FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Monitoring sítě v cloudovém systému
Student:	Petr Beneš
Vedoucí:	Ing. Jiří Chludil
Studijní program:	Informatika
Studijní obor:	Bezpečnost a informační technologie
Katedra:	Katedra počítačových systémů
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

1) Analyzujte potenciální útoky na webové aplikace a možnosti jejich detekce.
2) Analyzujte aplikaci Suricata a porovnejte ji s konkurenčními aplikacemi.
3) Navrhněte úpravy aplikace Suricata, které umožní
   a) automatickou instalaci na cílový virtuální stroj v cloudovém systému.
   b) jeho vzdálenou aktivaci/deaktivaci.
   c) centralizovaný sběr dat (včetně použitého API)
   d) detekovat správnou funkčnost aplikace
4) Implementujte navržené úpravy, tak aby šlo úpravy použít jako možné rozšíření projektu Suricata.
5) Navrhněte srovnávací studii pro vyhodnocení účinků upravené aplikace v cloudovém systému.
6) Srovnávací studii proveďte a zhodnoťte její výsledky

Seznam odborné literatury

Dodá vedoucí práce.

prof. Ing. Róbert Lórencz, CSc.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 2. ledna 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Monitoring sítě v cloudovém systému

Petr Beneš

Katedra počítačových systémů
Vedoucí práce: Ing. Jiří Chludil

10. května 2018

Poděkování

Chtěl bych poděkovat mému vedoucímu Ing. Jiřímu Chludilovi za podporu při psaní práce. Dále bych chtěl poděkovat firmě SIC s. r. o. za možnost vypracovat tuto práci a v neposlední řadě i jejím zaměstatncům, díky kterým jsem mohl mou praktickou část otestovat na systému Big Cloud.

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 10. května 2018 ……

Odkaz na tuto práci

Beneš, Petr. *Monitoring sítě v cloudovém systému*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Práce se zaměřuje na monitoring sítě a použití softwaru Suricata. Začátek je věnován útokům na webové aplikace. Jsou zde uvedené známé útoky jako například SQL Injection, Cross-site scripting a jiné. U každého útoku jsou popsány možnosti provedení, způsoby obrany a jejich možné následky. Následující sekce je věnována systémům IDS a IPS, jejich způsob ochrany a co ve skutečnosti svedou. V další části se již věnuji samotného softwaru Suricata, způsobu, jakým pracuje, jaké má konfigurační soubory a další. Je zde i vidět porovnání s konkurenčním systémem Snort. Po této části je vyhrazena sekce pro stručný popis systému Big Cloud, do kterého bude software Suricata integrován. Dále je popsáno, jak vypadá sběr záznamů uložených na jednotlivých virtuálních strojích, jakým způsobem se bude monitorovat software Suricata na virtuálních strojích a jak se budou ukládat záznamy do databáze. Po části s návrhem přichází oddíl, ve kterém je popsána instalace softwaru Suricata s postupy, jak správně software Suricata nakonfigurovat. Na závěr práce je uveden výsledek testování na systému Big Cloud. Je zde uvedeno a popsáno několik varovných hlášení, které se objevili během testování.

Klíčová slova Monitoring sítě, Cloudový systém, útoky na webové aplikace, IDS, IPS, Big Cloud, Suricata

Abstract

The thesis is focused on network monitoring using the software Suricata. Attacks on web applications are covered in the beginning. Several attacks like SQL Injection and Cross-site scripting are discussed. Options for exploitation are described for each attack, defense options and their possible consequences also. The next section is dedicated to IDS and IPS systems; how they work and what they can provide. Next, there is a description of how the software Suricata works, what configuration files look like and many other topics. Here is also a comparison with the competitive software Snort. After this section, there is a brief description of the system Big Cloud. The goal of the practice part of the work is a deployment of the software Suricata on the system Big Cloud. Pictures are used to describe the integration. In the final section of the work, some logs from the software Suricata are presented. Here is also described how to collect records stored on individual virtual machines, how Suricata software will be monitored on virtual machines and how records will be stored in the database. After the suggestion section, there is a section describing the installation of Suricata software with procedures for configuring Suricata software correctly. At the end of the thesis, the result of Big Cloud testing is presented. Several warning messages that appeared during testing are listed and described.

Keywords Network monitoring, Cloud system, attacks on web applications, IDS, IPS, Big Cloud, Suricata

Obsah

Seznam obrázků

Úvod

Bezpečnost cloudových sítí je čím dál důležitější, protože vznikají stále nové a nové servery poskytující tyto služby a služby jsou i čím dál více používané. S rostoucí popularitou těchto služeb se ale i zvyšuje hrozba možných útoků na ně. Průniky nejsou již ojedinělé a mohou nastat i několikrát denně. Následky útoků jsou například výpadky služeb, ztráta citlivých údajů a další. Narušení bezpečnosti se proto musí včas detekovat a co nejdříve na něj reagovat, aby dopady byly co nejnižší.

Moje práce je určená primárně pro firmu, ale poslouží také i jejich zákazníkům. Firmě to pomůže, protože budou mít více času zjistit průniky a mohou minimalizovat škody jimi způsobené.

Zákazníci díky tomu mohou pohodlněji využívat její služby a jejich údaje budou ve větším bezpečí. V práci se budu zabývat nasazením zadaného softwaru pro sledování sítě, analýzou sítě a zhodnocení výsledků softwaru Suricata. Dále pak i napíši postup instalace softwaru a jeho spuštění, následně pak i stručnou dokumentaci pro uživatele a správce sítě.

V první kapitole bakalářské práce se zabývám analýzou. Nejprve samotného problému útoku na webové aplikace. Popíši známé útoky, ukáži možnosti jejich provedení a způsoby obrany proti nim. V další části uvedu, co to jsou systémy IDS a IPS, jaký je v nich rozdíl, co mohou a nemohou dělat. Ve třetí části již popisuji samotný software Suricata, co to vlastně je, jakým způsobem pracuje a jaký je rozdíl oproti konkurenčnímu softwaru Snort. V poslední části se zabývám systémem Big Cloud, popíši jeho strukturu i na obrázcích a uvedu co je zač.

Kapitola 1

Cíl práce

Cílem rešeršní části práce je obeznámit čtenáře o potenciálních útocích na webové stránky, mít přehled v používání softwaru Suricata, seznámit se s nástrojem Suricatasc, pracujícím se softwarem Suricata, který dokáže například monitorovat jeho stav, popsat, jak vypadá server, na který se bude monitorovací software nasazovat, stručně popsat, co to je Big Cloud. Důležité je ale i pracování s výstupy softwaru Suricata, dokázat číst a zpracovat logy. Dále pak je potřeba dokázat bezpečným způsobem zapnout a komunikovat se softwarem. Vzhledem k povaze příkazů je nutné mít root práva a tato komunikace nesmí tvořit bezpečnostní riziko. Dalším cílem je porovnat software Suricata s konkurenčním softwarem Snort, zhodnotit klady a zápory obou softwarů. Cílem práce je i popsat, co to jsou systémy IDS a IPS, co dokážou, ale i čeho naopak nejsou schopné.

Cílem praktické částí práce je nasadit software Suricata na již existující systém, aby monitoroval dění na sítí. K tomu je zapotřebí vytvořit několik skriptů. Jeden, který dokáže jednoduše software Suricata nainstalovat, druhý, který inicializuje potřebné součásti pro správnou funkci, a třetí, který vytvoří nového uživatele Suricata, který bude mít administrátorská práva pouze na komunikaci se softwarem Suricata. Následně pak nasadit software na již existující systém, aby monitoroval dění na sítí. Dále vytvořím program, který bude periodicky posílat logy na určené místo.

Kapitola 2

Analýza

2.1 Analýza potenciálních útoků na webové aplikace

V této kapitole se budu věnovat útokům na webové aplikace. Uvedu několik známých útoků, které jsou způsobené interakcí webové aplikace a prohlížeče. K provedení útoků je potřeba alespoň základní znalost SQL, HTTP, XML a u některých dalších útoků i znalost javascriptu a PHP.

Na webové aplikace se útočí z mnoha důvodu, někteří to dělají pro peníze, jiní zase pro slávu, z nudy nebo chtějí zkusit své dovednosti.

2.1.1 SQL Injection

Tento typ útoku využívá špatného ošetření vstupu od uživatele při dosazování parametrů do předem připraveného dotazu mířícího do databáze. Problém nastává, pokud se do chráněné zóny pustí externí data, aniž by byla dostatečně ošetřena.

Útok může vypadat například tak, že server má předpřipraven dotaz `SELECT * FROM tabulka WHERE field = 'param';`, kde „param“ je vstup od uživatele. Uživatel může jako vstup zvolit například `'; DELETE FROM tabulka WHERE '1'='1`, čili výsledný kód, který se pošle do databáze vypadá takto:

```
SELECT * FROM tabulka
WHERE field = '';
DELETE FROM tabulka
WHERE '1'='1';
```

Tento vstup může způsobit ztrátu dat, pokud uživatel má na mazání z tabulky oprávnění.

Hrozba, kterou SQL injection představuje, může mít různé dopady podle nastavení databáze a především stupni oprávnění uživatele. Pokud by se používal vysoký stupeň oprávnění pro uživatele, tak může dojít ke ztrátě dat, neoprávněnému čtení z databáze nebo úpravě dat.

V PHP s MySQL výše uvedený problém nehrozí, pokud se použije funkce `mysqli::multi_query`, protože funkce provádějící SQL dotaz většinou provádí pouze první SQL dotaz. Dotaz `DELETE FROM tabulka WHERE '1'='1'` se neprovede. V PHP existuje ale funkce `mysqli::multi_query`, která podporuje více dotazů naráz, pokud se použije, riziko stále hrozí.

Proti SQL Injection se dá bránit několika způsoby:

1. Princip nejmenšího oprávnění – Uživatel ne vždy potřebuje právo na všechny typy dotazů, například u vyhledávačů mu stačí pouze právo na SELECT, žádná práva navíc by neměl mít. V databázi se dá také nastavit, uživatel měl přístup pouze jen k některým sloupcům tabulky.

2. Kontrola vstupů – Všechny vstupy uživatele představují riziko a musí být ověřeny. Existuje několik způsobů kontroly vstupu.

   a) Escapování znaků – Je potřeba provést escapování všech znaků, která mají v SQL speciální význam, by nemohly dotaz do databáze narušit. Escapování znaku znamená vložit před něj znak \. V jazyce PHP existuje speciální funkce `mysql_real_escape_string`, v C++ se dá použít escapeString z `mysql_connection.h`.

   b) Používání placeholderů – Použití parametrizovaných příkazů. Jsou to příkazy, které mají místo parametrů nevyplněné hodnoty a vyplní se později službami databázového systému. Na místa, kam přijdou parametry, se používá často znak `?` nebo `@jmeno`, záleží na databázovém systému. Parametrizovaný příkaz může vypadat takto:

   ```
   SELECT * FROM tabulka WHERE field = ?;
   ```

   c) Použití před připravených dotazů s parametrizovanými příkazy – Je vhodné je využít, protože kromě výhod parametrizovaných příkazů, přináší i výhodu v tom, že při opakovaném použití stejného dotazu vícekrát, klidně s odlišnými parametry, dosahují vyšší efektivity.

   d) Použití procedury – Procedury je třeba vytvářet opatrně, protože mohou způsobit SQL Injection. Chrání obdobně jako parametrizované příkazy, pokud jsou implementováný správně. Programátor by se měl vyhnout použití dynamicky generovaného SQL dotazu, jinak musí obsahovat validaci vstupu a hrozí SQL injection.

   e) Specializované nástroje – Pro bezpečnost databáze lze dotazy před vpuštěním otestovat. SQL Firewall porovnává strukturu dotazu

s připraveným seznamem povolených struktur. Problém tohoto nástroje je, že musíme vytvořit seznam povolených struktur. Dále existují nástroje pro testování, které napodobují chování útočníka, například sqlmap.

