ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Lejnar** Jméno: **Jan** Osobní číslo: **435645**

Fakulta/ústav: **Fakulta informačních technologií**

Zadávající katedra/ústav: **Katedra softwarového inženýrství**

Studijní program: **Informatika**

Studijní obor: **Softwarové inženýrství**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Nástroj pro forenzní analýzu serverových logů

Název bakalářské práce anglicky:

Tool for server log forensics

Pokyny pro vypracování:

Seznamte se s podobou serverových logů a proveďte analýzu využitelnosti obsažených údajů.
Navrhněte modulární nástroj, který zjednoduší bezpečnostní analýzu serverových logů. Nástroj by měl umět agregovat veškeré dostupné logy (např. přístupový a chybový log webového serveru, mailový log, autentikační log, ...). Z těchto informací nástroj vytvoří přehled, ze kterého je patrná aktivita v předem definovaném časovém intervalu. Přehled zobrazuje korelace (např. neoprávněný přístup na webovou stránku a pokus o autentizaci ze stejné zdrojové adresy).
Prototyp nástroje implementujte, otestujte a sepište uživatelskou dokumentaci.

Seznam doporučené literatury:

Dodá vedoucí práce.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Jan Baier, katedra teoretické informatiky FIT

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

,

Datum zadání bakalářské práce: **02.11.2016** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: _____

_____	_____	_____
Podpis vedoucí(ho) práce	Podpis vedoucí(ho) ústavu/katedry	Podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

_____	_____
Datum převzetí zadání	Podpis studenta

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Nástroj pro forenzní analýzu serverových logů

Jan Lejnar

Vedoucí práce: Ing. Jan Baier

12. května 2017

Poděkování

Mé poděkování patří zejména Ing. Janu Baierovi za cenné rady, věcné připomínky a vstřícnost při konzultacích v průběhu vypracovávání této bakalářské práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 12. května 2017 .....

Odkaz na tuto práci

Lejnar, Jan. *Nástroj pro forenzní analýzu serverových logů*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Tato práce se zabývá forenzní analýzou serverových logů, jak v nich obsažené údaje propojit a využít. Cílem je navrhnout modulární nástroj, který dokáže vytvořit přehled událostí v definovaném časovém intervalu. V přehledu jsou patrné korelace, jako podezřelé aktivity typu: neúspěšný pokus o přihlášení a následný opakovaný pokus o autentizaci ze stejné zdrojové adresy atd. Nástroj je možné využít ke zvýšení bezpečnosti a monitorování událostí na serveru. Aplikace je vybudována z části na existujícím řešení, které bylo nutné lehce vylepšit pro účel této práce. Druhou částí je grafická nadstavba, kterou jsem vytvořil ve *frameworku* Qt. Z pohledu programovacích jazyků se jedná o kombinaci C++ a Perlu. Výsledkem práce je funkční aplikace, která je připravena na možnost dalšího rozšíření. Pro správce serveru nebude problém dodefinovat si podezřelé aktivity, které by chtěl sledovat. Vytvořené řešení úspěšně shlukuje údaje ze serverových logů a provádí nad nimi forenzní analýzu.

Klíčová slova forenzní analýza, serverové logy, monitorování událostí, podezřelé aktivity, bezpečnost, modulární nástroj, Qt, C++, Perl

Abstract

This thesis deals with the forensic analysis of server logs, how to connect and utilize the data taken from these log files. The goal is to design a modular tool that has the ability to create an overview of events within a defined time interval. Correlations can be seen in the report. For example suspicious activities such as unsuccessful login attempt and following repeated attempts to authenticate from the same source address, etc. The tool can be used to increase security and monitor events on the server. The application is built in part on an existing solution that had been slightly improved for the purpose of this work. The second part is the graphic superstructure that I created in the Qt *framework*. It is a combination of C ++ and Perl from the perspective of programming languages. The result of the work is a functional application that is ready for further expansion. The server administrator will easy manage to specify suspicious activities he wants to track. The solution created successfully merges data from server logs and performs forensic analysis.

Keywords forensic analysis, server logs, event monitoring, suspicious activities, security, modular tool, Qt, C++, Perl

Obsah

Seznam obrázků

Seznam tabulek

Úvod

S roustoucí popularitou webových aplikací také roste význam bezpečnosti na internetu. Jednou z možností, jak zmenšit riziko zneužití, je nepodceňovat informace ze serverových logů, což jsou soubory, do kterých se na serveru zaznamenávají zajímavé události a aktivity. Tyto události se obvykle logicky rozčlení, a proto lze na serveru nalézt více typů logů. Například chybový a přístupový log.

Logovací soubory jsou cenným zdrojem informací pro forenzní analýzu, která umožňuje dohledat sled událostí, které vedly k nějakému nežádanému incidentu. Pokud dvě události korelují, znamená to, že spolu nějak souvisí. Hledání takovýchto korelací je přesně hlavním úkolem forenzní analýzy.

Seznámení s problematikou

V dnešní době většina softwaru, webových serverů atd. loguje. Programátory to nestojí mnoho úsilí. Otázkou je, jak moc jsou tyto logy následně využívány. Je potřeba říci, že logy jsou často velice nedoceněné a opomíjené. Není neobvyklé, že jsou ignorovány, dokud nezačnou zabírat na disku příliš mnoho místa. [1]

Logovací soubory nejčastěji ocení systémoví administrátoři, nicméně ruční procházení logů je pracné a zdlouhavé. Lze napsat skripty na vytažení konkrétního údaje. Pro agregaci zjištěných informací, s cílem zjistit vyšší smysl útržků, je již však lepší využít nějaký automatizovaný nástroj.

Avšak vytvořit nástroj pro forenzní analýzu není triviální úkol, protože nelze aplikovat mnoho obecných postupů. Hlavní důvody jsou dva:

1. Každý výrobce zpravidla používá svůj formát logu, což ztěžuje následnou snahu spojit údaje z více logů a hledání korelací.

2. Efektivní analýza vyžaduje znalost konkrétního prostředí. Je potřeba definovat, co je podezřelá aktivita a co není. Například informace o vypnutí serveru o víkendu může být v některém prostředí podezřelá, jinde ovšem zcela běžná.

Motivace

Do budoucnosti lze předpokládat, že objem logovacích dat ještě dramaticky naroste, stejně jako pestrost různých formátů logů. Po pravdě řečeno, již dnes mají některé organizace k dispozici kolem 1 PB logovacích dat, proto je výzvou vytvořit dostatečně rychlý a sofistikovaný nástroj, který dokáže nalézt korelace v rozumném čase. [1]

Bylo by ideální, kdyby se začal dodržovat nějaký standardizovaný formát logu, jako například *Common Log Format*. [11] Bohužel různí výrobci považují jiné informace za důležité a některé je pro ně zbytečné evidovat, proto definují vlastní formát logu.

Analýza logů je tedy nucena být připravena na tuto skutečnost. Existuje mnoho nástrojů, které dokážou zkoumat logy vybraných formátů. Mým cílem je vytvořit modulární aplikaci, která bude podporovat základní typy logů, ale také umožní hledat korelace i napříč logovacím souborem, který má atypický formát.

Struktura práce

Text práce je členěn do šesti kapitol. V první kapitole popisuji cíle práce. Druhá kapitola se zabývá literární rešerší již existujících nástrojů pro práci s logy. V této kapitole také představuji základní typy serverových logů. Třetí kapitola je věnována určitým fázím softwarovému procesu – od sběru požadavků, přes analýzu, až po implementaci. Čtvrtá kapitola obsahuje instalační a uživatelskou příručku. V páté kapitole je uvedeno, jakým způsobem probíhalo testování a jaké mělo výsledky. V poslední kapitole rekapituluji, jak jsem práci vypracoval a shrnuji přínosy vytvořeného nástroje.

Kapitola 1

Cíl práce

Cílem práce je navrhnout po softwarově-inženýrské stránce nástroj, který dokáže spojit více logů a jeho hlavním úkolem bude zobrazit přehled podezřelých aktivit v nějakém časovém intervalu. V přehledu budou patrné korelace, jako podezřelé aktivity typu: neúspěšný pokus o přihlášení a následný opakovaný pokus o autentizaci ze stejné zdrojové adresy atd.

Navíc je kladen důraz na modularitu aplikace, aby bylo možné dodefinovat pravidla popisující, co je podezřelá aktivita a jaká akce se má provést, pokud k této aktivitě dojde. Není žádoucí vytvořit aplikaci, která by podporovala pouze vybranou množinu typů logů. Kromě nejčastějších formátů si musí být navržený nástroj schopný poradit i s atypickým formátem logu.

Výsledkem práce bude funkční prototyp aplikace, připraven na možnost dalšího rozšíření. Pro správce serveru nebude problém si dodefinovat aktivity, které by chtěl sledovat, a také specifikovat požadovanou reakci (například odeslat upozorňovací e-mail).

Nástroj bude možné využívat ke zvýšení bezpečnosti a monitorování událostí na serveru.

Kapitola 2

Literární rešerše

2.1 Analýza formátů logů

Jak jsem již zmiňoval v úvodu, existuje mnoho rozmanitých formátů logů. Každý výrobce rád používá svůj specifický formát. Jaké jsou příklady známých serverových logů, jaké informace mohou obsahovat a kde se tyto informace nacházejí, popíšu v následující podkapitole.

2.1.1 Příklady logů

2.1.1.1 Linuxové systémové autorizační logy

Tento soubor typicky najdete v linuxových distribucích s absolutní cestou `/var/log/auth.log`. Sleduje používání mechanismů pro autorizaci uživatele sudo(příkaz `sudo`), vzdálené přihlášení k sshd atd. Jde o dobrý zdroj informací při zjišťování, zda se někdo neoprávněně nedostal na server. [12]

Listing 2.1: Formát události z linuxového autorizačního logu [12]

```
<měsíc> <den> <čas> <identifikace PC v síti>
<název démona>[<pid démona>]: <zpráva>
```

Listing 2.2: Příklad události (viz `auth.log`)

```
Nov  3 10:44:49 locust sshd[6905]: Failed password
for root from 221.229.172.76 port 60763 ssh2
```

2.1.1.2 E-mailové logy poštovního systému Postfix

„Umístění a název tohoto logu jsou definovány v konfiguračním souboru syslog.conf. Zpravidla to bývá buď /var/log/maillog, nebo /var/log/mail“. [4]

Tento log zaznamenává údaje o příchozí a odchozí e-mailové komunikaci. Zde se dají zjistit pokusy o rozesílání nebo doručování spamu apod.

Listing 2.3: Formát události e-mailového logu poštovního systému Postfix [4]

```
<měsíc> <den> <čas> <identifikace PC v síti>
<název démona>[<pid démona>]: <zpráva>
```

Listing 2.4: Příklad události doslova převzaný z [4]

```
Jan 10 16:34:11 robin postfix/smtpd[2175]: connect
from gmmr1.centrum.cz[46.255.225.252]
```

2.1.1.3 Chybové logy webového serveru Apache

Jde vůbec o nejdůležitější logovací soubor serveru Apache. Na toto místo bude Apache httpd ukládat diagnózy a všechny chyby, které se objeví při zpracovávání požadavku na serveru. Lze nastavit míru severity, která určuje, jaké typy událostí se budou logovat, například *error*, *warn*, *debug* atd. [13]

Listing 2.5: Formát události z chybového logu serveru Apache [13]

```
[<datum a čas>] [<severita hlášené chyby>] [client
<IP adresa klienta, který chybu vyvolal>] [<zpráva>]
```

Listing 2.6: Příklad události (viz `apache-access.log`)

```
[Sun Oct 30 06:26:24 2016] [error] [client 91.236.75.4]
File does not exist: /var/www/reader
```

2.1.1.4 Přístupové logy webového serveru Apache

Do tohoto logu se zaznamenávají všechny požadavky, který na Apache server přijdou. Nutno podotknout, že přesný formát logu je rozsáhle konfigurovatelný, a proto uvádím pouze jeho výchozí podobu, která odpovídá *Common Logfile Format*. [13]

Listing 2.7: Formát události z přístupového logu serveru Apache [14]

```
<vzdálená IP adresa nebo jméno hostitele> <jméno
vzdáleného uživatele> <jméno autentizovaného
uživatele> [<čas přijetí požadavku>] "<1. řádka
zprávy>" <finální návratový http kód>
<velikost odpovědi v bajtech>
```

Listing 2.8: Příklad události (viz `apache-access.log`)

```
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700]
"GET /apache/pb.gif HTTP/1.0" 200 2326
```

2.1.1.5 Audit logy webového aplikačního firewallu

ModSecurity je oblíbený *open-source* webový aplikační *firewall*, který může být nainstalován na nejpoužívanější servery jako jsou Apache, NGINX nebo IIS. Pokud přijde na server požadavek, který bude tímto *firewallem* vyhodnocen jako podezřelý, bude zalogován jak do chybového logu serveru, tak do audit logu tohoto *firewallu*. Jak demonstruje formát audit logu, v tomto souboru je událost popsána velice detailně. [15]

