost se na novou hrozbu připravit, jelikož se dá předpokládat, že nebude zaměřena jen na jednu konkrétní síť.

Bohužel do roku 2011, alespoň v České republice, nebylo možné ve větším měřítku takto informace sdílet. Bezpečnostní týmy tak byly postaveny před nelehké rozhodnutí, jelikož měly v zásadě tři možnosti, jak s daty naložit:

1. Sdílet data se správci, případně bezpečnostními týmy sítí, kterých se dotýkají.
2. „Zahodit“ data.
3. Poslat data do velkých sběrných míst (Shadowserver, Mynetwatchman, Team Cymru).

První možnost s sebou nese vysokou míru náročnosti a také může vytvořit tolik další režie, že ji daný tým nebude schopen s přijatelnými náklady zvládnout. Druhá cesta – zahození dat – pak přímo vede k neefektivnímu plýtvání užitečnými daty a v horším případě může vyústit i k rozvinutí bezpečnostní hrozby, jejíž počátky mohly být detekovány právě z dat, která byla zahozena. Poslední možnost s sebou přináší některé důležité otázky, jako je například ta, zdali existuje záruka, že data budou vhodně využita a nebudou zneužita. Nalezení odpovědi nemusí být vůbec snadné, a proto se jedná spíše o nouzové řešení.

Tato nepříznivá situace se změnila v roce 2011, kdy vznikl systém Warden.

[1] CERT/CSIR – Computer Emergency Response Team/Computer Security Incident Response Team

3.1 Warden

Ve vysokorychlostní síti CESNET2 existovaly dva akreditované CSIRT týmy, jenž iniciovaly vznik systému WARDEN. Jendalo se o tým CESNET-CERTS[2] poohlížející se po dalších zdrojích informací o bezpečnostních událostech, které by se týkaly sítě CESNET2. Dále zde byl tým CSIRT-MU[3], který disponoval událostmi z detekčních nástrojů provozovaných v síti Masarykovy univerzity, avšak pro data, která se přímo nevztahovala k síti Masarykovy univerzity, nenalézal smysluplné využití.

Hlavní myšlenkou systému Warden je zprostředkovat médium pro snadné a efektivní sdílení a využití informací CERT/SCIRT týmu, či dalším angažovaným bezpečnostním týmům. Tyto informace se týkají detekovaných anomáliích, jenž ostatní monitorované sítě odhalily svými nástroji, a týmy z nich mohou vytěžit důležité informace k zaopatření bezpečnosti sítě.

Systém Warden umožňuje jednoduše a efektivně týmu CERTS/CSIRT a dalším zapojeným bezpečnostním týmům rychlé sdílení a využití informací o detekovaných anomáliích, které byly zjištěny nasazenými nástroji v jimi monitorovaných sítích. Tato data jsou systémem předávána a poskytují týmům další užitečné informace potřebné pro zajištění bezpečnosti a monitoringu zdraví sítě. V současnosti je systému ve vývoji v rámci projektu Velká Infrastruktura CESNET, který spadá do činnosti sdružení CESNET bezpečnostního týmu CESNET-CERTS.

Ke dni 3. 4. 2019 činila suma všech událostí sdílených v systému Warden 50 915 235 událostí, které dohromady zabírají 73,32 GB a které jsou sdíleny mezi 26 klienty. Z toho lze vyvodit, že se jedná již o skutečně rozsáhlý systém, ve kterém půjde s vysokou pravděpodobností nalézt záznamy s podobnými časovými značkami, což byl jeden z důvodů, proč byl Warden pro tuto práci zvolen jako zdroj dat.

3.2 Mentat

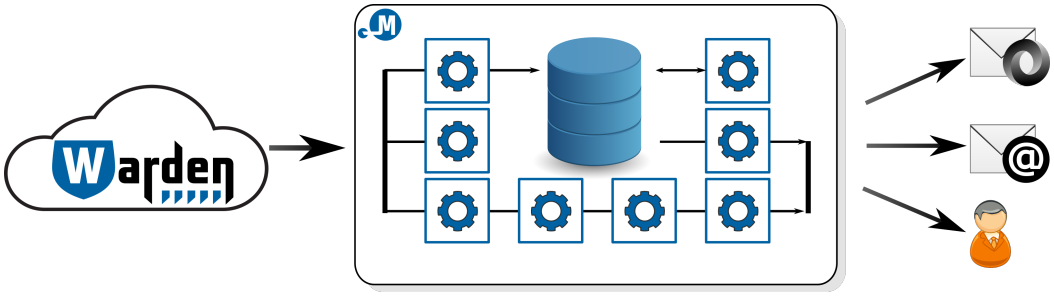
Informace v této sekci jsou přebrány ze zdroje [7]. Sdružení CESNET, které provozuje vysokorychlostní páteřní síť CESNET2, vyvinulo systém Mentat za účelem jednotného sběru událostí z různých heterogenních zdrojů. Události mohou být posléze kompaktně zpracovány či vyhodnocovány a lze v nich vyhledávat. Jedním z hlavních zdrojů těchto událostí je systém Warden. Systém Warden ukládá data do systému Mentat ve formátu IDEA (viz sekce 3.3). Tato skutečnost je grafiky znázorněná na obrázku 3.1.

Při vytváření systému Mentat byl zvolen modulární přístup, díky kterému může být systém rozdělen do menších logických celků (modulů). Každý z modulů zde vykonává „jednoduchý“ úkon, může fungovat paralelně s ostatními a lze jej jednoduše upravit, vylepšit či změnit. Jako implementační programovací jazyk je v celém systému použit jazyk Python3 a pro perzistentní ukládání dat autoři zvolili databázi PostgreSQL. Data jsou zde přenášena ve formátu IDEA.

3.3 Formát IDEA

Informace v této sekci jsou přebrány ze zdroje [6]. Anglický význam zkratky IDEA je *Intrusion Detection Extensible Alert*, což lze volně přeložit jako rozšiřitelná výstraha pro de-

[2]CESNET-CERTS – Bezpečnostní tým pro síť CESNET2
[3]CSIRT-MU – Bezpečnostní tým pro síť Masarykovy univerzity v Brně



Obrázek 3.1: Přehled systému Mentat [7]

tekci narušení. Jedná se o formát pro přenos dat, který je výsledkem snažení o vytvoření univerzálního komunikačního modelu, např. mezi honeypoty, který bude lehce rozšiřitelný s přihlédnutím k existujícím formátům, jejich výhodám a nevýhodám. Důvodem této snahy byla skutečnost, že existovalo mnoho administrátorů, kteří pro přenos a sdílení dat z bezpečnostních událostí používali různé formáty, což značně stěžovalo jejich sdílení. Formát IDEA se tyto problémy snaží řešit. Navíc jednotný formát pro sdílení dat umožňuje analýzu, při které je na vícero zdrojů nahlíženo jako na jeden, což může přinést další zajímavé poznatky.

Příklad formátu IDEA serializovaný ve formátu JSON je uveden v příloze A. Pro účely této práce je zde nutné zaměřit se na strukturu `DetectTime`, která nám poskytne čas detekce události, a dále na uzel `Source` a `Node`. V uzlu `Source` jsou zaznamenány zdrojové IPv4 a IPv6 adresy útočníka, přičemž je potřeba dbát na skutečnost, že IP adres zde může být více (např. když je detektorem zachycen útok botnetu včetně samotného příkazu od C&C serveru). V uzlu `Node` se nachází popis detektorů, které dané hlášení vytvořily, a také jich zde může být více.

Kapitola 4

Detektory systému Warden

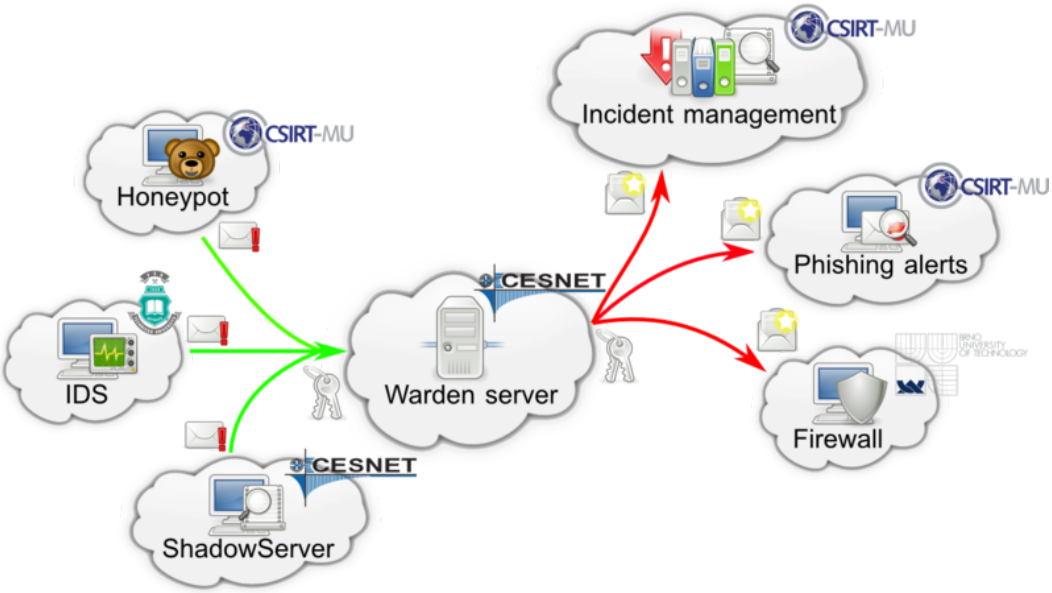
Do systému Warden je dnes zapojeno mnoho organizací, které přes něj sdílejí své bezpečnostní události. Tyto organizace mají interně zavedeny bezpečnostní systémy, které mohou detekovat bezpečnostní události různými způsoby. Mezi 2 nejdůležitější se řadí systémy typu *Honeypot* a *IDS*. Dále je do systému Warden ještě zapojena organizace *Shadowserver*, jak lze vidět na obrázku 4.1. Tato kapitola postupně všechny zmíněné systémy a organizaci představí.

4.1 Honeypot

Informace v této sekci jsou převzaty ze zdroje [24, str. 58 – 67]. Honeypot slouží jako označení bezpečnostním informačním systémům, jejichž účelem je, aby na síti byly předmětem skenování nebo kybernetických útoků, přičemž zdroj těchto aktivit pro honeypot není důležitý. Na čem tedy doopravdy záleží je, aby honeypot byl na síti zjistitelný a mohlo na něj být útočeno. Pokud se tak neděje, tak je jeho hodnota velmi malá či až žádná. Touto vlastností se honeypoty odlišují od většiny informačních systémů, které se ba naopak snaží všem kybernetickým útokům vyhnout.