Vhodné je ale i ošetřovat výstupy z databáze, jestli neobsahují spustitelný skript, například tag `<script>`. V PHP se dá docílit ochrany použitím `htmlspecialchars`.[3] [4] [5]

2.1.2 Injekce souboru (File Inclusion)

Zde se útočí na soubory, jejichž jméno je dynamické a je ovlivnitelné uživatelem. Dělí se na 2 typy, Remote File Inclusion (RFI) a Local File Inclusion (LFI). Remote File Inclusion nastane, pokud server stáhne a načte vzdálený soubor zadaný útočníkem. Local File Inclusion je podobný jako Remote File Inclusion s tím rozdílem, že server nestahuje žádné soubory, ale spouští lokální soubory. Nejčastěji se útočí na include file skriptu. Local File Inclusion může být provedeno například pomocí log poisoning. Ten funguje tak, že při pokusu o přístup k neexistujícímu souboru se událost zapíše do logu. Útočník může zadat kód exploitu jako název souboru, to se zapíše do logu a nakonec útočník může pomocí LFI si nechat vypsat z logu, čímž může dojít ke spuštění kódu.

Dopady útoku se odvíjejí od toho, jaká má uživatel práva. Může spustit skripty se svými přístupy. Pokud má práva administrátora, tak i jeho skripty budou mít administrátorská práva.

Útok může vypadat jednoduše. Když server čte parametry z URL, tak je stačí přepsat, příklad URL: `index.php?file=/etc/passwd` a server potom čte

```
readfile($_GET["file"]);
```

Základ ochrany je podobný jako u SQL Injection, je třeba si dát pozor na vstupy uživatele. Je nutné ověřit jaké soubory se includují do skriptů. Je vhodné vyhnout se přímým odkazům na soubory z URL. Nejlepší je nepoužívat vstupy uživatele jako názvy souborů. Pokud to nejde, tak alespoň mít white list povolených souborů. Remote File Inclusion se dá zabránit tím, že se vypne načítání vzdálených souborů. V PHP se dá nastavit `allow_url_include` na 0. [6] [7]

2.1.3 Session Hijacking

Tento útok zneužívá HTTP cookie oběti útoku pro získání neoprávněného přístupu k informacím nebo službám serveru.

Uvedu 4 možnosti, jak se zmocnit session. Všechny jsou založené na získání session ID z HTTP cookie.

1. Session Fixation – Útočník podstrčí vlastní session ID, nevytvořené serverem, některému uživateli, například v odkaze. Uživatel na odkaz klikne a přihlásí se na server, čímž se session ID vytvořené útočníkem stane platným.
2. Session Sidejacking – Útočník odposlouchává nešifrovanou komunikaci mezi serverem a webovým prohlížečem. Pro přihlášení se sice většinou používá šifrovaná komunikace, ale následná komunikace již šifrovaná být nemusí. Odtud útočník může zjistit session ID a tím získat identitu jiného uživatele.
3. Cross-site scripting – Útočník se snaží spustit kód na počítači uživatele. Kód se musí tvářit jako důvěryhodný a že pochází ze serveru. Tím může získat kopii cookie nebo provádět další činnosti.
4. Malware – Malwary dokáží ukrást cookie z webového prohlížeče a provozovat další operace. Z cookie může útočník zjistit session ID a změnit svou identitu.

Existuje několik programů, které Session Hijacking umožňují. Většinou byly i volně stažitelné například na Google Play, samozřejmě byly z tohoto obchodu smazané.

- Firesheep – Pro Mozillu Firefox byl vydán doplněk který umožnil uživatelům se jednoduše zmocnit session na nešifrovaných veřejných Wi-Fi. Jedná se pouze o zachytávání cookies uživatelů.
- WhatsApp sniffer – Tato aplikace umožňovala číst zprávy všech lidí na stejné síti, protože aplikace WhatsApp využívala nešifrovanou komunikaci, sítí se přenášely zprávy v plain textu.
- Droid Sheep – Nástroj využívající sidejacking. Poslouchá na nezabezpečené síti a zachytává HTTP pakety, ze kterých získá session ID.
- CookieCadger – Aplikace fungující na principu sidejackingu. Využívá nezabezpečené HTTP GET požadavky.

Metod pro prevenci útoku je opět více a je vhodné jich použít, co nejvíce. Několik z nich popíši.

1. Používat šifrovanou komunikaci pomocí SSL/TLS. Šifrovaná komunikace znemožní útok fungující na principu odposlouchávání paketů.
2. Používat dlouhé náhodně vygenerované čísla jako session key. To ztíží uhodnutí klíče nebo prolomení hrubou silou.
3. Vygenerovat nový session id po přihlášení uživatele. Tato možnost znemožní sessionfixation, protože útočník neví, jaký má cílový uživatel aktuálně session id.

4. Více faktorů kontroly uživateli. Webový server může kontrolovat IP adresu, ze které je požadavek odeslán. Může se použít i další údaje o uživateli například formou hashe. Při zcizení session id, je téměř vyloučené, že hash bude stejný.

5. Měnit cookie při každém požadavku. Pro útočníka bude obtížnější zmocnit se údajů.

6. Nastavení expirace pro každou session, mazání session po odhlášení uživatele.

[7]

2.1.4 XML external Entity Injection (XXE)

Útok je zaměřený na aplikace, které parsují XML vstup. Formát XML se často používá pro výměnu dat mezi aplikacemi, umožňuje definovat entity v rámci svého DTD. DTD znamená Document type definition, definuje strukturu, elementy a atributy XML dokumentu. Entity mohou být uloženy i v externím dokumentu. Aplikace v nich mohou číst data z externího dokumentu. Útočník ale může XML soubor zaměnit a zažádat o data z /etc/passwd. Špatně nakonfigurovaný parser mu to může umožnit. Útok může vést i k výpadku služeb (denial of service) tzv. XML bombou.

Příklad útoku, ve kterém se útočník snaží přečíst obsah souboru /etc/passwd:

```
<?xml version="1.0 ?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "file:///etc/passwd" >
]>
<foo>&xxe;</foo>
```

Příklad XML bomby znázorňující DoS útok:

```
<?xml version="1.0 ?>
<!DOCTYPE lolz [
<!ENTITY lol "lol">
<!ENTITY lol2 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
<!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
<!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
<!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
<!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">
<!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">
<!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">
<!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">
]>
<lolz>&lol9;</lolz>
```

Chránit se před XXE, lze několika způsoby, jako v předchozích případech se vyplatí použít, co nejvíce možností:

1. Zakázat zpracování DTD directiv v XML procesoru.
2. Zakázat zpracování externích entit v XML procesoru.
3. Použít XSD (alternativa pro DTD) pro ověření správného tvaru vstupních dokumentů.
4. Použít whitelisting pro povolené XML elementy ve vstupu.
5. Použít jednoduší datové formáty, např. JSON.

[7] [8] [9]

2.1.5 Cross-site scripting (XSS)

V tomto útoku se opět zneužívá především chyby špatně ošetřeného vstupu od uživatele. Útok spočívá v tom, že útočník dokáže na stránky dostat svůj javascriptový kód. Často využíván k phishingu, tedy k získání citlivých údajů ze stránek, kam by uživatel neměl mít přístup. Způsob útoku můžeme rozdělit na 3 typy:

1. Stored XSS Attack – Typ útoku, kdy škodlivý kód je permanentně uložen na serveru, například v databázi, ve zprávě na fóru, v komentáři a podobně. Oběti se spustí skript jakmile navštíví stránky (v případě umístění kódu například ve zprávě na fórum) nebo při vyžádání dat z databáze.
2. Reflected XSS Attack – Tento útok se zakládá na tom, že útočník pošle uživateli link na některou stránku a uživatel na něj musí kliknout. Útok se posílá jako parametr v URL, který se interpretuje do stránky. Může vypadat například takto:

   ```
   http://example.com/page?var=<script>alert('Útok')</script>.
   ```

3. DOM based Attack – Zde se opět posílá útok ve formě parametru v URL. Rozdíl oproti předchozímu typu je, že se útočí na javascriptový skript na serveru. Kód na stránkách může vypadat například takto:

   ```
   <script>
   var pos=document.URL.indexOf("var=")+6;
   document.write(document.URL.substring
   (pos,document.URL.length));
   </script>
   ```

   útočník do url napíše toto:

```
http://example.com/stranka.html?var=<script>
alert('XSS útok');
</script>
```

Možnosti obrany proti XSS:

1. Nikdy nevkládat neprověřená data. Všechny vstupy uživatele musí být ošetřené. Je vhodné escapovat všechen uživatelský vstup.
2. Používat whitelisting.
3. Používat Content Security Policy. Jedná se sice o nedokončený W3C standard, ale široce podporovaný v prohlížečích. Definuje HTTP hlavičku, pomocí které server sděluje prohlížeči, který externí obsah smí být načítán.
4. Používat knihovny, které jsou určeny ke zpracování vstupu od uživatele, například v javascriptu HtmlSanitizer nebo HTML Purifier v PHP.
5. Proti DOM Based útoku se dá chránit pomocí přiřazení do proměnné `element.textContent`. To zabrání ve spuštění škodlivého kódu.

[7] [10] [11] [12]

2.2 Systémy IDS a IPS

2.2.1 Intrustion Detection System (IDS)

IDS je obranný systém, který monitoruje systém či provoz na síti, vyhodnocuje podezřelé aktivity a vhodně doplňuje firewall. Firewall doplňuje tím způsobem, že zatímco firewall kontroluje pouze hlavičky TCP/IP, tak IDS prověřuje i obsah každého paketu. V případě detekce podezřelé aktivity vygeneruje varování (alert), zapíše událost do logu nebo jiným způsobem upozorní administrátora. Dokáže zjistit zda útok přišel z vnitřní nebo z externích sítí. Hlavním prvkem IDS je senzor, který obsahuje mechanismy pro detekci škodlivých a nebezpečných kódů, jeho hlavní činností je odhalování nebezpečí.

Systém IDS zajišťuje:

- Monitoruje a analyzuje aktivity systému a uživatele
- Kontroluje systémové konfigurace a zranitelnosti
- Posuzuje integritu kritických systému a dat
- Statisticky analyzuje aktivity založené na shodě se známými útoky
- Analýza abnormálních aktivit

- Zajišťuje kontrolu operačního systému

Intursion detection system obsahuje 3 hlavní komponenty:

1. Network Intrusion Detection System (NIDS) – Provádí analýzu provozu na celé síti. Porovnává provoz na síti s knihovnou známých útoků. V případě zjištění útoku varuje administrátora. Senzory jsou umístěny přímo na síťových prvcích např. na HUB, SWITCH a dalších.
2. Network Node Intrusion Detection System (NNIDS) – Provádí analýzu provozu na síti, který se uskutečňuje mezi sítí a určitým subjektem v síti. Rozdíl mezi NIDS a NNIDS je zřejmý, NIDS monitoruje celou síť, kdežto NNIDS provoz jen u jednoho hosta. Příkladem je instalace NNIDS na VPN zařízení, aby se zkoumala doprava poté, co je dešifrována.
3. Host Intrusion Detection System (HIDS) – Dělá snapshoty systému a porovnává je. V případě, že důležité systémové soubory byly upraveny nebo smazány, upozorní administrátora systému.

Inturosion Detection System není odpověď na všechny bezpečnostní problémy. Shrnu několik bodů, které objasní, co IDS má za úkol dělat a co naopak ne. IDS může poskytnout:

- přidá větší stupeň integrity do infrastruktury
- dokáže sledovat aktivitu uživatele, odkud kam posílá pakety
- dokáže rozpoznat a ohlásit změny dat
- automatizuje úlohu sledování internetu při vyhledávání nejnovějších útoků
- dokáže detekovat, když je systém pod útokem
- může detekovat chyby v konfiguraci systému
- může navádět správce systému při vytváření politik v systému
- může zajistit, aby bezpečnostní managment systému mohl provádět neodborný personál

Naopak co IDS neposkytuje:

- není možné tím kompenzovat slabé mechanismy identifikace a autentizace
- není možné zkoumat útoky bez zásahu člověka
- nekompenzuje slabiny síťových protokolů
- nekompenzuje problémy v kvalitě informací, které systém poskytuje

- nemůže analyzovat veškerý provoz v zaneprázdněné síti
- nedokáže se vždy vypořádat s útoky na úrovni paketů
- rozhodně není antivirový program, firewall

2.2.2 Intrusion Prevention System (IPS)

IPS jsou zařízení umožnující útokům předcházet. Jsou považovány za rozšíření systémů IDS. Rozdíl mezi těmito systémy je, že zatímco systémy IDS slouží primárně k detekci, IPS jsou schopny síť ochránit, díky schopnosti zahodit pakety, zamezit přístup nebo blokovat celé spojení.