Listing 2.9: Formát události z audit logu [16]

```
<jméno hostitele, nebo IP adresa> <IP adresa>
<jméno vzdáleného uživatele> <jméno lokálního
uživatele> [<čas přijetí>] "<1. řádka zprávy>"
<návratový kód odpovědi> <počet odeslaných bytů>
<referenční informace> <user-agent informace>
<ID transakce> <ID sezení> <relativní cesta k audit
logu> <offset audit logu> <velikost audit logu>
<MD5 hash audit logu>
```

Listing 2.10: Příklad události (viz `modsec_audit.log`)

```
www.multikemp.cz 127.0.0.1 - - [30/Oct/2016:06:26:50
+0100] "GET / HTTP/1.1" 403 202 "-" "-"
WBWEmlOn4fAAAFglO-EAAAAB "-" /20161030/20161030-0626/
20161030-0626 50-WBWEmlOn4fAAAFglO-\\EAAAAB
0 1215 md5:0d9ce155d7572a45b395ff32569326ec
```

2.1.1.6 Fail2ban logy

Fail2ban je zabezpečovací nástroj, který na určitou dobu zablokuje podezřelé IP adresy. Uživatel se stane podezřelým, pokud poruší nějaká pravidla, například překročení maximálního počtu neúspěšných přihlášení za časový interval. V logu tohoto nástroje lze nalézt stopy po *brute-force* útoku. [17]

Listing 2.11: Formát události z fail2ban logu [17]

```
<rok>-<měsíc>-<den> <hodiny>:<minuty>:<sekundy>,
<milisekundy> fail2ban.<komponenta>:
<severita události> <zpráva>
```

Listing 2.12: Příklad události (viz `fail2ban.log`)

```
2016-10-30 06:06:45,397 fail2ban.actions: WARNING
[ssh] Ban 221.229.172.75
```

2.1.1.7 Logy správce balíčků pro Debian

Balíčkovací systém Debianu vytváří log, který obsahuje informace o instalacích softwarových balíčků. Tento log lze využít ke zjištění, zda nebyl nějaký software nainstalován neoprávněně. [18]

Listing 2.13: Formát události z logu správce balíčků pro Debian [18]

```
<rok>-<měsíc>-<den> <hodiny>:<minuty>:<sekundy>
<typ události> <zpráva>
```

Listing 2.14: Příklad události (viz `dpkg.log`)

```
2016-09-05 10:05:21 status
installed libtiff4:amd64 3.9.6-11+deb7u1
```

2.1.2 Analýza využitelnosti, co je potřeba ke hledání korelací

Jak můžete vidět, některé události jsou jen velmi stručné, jiné samy o sobě poskytují tolik informací, že není potřeba hledat další korelující události.

Většina logovacích nástrojů umožňuje měnit formát a množství evidovaných informací pomocí logovacích stupňů. Není vhodné rovnou nastavit nejdetailnější stupeň logování, protože tím často nezískáte více cenných informací, jen několikanásobně zvětšíte velikost, kterou vám budou logy zabírat na disku. [2]

Pokud nějaká společnost potřebuje provádět forenzní analýzu, může logy předzpracovávat, normalizovat. Stačí znát formát svých logů, vědět, kde se které entity nacházejí, a poté je možné všechny typy logů namapovat na jeden společný formát. Pro více informací o normalizaci logů doporučuji [2] kapitola 4: „Deciding What to Capture and How to Do It“

Já se ovšem snažím navrhnout univerzální nástroj, proto si nemohu nějakou normalizaci dovolit. Nevím, jaké logy budu zpracovávat, natož co v nich je důležité a co je možné vynechat. Ano, napadla mě možnost, že bych donutil uživatele specifikovat, jak převést jeho formát logu na standardizovaný CLF formát. Tato cesta by ovšem znamenala, že konverzí nějakého logu bych přišel o potenciálně důležité informace. Pokud bych naopak definoval nějaký velice obecný formát, který by se snažil pokrýt co nejvíce elementů, které se mohou v logovacích souborech objevit, po konverzi by bylo mnoho elementů nedefinovaných, nemluvě o velikosti takového logu.

Budu tedy mít logy v různém formátu a s informacemi, které spolu vůbec nemusí souviset. Lze vůbec odhadnout, co lze očekávat v každém logu a co je nezbytné pro hledání korelací?

```
1  <měsíc> <den> <čas> <identifikace PC v síti> <název démona>[<pid démona>]: <zpráva>
2  Nov 3 10:44:49 locust sshd[6905]: Failed password for root from 221.229.172.76 port 60763 ssh2
3
4  <měsíc> <den> <čas> <identifikace PC v síti> <název démona>[<pid démona>]: <zpráva>
5  Jan 10 16:34:11 robin postfix/smtpd[2175]: connect from gmmr1.centrum.cz[46.255.225.252]
6
7  [<datum a čas>] [<severita hlášené chyby>] [<IP adresa klienta, který chybu vyvolal>] [<zpráva>]
8  [Sun Oct 30 06:26:24 2016] [error] [client 91.236.75.4] File does not exist: /var/www/reader
9
10 <vzdálená IP adresa nebo jméno hostitele> <jméno vzdáleného uživatele> <jméno autentizovaného uživatele>
11 [<čas přijetí požadavku>] "<1. řádka zprávy>"<finální návratový http kód> <velikost odpovědi v bytech>
12 127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0"200 2326
13
14 <jméno hostitele, nebo IP adresa> <IP adresa> <jméno vzdáleného uživatele> <jméno lokálního uživatele>
15 [<čas přijetí>] ”<1. řádka zprávy>” <návratový kód odpovědi> <počet odeslaných bytů>
16 <referenční informace> <user-agent informace> <ID transakce> <ID sezení> <relativní cesta k audit logu>
17 <offset audit logu> <velikost audit logu> <MD5 hash audit logu>
18 www.multikemp.cz 127.0.0.1 - - [30/Oct/2016:06:26:50 +0100] "GET / HTTP/1.1"403 202 "-""-"WBWEmlOn4fAAAF
19
20 <rok>-<měsíc>-<den> <hodiny>:<minuty><sekundy>,<milisekundy> fail2ban.<komponenta>: <severita události>
21 2016-10-30 06:06:45,397 fail2ban.actions: WARNING [ssh] Ban 221.229.172.75
22
23 <rok>-<měsíc>-<den> <hodiny>:<minuty><sekundy> <typ události> <zpráva>
24 2016-09-05 10:05:21 status installed libtiff4:amd64 3.9.6-11+deb7u1
```

Obrázek 2.1: Podobné entity napříč různými typy logů

Na obrázku 2.1 můžete vidět příklady všech výše zmíněných logů. Snažil jsem se nalézt entity, které se nachází v každém typu logu. Došel jsem k závěru, že v drtivé většině logů bude informace, kdy k události došlo. Tyto údaje jsou vyznačené modře. Dále je velice pravděpodobné, že v logu bude uvedeno, kdo danou událost vyvolal, popř. koho se týká. Touto informací ovšem může být IP adresa, název hostitele nebo uživatelské jméno. Navíc je tento údaj jen výjimečně povinným elementem (vyznačeno červeně). Pokud budeme mít štěstí, bude součástí popisu události (vyznačeno oranžově). Entita „kdo“ vyžaduje chápání kontextu konkrétního logu, a proto je lepší

ji zařadit spíše do skupiny elementů, které lze získat, ale obecně nejsou nezbytné k hledání korelací.

Pokud budu hledat nějaké korelace s časovými filtry nebo podmínkou, že události musí nastat v nějakém časovém intervalu, potřebuji získat z událostí *timestamp*. To je zřejmě jediné, co k hledání pokročilých korelací potřebuji. O tom, jakým způsobem zajistím získání času z libovolné události, se rozepíšu v návrhu své aplikace.

2.2 Srovnání existujících nástrojů pro práci s logy

Při vybírání vhodného nástroje pro analýzu logů jsem narazil na mnoho hotových řešení s různým zaměřením, například:

- XpoLog Center – Komplexní aplikace pro log management[1]. Dokáže přes SSH protokol agregovat logy jakéhokoliv typu. Pro práci s těmito logy podporuje například: hlášení nalezených událostí, monitorování, hledání korelací, sledování transakcí, ... [19]
- Splunk – Jedná se o komerční nadstavbu OSSEC, o kterém bude řeč později. Jeho hlavním přínosem je širší množina podporovaných formátů logů, v reálném čase vytvářené přehledy aktivit a široce nastavitelné upozorňování. [1]
- ClickTracks Optimizer – Slouží k analýze aktivit na webových stránkách. [20]
- Event Log Explorer – Robustní aplikace pro interaktivní zkoumání Windows událostí. [3]
- Log Parser – CLI nástroj od Microsoftu, který umožňuje vytvářet SQL dotazy nad širokým spektrem dat. Například *Event Logs* (soubory s přípomou `.evt`, popřípadě `.evtx`), které vytváří Windows. [3]

První tři jsou dostupné pod komerční licencí, popřípadě je dostupná trial verze. Poslední dva jsou určeny pouze pro Windows, ale má forenzní analýza bude probíhat na linuxovém serveru.

Mým cílem je vytvořit volně šiřitelný program, proto jsem se vydal spíše cestou *open-source* nástrojů.

[1] Obecný termín řízení logů zahrnuje činnosti: sběr, archivaci, agregaci a analýzu logů, popřípadě tvorbu upozornění.

Open-source nástroje sice často nejsou tak komplexní, ale přináší jiné výhody. Nulová pořizovací cena, jednoduchá přenositelnost a zejména možnost reálně upravit zdrojový kód. V případě problémů lze nahlédnout přímo do implementace. [3]

Při dalším hledání jsem narazil na nástroje, které nějakým způsobem pracují s logy, ale nepokrývají požadovanou funkcionalitu (neumožňují hledání korelací):

- rconvlog & convlog – Převádí formáty logů na CLF formát logu. [21]
- Chainsaw – GUI prohlížeč logů pro Log4j[2]. [22]
- Sematext Docker Agent – Jde o agenta, který má za úkol automatický sběr událostí. [23]
- Snare & syslog – Snare umožňuje, aby Windows události mohly být poslány přes syslog[3]. [1]

Pro statickou analýzu logů by jistě šlo využít nástrojů jako grep, sed, awk, head, tail. Tyto utility jsou standardní součástí většiny linuxových distribucí a jsou hojně využívány serverovými administrátory, kterým slouží jako stavební kameny pro tvorbu skriptů. Jejich použití vyžaduje ovšem příliš mnoho manuálního práce, hodí se na vytažení konkrétního údaje, nikoliv pro hledání korelací. [1]

Statické nástroje byly zavrhnuty, protože práce s nimi je komplikovaná při snaze hledání korelací z více logovacích souborů. Vývoj aplikace touto cestou by de facto znamenal začít na tzv. „zelené louce“. Výhodou by byla vysoká flexibilita, ale za cenu velké pracnosti.

Oproti statickým existují automatizované nástoje dvou typů: [5]

1. Offline monitorující – při spuštění začnou hledat chybové vzory, pomocí předem definovaných pravidel. Pokud je nějaký vzor nalezen, často dokážou také zareagovat podle specifikace v pravidlech pro daný vzor. Tyto nástroje mohou být spuštěny několikrát denně, jejich nevýhodou ovšem je, že preventivní reakce na sled událostí nemohou být provedeny ve správný čas.
2. Online monitorující – odstraňují předchozí omezení a umožňují průběžné sledování logů. Dokážou reagovat na události v reálném čase, ignorovat duplicitní události nebo dynamicky měnit pravidla, která definují podezřelé aktivity.

[2]Logovací *framework* pro Javu.

[3]Protokol sloužící k přeposílání zpráv z logů po síti.

Rozhodl jsem se tedy částečně využít nějakého již funkčního řešení. Ze zadání plyne, že je dostačující si vybrat nějaký offline automatizovaný nástroj, který dokáže analyzovat logy. Následuje seznam nejlepších nástrojů, ze kterých jsem vybral vítěze Simple Event Correlator.

2.2.1 Webalizer

Oblíbený nástroj pro analýzu serverových logů. [24, 25]

+ Vytváří grafické přehledy v HTML formátu, které je poté možné sledovat v jakémkoliv standardním webovém prohlížeči.

+ Je napsaný v jazyce C, což jej činí velice rychlým. Dokáže zpracovat cca 70 000 událostí za 1 vteřinu.

+ Zvládá mnoho formátů logů.

+ Podporuje asi 30 světových jazyků.

+ Není potřeba posílat informace o svém webu jiným serverům, jako tomu je například u konkurečního řešení Google Analytics.

− Určený spíše pro sledování aktivity uživatelů na webu.

− Jen kolem sta konfiguračních parametrů, které lze použít v konfiguračním souboru. To znamená, že tato aplikace neumožňuje dostatečně detailně definovat, co je podezřelá aktivita.

2.2.2 LogWatch

Tento nástroj se snaží události z logů interpretovat a kompletovat. Na základě této analýzy posílá e-mailem přehled důležitých událostí a příslušné statistiky. [26]

+ Definice podezřelých událostí a jejich interpretace zajišťují skripty napsané v Perlu, které tvoří konfiguraci tohoto nástroje.

+ Lze popsat regulárními výrazy, jaké události mají být ignorovány.

+ Je možné vymezit interval, který definuje, z jaké doby se mají události zpracovávat.

− Komplexnější definice podezřelých aktivit je nutné definovat v Perlu. LogWatch neposkytuje pokročilou konfiguraci bez znalosti programovacích jazyků.