Odlišnost honeypotů spočívá také v tom, že nejsou limitovány na řešení jediného, specifického problému. Na místo toho se jedná o vysoce flexibilní nástroj, který lze použít v mnoha různých situacích. Například dokáže odradit útočníky, což je cíl, který sdílí s firewallem. Honeypoty také někdy mají podobnou funkcionalitu jako IDS systémy – detekují útoky. Dále je lze využít k analýze automatických útoků, kterými jsou například počítačoví červi, nebo fungují jako nástroj včasné výstrahy před útoky. Dalších podobných příkladů by šlo uvést nespočet. Obecně lze tedy říci, že cíle honeypotů záleží na tom, k čemu byly jejich autory navrženy.

Honeypoty nemají žádnou produkční hodnotu, což znamená, že žádný uživatel nebo zařízení by s nimi nemělo komunikovat. Z tohoto důvodu data získaná honeypoty pochází ze všech interakcí s nimi, jelikož lze předpokládat, že jakákoliv aktivita směřující jejich směrem je přirozeně podezřelá. Množství těchto dat tak většinou není nijak velké, denně se jedná o několik megabytů či ještě méně, jelikož se nejedná o celé síťové toky, jejichž velikost se pohybuje v řádu gigabytů. To je považováno za jednu z velikých výhod honeypotů, jelikož analýza tak malého množství dat zabere menší dobu a stále vede ke kvalitním výsledkům.



Obrázek 4.1: Architektura systému Warden [4]

Honeypoty se dají rozdělit do dvou základních kategorií – *produkční* a *výzkumné* honeypoty. Toto rozdělení nemusí být vždy striktní. Některé honeypoty se svými vlastnostmi řadí do obou kategorií. Opět zde záleží na volbě autora honeypotu, jak jej využije.

4.1.1 Produkční honeypot

Produkční honeypot odpovídá představě slova honeypot u většiny lidí. Přidává na síle bezpečnosti specifické organizaci a pomáhá zmírnit rizika útoků. Produkční honeypoty jsou na rozdíl od těch výzkumných lehčí na implementaci a nasazení do systému, jelikož vyžadují méně funkcionality. Kvůli tomu však generují méně informací o útocích a útočnících. Tudíž se pomocí nich lze dozvědět např. o tom, odkud útoky přicházejí, ale ve většině případů neposkytují informace o potenciální vzájemné komunikaci mezi útočníky nebo o jejich nástrojích.

4.1.2 Výzkumný honeypot

Výzkumný honeypot je navržen tak, aby dokázal poskytovat co nejvíce možných informací o kyberkriminální komunitě. Nepřidává přímou hodnotu specifické organizaci. Jeho účel se zaměřuje na výzkum hrozeb, kterým může organizace čelit. Poskytuje tedy znalosti o původu útočníků, o tom, jak jsou organizovaní a nebo jaké druhy nástrojů k útoku použili a kde je získali. Díky všem těmto podrobným znalostem se mohou organizace lépe bránit proti budoucím útokům.

4.2 IDS

Informace v této sekci jsou převzaty ze zdroje [14]. Intrusion detection system[1] (IDS) je systém, který monitoruje počítačové systémové události za účelem odhalení nebezpečných nebo

[1] Intrusion detection system – systém detekce průniku

podezřelých aktivit v daném systému. Kvůli stále se rozvíjející kybernalitě se IDS staly důležitou a nezbytnou součástí organizací.

Běžné zabezpečovací techniky, jako jsou firewally nebo autentizační mechanismy, kterými jsou například hesla nebo různé šifrovací algoritmy, zajišťují jistou úroveň ochrany, nicméně nedokáží poskytnout obranu proti škodlivým kódům, interním útokům nebo nezabezpečeným modemům. Proto byly vyvinuty systémy detekce průniku, které na rozdíl od firewallů dokáží monitorovat síť zevnitř a v případě narušení bezpečnosti ihned podat zprávu administrátorovi sítě.

Na základě zdroje kontrolovaných dat lze IDS rozdělit do dvou kategorií: Host-based a Network-based IDS. Dále existuje druhé dělení IDS podle analyzovaných událostí, které má také dvě kategorie: Misuse based a Anomaly based IDS.

Host-based IDS (HIDS) detekují útoky, které směřují pouze na jeden systém. Nejčastěji jsou nasazeny v systémech, které jsou více zranitelné vůči útokům (např. webové servery). HIDS zde sbírají data ze systémových volání, operačních logů atd.

Network-based IDS (NIDS) se zaměřují na detekci útoků, jejichž cílem jsou různé systémy v síti. NIDS analyzují síťový provoz s cílem detekovat podezřelé aktivity v síti k prevenci nelegitimních přístupů k síťovým zdrojům.

Misuse based IDS používají různé techniky k vyhledávání shod mezi systémovými aktivitami a již známými kybernetickými útoky, které jsou uloženy v databázi.

Výhody

- Dokáží detekovat útoky bez falešně pozitivních výstrah.
- Slouží jako jednoduchý nástroj k použití, takže rozhodnutí o útocích je důvěryhodné.

Nevýhody

- Misuse based IDS nemohou detekovat nové druhy útoků.
- Systém musí být s každým nově objeveným útokem aktualizován.
- Pokud se objeví modifikovaná verze již známého útoku, tak ji tyto systémy neodhalí.

Anomaly based IDS analyzují chování uživatelů a systémových aktivit. Nejprve si vytvoří profily uživatelů, serverů a všech síťových připojení, které vykazují známé a legitimní aktivity. Poté vygenerují upozornění, která se spustí, pokud se nová data odchýlí od těch, která jsou již uložena v databázi uživatelů a systémových profilů.

Výhody

- Dokáží detekovat nové typy útoků.
- I když neexistuje kompletní informace o útoku, tak Anomaly based IDS jsou schopny daný útok detekovat.
- Dají se využít k vytváření databáze pro Misuse based IDS.

Nevýhody

- Anomaly based IDS generují spoustu falešně pozitivních zpráv, jelikož chování uživatelů a sítí není vždy známo dopředu.
- Tento přístup vyžaduje velké množství počátečních dat pro vymezení prahů, jaké budou sloužit k určení, které aktivity v systému lze považovat za legitimní.

4.3 Shadowserver

Nadace Shadowserver, jež byla založena v roce 2004, je americkou organizací, která se stala jednou ze světově hlavních organizací zabývajících se hlášeními o internetové bezpečnosti a prozkoumáváním škodlivých aktivit na síti [23]. Denně provádí skenování celé IPv4 části internetu. Jedná se o 4 miliardy adres, které jsou navíc za den prozkoumávány 42krát. Všechna takto získána data jsou každý den odesílána nadací ověřeným odběratelům, kteří je mohou využít ke zkvalitnění bezpečnosti jejich sítě [22].

Kapitola 5

Kybernetická kriminalita

Informace v následující části kapitoly jsou přebrány ze zdroje [16]. V předcházejících kapitolách byly často zmiňovány kybernetické útoky a možnosti jejich detekce. Nyní v této kapitole budou tyto útoky teoreticky vymezeny a dále představeny některé konkrétní příklady.

Činnost, jakou je nedodržování zákona v kybernetickém prostoru[1], ale i konání, které zde přesahuje hranice morálního chování, se nazývá kybernetická kriminalita – též kyberkriminalita či kybernalita. Množství těchto činností každoročně stoupá, což je mimo jiné zapříčiněno tím, že pachatelé operují v nehmotném globálním prostředí, které jim poskytuje velkou míru anonymity. Na rozdíl tak od běžné kriminality mohou snáze svou činnost maskovat a mají větší šanci vyhnout se odhalení. Dalším faktorem je jednoduchost, s jakou se lze do kybernetické kriminality zapojit. Například na internetovém serveru youtube je uložena spousta volně přístupných videí, které velmi názorně popisují, jak některou z aktivit spadající do kybernality provést.

Kybernalita s sebou přináší hrozby, které mohou být rozděleny do čtyř základních kategorii:

1. **Únik informace** – jedná se o případ, kdy subjekt bez patřičné autorizace získá přístup k informaci důvěrné povahy a nebo je jím tato informace odhalena. Únik informace může následně vyústit v přímý útok s vážným dopadem.

2. **Narušení integrity** – značí situaci, při které neautorizovaný činitel naruší soudržnost dat, přičemž se může stát, že se vytvoří data nová nebo že se data změní či vymažou.

3. **Potlačení služby** – jde o událost, kdy je se zřejmým úmyslem blokován přístup legitimního subjektu k obsahu, ke kterému by měl za běžných okolností přístup.

4. **Nelegitimní použití** – znamená, že subjekt, aniž by se autorizoval, využívá zdroje s omezeným přístupem a nebo jsou tyto zdroje používány neadekvátním způsobem.

Mimo těchto 4 základních kategorií existují ještě dvě další, které musejí stát mimo základní rozdělení. Jejich vztah k základním kategoriím bude popsán v následujících odstavcích a je také znázorněn na obrázku 5.1. Jedná se o *aktivační hrozby* a *podkladové hrozby*.

Aktivační hrozby dostaly svůj název z toho, že aktivují základní hrozby. To znamená, že při jejich realizaci bezprostředně dochází k vytvoření některé ze základních hrozeb, přičemž jsou samozřejmě také přímo ohroženy bezpečnostní parametry systému. Rozdělení aktivačních hrozeb do skupin je následující:

[1] Kybernetický prostor je virtuální prostor počítačových sítí.

- Penetrační hrozby:
  - **Maškaráda** – jedná se o jeden z nejobvyklejších způsobů, jak lze narušit bezpečnostní perimetr systému za pomocí toho, že se jedna entita vydává za jinou entitu. Útočící entita se snaží „přesvědčit“ systém, že vlastní všechna autorizační práva potřebná k manipulaci s chráněným obsahem, ačkoliv těmito právy ve skutečnosti nedisponuje.
  - **Obejití řízení** – případ, kdy útočník získá neautorizovaná práva či privilegia za pomocí bezpečnostních nebo systémových slabin.
  - **Narušení autorizace** – zdroj je zde za pomoci autorizovaného přístupu zneužíván k neautorizovaným účelům. Velikou odlišností útoku tohoto typu je, že nemůže být veden zvenčí, nýbrž zevnitř uživatelem, který vlastní přístupová práva k danému zdroji.
- Implantační hrozby:
  - **Trojský kůň** – jak samotné jméno napovídá, jedná se o případ, kdy se nějaký software navenek jeví jako uživatelsky užitečný, nicméně obsahuje uživateli skrytou část, která po spuštění naruší bezpečnostní prvky systému. Velmi běžným případem jsou aktualizace čteček PDF souborů, které při instalaci implantují trojského koně do systému nic netušícího uživatele. Implantovaný škodlivý software poté může například odesílat data o aktivitách uživatele na internetovou adresu, která je v kódu předem daná.
  - **Zadní vrátka** – tímto názvem je pojmenován kus systémového kódu, který po obdržení specifického datového řetězce na vstupu umožní obejití bezpečnostních prvků systému. Například v systému může existovat přihlašovací jméno, které po zadání získá plný přístup do systému bez jakýchkoliv dodatečných kontrol.