IPS systémy se dělí do čtyř kategorií:

1. Network-based Intusion Prevention System (NIPS) – provádí analýzu celé sítě
2. Wireless Intrusion Prevention System (WIPS) – monitoruje bezdrátovou síť na podezřelou aktivitu
3. Network Behavior Analysis (NBA) – zkoumá vnitřní síťový provoz kvůli identifikaci hrozeb, které generují neobvyklý provoz na síti, jako například útoky DDoS, určité formy malware
4. Host-based Intrusion Prevention System – monitoruje pouze jeden počítač na podezřelou aktivitu analýzou událostí, které se z počítače dějí

IPS po detekování podezřelé aktivity spouští příslušné akce. Několik z nich uvedu a stručně vysvětlím, co jednotlivé akce znamenají, pokud to nebude z názvu zřejmé.

1. Akce spouštějící výstrahu – Nejčastěji upozorňuje na útok pomocí vygenerované zprávy.
2. Akce zahazující provoz – Dokáže zastavit útok ještě předtím, než má příležitost zasáhnout cílovou oblast. IPS má 3 možnosti zahození provozu:
   a) Zahození paketu
   b) Zahození skupiny paketů daného spojení. Může být definováno na základě zdrojové IP adresy, cílové IP adresy, cílového portu a podobně.
   c) Zahození paketů na základě zdrojové IP adresy.
3. Akce záznamu podezřelého provozu – Pouze zaznamená podezřelou aktivitu k dalšímu prozkoumání.
4. Akce blokování – Dokáže zablokovat provoz na vzdálených místech v síti.

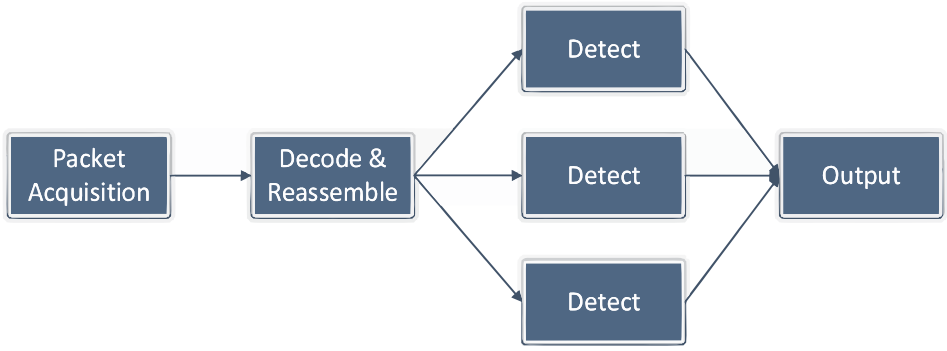
5. Akce resetování TCP spojení – Slouží k náhlému ukončení TCP spojení.
6. Akce povolení – Přidání výjimky do pravidel.

[13] [14]

2.3 Suricata

Suricata je software kombinující IDS i IPS. Jedná se o open source projekt, volně stažitelný z internetu. Software je napsán v jazyce C, ale obsahuje několik doplňků napsané v pythonu. Suricata pracuje na základě předem určených pravidel a obsahuje i podporu pro Lua skripty pro detekci komplexnějších hrozeb. Projekt Suricata a kód je ve vlastnictví neziskové organizace Open Information Security Foundation (OISF).

Software je podporován operačními systémy Linux, Windows, MacOS, FreeBSD a OpenBSD. Obsahuje Network Intrusion Detection System, Network Intrusion Prevention System a Netwrok Security Monitoring. Software se konfiguruje pomocí YAML konfiguračního souboru. Mezi hlavní výhody formátu YAML patří především lidsky čitelná forma. Velkou výhodou softwaru Suricata oproti jiným systémům je vícevláknové zpracování paketů, tím dosahuje vysoké výkonnosti, a proto se hodí k analýze vysokorychlostní sítě.



Obrázek 2.1: Zřetězené zpracování paketů [1]

Suricata funguje na základě zřetězeného zpracování (viz obr. 2.1). Každou část zpracovává jiné vlákno. Zpracování probíhá ve čtyřech krocích:

1. Zachycení paketu ze síťového rozhraní (Packet Acquisition) – Tato funkce je implementována ve více modulech, která podporují různá rozhraní pro zachycení paketu, např. pcap, Netmap.
2. Dekódování a sestavení síťového toku (Decode and Reassemble) – Zde se dekódují data a řídí se sestavení síťového toku.

3. Analýza paketu dle definovaných pravidel (detect)

4. Generování hlášení (Output)

2.4 Porovnání softwaru Suricata s konkurenčním Snortem

Suricata má výhodu ve vícevláknovém zpracování paketů. Díky tomu se pro analýzu sítě nemusí spouštět více nezávislých instancí softwaru. Také díky vícevláknovému zpracování dosahuje Suricata vyšší přesnosti v detekci hrozeb než Snort, při zahlcení systému jsou pakety nekontrolovatelně zahazovány, u jedno vláknového zpracování se zahazuje více paketu než při vícevláknovém. Suricata přebírá užitečné vlastnosti z jiných IDS nástrojů. Například skriptovací engine LuaJIT, který umožňuje vytvoření uživatelského skriptu, který se spouští při přijetí každého paketu a může tak detekovat i komplexnější hrozby, které nejsou obsaženy v pravidlech. Co se ale týká uživatelského rozhraní, tak Snort mi přišel uživatelsky přívětivější. Jednodušší instalace, především na operačním systému Windows, snazší výběr interface, na kterém naslouchá. Oba systémy podporují stejnou sadu pravidel, dají se použít pravidla ze stejné databáze. [15] [16] [17]

2.5 Big Cloud

Big Cloud je manažer cloudových úložišť vytvořený na základě cloud computingu. „*Cloud computing znamená, dodávání výpočetních služeb, jako jsou servery, úložiště, databáze, sítě, software, analytické nástroje a další, přes internet.*“ [18] Umožňuje správu virtuálních strojů, na kterých běží operační systémy Windows nebo GNU Linux. Při vytváření strojů má zákazník možnost vybrat si operační systém, velikost paměti RAM, velikost datového úložiště a další. [19]

2.6 Architektura systému Big Cloud

Při ukázce architektury systému Big Cloud nejprve uvedu vnější pohled na backendovou část systému Big Cloud a následně detailnější pohled.

2.6.1 Backend

Část webové aplikace, která slouží k administraci webu a ke zpracování dat. Jeho základní funkcí zde je na základě příkazů z ostatních částí systému vytvářet a konfigurovat virtuální stroje (VM) a zajišťovat jejich chod. Implementuje pouze potřebné funkce.

Na diagramu 2.3 je detailní pohled na backend po dekompozici.

Řídící node Stroj, na kterém je pouze software řídící chod cloudu.

Manažer stavu „*Spravuje data z výpočetních nodů zajišťuje běh periodických procesů.*“

Konfigurační server „*Zajišťuje distribuci konfiguračních dat od manažera stavu k jednotlivým konfiguračním klientům ve výpočetních nodech. Řídí také získání dat z výpočetních nodů.*“

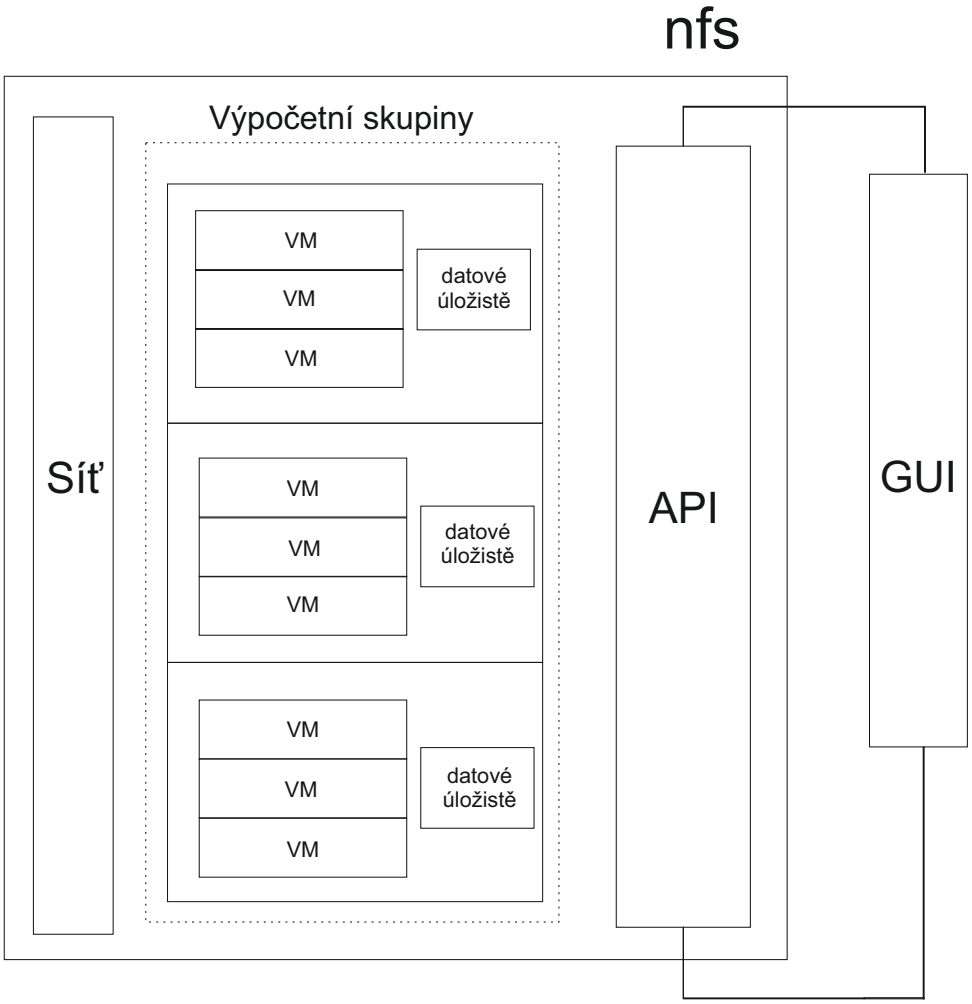
API Backend implementuje HTTP REST API pro komunikaci s ostatními části systému. Umožňuje vytvářet, mazat a měnit parametry virtuálních strojů. Poskytuje pro každý virtuální stroj informace o jeho stavu (zapnuto, vypnutou a umožňuje ho i měnit (zapnout, vypnout, restartovat). Je dostupné po síti zbytku systému. Informuje ostatní části systému o dostupných prostředcích (RAM, CPU, místa na disku).

Síťový subsystém „*Zajišťuje virtuálním strojům přístup k síti.*“ Rozděluje do VLANu.

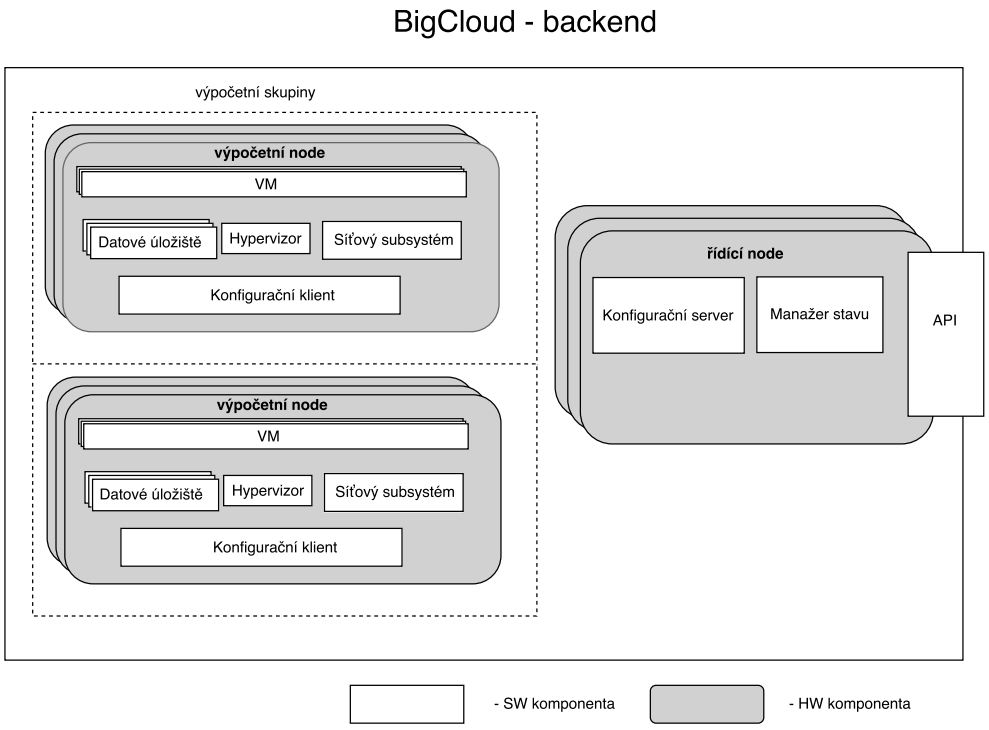
Hypervizor Základní součástí virtualizace. Software zajišťující běh virtuálních strojů a řídí přidělování prostředků k nim. Umožňuje běh několika operačních systémů.