2.2.3 ELK Stack

ELK Stack je kolekce tří *open-source* produktů: Elasticsearch, Logstash a Kibana. Logstash je nástroj, který slouží ke sběru a parsování logů. Dokáže sbírat logy z mnoha zdrojů, transformovat nebo filtrovat pomocí tzv. *Grok patternů* [28], které mi byly inspirací při tvorbě aliasů v mnou navrženém

nástroji. Elasticsearch je NoSQL[4] databáze, která umožňuje v datech vyhledávat a analyzovat je. Kibana je vizualizační vrstva, dovolující vykreslovat různé grafy apod. [29, 27]

+ Tento produkt vytlačil dřívějšího leadera Splunk, protože je *open-source* a přitom dosahuje stejných kvalit.
+ Široká možnost konfigurace.
+ *Grok patterny.*
+ Podpora NoSQL databáze.
+ Předpřipravené GUI.
− Těžkopádný nástroj.

2.2.4 OSSEC

OSSEC je výborný nástroj pro analýzu logů, jehož hlavní silou je generování upozornění, která dokáže zadržet a uložit. Jeho cílem je detekovat hrozby a zajistit, že budou včas rozpoznány a provede se příslušná akce. [1]

+ Přednastavená pravidla, možnost vlastních pravidel v XML.
+ Upozorňování v reálném čase.
+ Webové uživatelské rozhraní.
+ Odlehčený nástroj.
+ Podpora MySQL[5] a PostgreSQL[6] pro ukládání hlášených upozornění.
+ Funguje téměř ve všech OS.
− Ve výchozím nastavení zadržuje jen logy, které vyvolaly upozornění.
− Je nutné vyhradit volné místo na disku nebo použít databázi.
− Cílem je zvýšit bezpečnost, ne detailní hledání korelací. Patří mezi takzvané HIDS.

2.2.5 OSSIM

Open Source Security Information Management je kombinace více aplikací. Jednou z nich je již zmiňovaný OSSEC, oproti kterému se může OSSIM pyšnit vylepšeným bezpečnostním management systémem. Další výhodou je integrace programu Snort pro hledání korelací. Jde o tzv. SIEM nástroj. [1]

Podle mého názoru ovšem toto vylepšení nevyváží nárůst složitosti SIEM nástroje, proto bych dal přednost programu OSSEC.

[4] Alternativa relačních databází, která nevyžaduje definovat schémata. Proto jsou NoSQL databáze vhodnější pro ukládání nestrukturovaných dat, která se neukládají do řádků a sloupců, ale stylem klíč – hodnota.

[5] *Open-source* relační databáze.

[6] Objektově-relační databáze.

2.2.6 Tripwire

Tripwire naopak poskytuje téměř stejnou funkcionalitu, jako OSSEC. Je dostupný jak pod *open-source*, tak pod enterprise edicí. Pokud ovšem budu porovnávat jen jeho *open-source* verzi, pak se jedná špíše o slabší alternativu OSSEC, protože neexistuje žádná podpora pro Windows, a navíc Tripwire neumožňuje upozorňování v reálném čase. [30]

2.2.7 OpenNMS

OpenNMS je aplikace sloužící především k monitorování sítě. Provádí sběr událostí a generuje upozornění. Na tomto nástroji mne zaujalo, že vygenerované upozornění může spustit další hledání korelací. OpenNMS jsem ovšem zavrhnul ve chvíli, kdy jsem zjistil, že je napsaný v Javě. Nefunkčním požadavkem této práce je vyhnout se nutnosti instalovat virtuální stroj na server. [31]

2.2.8 LogHound

LogHound hledá opakující se vzory řádek v množině logovacích souborů. [1] Tento nástroj považuje každou řádku logu za transakci, která se skládá z elementů. Množinu elementů považuje za frekventovanou, pokud alespoň N transakcí obsahuje všechny tyto elementy. Konkrétní N lze definovat pomocí argumentů programu. [32]

LogHound mě přivedl na stránky autora (Risto Vaarandi), který vytvořil mnoho utilit pro práci s logy.

+ Jednoduchý nástroj napsaný v jazyce C.

+ Podpora regulárních výrazů pro substituce nebo definice oddělovačů elementů.

− Pouze jednoduchá analýza logů bez možnosti hledání složitějších korelací.

− Není možné definovat vlastní reakci na úspěšné nalezení frekventované množiny.

2.2.9 SEC

Simple Event Correlator je dalším nástrojem od pana Risto Vaarandi. Jak již název napovídá, SEC je program pro hledání pokročilých korelací. Může být využit k monitorování logů, řízení sítě nebo bezpečností forenzní analýze. [8, 5]

+ Dokáže číst data z mnoha zdrojů: ze souborů, ze standardního vstupu až po pojmenované roury.

+ Používá konfigurační soubor pro definici podezřelých aktivit. Tyto podezřelé aktivity dokáže popsat pomocí mnoha předpřipravených typů pravidel.

+ Pro administraci konfiguračního souboru není potřeba znalost programovacího jazyka.

+ Velice mocná je možnost specifikace jak podezřelých aktivit, tak reakce na ně pomocí Perlu.

+ Podporuje regulární výrazy, kterými lze popsat mnoho hledaných vzorů.

+ Umožňuje definovat kontexty, které dokážou spojit jednoduché regulární výrazy nebo definovat další události, které musí nastat, aby se jednalo o podezřelou aktivitu. Umožňují například přidat podmínky, ze kterého logu musí událost přijít, v jakém časovém intervalu apod.

+ Dokáže definovat, jaká reakce na podezřelou aktivitu se má provést. Obsahuje například zabudovanou možnost spouštět externí programy, např. `mail` – k odeslání upozorňovacího e-mailu správci serveru.

+ Výborná dokumentace.

+ Odlehčený nástroj.

+ Napsaný v Perlu, tedy platformově nezávislý.

− Absence GUI.

− Při offline čtení logovacích souborů nepodporuje pravidla pro práci s časem. (Nedokáže vytáhnout čas události, který je na právě zpracovávané řádce logu.)

+ Pokud tyto logy sleduje online, tedy pokud se o událostí dozví přesně ve chvíli, kdy se objeví ve sledovaném logu, dokáže s časem pracovat velmi dobře.

2.3 Představení vybraného nástroje

Aby bylo možné popisovat podezřelé aktivity, je nutné se nejdříve stručně seznámit s tvorbou pravidel v nástroji Simple Event Correlator.

Při forenzní analýze se prochází všechny řádky logů, kde každá řádka symbolizuje jednu událost. SEC se pro každou událost podívá na všechna pravidla a snaží se nalézt shodu, nějaké pravidlo, které právě na takovou událost čekalo.

Pravidla se zapisují do konfiguračního souboru a oddělují se prázdnými řádkami, bílými znaky nebo pomocí komentářů (začínají symbolem „#“). To znamená, že se tyto elementy uvnitř samotného pravidla vyskytovat

nemohou. Pokud je nějaká řádka pravidla příliš dlouhá, lze použít symbol „\“ a pokračovat na další řádce. [8]

Každé pravidlo obsahuje informaci, jakého je typu. SEC má zabudována například tato pravidla: [10, 8]

- Single – Pokud nastane shoda, okamžitě se provede akce (reakce na událost), která je specifikována v rámci tohoto pravidla. Jednoduché pravidlo, které nehledá žádné korelace.
- Suppress – Zde naopak pokud shoda nastane, tak se žádná akce neprovede. Hledání shody bude ovšem pokračovat procházením ostatních pravidel. Ve výchozím nastavení totiž jakmile SEC nalezne první shodu, pak už další v pravidlech nehledá. Toto pravidlo může sloužit k filtraci událostí.
- Calendar – Tento typ je specifický, protože jako jediný ignoruje vstupní logy. Umožňuje naplánovat nějakou akci na konkrétní čas.
- SingleWithScript – Při shodě se spustí definovaný skript a po jeho doběhnutí se vykoná jedna ze dvou akcí, v závislosti na výsledku tohoto skriptu. Na rozdíl od tohoto a předchozích, následující pravidla již dokážou hledat korelace.
- SingleWithSuppress – Toto pravidlo se chová podobně jako Single, ovšem při shodě také zahájí hledání korelací po dobu T sekund, kde T lze definovat. Korelací může být v tomto případě pouze stejná událost, která přijde později. Všechny takovéto události budou ignorovány po dobu T sekund. Teprve poté může shoda znovu nastat.
- SingleWithThreshold – Provede akci, pokud dojde k N opakovaným událostem během nějaké doby, viz 2.15. Hodnota N je specifikována polem `thresh`.
- EventGroup – Jde o zobecněnou verzi předchozího pravidla. Umožňuje napočítávat libovolný počet událostí různého typu v rámci společného intervalu (SEC používá termín okno – pole `window`).
- Pair – Párové pravidlo provede 1. akci, pokud je splněna 1. část a 2. akce se provede, pouze pokud událost popsaná v 2. části dorazí včas. Dobu hledání této korelace lze samozřejmě nastavit na nekonečno.

Následující pravidlo hledá ve vstupních souborech 3 neúspěšné pokusy o přihlášení pod stejnym uživatelským jménem během 1 minuty. Při shodě vypíše pole `desc` na standardní výstup.

Listing 2.15: Ukázka pravidla typu SingleWithThreshold

```
type = SingleWithThreshold
ptype = RegExp
pattern = sshd\[\d+\]: Failed .+ for (\S+) from \
[\d.]+ port \d+ ssh2
desc = Three SSH login failures within 1m for user $1
action=write -;
window = 60
thresh = 3
```

Každé pravidlo obsahuje řádky ve formátu `<klíč> = <hodnota>`, kde jako klíč lze použít pole, které daný typ pravidla podporuje. Některá pole jsou povinná, jiná volitelná.

Důležité je také pole `ptype`, které definuje, jakým způsobem bude specifikován vzor popisující shodu. SEC poskytuje například: [8]

- SubStr – V poli `pattern` bude očekáván podřetězec události, se kterou má vzniknout shoda. Tento způsob je časově efektivní, ale není možné událost volněji specifikovat.
- RegExp – Pomocí regulárních výrazů lze popsat většinu hledaných událostí. Lze zde efektivně využívat tzv. *capture groups*, viz [33].
- PerlFunc – Vzor může obsahovat funkci napsanou v jazyce Perl. Shoda nastane, pokud tato funkce vrátí *true*. Slouží k pokročilým popisům událostí tam, kde regulární výrazy nestačí.

Následující příklad byl převzán z [10].
Představte si vzor popsaný tímto regulárním výrazem:

```
(\S+) sshd\[\d+\]: Accepted.*for (\S+) from (\S+) port (\d+)\s
```

Shoda nastane, pokud se v analyzovaném logu vyskytne například tato událost:

```
Sep 16 17:46:47 spirit sshd[12307]: Accepted password for rik
from 204.176.22.9 port 59926 ssh2
```

Je důležité si uvědomit, že elementy uvnitř kulatých závorek (tzv. *capture groups*) se po shodě uloží do proměnných *$i*, kde *i* je pořadí skupiny.

- $1 = jméno hostitel (spirit)
- $2 = uživatelské jméno (rik)
- $3 = zdrojová IP adresa (204.176.22.9)
- $4 = číslo portu (59962)

Na tyto proměnné se lze v rámci pravidla odkazovat, například v popisu akce. V poli `action` lze popsat, jaká akce se má provést, pokud nastane shoda. Ve skutečnosti lze zde specifikovat seznam akcí, stačí je oddělit středníkem. SEC podporuje mnoho možností, například zápis do souboru, pojmenované roury, *socketu*, spustit externí skript, odeslat e-mail nebo poskytuje standardní CRUD operace s takzvanými kontexty. [8]

SEC není pouze sympatickým nástrojem pro hledání regulárních výrazů. Jeho hlavní silou je možnost využívat subrutiny napsané v Perlu, a to jak pro popis vzoru, tak pro popis akce. Kontexty jsou druhou doménou tohoto nástroje. Umožňují spojovat jednotlivá pravidla, a vytvářet tak pokročilá korelační schémata. Existují doporučení, jak korelační schémata uspořádat do konfiguračních souborů, viz [9].

Kontexty mohou mít časem omezenou životnost nebo mohou například vznikat při každém čtení pouze z konkrétního souboru. Časté použití kontextů je: Máme N pravidel, pro každé vytvoříme unikátní kontext, pokud došlo ke shodě. Vytvoříme ještě 1 pravidlo, jehož akce se provede, jen pokud bude zároveň aktivovaných těchto N kontextů. [8]

Mezi další nepovinná, ale užitečná pole, jako byl `context`, patří například `continue` nebo `varmap`. Jelikož účelem mé bakalářské práce není duplikovat jinak výborné manuálové stránky SEC, pro více informací vás na ně jen odkazuji [8].