Implantační hrozby nebývají ve většině případů aktivovány hned po implantaci, nýbrž až po čase, který uzná autor za vhodný.

Podkladové hrozby jsou hrozby, které, jak název napovídá, slouží jako podklad pro uskutečnění až několika základních hrozeb (nebo aktivační hrozby, která aktivuje základní hrozby). Na obrázku 5.1 je znázorněn vztah základních a podkladových hrozeb a v tabulce 5.1 je uveden popis jednotlivých hrozeb.

Hrozba	Popis
Porušení autorizace	Osoba, která je autorizovaná k použití zdroje pro jistý účel, jej použije k jinému, neautorizovanému účelu.
Obejití řízení	Útočník využije bezpečnostních mezer v systému nebo jeho slabin.
Potlačení služby	Omezení legitimního přístupu k informacím nebo jiným zdrojům v síti.
Nezákonný odposlech	Informace je získávána monitorováním přenosového kanálu.
Emisní nebo VF odposlech	Informace je extrahována z vysokofrekvenčního vyzařování nebo emisí či jiných elektromagnetických jevů, ke kterým dochází při provozu elektronického zařízení.

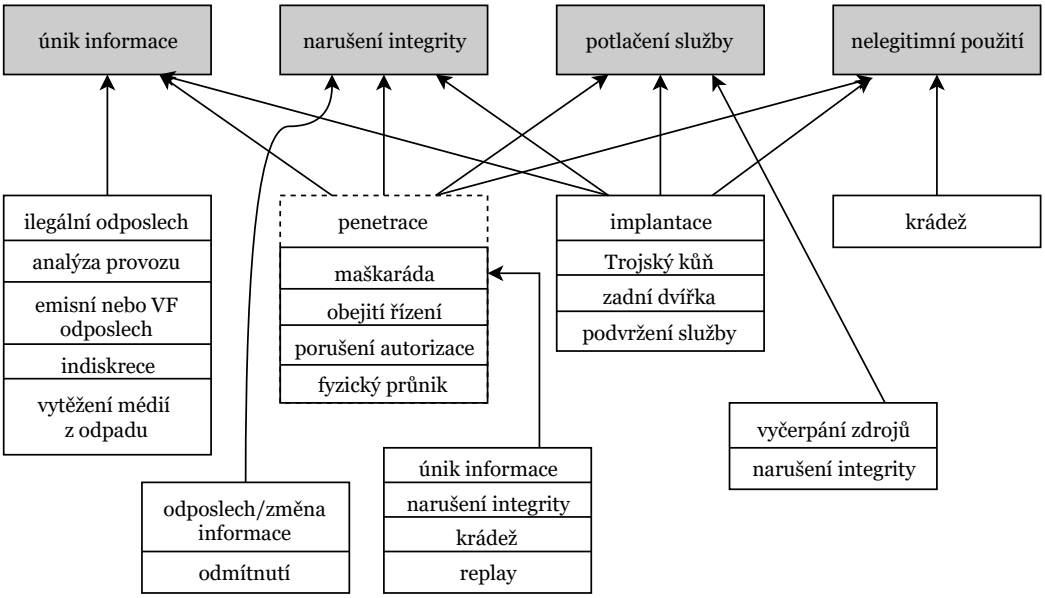
Nelegitimní použití	Zdroj je používán neautorizovanou osobou nebo neautorizovaným způsobem.
Indiskrece	Autorizovaná osoba prozradí důvěrnou informaci neautorizované osobě z neopatrnosti nebo za úplatu.
Únik informace	Získání důvěrné informace neautorizovanou osobou nebo systémem.
Narušení integrity	Konzistence dat je narušena jejich neautorizovaným vytvořením, úpravou nebo vymazáním.
Změna dat při přenosu	Přenášená data jsou během přenosu informačním kanálem změněna, odstraněna nebo zcela vyměněna.
Maškaráda	Jedna entita (osoba nebo systém) se představuje jako jiná entita.
Vytěžení odpadových médií	Informace je získávána z magnetických nebo papírových médií, vyhozených do odpadu.
Fyzický průnik	Útočník získá kontrolu nad systémem proniknutím k jeho ovládacím prvkům.
Replay	Zachycená kopie legitimní transakce je využita pro opětovný přenos s nelegitimním úmyslem.
Popření skutečnosti	Strana účastněná ve vzájemné komunikaci později popře, že k takové komunikaci došlo.
Vyčerpání zdrojů	Jistý zdroj, např. port, je úmyslně natolik zatížen, že je znemožněno používání služby, která je na něj vázána, řádnými uživateli.
Podvržení služby	Podvržený systém nebo systémová komponenta, které se vůči uživateli chovají jako běžná součást systému, slouží k získání citlivých informací od důvěřivého uživatele.
Krádež	Kritický prvek bezpečnostního systému (např. přístupová karta) nebo veškeré citlivé informace jsou zcizeny.
Analýza provozu	Informace je neautorizovanou entitou získána pomocí sledování provozu a výběrem podstatných jeho částí.
Zadní vrátka	Do systému je zabudována vlastnost nebo vložena součást, která při jisté konstelaci vstupních dat umožní obejít bezpečnostní mechanismy.
Trojský kůň	Software obsahuje zdánlivě nevinnou nebo neviditelnou část kódu, který – pakliže je spuštěn – ohrozí bezpečnost uživatele.

Tabulka 5.1: Výpis typických hrozeb [16]

Dále v této kapitole budou detailněji popsány některé příklady kybernetických útoků – informace o nich jsou převzaty ze zdroje [17], pokud není uvedeno jinak.

5.1 Spam

Pod pojmem spam čí spamming je označováno rozesílání nevyžádané elektronické pošty, která bývá nejčastěji reklamního charakteru. Množství rozeslané pošty je z podstaty spamu



Obrázek 5.1: Vztah základních a podkladových hrozeb [16]

obrovské, jelikož spam není cílený, nýbrž jeho cílem je, aby se dostal do elektronických pošt co nejvíce uživatelů.

Účelem spamu dnes nemusí být jen šíření reklamního materiálu, ale také implantace tzv. malware (viz sekce 5.2) do uživatelova zařízení.

5.1.1 Obrana proti spamu

Dnešní elektronické pošty disponují filtry, jenž nevyžádanou poštu automaticky odfiltrovávají do speciální složky, která je většinou pojmenována spam. Problémem je, že lidé rozesílající spam přicházejí se stále novými způsoby, jak tyto filtry obejít, a proto je také nutné dbát na prevenci, kterou je nikde zbytečně nezveřejňovat svou elektronickou adresu.

5.2 Malware

Malware, česky označován jako škodlivý kód, je počítačový program (nebo jeho část), který vzniká za účelem infikování systému. Z infikovaného systému může poté útočník odcizit data, špehovat uživatele nebo samotný systém poškodit.

Malware je velmi obecné označení škodlivého kódu, pod které spadá mj. spyware, adware, ransomware, keylogger nebo také trojští koně, počítačové viry a počítačoví červi.

Spyware

Spyware bez vědomí uživatele v napadeném systému sbírá a odesílá data. Data se mohou týkat informací, které slouží ke zlepšení cílené reklamy, tedy se může ku příkladu jednat o historii webového prohlížeče, ale také může jít o sběr osobních dat o uživateli.

Adware

Adware je druh malware, který v napadeném systému zobrazuje reklamu ve všech existujících formách (např. pop-up okna). Jedná se spíše o obtěžující škodlivý kód, který může využívat informace shromážděné přes spyware, avšak jejich spolupráce není nutná.

Počítačový vir

Počítačový vir nemá schopnost se sám šířit, a proto k tomu, aby pronikl do systému, využívá nejčastěji spustitelné soubory (EXE v OS[2] Windows), různé dokumenty, nebo funguje jako samospustitelná příloha v elektronické poště.

Počítačové viry vznikají za účelem poškodit uživatele a to například smazáním nějakého souboru. Vzhledem k tomu, že dnešní antivirové programy dokáží počítačové viry odhalit a smazat, jsou tyto viry na ústupu.

Počítačový červ

Počítačový červ, anglicky worm, je, co se týče možnosti šíření, propracovanější obdobou počítačového viru. Liší se v tom, že se může šířit sám a to pomocí programových chyb systému, které využívají např. ke změně běhu systému.

Ransomware

Ransomware, pokud se mu podaří proniknout do systému, v napadeném systému zašifruje soubory, tudíž se stanou pro uživatel daného systému nedostupnými. Za odšifrování souborů útočník požaduje zaplacení jistého finančního obnosu (dnes často v kryptoměně). Po zaplacení záleží jen na útočníkovi, zdali soubory skutečně odšifruje, nicméně většinou se tak stane.

5.3 DoS, DDoS

Informace v této sekci jsou převzaty ze zdroje [11]. DoS – Denial of Service a DDoS – Distributed Denial of Service jsou dva téměř totožné typy kybernetických útoků, které mají za cíl vyřadit internetovou službu dostupnou veřejnosti. Rozdílné jsou pouze v tom, že DoS je veden z jediného zdroje, zatímco DDoS má zdroje alespoň dva. Proto dále v této sekci, je-li uvedeno označení DoS, tak je ním myšlen nejen útok DoS, ale také DDoS.

Principiálně DoS využívá skutečnosti, že aby nějaká z internetových služeb mohla být veřejně přístupná, tak musí poskytovat jistou funkcionalitu všem uživatelům bez rozdílu (např. ping). Právě na některou z těchto funkcí (nebo i na všechny) se během útoku DoS zasílá enormní množství požadavků, což službu zahltí a kvůli tomu následně přestane korektně fungovat jako celek.

5.4 Skenování portů

Informace v této sekci jsou převzaty ze zdroje [25]. Skenování portů je metoda, pomocí níž lze zjistit, které porty síťového zařízení jsou otevřené a přijímají nová připojení. Jelikož většina služeb běží na standardním, tzn. zdokumentovaném portu, tak informace

[2] OS – operační systém

získané touto metodou mohou být využity ke zjištění, jaké služby jsou na daném zařízení spuštěny. Některé z metod skenování jsou popsány níže.