Virtuální stroj Nezávislé operační prostředí, které pracuje s hostitelským operačním systémem.

[2] [20]



Obrázek 2.2: Big Cloud backend - vnější pohled [2]



Obrázek 2.3: Big Cloud backend - detailnější pohled [2]

Kapitola 3

Návrh

3.1 Monitorování softwaru Suricata

Pro monitorování stavu softwaru Suricata, jsem se rozhodl nevyvíjet nový software, který by s softwarem Suricata komunikoval, ale použiji již existující Suricatasc. Nainstaluje se automaticky se softwarem Suricata. Popíši zde, jak software funguje, který se nainstaluje s verzí softwaru Suricata 4.0.4. Funkčnost Suricatasc se totiž často mění.

3.1.1 Suricatasc

Suricatasc je skript vytvořený v jazyce Python, který umožňuje komunikovat se softwarem Suricata za použití Unix soketů. Byl napsán, stejně jako software Suricata, neziskovou organizací Open Information Security Foundation. Funguje na bázi JSON protokolů. Pro správnou komunikaci se softwarem Suricata musí být v YAML konfiguračním souboru softwaru Suricata povolený `unix-command` takto:

```
unix-command:
  enabled: yes
  filename: /var/run/suricata-command.socket
```

V mém případě se software Suricata nainstaloval s tímto povolením automaticky. Zjistit, zda je, či není tato možnost povolená, se dá zjistit buď otevřením YAML konfiguračního souboru, nebo pomocí příkazu `suricata --build-info` a je potřeba najít `Unix socket enabled:`.

`/var/run/suricata-command.socket` je defaultní hodnota. Software Suricata se dá spustit s parametrem `--unix-socket`, čímž se specifikuje, na kterém poběží soketu, potom se musí změnit filename.

Suricatasc se dá spustit, jak v konzoli, tak i v interaktivním režimu. Teď udělám přehled několika užitečných příkazů. Všechny příkazy musí být provedené s administrátorskými právy.

- `suricatasc -c uptime` – vrátí se stav softwaru Suricata
- `suricatasc -c version` – vrátí verzi softwaru Suricata
- `suricatasc -c shutdown` – vypne software Suricata
- `suricatasc -c iface-list` – zobrazí seznam dostupných interface
- `suricatasc -c command-list` nebo `suricatasc -c help` – zobrazí seznam příkazů

Na jednotlivé stroje se půjde připojit pomocí SSH. Jelikož pro práci se Suricatasc jsou potřeba administrátorská práva, tak zde může být bezpečnostní riziko, aby někdo neoprávněný nemohl zneužít administrátorská práva. Je třeba vytvořit speciální uživatele, který bude mít právo na `sudo` a bude tím moct spustit pouze Suricatasc a Suricata s root právy.

3.2 Sběr logů ze softwaru Suricata

Software Suricata ukládá logy defaultně do `/var/log/suricata/` v této složce se nachází čtyři textové soubory, podrobněji o nich napíši v sekci dokumentace. Potenciální hrozby a události se zapisují v JSONu do souboru eve.json.

Pro tento účel vytvořím program napsaný v jazyce C++, který bude periodicky posílat na určité místo na serveru logy a pokud dopadne přenos úspěšně, tak je z virtuálním stroji smaže.

Využiji v programu Suricata novinky s názvem „Rotace logovacích souborů“. Rotace logovacího souboru eve.json funguje tak, že software Suricata sám po určité době, danou v konfiguračním souboru, začne zapisovat události do jiného souboru. Tato funkce mi umožní posílat pouze logy, které ještě nebyly poslány a zároveň budu mít jistotu, že software Suricata nebude chtít do těchto souborů nic zapsat, čili neriskuji ztrátu informací. Můj program bude využívat možnost ukládání logu, kde do názvu souboru se přidá timestamp a bude posílat všechny kromě nejnovějšího, protože do něj se ještě může zapisovat.

V případě, že v intervalu mezi odesíláním logů nenastane žádné varování, tak se budou odesílat prázdné zprávy. Jenže to může znamenat také, že software Suricata spadl. Proto bude mít můj program za úkol nejen odesílat logy, ale také bude hlásit stav softwaru Suricata. Bude pravidelně spouštět skript Suricatasc, na detekci stavu. Vzhledem k časté změně struktury tohoto skriptu, bude spouštění skriptu Suricatasc řešeno přes externí skript, defaultně pojmenovaný `SuricataAlive.sh`.

Odesílat na server se budou data také ve formátu JSON. Pokud vše proběhne, jak má, odešle se zpráva `{'message': @cislo, 'return': 'OK'}`, kde @cislo bude celočíselná proměnná, který se inkrementuje s každým voláním. Pokud software Suricata nebude fungovat správně pošle se `{'message': 0, 'return': 'FAILED'}`.

3.3 Návrh databáze pro uložení logů

Vzhledem k tomu, že logy ze softwaru Suricata se ukládají ve formátu JSON, tak pro architekturu databáze je vhodné použít MongoDB.

MongoDB je databáze, napsaná v jazyce C++, která se řadí mezi tzv. NoSQL databáze a místo relačních databází využívá dokumenty ve formátu JSON.

Každý virtuální stroj bude mít svůj záznam v tabulce. Databáze bude již na serveru vytvořena a program bude pouze do ní posílat data.

Tabulka bude vypadat takto:

- vm_id – integer – id virtuálního stroje, zjistí se podle api-key, který se bude posílat ve zprávě
- status – boolean – atribut, který ukazuje, zda v některé zprávě je přítomna událost typu alert
- time – timestamp – čas posledního zápisu
- messages – json – seznam zpráv

První 3 položky budou řízené serverem a jejich implementace není součástí mé bakalářské práce. Program bude posílat zprávy a ty se budou ukládat do proměnné messages. Nyní popíši, jak vypadá struktura události typu alert.

- timestamp – string – doba ve které událost nastala
- flow_id – integer – rozřazovací identifikátor pro toky
- in_iface – string – název interface
- event_type – string – typ události, například alert
- src_ip – string – optional – zdrojová IP adresa
- src_port – integer – optional – zdrojový port
- dest_ip – string – optional – cílová IP adresa
- dest_port – integer – optional – cílový port
- proto – string – optional – protokol, například ICMP, TCP
- icmp_type – integer – optional – pouze pokud je protokol ICMP
- icmp_code – integer – optional – pouze pokud je protokol ICMP
- tx_id – integer – optional – může se vyskytnout u protokolu TCP

- app_proto – string – optional – detekce protokolu aplikační vrstvy, například SSH, může ale skončit výsledkem `failed`, pokud software Suricata protokol nepozná nebo se pomocí pravidel paket zahodí dříve, než dojde k detekci

Jednotlivé protokoly mají různé parametry, například pro ICMP to je `icmp_type` a `icmp_code`.

Tyto atributy jsou společné pro většinu zpráv, vlastně pro všechny kromě statistiky. Teď vypíši seznam atributů, které se vyskytují v položce alert. Je to struktura opět ve formátu JSON.

- action – string – může obsahovat dvě hodnoty, buď allowed nebo blocked
- gid – integer – group id pravidla, defaultně nastaveno na 1
- signature_id – integer – sid pravidla
- rev – integer – číslo verze pravidla
- signature – string – shoduje se s položkou msg v pravidle
- category – string – optional – v pravidle je položka classType, která odkazuje do souboru `clasification.config`, ze kterého se použije název určující kategorii
- severity – integer – závažnost, zjistí se z položky classType stejně jako u položky category

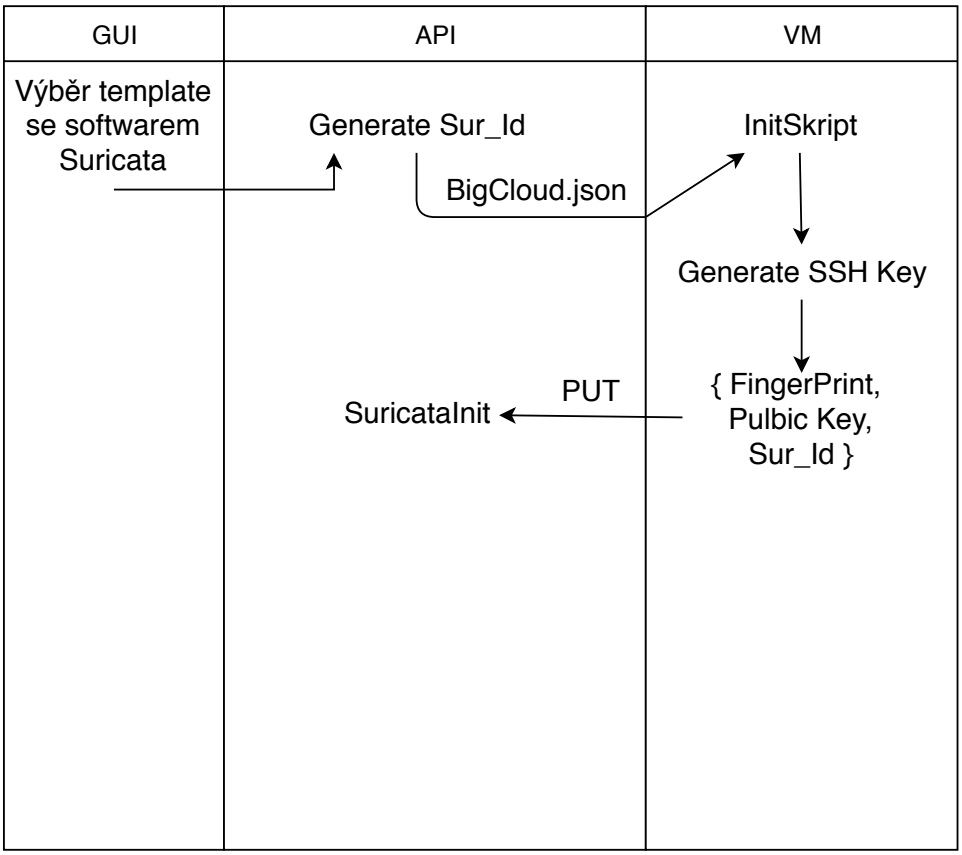
Struktura ostatních typů událostí bude pouze v příloze.

3.4 Nasazení na Big Cloud

V této části uvedu, jakým způsobem se software Suricata integruje do systému, a uvedu dva způsoby, jak si uživatel bude moct nainstalovat software Suricata na svůj virtuální stroj. Nejprve uvedu způsoby, jakým se to bude testovat, jelikož mám přístup do privátní sítě přes VPN, tak se budou data posílat na privátní IP. Ve skutečnosti se ale budou záznamy odesílat na veřejnou IP, což s sebou přináší jisté bezpečnostní riziko, jakým způsobem se implementuje na serveru ukáži později.

První možností, jak si uživatel bude moct nainstalovat software Suricata na svém systému je, že ve výběru šablon operačních systému budou systémy s již nainstalovaným softwarem Suricata. Když takto zvolí, tak API vrstva vygeneruje náhodné unikátní `sur_id`, které bude sloužit jako api-key pro identifikaci stroje, vloží ho do souboru BigCloud.json, který se pošle na nově vytvořený virtuální stroj. Na virtuálním stroji se spustí inicializační skript

`SuricataInit.sh` napsaný v jazyce Bash, který uloží api-key do konfiguračního souboru programu určeného k odesílání logů ze softwaru Suricata, vygeneruje SSH klíč a následně odešle zpátky na API do endpointu `SuricataInit` trojici api-key, veřejný klíč a fingerprint, která se tam uloží do databáze. Fingerprint se posílá proto, aby se SSH připojení neptalo na ověření fingerpritnu a mohlo být provedeno automaticky. Veřejný klíč je také kvůli automatizaci SSH připojení. Schéma komunikace je vidět na obrázku 3.1.