Následuje ještě jedna ukázka pravidla, které popisuje podezřelou aktivitu ze zadání, kdy dojde k neoprávněnému požadavku na webovou stránku a následný pokus o autentizaci ze stejné zdrojové adresy. V implementaci jsem využil párového pravidla:

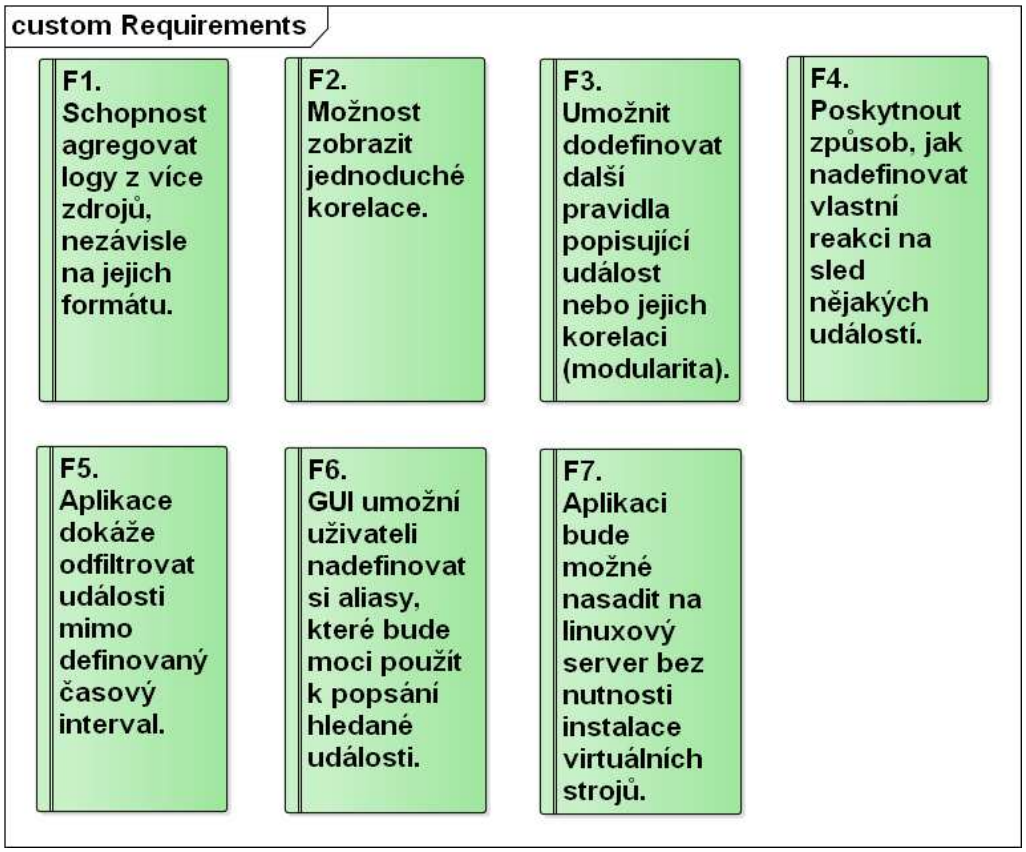
Listing 2.16: Ukázka pravidla typu Pair

```
type = Pair
ptype = RegExp
pattern = \S+ (\d.+) \S+ \S+ \[.*\] ".*" 40(?:1|3) \d+
desc = Unauthorized access to a website from \
IP adress: $1
action = none;
ptype2 = RegExp
pattern2 = (Failed|Accepted) password for (\S+) from \
$1 port (\d+) ssh2
desc2 = Correlation found from %1 port $3 -> \
$1 password for the user $2
action2 = write -;
```

Kapitola 3

Realizace

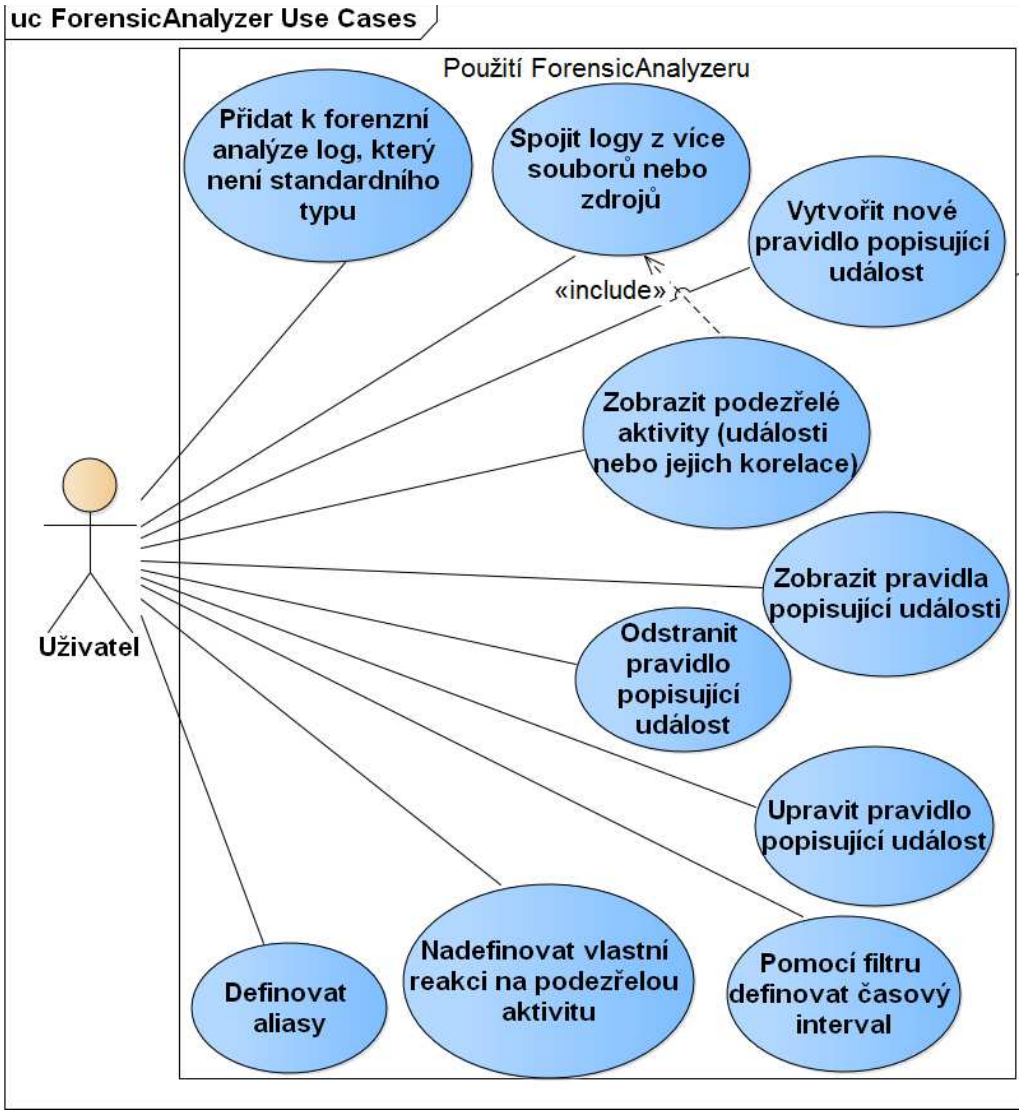
3.1 Sběr požadavků



Obrázek 3.1: Seznam požadavků

Po výběru tématu bakalářské práce jsem se pustil souběžně s rešerší do specifikace požadavků. Na obrázku 3.1 můžete vidět seznam požadavků, který jsem vytvořil v aplikaci Enterprise Architect. Po konzultaci s vedoucím práce jsem získal pár dalších požadavků, které přímo nevyplývaly ze zadání – viz funkční požadavky F4 a F6 a nefunkční požadavek F7.

3.2 Model případů užití



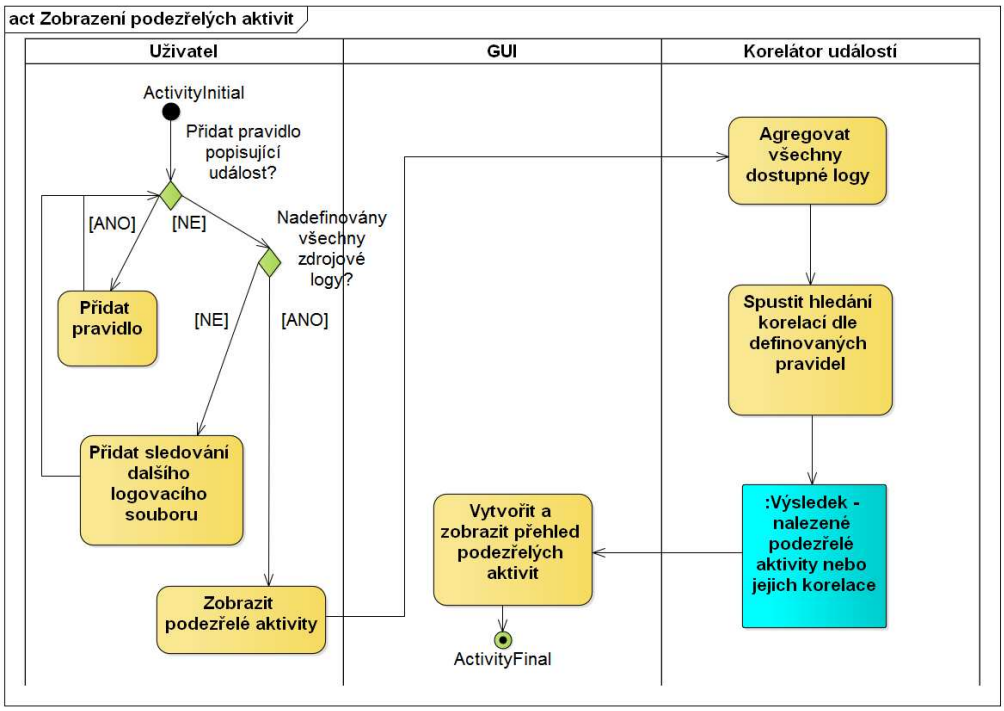
Obrázek 3.2: Model případů užití

Na obrázku 3.2 je model případů užití mého nástroje, který reflektuje seznam požadavků 3.1.

3.3 Návrh aplikace

3.3.1 Diagramy aktivit

Bylo klíčové detailně rozmyslet, co si představit pod těmi nejzajímavějšími modely případů užití. Na obrázku 3.3 je diagram aktivit popisující, jak si představuji *use case*: „Zobrazit podezřelé aktivity“



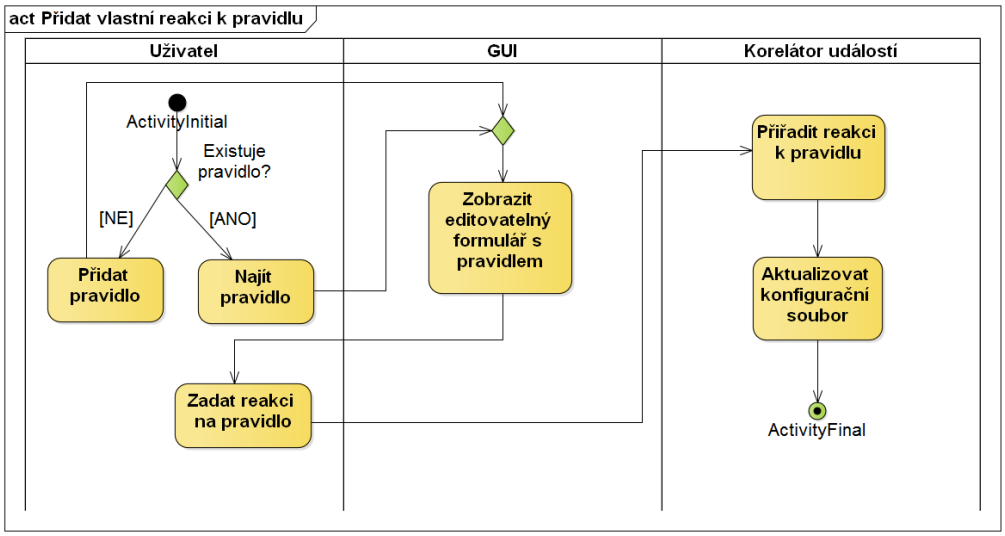
Obrázek 3.3: Zobrazení podezřelých aktivit (*activity diagram*)

Uživatel bude tedy moci nadefinovat pravidla popisující události, které chce sledovat a zadat zdrojové logy. Jakmile je všechny nadefinuje, tak další aktivitu už převezme „Korelátor událostí“, ten všechny logy spojí, spustí hledání korelací dle definovaných pravidel a vytvoří výstup, podle kterého už jen GUI zobrazí požadovaný přehled.

Můžete si všimnout, že logiku programu jsem rozdělil na dvě zóny zodpovědnosti, GUI a „Korelátor“. Učinil jsem tak, protože jsem se snažil na-

jít nějaký nástroj, který by mi pomohl, abych nemusel začínat na „zelené louce“ a mohl se více zaměřit na grafickou nadstavbu.

Obrázek 3.4 zachycuje, jakým způsobem jsem navrh realizaci požadavku F4 (Přidání reakce k pravidlu).



Obrázek 3.4: Přidání reakce k pravidlu (*activity diagram*)

Pro více diagramů týkajících se návrhu aplikace vás odkazuji na přiložené CD.

3.3.2 Plnění požadavků

Souběžně s rešeršní částí jsem se zamýšlel nad tím, jak splnit požadavek na modularitu aplikace. Jelikož jsem v té době procházel mnoho nástrojů, které používaly konfigurační soubory pro definici podezřelých aktivit, tak jsem se rozhodl je ideálně použít také. Tím jsem splnil požadavek na modularitu aplikace. Stačí jen zajistit, aby byla aplikace dostatečně konfigurovatelná a nebyla uzavřená, co se týče definovaných formátů logů nebo pravidel, které hledají podezřelé aktivity.