Skenování založené na plném a polovičním TCP spojení

Nejjednodušší forma skenování portů zahrnuje pokusy o navázání TCP spojení na všechny porty u zařízení, na které skenování směřuje. I když je tento způsob efektivní, tak je také lehce detekovatelný a zároveň pokud je spojení navázáno, tak služby za normálních okolností ukládají IP adresu zdroje. Aby se těmto skutečnostem útočníci vyhnuli, tak většinou používají důmyslnější verze této metody.

TCP SYN skenování se někdy v angličtině nazývá *half-open* scan, jelikož se zde nenavazuje plné TCP spojení. Celý proces navázaní TCP spojení zahrnuje nejprve odeslání SYN paketu, poté je poslán SYN/ACK paket zpět a nakonec se spojení potvrdí posláním ACK paketu zpět. Nicméně při TCP SYN skenování se nikdy ze strany útočníka neodešle poslední ACK paket, jelikož útočník už podle přijetí SYN/ACK paketu pozná, že je daný port otevřený. Na konci útočník posílá RST paket, kterým navazování spojení ukončí.

Stealth a Inderect skenování

Stealth skenování je na rozdíl od výše zmíněných technik velmi těžce detekovatelné, jelikož v jeho postupu není zahrnuto klasické navazování TCP spojení. Místo toho tato metoda používá FIN segmenty k odhalení otevřených portů, jelikož pokud je zaslán FIN segment směrem k uzavřenému portu, tak server ukončí spojení a odešle zpět RST paket. Naopak když se pošle FIN segment k otevřenému portu, tak server jen uzavře spojení a neodesílá RST paket. Obě varianty jsou pro útočníka detekovatelné a dokáže z nich poznat, jaké porty jsou na serveru otevřené. Stealth skenování existuje více variant, jako jsou např. Xmas Tree a Null skenování.

Při Inderect skenování se využívá (falešná) IP adresa třetího hostitele k zamaskování skutečného skenovacího systému. Jelikož skenovaný systém bude reagovat odesláním nebo neodesláním určitých segmentů falešnému hostovi, tak stačí monitorovat IP aktivitu falešného hosta k rozpoznání výsledků skenování.

Fragmented a Decoy skenování

Fragmented skenování je vylepšená obdoba TCP SYN a Stealth skenování používající malé, fragmentované IP pakety. TCP hlavičky jsou rozděleny do několika paketů, díky čemuž mají větší šanci, že je nezachytí filtry a ani IDS.

Decoy skenování spočívá v anonymizaci skenujícího stroje. K jejímu dosažení využívá „návnady“ – skenování probíhá zároveň z více zdrojů, takže i když jsou detekovány, tak nelze zjistit, které skenování pocházelo od útočníka a které z falešných zdrojů.

UDP skenování

Pomocí UDP skenování se zjišťuje, které UDP porty jsou otevřeny v hostitelském zařízení. Zakládají se na faktu, že pokud dorazí UDP paket k uzavřenému UDP portu, tak UDP spojení reaguje vygenerováním ICMP zprávy „port unreachable“. Tudíž typické UDP skenery odešlou prázdný UDP paket na vybrané porty a poté čekají na eventuální odpovědi. Pokud odpověď nedorazí, tak je port většinou otevřený.

Ident a Proxy skenování

Až doteď byly popsány skenery, které byly navrženy pouze k nalezení otevřených portů. Nicméně existují i skenery, které disponují další funkcionalitou, pomocí níž lze odhalit běžící démony přiřazené k otevřeným portům.

Ident skenování je zajímavým příkladem těchto extra možností. Zneužitím Identifikačního protokolu skenery dokáží zjistit název (Userid) démona běžícího na určitém portu. Dělají to tak, že na začátku ustanoví plné TCP spojení, poté odešlou tzv. ident žádost k identd (Identification Protocol daemon) na TCP portu 113.

FTP protokol podporuje možnost zvanou proxy FTP připojení. Původní myšlenka této volby dle RFC 959 byla umožnit klientovi, aby mohl současně navázat 2 FTP připojení s různými servery a poté převádět data mezi danými servery. Většina implementací může být lehce zneužita k tomu, aby server posílal data kamkoliv na Internetu. Tato slabina je zneužívána při proxy skenování, při kterém je FTP server využit jako proxy server ke skenování TCP portů.

5.5 Botnet

Informace v této sekci jsou převzaty ze zdroje [13]. Botnet je síť napadených počítačů nazývajících se „boti“ pod kontrolou člověka (operátora), kterému se říká „botmaster“. Termín „bot“ je odvozen od slova robot, jelikož podobně jako roboti jsou boti navrženi k automatizovanému výkonu předdefinované funkce. Jinými slovy individuální boti jsou softwarové programy běžící na hostitelském počítači, které umožňují botmasterovi vzdáleně kontrolovat hostitelovy aktivity. Botnety dnes představují významnou a rostoucí hrozbu proti kybernetické bezpečnosti, neboť poskytují distribuovanou platformu pro mnoho kybernetických zločinů (DDoS, šíření malware, phishing, rozesílání spamu atd.). Jelikož např. pro realizaci DDoS útoku musí být aktivováno více botů naráz, tak útoky od jednotlivých botů budou přicházet ve stejných nebo velmi podobných časech, což přesně zapadá do kategorie útoků, které hledá algoritmus představený v této práci, tudíž tento algoritmus disponuje potenciálem botnet nalézt.

Jedna z hlavních hodnot botnetu je schopnost poskytovat anonymitu pomocí vícevrstvé architektury velení a řízení (C&C). Navíc jednotliví boti nejsou fyzicky vlastněni botmasterem, takže se mohou nacházet na různých místech po celém světě. Rozdíly v časových pásmech, jazycích a zákonech ztěžují sledování škodlivých činností botnetu napříč mezinárodními hranicemi. Tato vlastnost činí botnet atraktivním nástrojem pro kybernetické zločince a ve skutečnosti představuje velkou hrozbu proti kybernetické bezpečnosti.

Kapitola 6

Rozpoznání podobnosti vektorů

V tabulce 6.1 můžeme vidět 4 základní podobné dvojčlenné skupiny vektorů, které by mělo jít nelézt za pomoci postupů, které jsou uvedeny níže v této kapitole. Jednotlivé řádky tabulky představují vektory označené písmeny A až H, zatímco sloupce značí body v čase – sloupec s označením „5“ znamená čas vzdálený o 5 sekund od počátečního času analýzy. Znak „X“ v tabulce vyjadřuje skutečnost, že pro danou IP adresu, která je reprezentována vektorem, existuje v daný čas záznam. Jmenovitě se tedy jedná o dvojice AG, BD, CE a FH. Důležitým faktem zde je, že dvojice si nemusí být podobné stoprocentně, jak lze v tabulce vidět, nýbrž musí být zvolená jistá tolerance pro jejich podobnost. Tato tolerance bude nazývána `treshold` a bude určena v experimentální části 7.3.

Dvojice BD a FH jsou si typově podobné. Obě se skládají z více podobných intervalů, avšak skupina FH je složena až na 1 výjimku v rámci tolerance ze stejně dlouhých intervalů, zatímco délky intervalů u skupiny BD nabývají různých velikostí.

Vektory C a E jsou si podobné pouze v jediném táhlém intervalu, což mají společné s podobnou dvojicí AG. Nicméně dvojice AG je speciální v tom, že se vektory, ze kterých se skládá, rozprostírají přes celou délku analyzovaného časového okna. Kvůli podobným skupinám typu skupiny AG je zapotřebí, aby metoda určující podobnost zajistila, že tyto skupiny nebudou označeny jako podobné se všemi ostatními. Zmíněná situace by mohla nastat, jelikož např. vektor A má společné všechny body výskytů se všemi ostatními vektory. Nicméně je třeba nezapomenout, že naopak tato skutečnost neplatí.

K samotnému určení, zdali si je dvojice vektorů podobná, použijeme ze začátku vzájemnou vzdálenost daných vektorů. V získaných dvojicích následně zkusíme nalézt podobné trojice, z těch pak podobné čtveřice atd. (viz sekce 6.4).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
A	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
B			X	X	X	X			X	X	X	X			X	X	X	
C				X	X	X	X	X	X	X	X	X	X					
D		X	X	X	X	X				X	X	X			X	X	X	
E				X	X	X	X		X	X	X	X	X	X				
F			X	X		X	X		X	X		X	X		X	X		
G	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
H	X		X	X		X	X		X	X		X	X		X	X		

Tabulka 6.1: Ukázka podobných vektorů

Abychom mohli použít některou z metod pro výpočet vzdálenosti, musí být množina analyzovaných dat vyjádřena v prostoru, nad kterým je metrika definována (viz sekce 2.1). To v našem případě, kdy jsme z databáze schopni zjistit pouze v jakých časech daná IP adresa vykazovala podezřelé aktivity, není splněno. Nicméně tyto údaje z databáze lze transformovat do časových řad, které již lze použít.

6.1 Časové řady

Časová řada je sekvence datových bodů indexovaných (nebo uvedených v grafu) v časovém pořadí (směr minulost → přítomnost). Nejčastěji jsou časové řady sekvence pořízené za sebou v časově rovnoměrných bodech. Jedná se tedy o sekvence diskrétních dat.

6.1.1 Transformace dat do časové řady

Data získaná ze systému Warden o jednotlivých IP adresách, která se před použitím zformátují do struktur ve tvaru `IP adresa : kolekce časových značek,` lze jednoduše transformovat do časové řady podle algoritmu 1.

Algoritmus 1: Vytvoření časové řady

Vstup: Struktura ve tvaru `IP adresa : kolekce časových značek detekovaných událostí pro danou IP adresu`

Výstup: Časová řada

```
1:  delka_kolekce ← velikost měřeného úseku + 1
2:  kolekce_casu o délce delka_kolekce ← naplnit nulami
3:  for casova_znacka in detekované časové značky ze vstupu do
4:      bod_v_case ← casova_znacka − počáteční čas měření
5:      bod_v_case ← čas v bod_v_case převedený na sekundy
6:      kolekce_casu[bod_v_case] ← 1
7:  end for
8:  casova_rada ← struktura ve tvaru IP adresa : kolekce_casu
9:  return casova_rada
```

Výsledkem algoritmu 1 bude tzv. řídká matice s oborem hodnot $0, 1$. Algoritmus 1 postupuje tak, že si na začátku deklaruje `kolekci`, jejíž délka odpovídá velikosti měřeného úseku v sekundách. Všechny prvky kolekce jsou následně inicializovány na hodnotu 0. Poté algoritmus postupně po jedné prochází časové značky získané z databáze, přičemž každou časovou značku převede na `bod v čase`, který odpovídá počtu sekund mezi časovou značkou a začátkem měření. Následně se do `kolekce` na indexu, který je shodný s `bodem v čase`, zapíše hodnota 1. Na konci algoritmus vrátí strukturu ve tvaru `IP adresa : kolekce`.