Obrázek 3.1: Schéma nasazení při vytvoření

Uživatel ale nemusí vybrat software Suricata ihned ze šablony, ale rozhodne se až poté, co si vytvoří virtuální stroj. V tomto případě bude mít možnost nainstalovat software pomocí balíčku, který bude umístění v repozitáři Big Cloudu s názvem `bigcloud-suricata`. Ten nejprve stáhne všechny potřebné skripty i program na odesílání logů. Pak se spustí instalační skript `SuricataInstall.sh`, který nainstaluje software Suricata a zároveň se zaregistruje v API vrstvě skriptem `SuricataReg.sh` a inicializuje program na posílání logů. Registrace bude to vypadat tak, že se odešle dvojice MAC ad-

resa a IP adresa na endpoint `SuricataReg`, kde se nejprve zkontroluje, zda se nepřijaly podvržené údaje, a pokud budou v pořádku, tak vygeneruje unikátní identifikátor `sur_Id`, který se pošle zpátky na virtuální stroj. Na virtuálním stroji se spustí inicializační skript `SuricataInit.sh`, stejný jako v případě, kdy si uživatel zvolí šablonu s předinstalovaným softwarem Suricata již v GUI při vytváření virtuálního stroje. Spustí se ale s parametrem `sur_Id`, který se použije později jako api-key. Další volání vypadají stejně jako v předchozím případě. Schéma komunikace je opět vidět na obrázku 3.2.

Jak jsem již zmínil, tento pohled není zcela bezpečný, pokud budou jednotlivé adresy, na které se odesílají zprávy, veřejné. Ve finální verzi se bude muset `SuricataAPI`, ve kterém bude MongoDB, endpointy `SuricataReg`, `SuricataLog` a `SuricataInit`, oddělit a komunikovat se bude přes firewall, aby se případní útočníci nemohli nikam neoprávněně dostat. Na obrázku 3.3 je vidět schéma komunikace. Virtuální stroje přistupují na `SuricataAPI` přes veřejnou IP adresu. O výměnu dat, validaci MAC adresy s IP adresou a generování api-key se stará GUI. GUI má také za úkol odesílat data ze `SuricataAPI` do API vrstvy, kde se budou data ukládat do již finální relační databáze.

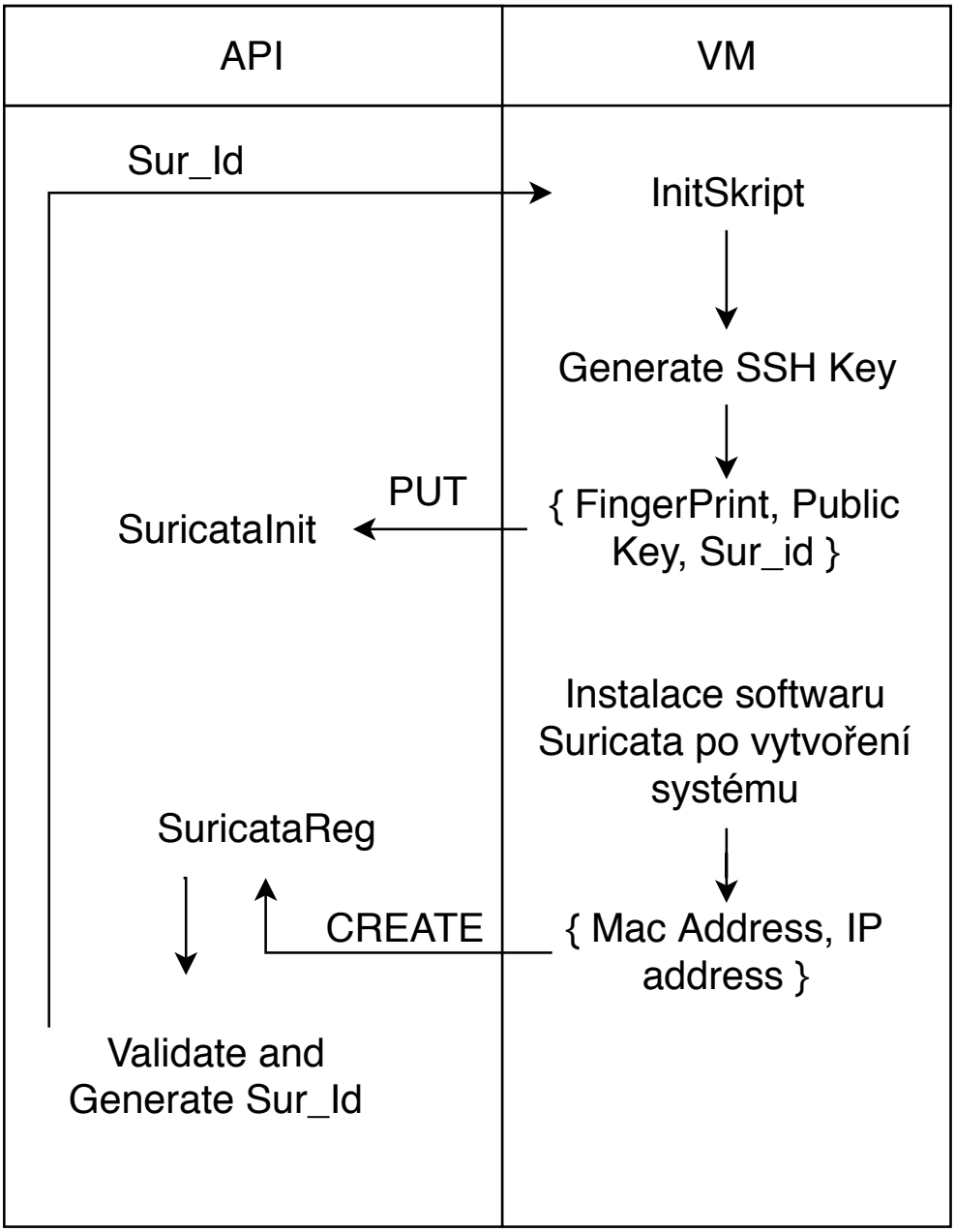
Při instalaci uživatelem po vytvoření virtuálního stroje, se může vygenerování api-key znázornit na stavovém diagramu 3.4. Uživatel spuštěním instalace softwaru Suricata, vytvoří požadavek na vytvoření api-key, který se odešle na endpoint `SuricataReg`. Obsahem požadavku je dvojice MAC adresa a IP adresa. V endpointu `SuricataReg` je požadavek připraven k validaci a odesílá se na GUI. V GUI dojde k samotné validaci a pokud není v pořádku, tak se api-key nevytvoří a je zde konec. Pokud ale vše dopadne správně, tak se vygeneruje api-key, který se odešle zpět na `SuricataReg` a odtud zpět na virtuální stroj. Na virtuálním stroji se uloží a odešle se spolu s vygenerovaným veřejným klíčem a fingerprintem na endpoint `SuricataInit`, kde se uloží do databáze.

Pro automatické spuštění softwaru Suricata nebudu vytvářet vlastní skript, ale použiji již existující z oficiálních stránek dokumentace softwaru Suricata. Umístí se do `/etc/init` a bude mít název `suricata.conf`. V tomto skriptu upravím ale dvě věci. Nejprve změním interface, aby sedělo s tím, které bude na virtuálním stroji. Druhá věc bude, že tam přidám spuštění mého programu určeného k odesílání logů. [22]

3.5 Zabbix

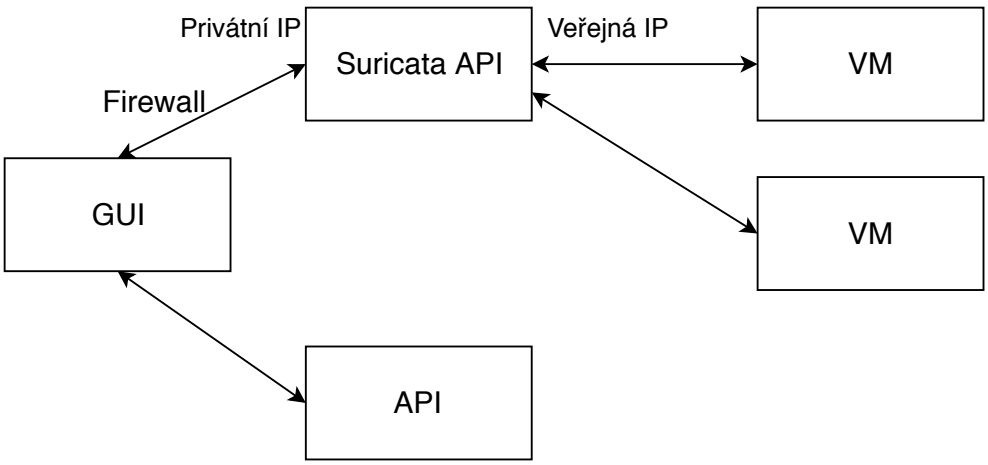
V případě, že by se objevila událost typu alert v logách, tak by se mohlo o této skutečnosti ihned informovat administrátora serveru a další členy. K tomu by právě mohl posloužit software Zabbix.

Zabbix je software určený k monitorování provozu na síti, je to open source distribuce, stejně jako software Suricata. Je možné do něj také integrovat své vlastní skripty.

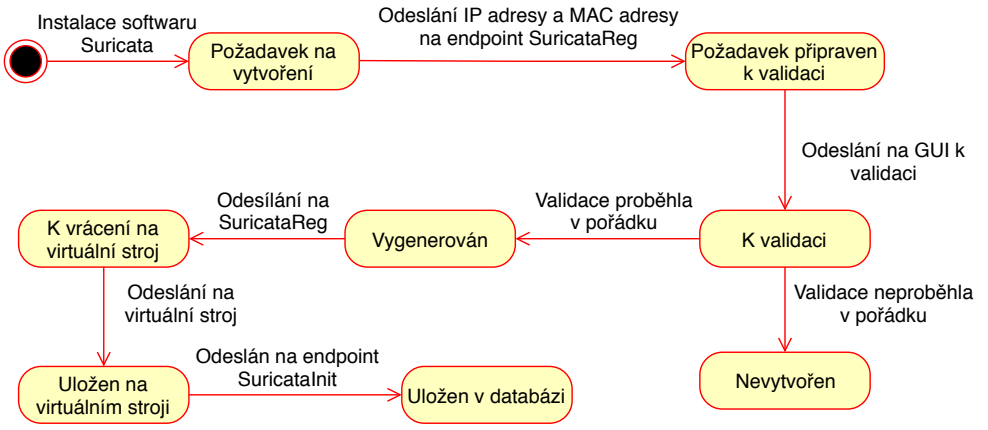


Obrázek 3.2: Schéma nasazení

Logovací program by posílal logy právě nejdříve přes Zabbix, který by procházel jednotlivé logy a v případě zjištění varování by mohl informovat administrátora například prostřednictvím SMS zprávy. [21]



Obrázek 3.3: Finální schéma komunikace



Obrázek 3.4: Diagram vytvoření api-key

Kapitola 4

Instalace a dokumentace

4.1 Popis instalace softwaru Suricata

Instalaci softwaru Suricata popíšu pro operační systémy Windows a Debian. Instalace bude zaměřena pro verzi systému Debian nasazených na systému Big Cloud a pro Windows 10.

4.1.1 Instalace Debian

Nejprve začnu s operačním systémem Debian. Pro instalaci je nutné mít administrátorská práva. V prvních pěti krocích připravuji operační systém, aby byla nejnovější stabilní verze softwaru Suricata.

1. `sudo sed -i 's/jessie/stretch/g' /etc/apt/sources.list` – v Jessie repozitáři se vyskytuje stará verze softwaru Suricata 2.0.7, ve Stretch je novější 4.0.4
2. `sudo echo "deb http://ftp.debian.org/debian stretch-backports main" >> /etc/apt/sources.list.d/backports.list`

   – software Suricata se bude instalovat z backportu stretch-backports
3. `sudo apt-get update` – první aktualizace systému, sice skončí chybou, ale musí být provedena
4. `sudo aptitude install debian-keyring debian-archive-keyring` – řešení problému s public key v předchozím updatu
5. `sudo apt-get update` – další aktualizace systému, tentokrát projde bez chyb
6. `sudo aptitude install suricata -t stretch-backports` – instalace softwaru Suricata

7. `oinkmaster -u http://rules.emergingthreats.net/open/suricata/emerging.rules.tar.gz -o /etc/suricata/rules`

   – příkaz oinkmaster stáhne pravidla do složky rules

8. `sudo mv -f /etc/suricata/rules/reference.config /etc/suricata/reference.config`
   `sudo mv -f /etc/suricata/rules/classification.config /etc/suricata/classification.config`

   – těmito příkazy se přetáhnou nově stažené konfigurační soubory

Tímto je software Suricata nainstalován a připraven ke spuštění. Všechny příkazy budou v jednom instalačním skriptu, aby bylo možné software Suricata pohodlně nainstalovat.