Po ukončení rešeršní části byl jako „Korelátor událostí“ zvolen program Simple Event Correlator, který jsem již představoval výše. Tento nástroj se perfektně hodí pro účel mé bakalářské práce, protože dokáže pokrýt požadavky F1, F2, F3 i F4.

Začal jsem tedy detailně studovat manuálové stránky tohoto nástroje. Byl jsem unešen funkcionalitou, kterou nabízí, ale potvrdil se nedostatek, který jsem již popisoval v rešeršní části (viz 2.2.9).

SEC má předpřipravená analyzační pravidla, která dokážou pracovat s časem, ovšem jen pro online monitorování logů. Při konzultaci s vedoucím práce jsme se shodli, že v praxi je ovšem daleko častější situace, kdy více zařízení loguje do sdíleného souborového systému a my bychom chtěli provádět offline analýzu pouze tam, nikoliv instalovat náš nástroj na každé zařízení.

Jak je uváděno v manuálu SEC [8]: *„Note that for event occurrence time SEC always uses the current time as returned by the time(2) system call, *not* the timestamp extracted from the event.“*. SEC tedy umožňuje offline čtení (parametr `--notail`), při kterém se ovšem nebere ohled na datum a čas, který je uveden v události, ale vezme se aktuální čas v době analýzy logů. Bylo tedy jasné, že budu muset tento nástroj lehce vylepšit, abych tento nedostatek opravil. Naplánoval jsem si, že při té příležitosti doimplementuji do SEC i požadavek F5.

Aplikace LogStash (viz ELK Stack 2.2.3) využívá takzvané *Grok patterny*, které jsem objevil v rešeršní části a byly mi inspirací k vyřešení požadavku F6, viz 4.1.

Požadavek F7 je nefunkční a byl klíčový pro zvolení technologie, ve které budu vytvářet GUI. Zavrhnul jsem .NET a Javu. Pro tvorbu GUI jsem zvolil *framework* Qt, napsaný v C++. SEC je napsaný v jazyce Perl, takže také požadavek F7 splňuje. Bude se tedy jednat o integraci C++ a Perlu.

3.4 Implementace

Implementační část měla 2 fáze:

1. Úprava SEC – výsledek jsem pojmenoval SEC+.
2. Vývoj GUI od nuly – název celkové aplikace je *Forensic Analyzer*.

3.4.1 SEC+

Z oficiálních stránek [7] jsem si stáhnul nástroj SEC v nejaktuálnější verzi 2.7.11. Nastudoval jsem jazyk Perl a zorientoval se v kódu, který obsahoval cca 13 000 řádek.

Mým cílem bylo nenásilnou cestou přidat funkcionalitu, ale přitom nenarušit původní způsob používání. Začal jsem nadefinováním vlastního přepínače `--parseEventTime`. Jak název napovídá, pokud se objeví tento přepínač, program se bude snažit získávat čas události přímo z dané řádky logu. Po zorientování v kódu jsem prošel všechna systémového volání `time()`

a tam, kde to bylo nutné, jsem modifikoval. Pokud je tedy definován přepínač `--parseEventTime`, dostanete se do mých řádek kódu, konkrétně do funkce `getEventTimeFromProcessingLine`. Jak tato funkce získá *timestamp* ze zpracovávané řádky logu?

3.4.1.1 Získávání timestampu z libovolného logu

SEC zadává vstupní soubory pomocí přepínače `--input=<cesta k souboru>[=<kontext>]`. Kontext je nepovinný, může to být libovolný identifikátor, na který se poté mohou odkazovat pravidla v konfiguraci (při každém čtení se totiž vytvoří interní kontext s tímto názvem). [8] Rozhodl jsem se využít této možnosti a přidat kontextu další význam – bude zároveň vyjadřovat název proměnné, jejíž hodnota popisuje, jak získat datum a čas z řádky logu tohoto typu.

Tyto názvy proměnných a jejich hodnoty budou v mnou definovaném konfiguračním souboru. Ve výchozím nastavení se jedná o soubor `res/logFormats.conf` a při spuštění SEC+ již předpokládá, že existuje a obsahuje všechny potřebné proměnné (kontexty). Pokud tomu tak není, program zareaguje chybovou hláškou.

Jako hodnotu proměnné očekávám regulární výraz, podle kterého dokážu z každé události získat datum a čas. Tento regulární výraz navíc musí obsahovat tzv. *named capture groups*. [33] Detailní popis požadovaných skupin zde 4.2.2.2. Vložením `?<var>` do regulárního výrazu přikážete Perlu, aby v případě nalezené shody uložil tuto skupinu do proměnné `$+{var}`. Aplikace spoléhá na naplnění těchto proměnných, vypočítává z nich *timestamp* pro každou událost.

Pro práci s časem jsem přidal knihovnu Date::Parse. [35] Perl obsahuje i sofistikovanější knihovny, které dokážou odhadnout z řetězce datum, ale tato cesta není stoprocentní a jako univerzálnější řešení mi přišlo donutit uživatele, aby mi sdělil, co kde najdu. Také je možné, že uživatel přidá svůj formát logu, ve kterém je na začátku datum a až na konci je čas apod. Z takové řádky by ani ty nejlepší knihovny nedokázaly jednoznačně odhadnout datum a čas.

Zpátky k funkci `getEventTimeFromProcessingLine`. Při čtení řádky vím, ze kterého je souboru a také, jaký je kontext. Pokud ještě neznám regulární výraz pro tento kontext, najdu ho v konfiguračním souboru. Poté už snadno získám datum a čas, který vracím jako *timestamp*.

3.4.1.2 Dodržování čtení událostí dle jejich timestampu

Tímto mělo být jednoduché rozšíření dovršeno. Při testování jsem však narazil na nečekaný problém. Pokud má SEC definováno více vstupních logů a čte je offline, pak se snaží být maximálně spravedlivý. To se projevuje tak, že pokud má dva soubory, tak vždy vezme nejprve 1. řádku z 1. logu, poté 1. řádku z 2. logu, 2. řádku z 1. logu atd. Já bych ovšem potřeboval, aby události bral v pořadí, které respektuje čas jednotlivých událostí.

První řešení, které mě napadlo, bylo předzpracovat všechny události – vytvořit jeden sloučený vstupní log, ve kterém jsou události seřazeny. Tímto bych ovšem přišel o již zabudovanou možnost psát pravidla, která se použijí, jen pokud se čte řádka z konkrétního souboru.

Jelikož lze předpokládat, že jednotlivé logy jsou seřazené podle času, stačí se jen podívat na první nezpracovanou řádku v každém logu a najít nejaktuálnější. Tuto logiku najdete ve funkci `readRecentLineToReadbuffer`. Použil jsem tedy myšlenku slévání, podobnou jaká je v algoritmu MergeSort.

Poslední zásadní část, kterou jsem musel modifikovat, je právě ukládání řádky do *bufferu*, se kterým SEC pracuje. SEC čte z každého zdroje fixní počet bajtů. Následně se snaží v *bufferu* nalézt *newline*, což má mimo jiné neočekáváné chování v situaci, kdy soubor obsahuje pouze jedinou událost bez *newline*.

Jakmile naleznu událost, která má z časového hlediska přijít nejdříve, uložím ji do *bufferu* a na její místo posunu následující událost ze stejného souboru. Stačí mi tedy číst po řádcích, ne po bajtech. To byla modifikace, kterou jsem musel provést. Celá tato logika bude aplikována, pokud bude SEC+ puštěn také s přepínačem `--parsedTimeOrder`.

Vytvořil jsem další oddělený přepínač, protože jsem si vědom toho, že čtení po řádcích je časově méně efektivní a pokud uživatel potřebuje pouze získávat datum a čas z událostí a přitom nepotřebuje dodržovat jejich pořadí, potom tento parametr nebude specifikovat a dosáhne tak vyšší efektivity.

Přidáním přepínače `--dontParseEventTime` je možné zrušit případné použití přepínačů `--parseEventTime` a `--parsedTimeOrder`.

3.4.1.3 Přidání časových filtrů

Následovalo přidání posledních dvou přepínačů: `--timeFilterFrom` a `--timeFilterTo`, které slouží k zahození událostí, které nastaly dříve, resp. později než zadaný *timestamp*. *Timestamp* vypočítám z hodnoty tohoto přepínače, který je pro oba filtry očekáván ve formátu:
`<měsíc>/<den>[/<rok>]:<hodin>:<minut>:<vteřin>`

Kód jsem doplnil o validace, komentáře a také jsem všechny své hlavní funkce anotoval svým jménem, aby bylo jasně poznat, co je mým dílem. Díky tomu, že jsem průběžně verzoval, jsem mohl přidat do přílohy *git-diff*, který přehledně zobrazuje rozdíly naklonovaného oficiálního repozitáře SEC a mojí verze SEC+.

3.4.2 Forensic Analyzer

Hlavním cílem bylo vytvořit grafickou nadstavbu SEC+, která by umožňovala v GUI generovat potřebné parametry, spravovat vstupní logovací soubory, editovat konfigurační soubory atd.

GUI jsem se snažil navrhnout tak, aby bylo prakticky využitelné, proto jsem se rozhodl, že si *Forensic Analyzer* bude i po vypnutí pamatovat již definované vstupní soubory, konfigurační soubory a aliasy. Při spuštění programu se všechny definované aliasy načtou a jsou připraveny k použití.

Jak vstupní logovací soubory, tak konfigurační soubory, mohou být přidány nebo odebrány z *Forensic Analyzeru*. Tuto možnost jsem do nástroje přidal, aby uživatel nemusel vždy znovu definovat, který log chce použít a jaký je formát obsažených událostí. Z tohoto důvodu je v nástroji možnost aktivovat a deaktivovat soubor. Pokud je soubor deaktivovaný, bude při spuštění analýzy ignorován. O dynamické přidávání a odebírání souborů se stará třída `QtFileController`.

Pro modifikaci vstupního logovacího souboru jsem navrh dialog, který představuje třída `LogFileDialog`. V tomto dialogu lze změnit regulární výraz popisující formát událostí.

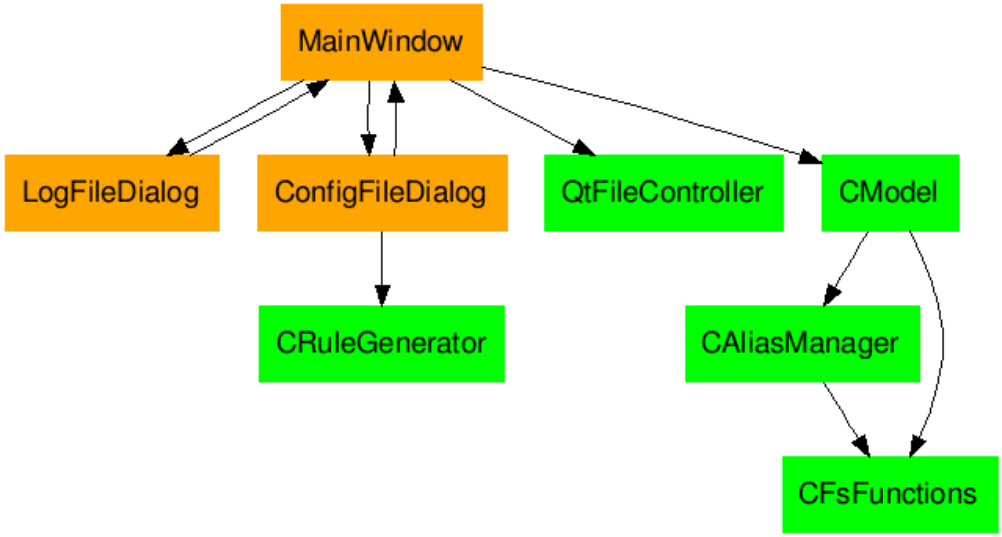
Protože formát může být velice komplikovaný a nepřeledný, je v tomto dialogu také možnost používat a editovat aliasy (více viz 4.1).

Třída `CAliasManager` spravuje také druhý seznam aliasů, který můžete nalézt při modifikaci konfiguračního souboru – tento dialog je implementován třídou `ConfigFileDialog`. Na rozdíl od dialogu pro správu logů zde lze přímo editovat obsah souboru.

Rozhodl jsem se, nad rámec sjednaného rozsahu, přidat generování pravidel, které SEC poskytuje. Uživatel si zvolí typ pravidla a *Forensic Analyzer* dokáže následně vygenerovat všechna povinná pole, která musí být definována pro daný typ pravidla.

Hlavní okno aplikace jsem navrhnul responzivně a slouží k přehledu aktivních souborů a parametrů pro SEC+, k samotnému spuštění analýzy a k zobrazení nalezených výsledků.

Poslední zmíněná, ovšem ne co do důležitosti, je třída `CModel`, která spravuje zejména aktivní a neaktivní soubory a generuje parametry pro SEC+.