Jelikož obvyklá četnost výskytu IP adresy v měřeném úseku nepřekračuje 50 procent, bude většina kolekce, kterou získáme na výstupu algoritmu 1, naplněná nulami, které značí, že v daném časovém bodě neexistuje pro danou IP adresu záznam, a menší část jedničkami, které značí přesný opak. Tahle skutečnost je první nevýhodou algoritmu 1. Druhou nevýhodou je fakt, že algoritmus potřebuje dopředu informaci o délce měřeného úseku, která v rámci této práce není malá, což znamená, že kolekce pro uložení bodů bude mít velikou délku. Důsledkem toho je, že zpracovat výstup algoritmu trvá nepřiměřeně dlouho, jak bylo experimentálně dokázáno v sekci 7.1.

Z těchto důvodů je nutné algoritmus 1 modifikovat tak, aby jeho výstupem byly pouze indexy časových bodů, ve kterých pro danou IP adresu existují časové záznamy. Modifikovaným algoritmem je algoritmus 2. Algoritmus 2 postupuje velmi podobně jako algoritmus 1, avšak na začátku vytvoří prázdnou `kolekci` nulové délky, která se v případě potřeby bude inkrementovat o jedna. Výpočet `bodu v čase` probíhá stejně, avšak `bod v čase` je přímo zapsán na konec `kolekce`. Výstupem algoritmu je nakonec struktura ve tvaru `IP adresa : kolekce`.

Tímto způsobem se zbavíme všech nevýhod algoritmu 1, i když výpočet vzdálenosti vektorů se o něco ztíží (viz sekce 6.2). Nicméně výhody dalece převažují nevýhody.

Algoritmus 2: INDEXOVÝ ZÁPIS ČASOVÉ ŘADY

Vstup: Struktura ve tvaru `IP adresa : kolekce detekovaných časových značek detekovaných událostí pro danou IP adresu`

Výstup: Indexový zápis časové řady

```
1:  kolekce_indexu ← prázdná kolekce
2:  i ← 0
3:  for casova_znacka in detekované časové značky ze vstupu do
4:      bod_v_case ← casova_znacka − počáteční čas měření
5:      bod_v_case ← čas v bod_v_case převedený na sekundy
6:      kolekce_indexu[i] ← bod_v_case
7:      i ← i +1
8:  end for
9:  indexovy_zapis ← struktura ve tvaru IP adresa : kolekce_indexu
10: return indexovy_zapis
```

6.2 Výpočet vzdálenosti dvou vektorů

U výpočtu vzdálenosti vyjdeme z Hammingovy vzdálenosti, jejíž způsob výpočtu je však nutné vhodně upravit (viz sekce 6.2.1), aby ji bylo možné aplikovat na indexové zápisy časových řad z algoritmu 2.

Ještě před uvedením zmíněných úprav je zapotřebí zmínit, že kdybychom Hammingovu metriku aplikovali na výstup z algoritmu 1, nebylo by zapotřebí výpočet nijak upravovat. Výstupem metriky by byl součet hodnot (jedniček), které se ve vektorech nenachází na stejném místě. Tato myšlenka je demonstrována v příkladu 1.

Příklad 1

Nechť jsou dány záznamy o IP adrese `A` v časech 12:00:00 a 12:00:03, a o IP adrese `B` v časech 12:00:01 a 12:00:03. Zkoumaný časový úsek je 12:00:00 - 12:00:05.

Z výstupu algoritmu 1 získáme 2 časové řady, tedy $casova\_rada\_A = (1, 0, 0, 1, 0, 0)$ a $casova\_rada\_B = (0, 1, 0, 1, 0, 0)$.

Aplikací Hammingovy metriky nad časovými řadami[1] získáme součet $1 + 1$, z čehož vyplývá, že každý detekovaný čas IP adresy `A`, který se nevyskytuje u IP adresy `B` (a naopak), se projeví přičtením 1 do výsledné vzdálenosti. Je zřejmé, že čím vyšší vzdálenost vyjde, tím méně si jsou IP adresy podobné.

[1] Budeme-li na časové řady nahlížet jako na binární vektory, tak pro výpočet Hammingovy vzdálenosti stačí použít operaci `XOR`

6.2.1 Způsob výpočtu vzdálenosti

Finální způsob výpočtu vzdálenosti dvou vektorů vychází z informací uvedených výše v sekci 6.2. Je nutné si uvědomit, že algoritmy 1 a 2 nám poskytují významově zcela identické výstupy, které jsou jen jiným způsobem reprezentované. Proto lze vzdálenost počítat algoritmem 3, který vychází z Hammingovy metriky.

Algoritmus 3 nahlíží na indexové zápisy časových řad získané z výstupu algoritmu 2 jako na množiny. Vzdálenost mezi 2 vektory tedy počítá tak, že z obou množin, které reprezentují analyzované vektory, odstraní průnik daných množin. Zdůvodnění tohoto kroku je, že prvky, které mají obě množiny společné, by se v Hammingově vzdálenosti nijak neprojevily (viz příklad 1), a proto jsou pro její výpočet irelevantní. Vzdálenost je poté rovna počtu zbylých prvků v obou množinách.

Výsledná vzdálenost se následně porovná s empiricky určeným prahem, který je nazýván `treshold`. Pro tyto hodnoty platí tvrzení 1.

Určování podobnosti dvou vektorů způsobem, který je popsán v této sekci, je sice možné, avšak má jednu zásadní nevýhodu, která je popsána v sekci 6.3. V sekci 6.3 je také popsán finální způsob určení podobnosti dvou vektorů, který je kvůli zmíněné nevýhodě k získání kvalitních výsledků nutné použít, avšak tento způsob vychází z myšlenek uvedených v této sekci.

Algoritmus 3: Výpočet vzdálenosti 2 vektorů

Vstup: Indexové zápisy časových řad získané algoritmem 2
Výstup: Vzdálenost vektorů

1: $vzdalenost \leftarrow 0$
2: $prunik \leftarrow$ *1. indexový zápis* $\cap$ *2. indexový zápis*
3: *1. indexový zápis* $\leftarrow$ *1. indexový zápis* $-$ *prunik*
4: *2. indexový zápis* $\leftarrow$ *2. indexový zápis* $-$ *prunik*
5: $vzdalenost \leftarrow |$*1. indexový zápis*$| + |$*2. indexový zápis*$|$
6: **return** *vzdalenost*

Tvrzení 1. *Je-li Hammingova vzdálenost 2 časových řad větší než treshold, potom si řady nejsou podobné. V opačném případě si podobné jsou.*

6.3 Určení podobnosti dvou vektorů

V sekci 6.2.1 bylo nastíněno, že určovat podobnost dvou vektorů pomocí jejich vzdálenosti má jednu vadu. K pochopení oné vady nám pomohou příklady 2 a 3.

V obou příkladech figurují dvě IP adresy o stejném počtu záznamů, což není pro funkčnost algoritmu 3 podmínkou, ale pro účely demonstrace je to výhodné. V příkladu 2 je demonstrována situace, u které lze říci, že určení podobnosti IP adres pomocí vzdálenosti funguje korektně, jelikož IP adresy, které se shodují v 10 případech z 15, nelze považovat za podobné.

Na druhou stranu v příkladu 3 je evidentně nesprávné tvrdit, že IP adresy, které se shodují v 95 případech ze 100, si nejsou podobné. Tento problém je zde způsoben tím, že nehledě na počet záznamů o IP adrese je pro všechny IP adresy nastaven stejný `treshold`.

Příklad 2
Nechť jsou dány dvě IP adresy `A` a `B`, o kterých existuje 15 záznamů (dohromady tedy

30). Množství shodných záznamů obou IP adres je rovno 10. Aplikací algoritmu 3 na tyto IP adresy zjistíme, že vzdálenost IP adres se rovná 10.

Je-li `treshold` zvolen jako číslo 9, tak tyto IP adresy si nejsou podobné, jelikož jejich vzdálenost je větší jak `treshold`.

Příklad 3
Nechť jsou dány dvě IP adresy `A` a `B`, o kterých existuje 100 záznamů (dohromady tedy 200). Množství shodných záznamů obou IP adres je rovno 95. Aplikací algoritmu 3 na tyto IP adresy zjistíme, že vzdálenost IP adres se rovná 10.

Je-li `treshold` zvolen jako číslo 9, tak tyto IP adresy si nejsou podobné, jelikož jejich vzdálenost je větší jak `treshold`.

Pokud bychom i přesto chtěli určovat podobnost mezi dvěma IP adresami pomocí jejich vzdálenosti, tak by `treshold` nesměl být nastaven implicitně na jednu danou hodnotu, ale musel by se explicitně nastavovat podle toho, kolik záznamů o IP adrese existuje. Řešení by tedy vyžadovalo existenci nějaké převodové tabulky, podle které by se podle počtu záznamů o IP adrese nastavil `treshold`. Vytvoření zmíněné převodové tabulky by kladlo veliké nároky na validitu, kterou by nebylo nijak lehké ověřit, jelikož hodnoty v převodové tabulce by šlo získat pouze pomocí experimentů. Navíc zde vyvstává další problém při určování podobnosti dvou IP adres, o kterých existují v řádu alespoň desítek rozdílné počty záznamů, jelikož algoritmus musí pracovat obecně. Například by se jednalo o IP adresu `A`, o které existuje 12 záznamů, a o IP adresu `B`, o které existuje 134 záznamů. V takovémto případě by bylo velmi nesnadné zvolit `treshold`.

6.3.1 Způsob určení podobnosti dvou vektorů

Finální způsob určení podobnosti dvou vektorů, které reprezentují záznamy o 2 IP adresách, zakládá na postupu v sekci 6.3, kde se podobnost 2 vektorů určuje pomocí vzdálenosti, která odpovídá počtu neshodných časových okamžiků obsažených v daných vektorech. Bude zde také použita empiricky zvolená hodnota `treshold`, avšak ta nebude porovnávána se vzdáleností vektorů.

Hodnota `treshold` bude použita pro porovnání podílu celkového počtu záznamů o IP adrese a počtu shodných záznamů obou IP adres, přičemž tyto podíly (pro obě IP adresy) musí být menší nebo rovny hodnotě `treshold`. Tato myšlenka je zapsána v tvrzení 2.