4.1.2 Instalace Windows 10

Aby software Suricata mohl fungovat na operačním systému Windows, je nezbytně nutné nainstaloval WinPcap, například odtud `https://www.winpcap.org/install/default.htm`. Popíši instalaci stažením instalačního souboru ze stránek softwaru Suricata.

Stažení instalačního souboru. Ke dni 25.3. je nejnovější verze 4.0.4, jenže bohužel mají momentálně na stránkách `https://suricata-ids.org/download/` chybný instalační soubor, takže nepůjde použít. Jde ale stáhnout odjinud `https://redmine.openinfosecfoundation.org/projects/suricata/files`.

Stažení pravidel. Pravidla se dají pohodlně stáhnout odsud `http://rules.emergingthreats.net/open/suricata/emerging.rules.tar.gz`. Obsah poté extrahovat do kořenové složky Suricata. V archivu je složka rules, tak tou nahradit stávající složku rules v kořenové složce Suricata.

Stažení threshold.config Soubor threshold.config chybí a je vyžadován, proto je třeba ho také stáhnout, například odtud `https://github.com/StamusNetworks/Amsterdam/blob/master/src/config/suricata/threshold.config` a umístit ho do kořenové složky softwaru Suricata.

Vytvoření logovacích souborů. V některých verzích softwaru Suricata se nevytváří automaticky logovací soubory a je třeba je vytvořit ručně. Ve složce Suricata chybí logsuricata.log a potom ve složce Suricata/log pro změnu fast.log, eve.json a stats.log.

4.2 Suricata package

Jak jsem již psal software Suricata si bude moct uživatel stáhnout z repozitáře Big Cloud. Vytvořil jsem skript `SuricataCreatePackage.sh`, který vytvoří debian balíček pomocí příkazu fpm. Nainstalováním balíčku se nastaví všechny důležité části pro správnou funkci.

4.3 Použití softwaru Suricata

Jakmile je software Suricata nainstalován, tak vyzkouším, zda je správně nakonfigurován. K tomu slouží příkaz `suricata -T`, parametr -T spustí software Suricata v testovacím módu. V případě, že je změněno umístění konfiguračního souboru nebo software Suricata není nainstalován na defaultním místě, tak je nutné ještě specifikovat umístění konfiguračního souboru parametrem -c například takto `suricata -T -c suricata.yaml`.

Verzi softwaru Suricata také mohu zjistit z příkazové řádky a to příkazem `suricata -V`. Ke dni 26.3.2018 je verze 4.0.4, sice se už připravuje i verze 4.1.0, ale ta je zatím pouze v beta verzi.

Software Suricata se spouští na daném interface, ten se určuje parametrem -i. Pokud chci spustit software Suricata, aby naslouchal na eth0 a určuji i umístění konfiguračního souboru, tak příkaz vypadá takto:

```
suricata -c /etc/suricata/suricata.yaml -i eth0
```

Takto spuštěný software Suricata blokuje příkazovou řádku a pokud se příkazová řádka vypne, ukončí se i běh softwaru Suricata. Aby se to nestalo existuje přepínač -D, který spustí software Suricata jako daemon, takže poběží na pozadí, neblokuje příkazovou řádku a ani s ukončením příkazové řádky se software Suricata nevypne.

Pokud je software Suricata spuštěn v daemon módu, tak se může nastavit přepínačem –pid-file soubor, který bude uchovávat ID procesu, pokud se tento přepínač nepoužije, tak se vezme soubor z konfiguračního souboru.

4.4 Konfigurační soubor

Konfigurační soubor se defaultně jmenuje Suricata.yaml a nachází se ve složce s nainstalovaným softwarem Suricata. Je rozdělený na několik částí označených jako „step“.

V prvním kroku je nastavení sítě, na které bude Suricata pracovat. Nastavují se zde adresy `HOME_NET`, `EXTERNAL_NET`, `HTTP_SERVERS`, `SMTP_SERVERS`…. Dále se zde nastavují porty, `HTTP_PORTS`, `SSH_PORTS`, `FTP_PORTS`…

V druhém kroce je nastavení pravidel. Nejprve je zde možnost změnit cestu k pravidlům. Potom tu je seznam pravidel, musí se tu zakomentovat

všechny pravidla, které nejsou k dispozici, komentuje se znakem `#`. Pod seznamem pravidel se ještě vyskytují cesty k dalším konfiguračním souborům `classification.config`, `reference.config` a `threshold.config`.

V kroce číslo tři se nastavují výstupy ze softwaru. Nejdříve se nastavuje cesta ke složce, ve které budou soubory, do kterých se ukládají logy. Následuje tu základní nastavení ukládání statistik a pak dalších výstupů. Je to strukturovaný a dobře čitelný oddíl. Nachází se zde zapisování událostí, http, tls, dns, pcap a dalších. Většina z nich jede defaultně vypnuta, pozná se to tak, že pod každým typem logu je položka enabled, do které se přiřazují hodnoty yes nebo no. Parametrem, který nemusí být správně pochopen je append, který když je nastavený na `yes`, tak Suricata po každém zapnutí logy na sebe nabaluje, takže po nějaké době mohou být opravdu velké.

Položka eve.log je trochu rozsáhlejší a řekl bych také nejdůležitější. Proměnné jsou dobře pojmenované a srozumitelné, tak nebude třeba každou detailně popisovat. Bohužel ale některé zajímavé chybí jako například `rotate-interval`, který umožňuje pravidelnou změnu výstupu `eve-logu` a já této funkcionality využívám v programu na odesílání logů. Událostí je více jejich seznam je pod položkou types, jimiž jsou například alert, http, dns, tls a další. Všechny jsou v konfiguračním souboru srozumitelně popsány. Vypnout se dají zakomentováním názvu typu a celé jeho struktury. U některých se zde vyskytuje proměna extended, která umožňuje delší výstupy do logu, strukturu jsem popsal výše v části pojednávající o návrhu databáze.

V předchozích kapitolách jsem psal, že se zmíním o zapisování statistik. Statistiky se zapisují defaultně na 2 místa do stats.log a eve.json. Doba mezi zápisy se nastavuje do proměnné `interval`, defaultně je nastaveno 8 sekund, ale vyskytují se zde odchylky, mně při testech vycházela doba o sekundu nižší. Rozdíl mezi výstupem eve.json a stats.log je, že do stats.log se nevypisují výchozí hodnoty, píšou se tam pouze změny, kdežto v eve.json se vypisují všechny proměnné. Ve stats.log je to navíc i dobře čitelné pro člověka.

Ve čtvrtém kroce přichází obecná konfigurace zachytávání paketu, je potřeba se ještě podívat do rozšíření nastavení.

V pátém kroce je nastavení protokolů aplikační vrstvy. U jednotlivých protokolů se dá určit, zda se budou nebo nebudou zachytávat a zpracovávat. Je zde ještě možnost `detection-only`, která způsobí, že se protokoly budou jen detekovat, ale nebudou se nijak zpracovávat. U některých protokolů je možnost nastavení portů a dalších atributů.

Úplně na závěr se zde vyskytují už jen rozšiřující nastavení. Některá nastavení rozšiřují předchozí konfiguraci. V této části se dá upravit například pořadí v jakém se vyhodnocují pravidla, přidat vlastní konfigurační soubory, zlepšit výkon, zapnout profilaci pravidel, přidat vlastní potenciálně nebezpečné IP adresy, určit pod jakým uživatelem a skupinou se bude software Suricata spouštět a další specifikace.

Konfigurační soubory `classification.config` a `reference.config` slouží k tomu, že se do nich odkazují pravidla. Do `classification.config` se od-

kazuje atribut `classtype` a do `reference.config` atribut reference.

Posledním konfiguračním souborem je `threshold.config`. Ten slouží k tomu, aby redukoval generované varování kvůli pravidlům, které často hlásí do logu.

4.5 Pravidla

Pravidla jsou základem pro monitorovací zařízení. Jednotlivé pakety se porovnávají s pravidly a vyhodnocují se příslušné akce.

V této části popíši, jak vypadají pravidla, se kterými pracuje nejen software Suricata, ale i například software Snort.

Struktura pravidla se skládá ze tří částí:

1. akce – určuje, co se stane
2. hlavička – definuje protokol, ip adresy, porty a směr komunikace
3. možnosti pravidla – definuje specifikace pro pravidla

Jsou čtyři akce a vyhodnocují se podle daného pořadí. Pořadí se dá změnit v konfiguračním souboru.

- pass – přestane se skenovat paket, další pravidla se pro tento paket nebudou aplikovat
- drop – paket se zahodí, nepošle se dále, funguje pouze pokud je software Suricata zapnut v `IPS/inline módu`, příjemce nedostane žádnou zprávu a vygeneruje se alert
- reject – vyšle se reject paket odesílateli i příjemci, vygeneruje se alert, pokud je zapnuto v `IPS/inline módu`, tak se také zahodí paket
- alert – zapíše se varování do logu

V hlavičce je nejprve uveden protokol například tcp, udp, icmp, ip, http, ftp…. Následuje dvojice adresa, port zdroje a adresa, port cíle. Mezi těmito adresami se ještě musí určit směr, buď -> (od zdroje k cíli) nebo <> (oba směry).

Příklad:

```
alert tcp 127.0.0.1 any -> any !80
```

Zbytek pravidla už jsou jen možnosti, píší se ve formátu `<keyword> : <settings>;` nebo může chybět položka `<settings>`.

Pravidlo, se kterým se pracuje vypadá například takto:

```
alert tcp $EXTERNAL_NET 25565 -> $HOME_NET any
(msg:"ET GAMES MINECRAFT Server response inbound";
flow:established,from_server; content:"|7B 22|"; depth:10;
classtype:policy-violation; sid:2021701; rev:1;
metadata:created_at 2015_08_21, updated_at 2015_08_21;)
```

Akcí pravidla je opět alert. Protokol, který se zachytává je tcp. Definice pro jednotlivé IP adresy `EXTERNAL_NET` a `HOME\_NET` se nachází v konfiguračním souboru. Teď popíši jednotlivé atributy v poslední části pravidla. Msg je zpráva, která se přiřadí v souboru s událostmi do položky `signature`. Flow určuje datový tok, `established` znamená, že už musí být vytvořen, `from_server` určuje směr. V položce content je string, který se bude vyhledávat v obsahu paketu. Poté následuje depth, což je rozšiřující atribut pro content a určuje kolik bytů se bude číst z jednotlivých paketů. Classtype je reference do souboru `classification.config`, v tomto případě do souboru s událostmi přiřadí do `category` zpráva „Potential Corporate Privacy Violation“ a do položky `severity` číslo 1 neboli nízká hrozba. Sid značí číslo pravidla. Rev určuje verzi pravidla. Metadata jsou přídavná data o pravidle. [23] [24]

4.6 Práce do budoucna

Mnou navržené řešení podporuje pouze jeden interface a to eth0. Jako práce do budoucna by mělo být možné uživateli umožnit přidat podporu pro více interface a nebo i pro jedno různé od eth0. Pro toto rozšíření je zapotřebí upravit skript suricata.conf, konfigurační soubor suricata.yaml a skript SuricataReg.sh. Ve všech 3 souborech je momentálně napevno nastavena hodnota eth0.

Kapitola 5

Testování

Nejprve si vytvořím základní test vygenerování varování, abych se ujistil, zda software Suricata hlásí, jak má. Nejprve je třeba se ujistit, zda je správně nakonfigurovaný `HOME_NET` v konfiguračním souboru `suricata.yaml`.