Obrázek 3.5: Zjednodušený *class diagram*

Na obrázku 3.5 můžete vidět zjednodušený *class diagram Forensic Analyzeru*. Tento graf zobrazuje zmiňované třídy a jejich vzájemné závislosti. Třídy označené oranžově jsou vázány na konkrétní formuláře, které jsem vytvořil ve vývojovém prostředí Qt Creator. Zeleně vyznačené jsou naopak klasické C++ třídy.

Pro detailnější popis implementace nebo diagramy vřele doporučuji vygenerovat programátorskou příručku pomocí nástroje Doxygen (viz 4.1.3).

Za zmínku stojí, jak jsem vyřešil integraci *Forensic Analyzeru* a SEC+. Jakmile uživatel spustí analýzu, *Forensic Analyzer* vygeneruje *shellový* příkaz pro spuštění SEC+ se správnými parametry. Poté vytvoří nový proces pro SEC+ a přesměruje se jeho standardní výstup na zápisový konec nepojmenované roury. Na čtecím konci roury je připojen *Forensic Analyzer*, který data zobrazuje v GUI. Využil jsem knihovní funkci POSIXu – `popen`. [34]

Kapitola 4

Instalační a uživatelská příručka

4.1 Instalační příručka

4.1.1 Požadavky

Aplikace *Forensic Analyzer* je určená pro operační systém Linux. Pro úspěšnou instalaci je potřeba mít nainstalované tyto balíčky:

- make
- g++
- qt-sdk
- doxygen
- graphviz

Balíček instalujte příkazem: `sudo apt-get install <název balíčku>`

4.1.2 Instalace

Pro instalaci otevřete příkazový řádek v adresáři se souborem `Makefile` a zadejte příkaz `make`. Součástí instalace je také snaha přiřadit práva na spouštění přidružené aplikace SEC+ s relativní adresou `./res/sec+`. Tento soubor, prosím, nemažte ani jej nepřesouvejte na jiné umístění.

4.1.3 Možnosti Makefilu

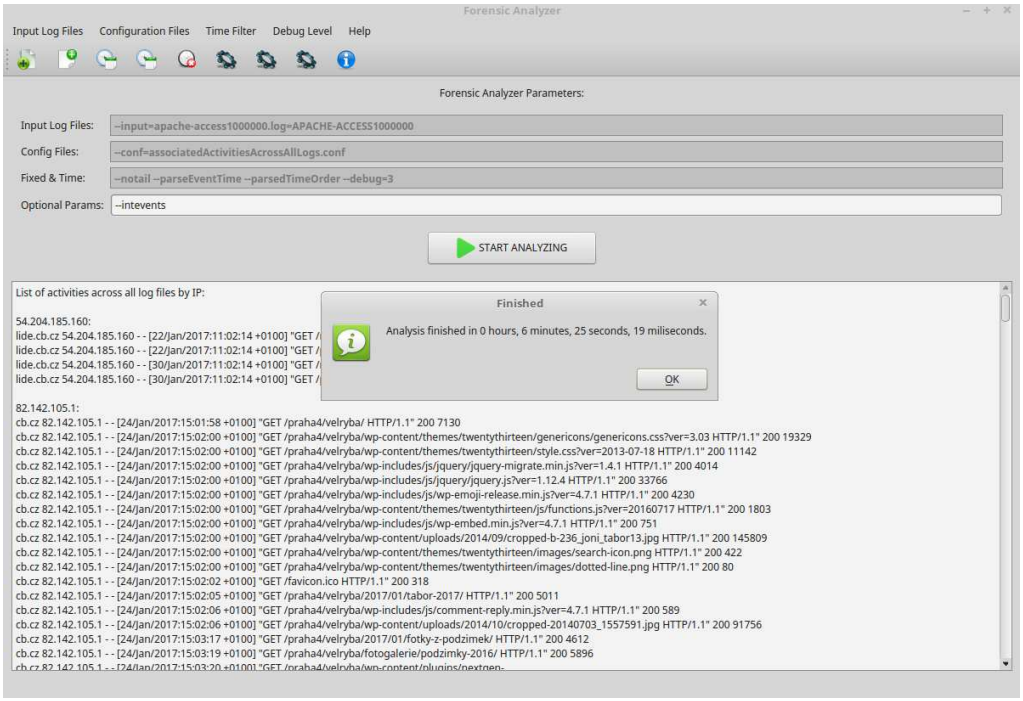
`Makefile` má definováno více zajímavých návěští, které můžete připsat za `make`:

- all – Výchozí návěští, pokud není explitně specifikováno jinak. Nainstaluje aplikaci a vygeneruje programátorskou příručku.
- run – Spustí aplikaci. Při spuštění již nejsou potřeba administrátorská práva.
- doc – Vygeneruje programátorskou příručku do složky `doc`. Pro spuštění HTML verze příručky otevřete soubor `doc/index.html`.
- count – Počítá počet řádků kódu všech souborů s příponami `.h` a `.cpp`
- clean – Odinstaluje aplikaci a odstraní programátorskou příručku.

Př.: Pouze pro vygenerování programátorské příručky zadejte: `make doc`

4.2 Uživatelská příručka

4.2.1 Hlavní obrazovka



Obrázek 4.1: Hlavní okno aplikace

Po spuštění *Forensic Analyzeru* se dostanete na hlavní obrazovku, viz 4.1.

Tato hlavní obrazovka je rozdělena na dvě části:

1. Část s parametry, které definují, jak bude forenzní analýza probíhat.
2. Okno pro zobrazení výsledků, ať už čistě pro zobrazení nalezených korelací nebo detailnější záznam průběhu analýzy.

Až budete mít všechny parametry specifikované, stiskněte tlačítko `START ANALYZING`, popřípadě klávesovou zkratku `Ctrl + R`. Následně proběhne forenzní analýza zadaných logů dle definovaných konfiguračních souborů. Po dokončení analýzy se objeví výsledky a informace o době běhu.

4.2.1.1 Parametry Forensic Analyzeru

Parametry jsou logicky rozdělené do 4 kategorií:

1. Parametry popisující jaké jsou vstupní logovací soubory, ve kterých se budou podezřelé aktivity hledat.
2. Parametry popisující z jakých konfiguračních souborů se bude vytvářet sada pravidel, která definuje podezřelé aktivity.
3. Fixní parametry, což jsou přednastavené parametry, které zaručují správnou funkcionalitu pro offline analýzu logů.
4. Volitelné parametry, které slouží pro pokročilé uživatele se znalostmi další parametrizace aplikace SEC, které zde lze uplatnit.

Můžete si všimnout, že parametry z kategorie 1–3 jsou šedé. To značí, že je nelze ručně editovat. Generují se automaticky dle aktuálního stavu aplikace.

Parametry z kategorie 4 je možné editovat. Lze použít veškeré parametry příkazové řádky, které podporuje nástroj SEC – viz [7, 8]. Ve výchozím nastavení je nastaven parametr `--intvents`, který vynutí generování interních událostí SEC, kterých lze také využívat při psaní pravidel v konfiguračních souborech.

4.2.1.2 Menu lišta a panel nástrojů

Panel nástrojů poskytuje rychlejší přístup k některým prvkům menu lišty. Tento panel nástrojů není fixní a lze jej přemístit na jiný okraj aplikace. Většina operací, které zde nebo v menu najdete, má přiřazené alternativní klávesové zkratky, které mohou zrychlit používání aplikace.

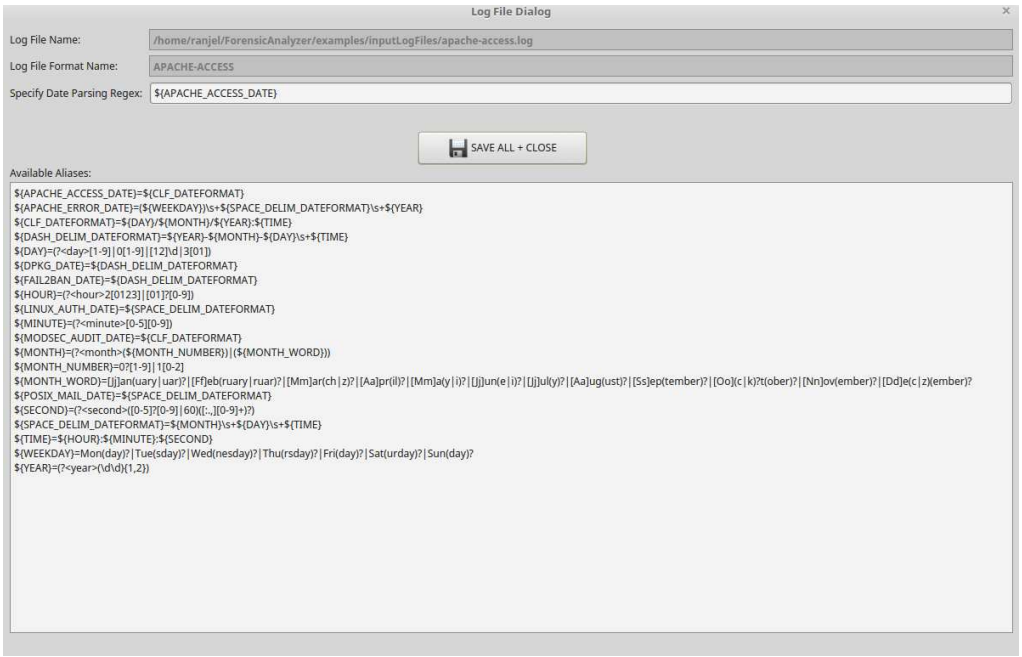
V menu se nachází možnost specifikovat časové filtry, které umožní definovat interval, ve kterém chcete hledat podezřelé aktivity a jejich korelace. Události mimo tento interval budou ignorovány. Pomocí kalendáře lze specifikovat horní nebo dolní mez, popřípadě obě současně.

Další položkou je volba podrobnosti výpisu. Čím vyšší úroveň, tím více informací o průběhu analýzy bude zobrazeno. Výchozím nastavením je úroveň 3. Úroveň 1 zobrazuje pouze nalezené podezřelé aktivity a druhým extrémem je úroveň 6, která detailně popisuje všechny důležité momenty analýzy.

Jak podrobnost výpisu, tak časové filtry se připisují do parametrů kategorie 3. Pokud budete chtít přidávat parametry kategorie 4 může být praktické si zobrazit veškerou dostupnou parametrizaci SEC+ (například pomocí klávesové zkratky `Ctrl + U`).

Zbývají 2 nejdůležitější položky menu, které si rozebereme podrobněji.

4.2.2 Správa vstupních logovacích souborů



Obrázek 4.2: Dialog se vstupním logovacím souborem

V první záložce na hlavním okně naleznete přehled aktivních a neaktivních vstupních logů. Aktivní logy jsou zaškrtnuté a tedy se také nachází v parametrech kategorie 1. První tři akce tohoto menu jsou vyhrazené pro

možnost přidat, editovat nebo smazat vstupní log. Akce smazat log (popřípadě konfigurační soubor) pouze odstraní evidenci tohoto souboru z aplikace, neprovádí fyzické smazání z disku.

Pokud zvolíte akci přidat log, zobrazí se vám dialog, ve kterém vyberte logovací soubor, který budete chtít zkoumat. Následovat bude dialog z obrázku 4.2, ve kterém je potřeba doplnit potřebné informace o logovacím souboru.

4.2.2.1 Aliasy

Největší část je věnována pro uživatelem definované aliasy, které slouží k zjednodušení a zvýšení přehlednosti zápisu regulárních výrazů.

Alias se definuje zápisem `${<jmémo proměnné>}=<hodnota>`. Vyhodnocování probíhá rekurzivně, dokud `<hodnota>` není obyčejný řetězec.

Listing 4.1: Příklad definice tří aliasů

```
${SECOND}=(?<second>([0-5]?[0-9]|60)([:.,][0-9]+)?)
${MINUTE}=(?<minute>[0-5][0-9])
${TIME}=(?<hour>2[0123]|[01]?[0-9]):${MINUTE}:${SECOND}
```

4.2.2.2 Popis data a času

Je nutné specifikovat regulární výraz, podle kterého bude možné z každé události získat datum a čas. Tento regulární výraz navíc musí obsahovat tzv. *named capture groups*. [33]

Vyžaduji specifikovat přesně tyto skupiny:

- year[7]
- month
- day
- hour
- minute
- second

Každá skupina musí být obalena zleva `?<` a zprava `>`.

[7] Tato skupina je nepovinná.

Listing 4.2: Příklad regulárního výrazu získávající datum a čas z události

```
(?<month>(0?[1-9]|1[0-2]))\s+(?<day>[1-9]|0[1-9]|
[12]\d|3[01])\s+(?<hour>2[0123]|[01]?[0-9]):(?<minute>
[0-5][0-9]):(?<second>([0-5]?[0-9]|60)([:.,][0-9]+)?)
```

Pokud bude skupina pojmenovaná například `month`, aplikace očekává, že v této skupině nalezne údaj o tom, jaký je právě měsíc. Nemusí to být ovšem jen číselná informace. Validní je také například „Jan“ nebo „January“.

Pojmenovaná skupina `year` je jediná nepovinná, protože některé logy zkrátka rok neevidují. Pokud tato informace není dostupná, aplikace vezme nejpravděpodobnější rok dle dnešního data. Pokud je například 16. dubna 2017 a bude analyzován log, ve kterém je měsíc specifikován jako „Nov“, pak aplikace vyhodnotí, že se jedná o listopad roku 2016, ne roku 2017.