Tvrzení 2. *Nechť je dána IP adresa A s počtem záznamů α a IP adresa B s počtem záznamů β. Dále nechť číslo κ je rovno počtu záznamů, které mají IP adresy A a B společné.*

Platí-li nerovnost $\frac{\alpha}{\kappa} \leq$ `treshold` a zároveň $\frac{\beta}{\kappa} \leq$ `treshold`, potom jsou si IP adresy A a B podobné.

6.3.2 Práce s časovým posunem

Při určování podobnosti 2 vektorů je také zapotřebí brát v zřetel, že detektory, které daly vzniknout zkoumaným vektorům, nepracují vždy se stoprocentní časovou přesností. To znamená, že časová značka, kterou události přidělí, se může oproti realitě lišit. Navíc 2 anomálie, které byly na síti generovány ve stejný čas se stejným cílem, tudíž jsou si podobné, nemusejí do cíle dorazit ve stejný čas, jelikož se může některá z nich z mnoha různých důvodů zdržet na síťovém zařízení po cestě.

Ze zmíněných důvodů je tedy nutné k počtu záznamů κ z tvrzení 2, přičíst ty záznamy, jejichž rozdíl je menší nebo roven empiricky zvolenému `časovému posunu`.

6.4 Hledání podobných skupin

Jak již bylo řečeno na začátku kapitoly, podobné skupiny IP adres budeme hledat postupně, tedy v nalezených dvojicích se pokusíme nalézt trojice, dále v získaných trojicích zkusíme najít čtveřice atd. dokud bude možné poskládat o 1 větší n-tici oproti předchozímu kroku. Příklady 4, 5 a 6 demonstrují, co musí platit, abychom z menších skupin mohli vytvořit o 1 větší skupinu. Obecně je tedy při určování, zdali IP adresy tvoří podobnou skupinu o velikosti n, nutné, aby byly splněny dvě podmínky. Tou první je, že musí existovat všechny skupiny IP adres o velikosti $n-1$, které lze vytvořit z dané skupiny o velikosti n. Tyto skupiny musí být dále všechny navzájem podobné, což je druha ze zmíněných podmínek.

Učiníme tak podle algoritmu 4, který funguje tak, že si pomocí množinových operací vytváří `podezřelé skupiny`, což jsou skupiny, které vznikají z o 1 menších skupin. Při prvním nálezu `podezřelé skupiny` se vytvoří struktura ve tvaru `podezřelá skupina : 1`, v níž číslo jedna reprezentuje počet výskytů `podezřelé skupiny`. Následně pokud se nějaká „podezřelá skupina“ vyskytne v `dostatečném počtu`, tak je prohlášena za podobnou, jinak se její počet výskytu inkrementuje o jedna.

Pochopit, jaké hodnoty musí `dostatečný počet` nabýt, nám pomůže obrázek 6.1 (spolu s příkladem 6), na kterém vidíme postup algoritmu 4 při vytváření pětice. Každá spojnice, ze které vycházejí šipky, reprezentuje průchod vnějšího cyklu, který začíná na řádku číslo 5, přičemž směr spojnic je shora dolů. Samotné šipky jsou pak ukázkou průchodu vnitřního cyklu, který začíná na řádku 6. V souhrnu se tedy při hledání pětice stane to, že se „podezřelá množina“ v algoritmu 4 vyskytne právě $(4 + 3 + 2 + 1)$krát, tj. desetkrát.

Obecně, hledáme-li skupinu o velikosti n, tak `dostatečný počet` se vypočítá podle vztahu v rovnici 6.1.

$$dostatecny\_pocet = \sum_{k=1}^{n-1} k \tag{6.1}$$

Příklad 4
Nechť je dána trojice `ABC`, která se skládá ze 3 různých podobných IP adres `A`, `B` a `C`. Poté musí existovat 3 podobné dvojice `AB`, `AC` a `BC`.

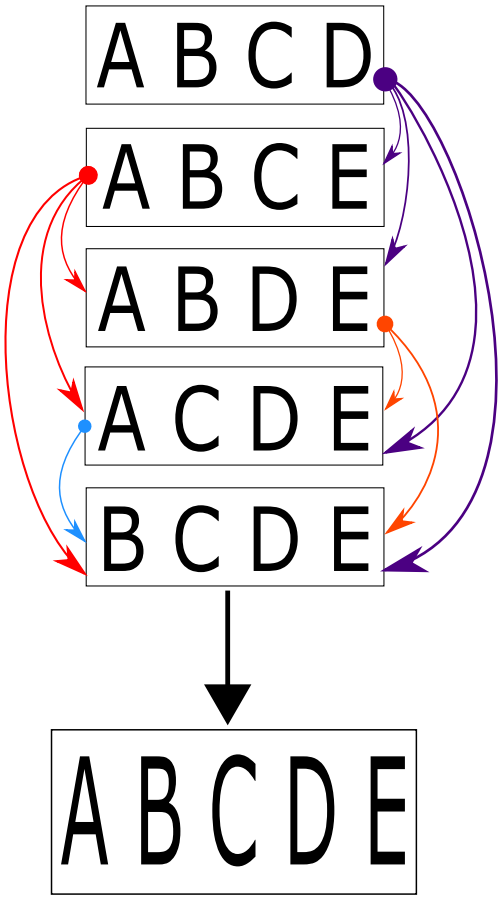
Příklad 5
Nechť je dána čtveřice `ABCD`, která se skládá ze 4 různých podobných IP adres `A`, `B`, `C` a `D`. Poté musí existovat 4 podobné trojice `ABC`, `ABD`, `ACD` a `BCD`.

Příklad 6
Nechť je dána pětice `ABCDE`, která se skládá z 5 různých podobných IP adres `A`, `B`, `C`, `D` a `E`. Poté musí existovat 5 podobných čtveřic `ABCD`, `ABCE` a `ABDE`, `ACDE` a `BCDE`.

Algoritmus 4: Nalezení podobných skupin

Vstup : Kolekce podobných n-tic
Výstup : Podobné skupiny

```
1:  skupiny ← prázdná kolekce
2:  do
3:      podezrele_skupiny ← prázdná kolekce
4:      nove_skupiny ← prázdná kolekce
5:      for ntice_1 in podobné n-tice do
6:          for ntice_2 in podobné n-tice od ntice_1 z nadřazeného cyklu
            do
7:              prunik ← ntice_1 ∩ ntice_2
8:              if prunik je neprázdný then
9:                  podezrela_skupina ← ntice_1 ∪ ntice_2
10:                 if podezrela_skupina ∉ podezrele_skupiny then
11:                     pocet_vyskytu ← 1
12:                     nova_struktura ← struktura ve tvaru
                         podezrela_skupina : pocet_vyskytu
13:                     vlož nova_struktura do podezrele_skupiny
14:                 else
15:                     pocet_vyskytu ← pocet_vyskytu +1
16:                     if pocet_vyskytu je dostatečný then
17:                         přidej podezrela_skupina do nove_skupiny
18:                     else
19:                         aktualizuj pocet_vyskytu u podezrela_skupina
20:                     end if
21:                 end if
22:             end if
23:         end for
24:     end for
25:     přidej nove_skupiny do skupiny
26: while Byla nalezena alespoň 1 nová skupina;
27: return skupiny
```



Obrázek 6.1: Ukázka algoritmu 4 při vyhledávání pětice ze čtveřic

Kapitola 7

Experimenty

Experimenty, které budou dále popsány v této kapitole, byly všechny prováděny na stejném výpočetním stroji s OS Windows 10 v programovacím jazyce Python a nad daty získanými pomocí systému Warden v časovém období od 30. 5. 2018 22:38:08 do 31. 5. 2018 1:59:59, ve kterém se nachází záznamy o bezpečnostních incidentech obsahující 4702 IP adres, pokud není uvedeno jinak. Ve všech experimentech je součástí implementace redukce šumu, protože šum zde zapříčiňoval nálezy mnoha dvojic, které se shodovaly např. ve 3 časech, což zbytečně zpomalovalo běh programu. Redukce je dosaženo tak, že jsou analyzovány pouze IP adresy, o nichž existuje alespoň 10 detekcí v různých časech. Po aplikaci redukce se počet analyzovaných IP adres snížil na 1775. Dále všechny časové údaje jsou výsledkem zprůměrování 10 pokusů, přičemž časy se lišily nanejvýš v řádu jednotek sekund, opět pokud není uvedeno jinak. Vzhledem k tomu, že přenosový formát IDEA vyžaduje, aby časové značky byly reprezentovány v čase UTC [5], tak časový formát není brán v zřetel (předpokládaná podoba časových značek je `rok-měsíc-den hodiny-minuty-sekundy`).

7.1 Experimenty s algoritmem 1

Zpracování časového okna zabralo 1,104 sekund. Za tuto dobu se vygeneroval výstup, na jehož zpracování bylo zapotřebí 21 498 800 záznamů (bez 1775 záznamů pro samotné IP adresy). To znamená, že na jednu IP adresu bylo použito 12 112 záznamů, což přesně odpovídá charakteru algoritmu, jelikož toto číslo zároveň reprezentuje délku analyzovaného okna v sekundách.

Čas potřebný ke spočtení pouze Hammingovy vzdálenosti mezi vektory reprezentující IP adresy, tudíž bylo vynecháno její porovnávání s hodnotou `treshold` a také práce s časovým posunem, byl vyměřen na 1:12:13. Vzhledem k délce tohoto času se jedná o výsledek zaokrouhlení pouze tří pokusů, jejichž odlišnost se pohybovala v řádu desítek sekund.

Hammingova vzdálenost byla počítána pomocí operace XOR mezi jednotlivými body v čase v časových řadách z algoritmu 1. Kdybych v daných datech určoval podobnost pomocí finální metody uvedené v sekci 6.3.1, tak by místo operace XOR musela být použita operace XNOR (negace XOR), která ke své realizaci přirozeně potřebuje o jeden krok navíc, tudíž by celý proces trval ještě déle. Navíc práce s časovými posuny by zde nešla implementovat přímočaře pomocí rozdílů dvou bodů v čase, takže by vedla k dalšímu vysokému prodloužení. Z těchto důvodů jsem se rozhodl algoritmus 1 dále nepoužívat.

7.2 Experimenty s algoritmem 2

Časové okno bylo zpracováno za 0,726 sekund. Po tomto časovém intervalu algoritmus 2 poskytuje výstup, který při svém zpracování spotřeboval 43 314 záznamů (opět bez 1 775) a to odpovídá zhruba 24,4 záznamům na IP adresu. Zde lze již poprvé vidět, že algoritmus 2 pracuje mnohem efektivněji v oblastni množství výstupních dat, což se výrazně projeví na rychlosti jejich zpracování.