Nyní přejdu k základnímu testu. Vytvořím si nové pravidlo ve složce `/etc/suricata/roles` například `local.rules` a do něj vepíši `alert icmp any any -> 185.88.72.18 any (msg: "ICMP detected"; sid:10000001;)`. Tímto pravidlem říká, že když se někdo snaží poslat ICMP zprávu, například ping, na adresu 185.88.72.18, adresa z mé sítě, tak se vyvolá alert do logu. V konfiguračním souboru `suricata.yaml` je ale ještě třeba přidat toto pravidlo, ve druhé části pod položkou `rule-files`. [25]

Teď už jen stačí poslat ping na adresu 185.88.72.18 a za každý úspěšný ping se objeví v `eve.json` alert, který vypadá takto:


```
{
    "timestamp": "2018-04-19T14:15:26.157369+0200",
    "in_iface": "eth0",
    "event_type": "alert",
    "src_ip": "185.88.72.17",
    "dest_ip": "185.88.72.18",
    "proto": "ICMP",
    "icmp_type": 8,
    "icmp_code": 0,
    "alert":
    {
        "action": "allowed",
        "gid": 1,
        "signature_id": 10000001,
        "rev": 0,
        "signature": "ICMP detected",
```

```
        "category": "",
        "severity": 3
    }
}
```


Při testech na Big Cloudu po více než 10 minutách od zapnutí softwaru Suricata obsahoval log přes 124KB, což se rovná 293 záznamům.

Jedním z nejčastějších útoků, který je vedený na moji IP adresu, se zdá být útok, který je v pravidlech pojmenován jako „ET SCAN SSH BruteForce Tool with fake PUTTY version“. Ověřit, že se nejedná o chybný poplach, se dá například v logu `/var/log/auth.log`, ve kterém je vypsána spousta pokusů o ssh připojení.

Existuje několik způsobů jakým provést útok typu BruteForce SSH Scan a jeden z nich popíši. Využiji již vytvořený skript, volně dostupný na internetu, napsaný v jazyce Python. Skript využívá slovníku, podle kterého se snaží uhádnout heslo.

- `wget http://zeldor.biz/other/bruteforce/brutessh.zip` – skriptu včetně slovníku a dalších
- `apt-get install python-crypto python-paramiko` – nainstalování Pythonu

Po těchto příkazech je vše připravené, v `brutessh.zip` se nachází složka `brutessh` a v ní skript `brutessh.py`. Útok se spouští příkazem

```
python brutessh.py -h 185.88.72.17 -u root -d passlist.txt
```

parametrem -h určím adresu, na kterou se útočí, parametrem -u nastavuji uživatele, pod kterým se chci přihlásit a nakonec parametrem -d určím slovník, který obsahuje seznam hesel, která se budou zkoušet. [26]

Po spuštění skriptu software Suricata zareagoval tímto varováním:


```
{
    "timestamp": "2018-04-19T18:49:46.359320+0200",
    "flow_id": 420987471035288,
    "in_iface": "eth0",
    "event_type": "alert",
    "src_ip": "46.13.19.197",
    "src_port": 53588,
    "dest_ip": "185.88.72.17",
    "dest_port": 22,
    "proto": "TCP",
    "alert":
    {
        "action": "allowed",
```

```
        "gid": 1,
        "signature_id": 2001219,
        "rev": 20,
        "signature": "ET SCAN Potential SSH Scan",
        "category": "Attempted Information Leak",
        "severity": 2
    },
    "flow":
    {
        "pkts_toserver": 1,
        "pkts_toclient": 0,
        "bytes_toserver": 66,
        "bytes_toclient": 0,
        "start": "2018-04-19T18:49:46.359320+0200"
    }
}
```


Časté jsou ale i varování typu `Active Threat Intelligence Poor Reputation`. Tyto hlášky znamenají komunikaci s blacklistovanou IP, například adresa 58.218.213.146 podle stránek `https://www.abuseipdb.com/check/58.218.213.146` vypadá, že se zaměřují hlavně na port scan.

Port scanning je dalším zdrojem varování v logu. Pro demonstraci jsem použil Kali Linux a nástroj `nmap`. Příkaz vypadal takto: `nmap 185.88.72.17 -p 1-65350`, tímto jsem skenoval porty 1–65350 na adrese 185.88.72.17. Software Suricata zjistil, že se někdo pokouší napojit na mySQL port 3306 a do logu vypsal:


```
{
    "timestamp": "2018-04-19T19:20:18.174285+0200",
    "flow_id": 376835327305933,
    "in_iface": "eth0",
    "event_type": "alert",
    "src_ip": "46.13.19.197",
    "src_port": 63317,
    "dest_ip": "185.88.72.17",
    "dest_port": 3306,
    "proto": "TCP",
    "alert":
    {
        "action": "allowed",
        "gid": 1,
        "signature_id": 2010937,
        "rev": 3,
```

```
        "signature": "ET SCAN Suspicious inbound
         to mySQL port 3306",
        "category": "Potentially Bad Traffic",
        "severity": 2
    },
    "flow":
    {
        "pkts_toserver": 1,
        "pkts_toclient": 0,
        "bytes_toserver": 60,
        "bytes_toclient": 0,
        "start": "2018-04-19T19:20:18.174285+0200"
    }
}
```


V logu se ale často útoky pomocí tzv. SIPVicious softwarů. SIP (Session Initiation Protocol) je protokol sloužící k vytvoření spojení mezi 2 objekty, hlavně využívaný v telefonii. [27] Tento typ útoku nehrozí pokud není podpora SIP. Hlavní snahou tohoto útoku je prolomit heslo.

Software Suricata ho indikuje, když packety jsou podepsané user agentem „friendly-scanner“. Otestoval jsem to opět pomocí Kali Linuxu a softwaru svmap, dá se použít i například svcrack. `svmap 185.88.72.17`. Tento příkaz hledá SIP komponenty v síti [28]. Software Suricata vypsal do logovacího souboru následující varování:


```
{
    "timestamp": "2018-04-19T19:55:28.856928+0200",
    "flow_id": 2181185586035533,
    "in_iface": "eth0",
    "event_type": "alert",
    "src_ip": "46.13.19.197",
    "src_port": 5060,
    "dest_ip": "185.88.72.17",
    "dest_port": 5060,
    "proto": "UDP",
    "alert":
    {
        "action": "allowed",
        "gid": 1,
        "signature_id": 2011716,
        "rev": 3,
        "signature": "ET SCAN Sipvicious User-Agent Detected
         (friendly-scanner)",
        "category": "Attempted Information Leak",
```

```
        "severity": 2
    },
    "app_proto": "failed",
    "flow":
    {
        "pkts_toserver": 2,
        "pkts_toclient": 1,
        "bytes_toserver": 895,
        "bytes_toclient": 476,
        "start": "2018-04-19T19:55:23.424781+0200"
    }
}
```


Software Suricata neslouží ale pouze k nahlašování varování. Může také sloužit ke sledování zaměstnanců. Například pokud se přípojí na internetové stránky `www.seznam.cz`, tak v případě logování TLS spojení se objeví v event logu následující.


```
{
    "timestamp": "2018-04-29T14:45:17.697643+0200",
    "flow_id": 1685975426006744,
    "in_iface": "enp0s3",
    "event_type": "tls",
    "src_ip": "192.168.1.19",
    "src_port": 35118,
    "dest_ip": "77.75.77.53",
    "dest_port": 443,
    "proto": "TCP",
    "tls":
    {
      "subject":"unknown=CZ, O=Seznam.cz, a.s., C=CZ, ST=Hlavni
      mesto Praha, L=Praha 5, unknown=Private Organization,
      serialNumber=26168685, OU=System Support,
      CN=www.seznam.cz",
      "issuerdn":"C=US, O=thawte, Inc., CN=thawte EV SSL CA - G3",
      "serial":"5E:31:0A:5B:0B:72:84:C5:38:08:B1:79:DC:79:4D:68",
      "fingerprint":"3a:db:b1:5a:01:f7:ac:ba:70:1f:16:62:70:28:1a:
      4c:84:28:a4:02",
      "sni":"www.seznam.cz",
      "version":"TLS 1.2",
      "notbefore":"2017-10-27T00:00:00",
      "notafter":"2018-12-21T23:59:59"
    }
}
```

Vzhledem k tomu, že software Suricata dokáže vyprodukovat stovky MB dat za den, záleží na činnosti uživatele, tak se budou zapisovat pouze události typu alert, aby nedocházelo k zahlcení sítě, daty o činnosti uživatele. Některé záznamy ale obsahují více typů událostí najednou, například u události typu alert se může vypsat i datový tok v události flow.

Závěr

Cílem práce bylo provést analýzu potenciálních útoků na webové aplikace, naučit se pracovat se softwarem Suricata, porovnat jeho výhody a nevýhody oproti konkurenčnímu programu Snort, vytvořit další program a skripty, které usnadní práci s ním. Vytvořil jsem program, který dokáže periodicky posílat logy na určenou adresu, je vytvořen v jazyce C++ a má u sebe i konfigurační soubor, ve kterém se dá nastavit adresa, na kterou se budou logy posílat, interval mezi posíláním jednotlivých logů. Skript na inicializaci, instalaci a vytvoření nového uživatele Suricata, jsou vytvořeny v jazyce Bash. Pro správnou funkci všechny tři skriptů, je zapotřebí je spouštět s administrátorskými právy. Dalším cílem bylo zpracovat dokumentaci pro rychlou orientaci v konfiguračním souboru, pro orientaci v softwaru Suricata a ukázat možnosti ovládání přes program Suricatasc, který je napsán v Pythonu a je součástí softwaru Suricata. Poslední cíle, ve kterých sem popisoval, co to je Big Cloud, systémy IDS a IPS jsem v práci také zahrnul. Pro lepší představu, jak vypadá Big Cloud, jsou v práci i dva obrázky znázorňující systém. Popsal jsem také možnosti, jakými uživatel může software Suricata na svůj virtuální stroj na instalovat, rovnou při vytvoření nebo až po vytvoření. Obě možnosti jsou popsané a znázorněné na diagramu.

Z práce si odnáším hlavně znalost programu Suricata, ale i při vytváření softwaru pro periodické zasílání logů jsem se musel naučit jakým způsobem se dá v C++ komunikovat pomocí http nebo https, dále pak práci s MongoDB.

Jako práce do budoucna je možné vytvořit balíček a umístit ho na server Big Cloud, pomocí kterého uživatel nainstaluje software Suricata příkazem apt-get a nebude muset spouštět instalační skript.

Literatura

[1] Zřetězené zpracování packetů. [online], prosinec 2017, [cit. 2018-03-31]. Dostupné z: https://xbu.me/article/performance-characterization-of-suricata-thread-models/

[2] Gregor, P.: BigCloud - backend pro veřejný cloudový systém. [online], květen 2015, [cit. 2018-04-05]. Dostupné z: https://alfresco.fit.cvut.cz/share/proxy/alfresco/api/node/content/workspace/SpacesStore/d558175e-4886-4bff-8505-c0406e08c9ce

[3] Ferschmann, P.: Bezpečnost na webu – přehled útoků na webové aplikace. [online], listopad 2008, [cit. 2018-03-25]. Dostupné z: https://www.zdrojak.cz/clanky/prehled-utoku-na-webove-aplikace/

[4] Zahradnický, T.; Kokeš, J.: Databáze a bezpečnost. [online], březen 2018, [cit. 2018-03-25]. Dostupné z: https://edux.fit.cvut.cz/courses/BI-BEK/_media/lectures/bek-08.pdf

[5] Wichers, D.; Manico, J.; Seil, M.; aj.: SQL Injection Prevention Cheat Sheet. [online], únor 2018, [cit. 2018-03-25]. Dostupné z: https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

[6] What is the Remote File Inclusion vulnerability? [online], [cit. 2018-03-25]. Dostupné z: https://www.netsparker.com/blog/web-security/remote-file-inclusion-vulnerability

[7] Zahradnický, T.; Kokeš, J.: Bezpečnost webových aplikací. [online], březen 2018, [cit. 2018-03-25]. Dostupné z: https://edux.fit.cvut.cz/courses/BI-BEK/_media/lectures/bek-10.pdf

[8] Wichers, D.; Wang, X.; Jardine, J.; aj.: XML External Entity (XXE) Prevention Cheat Sheet. [online], březen 2018, [cit. 2018-03-26]. Dostupné z: https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Prevention_Cheat_Sheet

[9] Prakash, T.: Xml eXternal Entity (XXE) Attack. [online], leden 2018, [cit. 2018-03-26]. Dostupné z: https://www.secpod.com/blog/xxe-xml-external-entity-attack/

[10] Cross-site Scripting (XSS). [online], březen 2018, [cit. 2018-03-26]. Dostupné z: https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

[11] DOM Based XSS. [online], červen 2015, [cit. 2018-03-26]. Dostupné z: https://www.owasp.org/index.php/DOM_Based_XSS