Lze zde využít výše zmíněné aliasy.

4.2.2.3 Formát logu

V horní části okna se nachází další dvě pole. V prvním je absolutní cesta k právě spravovanému souboru. V druhém poli je generovaný název formátu logu, pro který se automaticky vytváří alias `${<název formátu>}=<zadaný vstup od uživatele>`, který bude možné použít později pro jiný logovací soubor stejného formátu.

4.2.2.4 Uložení změn a ukončení

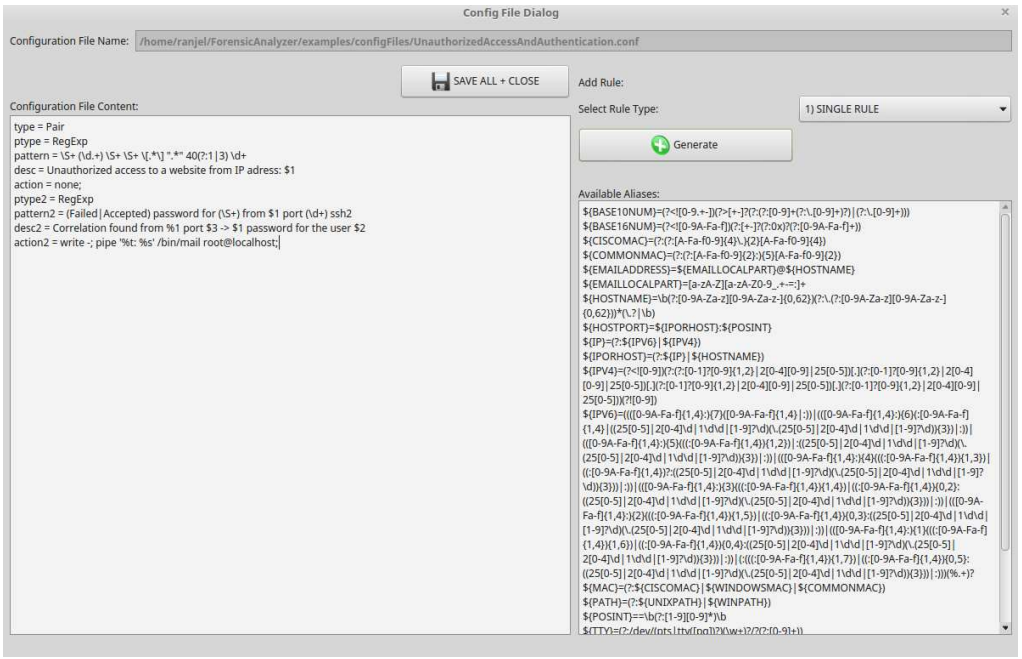
Uložení změn v tomto okně se provede příslušným tlačítkem `SAVE ALL + CLOSE`, případně klávesovou zkratkou `Ctrl + S`. Pokud okno zavřete křížkem v pravém horním rohu, změny nebudou uloženy.

Nově přidaný log je automaticky považován za aktivní.

4.2.3 Správa konfiguračních souborů

Logika správy konfiguračních souborů je velice podobná správě vstupních logovacích souborů. Zmíním tedy jen odlišnosti.

Na obrázku 4.3 je zachycen dialog po výběru konfiguračního souboru. Vpravo naleznete opět seznam aliasů. Tyto aliasy jsou oddělené od předchozích a lze je uplatnit v levé části, která zobrazuje obsah daného konfiguračního souboru.



Obrázek 4.3: Dialog s konfiguračním souborem

4.2.3.1 Pravidla z konfiguračního souboru

Jednotlivá pravidla musí být oddělena minimálně jedním prázdným řádkem. Pravidla mohou být různých typů, viz [8]. Lze si zvolit typ pravidla, které chcete přidat, a kliknout na tlačítko `Generate`, popřípadě stisknout klávesovou zkratku `Ctrl + G`. Takto je možné vygenerovat všechny povinné položky, které dané pravidlo vyžaduje specifikovat.

Můžete si všimnout, že je automaticky generována akce „`write -`“[8]. Tuto akci nemažte, jinak hledanou podezřelou aktivitu program sice nalezne, ale v přehledu ji nezobrazí. Je možné přidávat více akcí, stačí je oddělovat středníkem.

4.2.4 Zálohovací soubory

Aplikace si i po vypnutí pamatuje mnoho informací z předchozího běhu a automaticky je načítá při spuštění. Jedná se například o seznam definovaných logovacích a konfiguračních souborů nebo seznam definovaných aliasů. Tato evidence je uložena v zálohovacích souborech ve složce `backup`.

[8] Spojovník (`-`) je zkratka pro standardní výstup.
Akce `write` má signaturu: `write <soubor> [<obsah>]`. Podrobněji zde [8].

Pokud chcete rychle získat čistou verzi *Forensic Analyzeru*, můžete celou tuto složku smazat. Aplikace je ovšem dodána s přednastavenými aliasy, které by byla škoda nevyužít.

Kapitola 5

Testování

Pro účel testování jsem připravil aliasy pro všechny typy logovacích souborů, které mi vedoucí práce poskytl. Přidání těchto logů do *Forensic Analyzeru* bude tedy triviální (postupujte analogicky k obrázku 4.2). Tyto logy najdete na přiloženém CD ve složce `examples/inputLogFiles`.

Ve složce `examples` je také složka `configFiles`, která obsahuje příklady jednotlivých pravidel včetně vysvětlivek. Ukázkové konfigurační soubory lze aplikovat na log `examples/inputLogFiles/secRulesExamples.log`, který demostruje použití různých typů pravidel. Můžete se přesvědčit, že *Forensic Analyzer* zachovává původní způsob používání programu Simple Event Correlator a navíc umožňuje offline práci s časem, řazení událostí, filtry apod.

Nyní je potřeba dalšími pravidly popsat, co je podezřelá aktivita ve vašem kontextu. Podařilo se mi vytvořit jeden univerzální konfigurační soubor, podle kterého se bude hledat korelace IP adres napříč libovolnými logy, které do *Forensic Analyzeru* přidáte. Tento konfigurační soubor vytváří přehled, ze kterého je patrná aktivita jednotlivých uživatelů. Můžete si ho detailně prohlédnout na obrázku 5.1.

První pravidlo shlukuje události, které obsahují stejnou IP adresu. Toho dosahuje pomocí kontextů a takzvaného *event store*, který má každý kontext k dispozici. Za zmínku stojí také praktické využití aliasu `$IP`, který bude po vyhodnocení popisovat vzor tohoto pravidla sice komplexním, ale poměrně nepřehledným způsobem (viz `backup/formatAliasesBackup.txt`).

Druhé pravidlo demonstruje integraci Perlu. Využívá tzv. interních událostí, které jsou generovány pouze s přepínačem `--intevents`. S tímto přepínačem bude po přečtení všech logovacích souborů vyvolána interní událost `SEC_SHUTDOWN`, na kterou přesně druhé pravidlo čeká. Úkolem tohoto pravidla je vytvořit výpis, který bude zobrazen uživateli.

Listing 5.1: Konfig. soubor `associatedActivitiesAcrossAllLogs.conf`

```
type = Single
ptype = RegExp
pattern =  (${IP})
desc = $0
action = add context_$1 $0

type = Single
ptype = SubStr
pattern = SEC_SHUTDOWN
context = SEC_INTERNAL_EVENT
desc = SEC_SHUTDOWN triggered!
action = lcall %ret -> ( sub { \
  print "List of activities across all log files by \
  IP:\n"; \
  foreach $contextName (keys %main::context_list) { \
      if ($contextName ne "SEC_INTERNAL_EVENT") { \
      print "\n".substr($contextName, 8).":\n"; \
      foreach $event (@{ \
        $main::context_list{$contextName}->{"Buffer"} \
      }){ \
        print "$event\n"; \
      } } } } )
```

V rámci testování doby běhu jsem použil tento konfigurační soubor, který jsem aplikoval na různě veliké logovací soubory `apache-access<počet řádek logu>.log`. Naměřené zprůměrované výsledky jsou v tabulce 5.1.

Doba běhu je udávána ve formátu: `<minuty>:<sekundy>:<milisekundy> [<odpovídající přepočet na milisekundy>]`.

V tabulce 5.1 jsou tři sloupce. V každém sloupci se jedná o testování s jinou parametrizací nástroje *Forensic Analyzer*.

1. SEC – V tomto případě byla prováděna forenzní analýza s volitelným parametrem `--dontParseEventTime`. Připomínám, že pomocí tohoto přepínače lze zrušit fixní parametry `--parseEventTime` a `--parsedTimeOrder`, tedy lze vynutit nasimulování původního programu Simple Event Correlator (odtud i název sloupce).

2. FA – FA je zkratka pro *Forensic Analyzer*. Analýza spouštěna s prázdnými volitelnými parametry. Jedná se o výslednou složitost vytvořeného nástroje *Forensic Analyzer*. Doba běhu této mé nadstavby je zhruba 150% vůči původní nerozšířené verzi SEC. Za tento nárůst

časové složitosti získáte mnohem lepší offline analýzu – *timestamp* je možné získávat přímo z události v logu, události je možné brát spravedlivě v pořadí dle původních časových značek, možnost filtrovat události a v neposlední řadě využívat GUI, které značně usnadňuje tvorbu pravidel a generování parametrů.

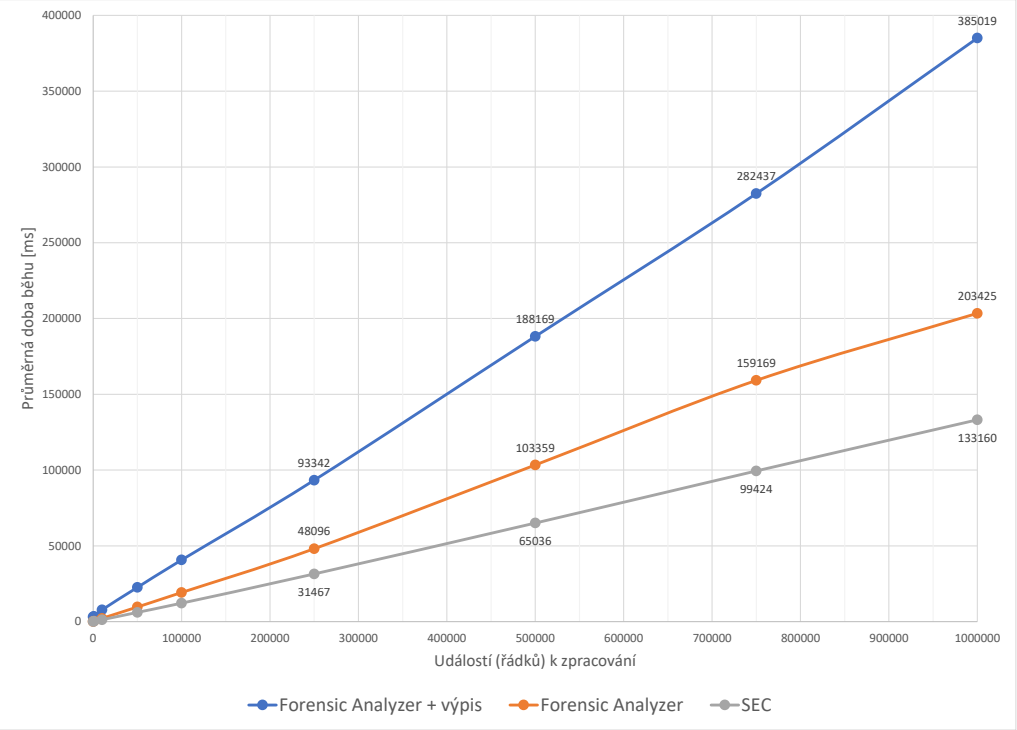
3. FA + výpis – V tomto testování jsem přidal přepínač `--intevents` do volitelných parametrů. Na rozdíl od obou předchozích se pouze v tomto případě vypíše přehled, ze kterého je patrná aktivita jednotlivých uživatelů. Tento sloupec reprezentuje reálnou složitost, tedy k složitosti forenzní analýzy je nutné navíc připočítat složitost zobrazení výstupu v GUI. Doba běhu vychází přibližně 200% ve srovnání s předchozí parametrizací. Nutno podotnout, že tento dvojnásobný nárůst časové složitosti je způsoben tím, že tento konfigurační soubor generuje velké množství výstupů. Pokud v akci `write` nahradíte standardní výstup například souborem, budete se spíše blížit složitosti z předchozího bodu.

Na obrázku 5.1 je graf, který odráží tabulku 5.1.

Podařilo se mi vytvořit nástroj, který zachovává lineární asymptotickou časovou složitost.