Vyměření Hammingovy vzdálenosti mezi jednotlivými body v čase v časových řadách z algoritmu 2 bylo provedeno podle algoritmu 3. Čas k tomu potřebný se rovná 5,475 sekund, což je obrovská úspora oproti algortimu 1. Tento čas se týká pouze samotného výpočtu vzdálenosti, nicméně zde je již také proveditelné, vzhledem k optimálně malému času výpočtu vzdálenosti, vést experiment podle postupu v sekci 6.3.1.

Nalezení podobných dvojic dle finální metody zabralo 8,585 sekund. Je však nutné dodat, že tohoto času bylo dosaženo bez práce s časovými posuny. Po začlenění časových posunů algoritmus nalezl všechny podobné dvojice za 96,089 sekund.

7.3 Nastavení hodnoty treshold a časového posunu

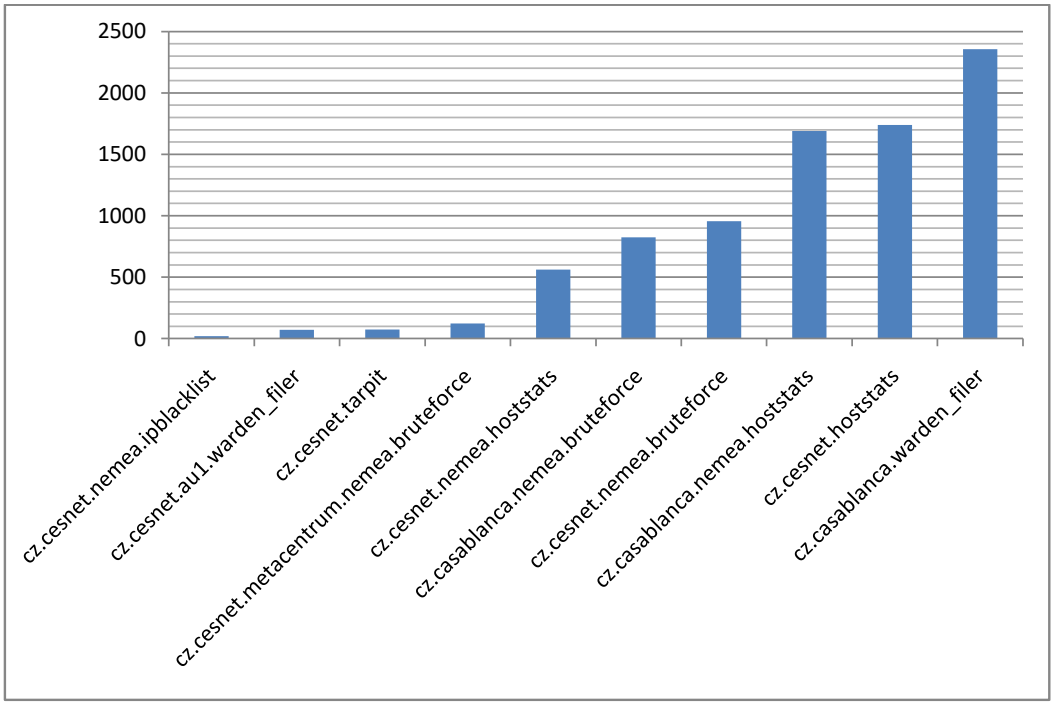
Hledání optimální hodnoty treshold a délky časového posunu musí probíhat současně, jelikož se stejným způsobem podílejí na ovlivňování výsledků. Například, bude-li treshold, respektive časový posun nastaven na příliš vysoké číslo, tak v lepším případě bude výstup programu obsahovat velké množství falešně pozitivních nálezů podobných skupin. Horší variantou je, že programu může trvat nepřiměřeně dlouho dokončit svůj běh a nebo by musel být uživatelem ukončen. Naopak, pokud by dané hodnoty byly nastaveny příliš nízko, tak by se z výstupu vytratily skupiny, jejichž podobnost není stoprocentní, avšak stále je lze považovat za podobné.

Na začátku jsem náhodně určil hodnotu treshold jako 1,1 a míru časového posunu jako 40 sekund. S těmito hodnotami program nalezl v analyzovaných datech dohromady 942 skupin s podobnými IP adresami, přičemž největší skupina obsahovala 16 IP adres. Nicméně tohoto výsledku bylo dosaženo za 1:57:3, což nelze považovat za přijatelnou dobu[1]. Tento pokus byl proveden pouze jedenkrát.

Ve druhém pokusu jsem nechal treshold na stejné hodnotě, avšak časový posun jsem snížil na 20 sekund. Výsledků bylo dosaženo za 122,562 sekund, přičemž program nalezl 330 skupin s podobnými IP adresami – největší skupina byla sedmiprvková. Po detailnějším pohledu na nalezené skupiny jsem dospěl k závěru, že tento treshold umožňuje výskyt falešně pozitivních nálezů a tudíž musí být snížen.

Při třetím experimentu byl časový posun navrácen na hodnotu 40 sekund a treshold byl snížen na 1,05. S danými parametry program našel 208 skupin s podobnými IP adresami za 163,580 sekund. Největší skupina čítala 11 IP adres. V porovnání s druhým pokusem zde bylo nalezeno více větších skupin a méně těch menších. Po porovnání nalezených větších skupin se ukázalo, že jejich označení za podobné lze považovat za korektní. Výskyty falešně pozitivních nálezů jsem neodhalil. Z těchto důvodů jsem se rozhodl dané hodnoty již ponechat.

[1] Za přijatelnou dobu považuji čas do 5 minut.



Obrázek 7.1: Počty detekovaných záznamů o IP adresách jednotlivými detektory

7.4 Vyhodnocení výsledků

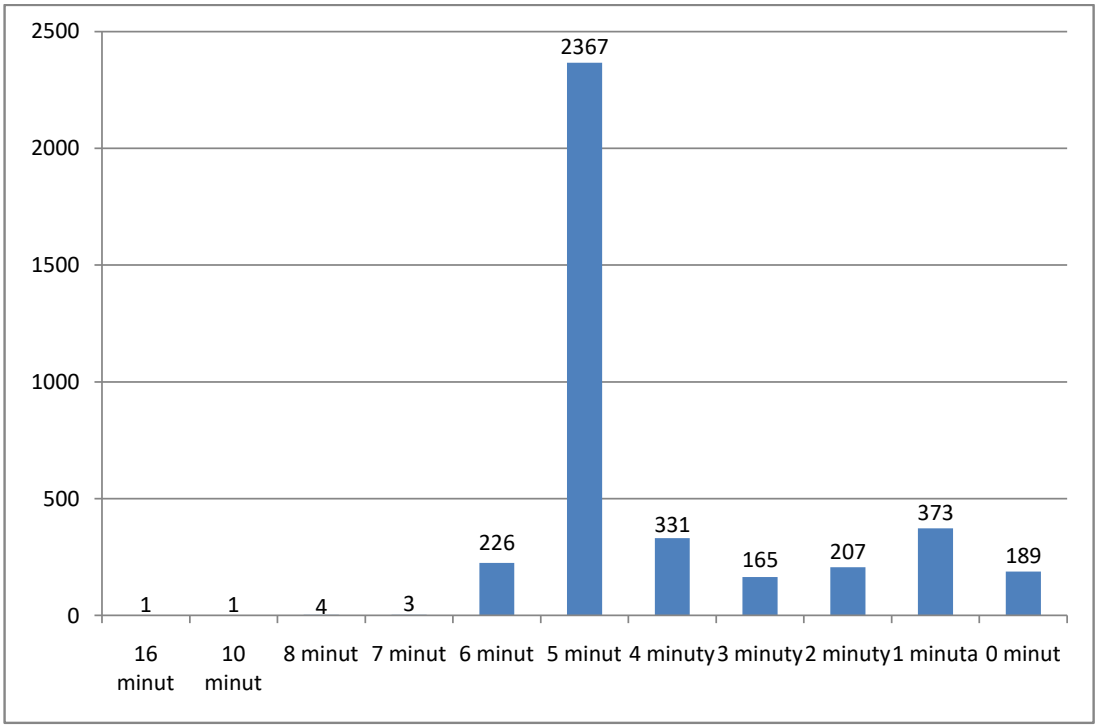
Jak již bylo zmíněno výše, program nalezl 208 skupin s podobnými IP adresami. Nejvíce záznamů o daných IP adresách poskytl *cz.casablanca.warden_filer*, nejméně pak *cz.cesnet.nemea.ipblacklist*. Podrobnější informace byly zaneseny do grafu na obrázku 7.1, který se týká pouze detekcí IP adres, které se vyskytují v nalezených skupinách, a jehož svislá osa představuje souhrnný počet detekovaných záznamů o IP adresách a na vodorovné ose jsou pak vyobrazeny jednotlivé detektory.

Ze statistických údajů o nalezených skupinách lze uvést, že průměrně se po zaokrouhlení vyskytovalo ve skupině 5 IP adres, střední hodnota počtů výskytů byla také 5 a nejčetnější skupiny byly dvouprvkové, kterých se na výstupu programu nachází 53. Velikosti nalezených skupin se příliš nelišily, jak dokládá hodnota směrodatné odchylky, která byla rovna 2,739.

Dalšími statistickými údaji jsou informace o počtech časů, ve kterých jsou si IP adresy v jednotlivých skupinách podobné. Průměrně se po zaokrouhlení jedná o 20 takovýchto časů, medián je roven šestnácti a v nejvíce případech se v nalezených skupinách vyskytovaly IP adresy, které si byly podobné v 16 časech – to se přesně týkalo 132 skupin. Směrodatná odchylka zde dosahuje hodnoty 7,253, tudíž rozdíly jsou tu již o něco větší.

Z grafu, jehož svislá osa značí počet výskytů a vodorovná osa délku intervalů, na obrázku 7.2 lze vyčíst, že v drtivé většině případů se prodleva mezi jednotlivými detekcemi podobných IP adres pohybuje okolo 5 minut. Jedná se o přibližný údaj, jelikož hodnoty byly počítány bez sekund. To znamená že interval „0 minut“ odpovídá 0 až 59 sekundám. Z toho vyplývá, že všechny podobnosti jsou v podstatě stejného charakteru – intervaly, ve kterých jsou si IP adresy podobné, mají mezi s sebou jen krátké rozestupy. Výjimkou jsou pouze šestnácti a desetiminutové prodlevy, které se v datech nachází každá po jedné.