[12] Cross-site Scripting (XSS). [online], březen 2018, [cit. 2018-03-26]. Dostupné z: https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

[13] Rozenblum, D.: Understanding Intrusion Detection Systems. [online], srpen 2001, [cit. 2018-03-28]. Dostupné z: https://www.sans.org/reading-room/whitepapers/detection/understanding-intruson-detection-systems-337

[14] Chapčák, D.: BEHAVIORÁLNÍ ANALÝZA SÍŤOVÉHO PROVOZUA DETEKCE ÚTOKŮ (D)DOS. [online], 2017, [cit. 2018-03-28]. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=147879

[15] Suricata. [online], [cit. 2018-03-31]. Dostupné z: https://suricata-ids.org/

[16] Piecek, A.: AKCELERACE DETEKCE BEZPEČNOSTNÍCH HROZEB V SÍTI. [online], 2017, [cit. 2018-03-31]. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=159252

[17] Day, D. J.; Burn, B. M.: A Performance Analysis of Snort and Suricata Network Intrusion Detection and Prevention Engines. [online], prosinec 2017, [cit. 2018-03-31]. Dostupné z: http://www.thinkmind.org/download.php?articleid=icds_2011_7_40_90007

[18] Co je cloud computing? [online], [cit. 2018-05-1]. Dostupné z: https://azure.microsoft.com/cs-cz/overview/what-is-cloud-computing/

[19] Co je Big Cloud? [online], [cit. 2018-04-13]. Dostupné z: https://www2.bigcloud.cz/#bigcloud

[20] redakce: Co je to virtualizace. [online], duben 1999, [cit. 2018-04-05]. Dostupné z: https://businessworld.cz/ostatni/co-je-to-virtualizace-7158

[21] What is Zabbix. [online], [cit. 2018-04-15]. Dostupné z: https://www.zabbix.com/documentation/3.4/manual/introduction/about

[22] Init Scripts. [online], [cit. 2018-05-03]. Dostupné z: http://suricata.readthedocs.io/en/suricata-4.0.4/initscripts.html

[23] OISF: Suricata Rules. [online], 2016, [cit. 2018-04-19]. Dostupné z: http://suricata.readthedocs.io/en/latest/rules/index.html

[24] OISF: Suricata.yaml. [online], 2016, [cit. 2018-04-19]. Dostupné z: http://suricata.readthedocs.io/en/latest/configuration/suricata-yaml.html#suricata-yaml-action-order

[25] Chandel, R.: How to Configure Suricata IDS in Ubuntu. [online], leden 2018, [cit. 2018-04-19]. Dostupné z: http://www.hackingarticles.in/configure-suricata-ids-ubuntu/

[26] Drobot, I.: How to Bruteforce SSH. [online], leden 2011, [cit. 2018-04-19]. Dostupné z: https://zeldor.biz/2011/01/how-to-bruteforce-ssh/

[27] What is SIP – Session Initiation Protocol? [online], [cit. 2018-04-19]. Dostupné z: https://www.3cx.com/pbx/sip/

[28] SIPVicious. [online], únor 2014, [cit. 2018-04-19]. Dostupné z: https://tools.kali.org/sniffingspoofing/sipvicious

Dodatek A

Seznam použitých zkratek

API Application Programing Interface

DTD Document type definition

GUI Graphical user interface

HTTP Hypertext Transfer Protocol

IDS Intrusion Detection System

IPS Intrusion Prevention System

LFI Local File Inclusion

MAC adresa Media Access Control

RFI Remote File Inclusion

SIP Session Initiation Protocol

SSH Secure Shell

XML Extensible markup language

XSS Cross-site scripting

XXE XML external Entity Injection

Dodatek B

Dokumentace programu odesílajícího logy

Program pro odesílání logů má konfigurační soubor s názvem `SuricataLogger.config`. Popíši jednotlivé parametry:

- Event Log Path – cesta k souborům s událostmi
- Common Prefix for Log Files – společný prefix všech souborů s událostmi
- Suricata Shell Script – cesta ke skriptu, který bude zjišťovat stav softwaru Suricata
- URL – adresa endpointu, kam se budou odesílat logy
- Secured Port – port pro zabezpečenou komunikaci HTTPS
- Unsecured Port – port pro nezabezpečenou komunikaci HTTP
- Log Sending Period (in seconds) – doba v sekundách mezi jednotlivými logy
- ApiKey – speciální identifikátor aplikace, přiřadí se při inicializaci

Sestavuje se příkazem `make` nebo je možno zkompilovat ručně pomocí

```
g++ -std=c++14 Config.cpp Main.cpp LogSender.cpp Timer.cpp Time.cpp
-lcurl -lstdc++fs.
```

Dodatek C

Struktura událostí

V této části přílohy vypíši atributy, které se vyskytují v souboru s událostmi. U každého atributu uvedu datový typ, pokud se nemusí objevit, tak tam bude napsáno optional a u některým popíši, co daný atribut znamená.

- http – u http záleží na typu logování, buď rozšířené nebo zkrácené, zkrácené má pouze 4 položky
  - hostname – string
  - url – string
  - http_user_agent – string – například Mozilla
  - http_content_type – string – typ vrácených dat, například application/json, application/x-gzip
  - http_refer – string – optional, jen pokud je rozšířené logování
  - http_method – string – optional, jen pokud je rozšíření logování – HTTP metoda, GET, POST, HEAD…
  - protocol – string – optional, jen pokud je rozšíření logování – protokol nebo verze HTTP
  - status – string – optional, jen pokud je rozšíření logování – HTTP statuscode
  - length – integer – optional, jen pokud je rozšíření logování – velikost obsahu těla HTTP
- DNS – u DNS záleží na typu, může být buď query, nebo answer
  - version – integer – optional, pouze u typu „answer“
  - type – string – může být buď „answer“, nebo „query“
  - id – integer
  - rcode – string – optional

  - rrname – string – optional – Resource Record Name
  - rrtype – string – optional – Resource Record Type
  - ttl – integer
  - rdata – string
  - flags – string – optional – DNS answer flag v hexadecimální soustavě
  - qr – boolean – optional – indikázor Query/Response flagy
  - aa – boolean – optional – indikátor Authoritative Answer flagy
  - rd – boolean – optional – indikátor Recursion Desired flagy
  - ra – boolean – optional – indikátor Recursion Available flagy
  - tc – boolean – optional – indikátor Truncation flagy
  - answers – endpoint
    * rrname – string
    * rrtype – string
    * ttl – integer – Time-To-Live
    * rdata – string – Recource Data

- TLS – dvojí způsob logování, kratší a rozšířený, kratší má 2 nebo jen 1 položku, 1 položku má v případě, že je hodnota session_resumed nastaveno na true, potom chybí subject a issuer
  - subject – string – optional
  - issuerdn – string – optional – položka issuer z TLS certifikátu
  - session_resumed – boolean – optional – nastavuje se na true/false, pokud TLS spojení bylo navázáno přes session id
  - serial – string – optional
  - fingerprint – string – optional – fingerprint TLS certifikátu
  - sni – string – optional – Server Name Indication
  - version – string – optional – verze použitého TLS
  - notbefore – string – optional – NotBefore položka v TLS certifikátu
  - notafter – string – optional – NotAfter položka v TLS certifikátu

- TFTP
  - packet – string – optional – může mít hodnoty read, write nebo error
  - file – string – optional – název posílaného souboru
  - mode – string – optional

- flow

  - pkts_toserver – integer
  - pkts_toclient – integer
  - bytes_toserver – integer
  - bytes_toclient – integer
  - start – string – čas začátku, například `2018-04-19T18:53:25.647290+0200`
  - end – string – optional
  - age – integer – optional
  - state – string – optional
  - reason – string – optional
  - alerted – boolean – optional
  - tcp – endpoint – optional – vyskytuje se pokud se jedná o TCP u UDP se neposílá
    * tcp_flags – string
    * tcp_flags_ts – string
    * tcp_flags_tc – string
    * syn – boolean
    * fin – boolean
    * psh – boolean
    * ack – boolean
    * state – string

- stats – další položkou je stats, frekvence logování se nastavuje v konfiguračním souboru, vysvětlím později jak
  - uptime – integer
  - capture – endpoint
    * kernel_packets – integer
    * kernel_drops – integer
  - decoder – endpoint
    * pkts – integer
    * bytes – interger
    * invalid – integer
    * ipv4 – integer
    * ipv6 – integer
    * ethernet – integer
    * raw – integer

        * null – integer
        * sll – integer
        * icp – integer
        * upd – integer
        * sctp – integer
        * icmpv4 – integer
        * icmpv6 – integer
        * ppp – integer
        * pppoe – integer
        * gre – integer
        * vlan – integer
        * vlan_qinq – integer
        * ieee8021ah – integer
        * teredo – integer
        * ipv4_in_ipv6 – integer
        * ipv6_in_ipv6 – integer
        * mpls – integer
        * avg_pkt_size – integer
        * max_pkt_size – integer
        * erspan – integer
        * ipraw – endpoint
            · invalid_ip_version – integer
        * ltnull – endpoint
            · pkt_too_small – integer
            · unsupported_type – integer
        * dce – endpoint
            · pkt_too_small – integer
    - flow – endpoint
        * memcap – integer
        * tcp – integer
        * upd – integer
        * icmpv4 – integer
        * icmpv6 – integer
        * spare – integer
        * emerg_mode_entered – integer
        * emerg_mode_over – integer
        * tcp_reuse – integer
        * memuse – integer

- defrag – endpoint
  - ipv4 – endpoint
    - fragments – integer
    - reassembled – integer
    - timeouts – integer
  - ipv6 – endpoint
    - fragments – integer
    - reassembled – integer
    - timeouts – integer
  - max_frag_hits – integer
- tcp – endpoint
  - sessions – integer
  - ssn_memcap_drop – integer
  - pseudo – integer
  - pseudo_failed – integer
  - invalid_checksum – integer
  - no_flow – integer
  - syn – integer
  - synack – integer
  - rst – integer
  - segment_memcap_drop – integer
  - stream_depth_reached – integer
  - reassembly_gap – integer
  - overlap – integer
  - overlap_diff_data – integer
  - insert_data_normal_fail – integer
  - insert_data_overlap_fail – integer
  - insert_list_fail – integer
  - memuse – integer
  - reassembly_memuse – integer
- detect – endpoint
  - alert – integer
- app_layer – endpoint
  - flow – endpoint
    - http – integer
    - ftp – integer
    - smtp – integer

            - tls – integer
            - ssh – integer
            - imap – integer
            - msn – integer
            - smb – integer
            - dcerpc_tcp – integer
            - dns_tcp – integer
            - failed_tcp – integer
            - dcerpc_udp – integer
            - dns_udp – integer
            - failed_udp – integer
        * tx – endpoint
            - http – integer
            - ftp – integer
            - smtp – integer
            - tls – integer
            - ssh – integer
            - imap – integer
            - msn – integer
            - smb – integer
            - dcerpc_tcp – integer
            - dns_tcp – integer
            - failed_tcp – integer
            - dcerpc_udp – integer
            - dns_udp – integer
            - failed_udp – integer
    - flow_mgr – endpoint
        * closed_pruned – integer
        * new_pruned – integer
        * est_pruned – integer
        * bypassed_pruned – integer
        * flows_checked – integer
        * flows_notimeout – integer
        * flows_timeout – integer
        * flows_timeout_inuse – integer
        * flows_removed – integer
        * rows_checked – integer
        * rows_skipped – integer

        * rows_empty – integer
        * rows_busy – integer
        * rows_maxlen – integer
    - file_store – endpoint
        * open_files – integer
    - dns – endpoint
        * memuse – integer
        * memcap_state – integer
        * memcap_global – integer
    - http – endpoint
        * memuse – integer
        * memcap – integer
- ssh
    - client – endpoint
        * proto_version – string
        * software_version – string – například PUTTY
    - server – endpoint
        * proto_version – string
        * software_version – string – například OpenSSH_6.7p1 Debian-5+deb8u3
- vars
    - flowbits – endpoint – nastavují se v pravidlech, slouží k tomu, pokud chceme porovnávat sekvenci paketů
        * ET.EVIL – boolean – optional
        * ET.DshieldIP – boolean – optional
        * ET.TorIP – boolean – optional

Dodatek D

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `SuricataLogger`
  - `Source` ..... zdrojový kód implementace
- `Scripts` ..... skripty pro práci se softwarem Suricata
- `benesp32_thesis.pdf` ..... text práce ve formátu PDF