Tabulka 5.1: Testování doby běhu dle různého počtu zpracovaných událostí

# řádek	FA+výpis [~ms]	FA [~ms]	SEC [~ms]
5	00:03:073 [3073]	00:00:076 [76]	00:00:078 [78]
1000	00:03:482 [3482]	00:00:266 [266]	00:00:187 [187]
10000	00:07:775 [7775]	00:02:148 [2148]	00:01:242 [1242]
50000	00:22:576 [22576]	00:09:658 [9658]	00:06:118 [6118]
100000	00:40:702 [40702]	00:19:228 [19228]	00:12:192 [12192]
250000	01:33:342 [93342]	00:48:096 [48096]	00:31:467 [31467]
500000	03:08:169 [188169]	01:43:359 [103359]	01:05:036 [65036]
750000	04:42:437 [282437]	02:39:169 [159169]	01:39:424 [99424]
1000000	06:25:019 [385019]	03:23:425 [203425]	02:13:160 [133160]



Obrázek 5.1: Graf porovnávající dobu běhu dle počtu zpracovaných událostí a použité parametrizace vytvořeného nástroje

Kapitola 6

Zhodnocení

V souladu se zadáním práce jsem nejprve nastudoval literaturu s problematikou logů a forenzní analýzy. V aplikaci Enterprise Architect jsem po konzultaci s vedoucím práce zachytil seznam požadavků, sestavil model případů užití a namodeloval diagramy aktivit. Poté jsem v rešeršní části prošel mnoho nástrojů a vybral jsem nejvhodnějšího kandidáta – Simple Event Correlator.

Tento nástoj jsem úspěšně rozšířil, přičemž jsem zachoval původní funkcionalitu a způsob používání. Z tohoto důvodu se domnívám, že bych mohl vytvořit *pull request* do originálního projektu na Githubu s vysokou šancí začlenění mých změn do programu Simple Event Correlator. S cílem zvládnout toho rozšíření jsem se musel detailně seznámit s dokumentací tohoto nástroje [8], zorientovat se v kódu, který čítal cca 13 000 řádek, a nastudovat samotný jazyk Perl, ve kterém je tato aplikace napsaná. Rozšíření přináší podporu pro offline forenzní analýzu, která zkoumá *timestamp* jednotlivých událostí.

Dále jsem si vyzkoušel také vývoj od nuly. Vytvořil jsem grafickou nadstavbu, která značně usnadňuje tvorbu pravidel, umožňuje používat aliasy, pamatuje si již použité logovací nebo konfigurační soubory, generuje parametry a celkově odstiňuje uživatele od příkazové řádky.

K tomuto softwaru jsem vytvořil dokumentaci, která zahrnuje programátorskou, instalační a uživatelskou příručku.

Při testování se ukázalo, že můj produkt má lineární časovou složitost vůči počtu zpracovaných událostí.

Ve výsledku vznikl volně šiřitelný nástroj *Forensic Analyzer*, který umožňuje analyzovat podezřelé aktivity na serveru.

Závěr

Cílem této práce bylo navrhnout nástroj, který dokáže spojit více logů a zobrazit přehled podezřelých aktivit. Nástroj byl úspěšně navržen a implementován, všechny požadavky, které byly na aplikaci kladeny, byly splněny.

Pomocí konfiguračních souborů je možné dodefinovat pravidla popisující podezřelé aktivity a jaká reakce se má provést, pokud k této aktivitě dojde. Aplikace je připravena na přidání nového logovacího souboru s atypickým formátem a podporuje také základní formáty serverových logů, se kterými jsem se seznámil a vytvořil analýzu využitelnosti obsažených údajů. Sepsal jsem podrobnou uživatelskou dokumentaci a předpřipravil ukázky pravidel, které hledají různé korelace. Nakonec byl nástroj podroben testování.

Výsledkem práce je funkční aplikace, která zjednodušuje bezpečnostní analýzu serverových logů. Poskytuje rozsáhlé možnosti konfigurace, její modularita umožňuje praktické využití a otevírá možnosti k dalšímu rozříření.

Možnosti budoucího rozvoje

Podařilo se mi vytvořit plně funkční aplikaci, ačkoli zadání požadovalo pouze prototyp aplikace. Nicméně prostory k vylepšení vidím v grafickém uživatelském rozhraní. Napadlo mě například přidat generování grafických přehledů podezřelých aktivit, které byly odhaleny při forenzní analýze.

Literatura

[1] CHUVAKIN, Anton., SCHMIDT, Kevin J., PHILLIPS, Chris a MOULDER, Patricia. *Logging and log management: the authoritative guide to understanding the concepts surrounding logging and log management.* Amsterdam: Elsevier/Syngress, 2013. ISBN 1597496359.

[2] MAIER, Phillip Q. *Audit and trace log management: consolidation and analysis.* Boca Raton, FL: Auerbach Publications, 2006. ISBN 978-0849327254.

[3] ALTHEIDE, Cory. a CARVEY, Harlan A. *Digital forensics with open source tools.* Burlington, MA: Syngress, 2011. ISBN 978-1597495868.

[4] LACINA, Miroslav. *Agregace a prohledávání poštovních logů.* Praha, 2014. bakalářská práce. České vysoké učení technické v Praze, fakulta Informačních technologií. vedoucí práce Ing. Jan Baier

[5] AMBRE, Amruta a SHEKOKAR, Narendra. *Insider Threat Detection Using Log Analysis and Event Correlation.* Procedia Computer Science [online]. 2015, 436-445 [cit. 11. prosince 2016]. DOI: 10.1016/j.procs.2015.03.175. ISSN 18770509. Dostupné z: `http://linkinghub.elsevier.com/retrieve/pii/S1877050915004184`

[6] JANSEN, Bernard J. *Search log analysis: What it is, what's been done, how to do it.* Library & Information Science Research [online]. 2006, 407–432 [cit. 2. dubna 2017]. ISSN 0740-8188. Dostupné z: `http://www.sciencedirect.com/science/article/pii/S0740818806000673`

[7] VAARANDI, Risto. *SEC - simple event correlator v2.7.11* [software]. 3. února 2017. Dostupné z: `http://simple-evcorr.github.io/`. Požadován Perl 5.8 a vyšší

[8] VAARANDI, Risto. *SEC manpage* [online]. 2017. [cit. 2. dubna 2017]. Dostupné z: http://simple-evcorr.github.io/man.html

[9] VAARANDI, Risto. *Simple Event Correlator - Best Practices for Creating Scalable Configurations* [online]. 2015. [cit. 2. dubna 2017]. Dostupné z: http://ristov.github.io/publications/cogsima15-sec-web.pdf

[10] LANG, David. *Using SEC* [online]. 2013. [cit. 2. dubna 2017]. Dostupné z: https://www.usenix.org/system/files/login/articles/09_lang-online.pdf

[11] *Logging Control In W3C httpd* [online]. World Wide Web Consortium. 1995. [cit. 2. dubna 2017]. Dostupné z: https://www.w3.org/Daemon/User/Config/Logging.html#common-logfile-format

[12] *LinuxLogFiles* [online]. Official Ubuntu Documentation. (2015). [cit. 8. dubna 2017]. Dostupné z: https://help.ubuntu.com/community/LinuxLogFiles#Authorization_Log

[13] *Log Files* [online]. The Apache Software Foundation. [cit. 8. dubna 2017]. Dostupné z: https://httpd.apache.org/docs/1.3/logs.html

[14] *Apache Module mod_log_config* [online]. The Apache Software Foundation. [cit. 8. dubna 2017]. Dostupné z: http://httpd.apache.org/docs/current/mod/mod_log_config.html

[15] *Linux : ModSecurity log analysis with Modgrep* [online]. The Endless Geek. (2014). [cit. 8. dubna 2017]. Dostupné z: http://endlessgeek.com/2014/02/search-modsecurity-audit-log-analysis

[16] *ModSecurity 2 Data Formats* [online]. SpiderLabs. (2012). [cit. 8. dubna 2017]. Dostupné z: https://github.com/SpiderLabs/ModSecurity/wiki/ModSecurity-2-Data-Formats

[17] *Fail2Ban setup for Apache* [online]. miniwark. (2013). [cit. 9. dubna 2017]. Dostupné z: https://github.com/miniwark/miniwark-howtos/wiki/Fail2Ban-setup-for-Apache

[18] *dpkg man page* [online]. Ubuntu Manpage Repository. [cit. 9. dubna 2017]. Dostupné z: http://manpages.ubuntu.com/manpages/xenial/man1/dpkg.1.html

[19] *XpoLog Center Log Management Solution* [online]. XpoLog.com. (2009). [cit. 9. dubna 2017]. Dostupné z: http://www.cressidatechnology.com/pdfs/products/xplg/XpoLogCenterBro.pdf

[20] *ClickTracks Optimizer* [online]. clicktracks.com. [cit. 9. dubna 2017]. Dostupné z: http://www.clicktracks.com/products/optimizer/index.php

[21] *RConvlog - CONVLOG replacement* [online]. Rebex Labs. (2014). [cit. 9. dubna 2017]. Dostupné z: https://labs.rebex.net/RConvLog

[22] *Chainsaw v2* [online]. The Apache Software Foundation. (2007). [cit. 13. dubna 2017]. Dostupné z: https://logging.apache.org/chainsaw

[23] *Docker Operational Intelligence* [online]. Sematext Cloud. [cit. 13. dubna 2017]. Dostupné z: https://sematext.com/docker

[24] BARRETT, Bradford L. *The Webalizer* [online]. webalizer.org. (2014). [cit. 13. dubna 2017]. Dostupné z: http://www.webalizer.org

[25] BARRETT, Bradford L. *The Webalizer - A web server log file analysis tool* [online]. mrunix.net. (Copyright 1997-2013). [cit. 13. dubna 2017]. Dostupné z: ftp://ftp.mrunix.net/pub/webalizer/README

[26] DOČEKAL, Michal. *Správa linuxového serveru: Sledování logů pomocí nástroje logwatch* [online]. linuxexpres. (2011). [cit. 13. dubna 2017]. Dostupné z: https://www.linuxexpres.cz/praxe/sprava-linuxoveho-serveru-sledovani-logu-pomoci-logwatch

[27] PARSONS, Trevor. *The Pros and Cons of Open Source Logging* [online]. logentries. (2014). [cit. 14. dubna 2017]. Dostupné z: https://blog.logentries.com/2014/09/the-pros-and-cons-of-open-source-logging

[28] *A Beginner's Guide to Logstash Grok* [online]. logz.io. [cit. 14. dubna 2017]. Dostupné z: https://logz.io/blog/logstash-grok

[29] *The Complete Guide to the ELK Stack* [online]. logz.io. [cit. 14. dubna 2017]. Dostupné z: https://logz.io/learn/complete-guide-elk-stack

[30] *Tripwire Open Source vs OSSEC : Which Is Right For You?* [online]. UpGuard. (2015). [cit. 14. dubna 2017]. Dostupné z:

https://www.upguard.com/articles/tripwire-open-source-vs.-ossec-which-is-right-for-you

[31] *OpenNMS (Open Network Management System)* [online]. searchnetworking.techtarget.com. (2006). [cit. 14. dubna 2017]. Dostupné z: http://searchnetworking.techtarget.com/definition/OpenNMS

[32] VAARANDI, Risto. *loghound* [online]. ristov.github.io. (2004). [cit. 14. dubna 2017]. Dostupné z: http://ristov.github.io/loghound/loghound.html

[33] *perlre – Capture Groups* [online]. Perl Programming Documentation. [cit. 16. dubna 2017]. Dostupné z: https://perldoc.perl.org/perlre.html#Capture-groups

[34] *popen - initiate pipe streams to or from a process* [online]. The Open Group. (Copyright 1997) [cit. 16. dubna 2017]. Dostupné z: http://pubs.opengroup.org/onlinepubs/007908799/xsh/popen.html

[35] BARR Graham, The Comprehensive Perl Archive Network. *TimeDate-2.30 , Date::Parse* [software]. (Copyright 1995-2009). Dostupné z: http://search.cpan.org/~gbarr/TimeDate-2.30/lib/Date/Parse.pm

Příloha **A**

Seznam použitých zkratek

CLI Command-Line Interface

CLF Common Log Format

CRUD Create, Read, Update and Delete

GUI Graphical User Interface

HIDS Host-based Intrusion Detection Systems

HTML HyperText Markup Language

HTTPD HyperText Transfer Protocol Daemon

IP Internet Protocol

OS Operating System

POSIX Portable Operating System Interface

SIEM Security Information and Event Management

SQL Structured Query Language

SEC Simple Event Correlator

SSH Secure Shell

XML Extensible Markup Language

Příloha **B**

Obsah přiloženého CD

- `backup` ..... zálohovací soubory *Forensic Analyzeru*
- `examples` ..... příklady použití *Forensic Analyzeru*
  - `configFiles` ..... příklady konfiguračních souborů
  - `inputLogFiles` ..... příklady logovacích souborů
- `man` ..... instalační a uživatelská příručka
- `other`
  - `bin` ..... spustitelná forma implementace
  - `enterpriseArchitect` ..... Enterprise Architect dokumentace
  - `gitDiffSEC+` ..... porovnání SEC+ s oficiální verzí SEC
- `res` ..... významné soubory pro *Forensic Analyzer*
  - `sec+` ..... program SEC+ (součást *Forensic Analyzeru*)
- `src` ..... zdrojové kódy *Forensic Analyzeru*
- `thesis` ..... text práce
  - `src`
    - `BP_Lejnar_Jan_2017.tex` ..... text práce ve formátu LaTeX
  - `text`
    - `BP_Lejnar_Jan_2017.pdf` ..... text práce ve formátu PDF
- `Makefile` ..... makefile *Forensic Analyzeru*
- `readme.pdf` ..... stručný popis nejdůležitějšího obsahu CD