Některé z nalezených skupin jsem podrobil podrobnému průzkumu. Nejprve se jednalo o 3 skupiny složené z 11 IP adres – 8 IP adres měly všechny skupiny společné a časy detekcí



Obrázek 7.2: Souhrn délek intervalů mezi jednotlivými detekcemi

se shodovaly všechny. Dohledal jsem typy útoků v nalezených časech, kterých bylo dohromady 16, a až na výjimky se vždy jednalo o mnohonásobné pokusy o přihlášení přes protokol SSH, což pravděpodobně odpovídá hádání hesla. Následně jsem prozkoumal všechny 4 desetiprvkové skupiny z nichž 2 a 2 měly stejné časy detekcí. U prvních dvou se jednalo o mnohonásobné pokusy o přihlášení, zatímco u zbylých 2 šlo o skenování portu. Ani u dalších zhruba 30 nalezených skupin, které jsem prošel, se charakter nálezů nijak nelišil. Útoky pocházející od detekovaných IP adres byly vždy typu skenování portů nebo mnohonásobných pokusů o přihlášení. Dále jsem zaznamenal, že počty časů, ve kterých jsou si IP adresy podobné, se většinou shodují s množstvím časových značek vztahujících se k jednotlivým IP adresám, přičemž rozdíly v těchto množstvích jsem nenalezl větší než 1. Tato skutečnost je dána nastavenou hodnout `treshold`.

Kapitola 8

Závěr

Cílem této práce bylo v bezpečnostních síťových hlášeních nalézt IP adresy, které byly detekovány ve stejných nebo velmi podobných časech. Byly zde vysvětleny dva postupy, jak lze daného cíle dosáhnout. Jejich společnou myšlenkou je, že se nejprve naleznou podobné dvojice, v nalezených dvojicích se hledají podobné trojice, v těch pak podobné čtveřice atd. Nicméně postupy se liší ve způsobu hledání podobných dvojic.

První metoda hledá podobné dvojice za pomoci Hammingovy vzdálenosti, která je po jejím výpočtu porovnána s hodnotou `treshold`, což je vlastně práh, který je-li překročen, tak si dvě IP adresy nejsou podobné. Avšak právě určení onoho prahu, jak bylo v textu vysvětleno, je velmi náročné a vedlo k vytvoření druhé, finální metody.

Druhá metoda posuzuje podobnost dvou IP adres za pomoci dvou poměrů porovnaných s hodnotou `treshold`. Tyto dva poměry představují podíl celkového počtu záznamů pro IP adresu ku počtu společných záznamů obou IP adres. Jelikož IP adresy nemusejí mít stejný počet záznamů, tak se jedná o právě 2 zmíněné poměry. Dále pokud jsou oba výsledné poměry menší nebo rovny hodnotě `treshold`, tak jsou si IP adresy podobné. Výhodou tohoto postupu je, že na rozdíl od první metody zohledňuje počet záznamů pro IP adresu, jelikož na podobnost IP adres se 100 záznamy nelze nahlížet stejně jako na podobnost IP adres s 10 záznamy, což v 1. metodě není dodrženo. Touto metodou bylo nalezeno 208 podobných skupin, z nichž největší (3) obsahovaly 11 IP adres. U nalezených skupin podrobených analýze se vždy jednalo o kybernetické aktivity typu skenování portů nebo mnohonásobné pokusy o přihlášení. Počty časů, ve kterých si byly IP adresy podobné, se většinou shodovaly s množstvím časových značek vztahujících se k jednotlivým IP adresám.

Budoucí vývoj této práce by se měl zaměřit na efektivnější práci s časovými posuny, jelikož při určování dvojic dle finální metody tato část zabírá celých 91 % času. Dále je zde možnost zohledňovat při určování podobnosti nejen čas, ale i o jaký typ útoku se jedná, což by vedlo k přesnějšímu určení, zdali se nalezená skupina dá řadit pod botnet.

Literatura

[1] *Data mining*. [Online; navštíveno 14.04.2019].
URL https://cs.wikipedia.org/wiki/Data_mining

[2] *Internet*. [Online; navštíveno 01.04.2019].
URL https://cs.wikipedia.org/wiki/Internet

[3] *Number of internet users worldwide 2005-2018*. [Online; navštíveno 01.04.2019].
URL https://www.statista.com/statistics/273018/number-of-internet-users-worldwide/

[4] CESNET: *Architektura*. [Online; navštíveno 26.04.2019].
URL https://warden.cesnet.cz/cs/architecture

[5] CESNET: *Design decisions*. [Online; navštíveno 29.04.2019].
URL https://idea.cesnet.cz/en/design

[6] CESNET: *Intrusion Detection Extensible Alert*. [Online; navštíveno 06.04.2019].
URL https://idea.cesnet.cz/en/index

[7] CESNET: *Mentat*. [Online; navštíveno 07.04.2019].
URL https://mentat.cesnet.cz/cs/index

[8] CESNET: *O nás*. [Online; navštíveno 02.04.2019].
URL https://www.cesnet.cz/sdruzeni/

[9] CESNET: *O projektu*. [Online; navštíveno 02.04.2019].
URL https://warden.cesnet.cz/cs/about_project

[10] CESNET: *Warden*. [Online; navštíveno 02.04.2019].
URL https://warden.cesnet.cz/cs/index

[11] CSIRT.CZ: *Základní principy DoS útoku*. Duben 2015, [Online; navštíveno 23.04.2019].
URL https://www.csirt.cz/page/2790/zakladni-principy-dos-utoku/

[12] Došlá, Z.; Došlý, O.: *Metrické prostory*. Masarykova univerzita, 2006, ISBN 80-210-4160-9.

[13] Feily, M.; Shahrestani, A.; Ramadass, S.: *A survey of botnet and botnet detection*. In *2009 Third International Conference on Emerging Security Information, Systems and Technologies*, IEEE, 2009, s. 268–273.

[14] García-Teodoro, P.; Díaz-Verdejo, J.; Maciá-Fernández, G.; aj.: Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, ročník 28, č. 1, 2009, ISSN 0167-4048, doi:https://doi.org/10.1016/j.cose.2008.08.003, [Online; navštíveno 05.05.2019]. URL http://www.sciencedirect.com/science/article/pii/S0167404808000692

[15] Holčík, J.; Komenda, M.: *Matematická biologie: e-learningová učebnice [online].* Masarykova univerzita, 2015, ISBN 978-80-210-8095-9.

[16] Jirovský, V.: *Kybernetická kriminalita nejen o hackingu, crackingu, virech a trojských koních bez tajemství.* Grada Publishing, a.s., 2007, ISBN 978-80-247-1561-2.

[17] Kohout, R.; Karchňák, M. R.: *Bezpečnost v online prostředí.* BIBLIO KARLOVY VARY, z.s., 2016, ISBN 978-80-260-9543-9.

[18] Kuřina, F.: *Pojem vzdálenosti v geometrii.* [Online; navštíveno 28.01.2019]. URL http://mfi.upol.cz/files/25/2504/mfi_2504_241_245.pdf

[19] Mueen, A.; Nath, S.; Liu, J.: *Fast Approximate Correlation for Massive Time-series Data.* [Online; navštíveno 14.04.2019]. URL http://alumni.cs.ucr.edu/~mueen/pdf/sigmod2010.pdf?fbclid=IwAR2W12zdRfQ-axKmkxc6wwcX1qgKG5aVoGQR6E15G1SgLCwrmF3iROEAqWo

[20] Olšák, P.: *Eukleidovský prostor.* [Online; navštíveno 16.04.2019]. URL http://petr.olsak.net/bilin/eprostor4.pdf

[21] Policie České republiky: *Kyberkriminalita.* [Online; navštíveno 01.04.2019]. URL https://www.policie.cz/clanek/kyberkriminalita.aspx

[22] Shadowserver Foundation: *What We Do.* [Online; navštíveno 28.04.2019]. URL https://www.shadowserver.org/what-we-do/

[23] Shadowserver Foundation: *Who We Are.* [Online; navštíveno 28.04.2019]. URL https://www.shadowserver.org/who-we-are/

[24] Spitzner, L.: *Honeypots: Tracking Hackers.* Addison Wesley, 2002, ISBN 0-321-10895-7.

[25] de Vivo, M.; Carrasco, E.; Isern, G.; aj.: *A Review of Port Scanning Techniques. SIGCOMM Comput. Commun. Rev.*, ročník 29, č. 2, Duben 1999: s. 41–48, ISSN 0146-4833.

Příloha A

Příklad formátu IDEA

Příklad je přebrán ze zdroje [6].


```
{
  "Format" : "IDEA0",
  “ID"; : "4390fc3f-c753-4a3e-bc83-1b44f24baf75",
  "CreateTime" : "2012-11-03T10:00:02Z",
  "DetectTime" : "2012-11-03T10:00:07Z",
  "WinStartTime" : "2012-11-03T05:00:00Z",
  "WinEndTime" : "2012-11-03T10:00:00Z",
  "EventTime" : "2012-11-03T07:36:00Z",
  "CeaseTime" : "2012-11-03T09:55:22Z",
  "Category" : ["Fraud.Phishing"],
  "Ref" : ["cve:CVE-1234-5678"],
  "Confidence" : 1,
  "Note" : "Synthetic example",
  "ConnCount" : 20,
  "Source" : [
    {
      "Type" : ["Phishing"],
      "IP4" : ["192.168.0.2-192.168.0.5", "192.168.0.10/25"],
      "IP6" : ["2001:0db8:0000:0000:0000:ff00:0042::/112"],
      "Hostname" : ["example.com"],
      "URL" : ["http://example.com/cgi-bin/killemall"],
      "Proto" : ["tcp", "http"],
      "AttachHand" : ["att1"],
      "Netname" : ["ripe:IANA-CBLK-RESERVED1"]
    }
  ],
  "Target" : [
    {
      "Type" : ["Backscatter", "OriginSpam"],
      "Email" : ["innocent@example.com"],
      "Spoofed" : true
    },
    {
      "IP4" : ["10.2.2.0/24"],
```

```
        "Anonymised" : true
      }
    ],
    "Attach" : [
      {
        "Handle" : "att1",
        "FileName" : ["killemall"],
        "Type" : ["Malware"],
        "ContentType" : "application/octet-stream",
        "Hash" : ["sha1:0c4a38c3569f0cc632e74f4c"],
        "Size" : 46,
        "Ref" : ["Trojan-Spy:W32/FinSpy.A"],
        "ContentEncoding" : "base64",
        "Content" : "TVpqdXN0a2lkZGluZwo="
      }
    ],
    "Node" : [
      {
        "Name" : "cz.cesnet.kippo-honey",
        "Type" : ["Protocol", "Honeypot"],
        "SW" : ["Kippo"],
        "AggrWin" : "00:05:00"
      }
    ]
  }
```