Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Inženýrská informatika
Informační systémy

Bakalářská práce

Pokročilé zabezpečení heterogenní IT infrastruktury pomocí Log IDS

Autor : Martin Beránek Vedoucí práce : Pavel Král

Plzeň, 2014

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 26. června 2014

.....

Martin Beránek

Abstract

This work deals with the automatic log analysis using the Open Source Host-based Intrusion Detection System (OSSEC). This topic was chosen due to its multidisciplinary nature, which includes computer networks, computer security and work in various systems. The OSSEC system was chosen because of its an open source system. The main goal was to analyze and to test it in different environments used in praxis. Although there were some complications, the task was fulfilled. My conclusion is that the OSSEC system is able to handle a vast majority of logs generated in typical informational system as well as increasing its overall security.

Tato práce se zabývá automatickou analýzou logů za použití Open Source Host-based Intrusion Detection System (OSSEC). Toto téma bylo zvoleno pro svoji mezioborovou povahu, která zahrnuje počítačové sítě, počítačovou bezpečnost a práci v různých systémech. OSSEC byl zvolen, protože se jedná o open source systém. Hlavním cílem byla analýza a otestování tohoto systému v různých prostředích, která jsou používána v praxi. Přestože se vyskytly nějaké komplikace, úkol byl splněn. Mým závěrem je, že systém OSSEC je schopen zvládnout naprostou většinu logů, generovaných typickým informačním prostředím, stejně jako zvýšit jeho celkovou bezpečnost.

Poděkování

Děkuji Ing. Pavlu Královi, Ph.D. za možnost dělat právě tuto bakalářskou práci. Děkuji také za existenci předmětů KIV/BIT, KIV/UPS a KIV/ZOS, které mi daly určitý teoretický základ k jejímu vypracování.

Obsah

Kapitola 1

Úvod

Zpracování systémových a aplikačních logů může být u větších podnikových prostředí problémem, protože tyto logy často nejsou uzpůsobeny pro čtení člověkem nebo obsahují množství informací, které jsou pro správce systému irelevantní. Pro existují systémy pro správu a automatické reagování na konkrétní logy. Jedním takovým systémem je Open Source Security (dále jen OSSEC), kterým se zabývá tato práce.

OSSEC je také Host-Based Intrusion Detection System (HIDS). To znamená, že s pomocí systémových agentů monitoruje činnost sledovaných systémů a loguje vybrané změny a přístupy do systému. Zároveň nabízí OSSEC i částečnou funkcionalitu Intrusion Prevention System (IPS), protože umožňuje automaticky zařazovat do blacklistu IP, ze kterých zachytí například sken portů.

Čtenář by po přečtení této práce měl porozumět problematice zpracování systémových a aplikačních logů. Měl by být také schopen nainstalovat a spravovat systém OSSEC, včetně správy uživatelských (neproprietárních) logů a definice vlastních pravidel.

1.1 Struktura práce

V druhé kapitole svojí práce se věnuji obecně problematice síťové bezpečnosti, analýzy logů a systému pro její automatizaci OSSECu. Dále moje práce řeší instalaci samotného systému, jeho správu, připojování sledovaných zařízení a webové grafické rozhraní. Čtvrtá kapitola se podrobně zabývá funkcí dekodérů logů a pravidel k jejich vyhodnocení. Pátá kapitola je věnována dostupnosti podpory pro OSSEC. V kapitole šest popisuji popisuji testovací prostředí, jeho konfiguraci a prováděné testy. V příloze je pak obsah optického disku, výstupy z testování a vlastní kód, který bylo třeba dopsat pro požadovanou funkčnost.

Kapitola 2

Počítačová bezpečnost a OSSEC

2.1 Bezpečnostní hrozby

2.1.1 Sociální inženýrství

Sociální inženýrství je způsob, jak si získat důvěru zejména privilegovaného uživatele. Typickým způsobem je vykonstruování profilu atraktivní mladé ženy na sociální síti a následné získání přístupových údajů od důvěřivých uživatelů zabezpečeného systému. V takovém případě je útok, pokud při něm nedojde k poškození dat, prakticky nezjistitelný, ale zároveň dokáže způsobit rozsáhlý únik dat, která jsou nejčastějším cílem útoků. Obrana proti takovému útoku je složitá a zahrnuje zejména školení uživatelů, protože jiné zabezpečovací prvky zde nejsou příliš účinné.

2.1.2 Fyzické poškození

Fyzické poškození zahrnuje jakékoliv akce, které vedou ke zničení nebo znefunkčnění hardwaru či infrastruktury. Může se jednat o cokoliv od přestřižení kabelu až po bombardování datového skladu. Obrana proti takovému útoku musí být opět fyzická a příliš se netýká síťové bezpečnosti, která je předmětem této práce.

2.1.3 Útok zvnějšku

Útok z vnější sítě je nejtypičtějším druhem útoku, proti němuž se lze bránit s využitím systémů pro detekci vniknutí. Jedná se o akce směřující k získání informací o cílové síti, jejímu vyřazení z provozu, odříznutí od vnějšího světa, průniku dovnitř a odcizení nebo poškození dat. Takové útoky zanechávají typické otisky, obvykle je možné je zaznamenat a na jejich základě podniknout příslušné kroky k zastavení útoku a/nebo lokalizaci útočníka. Protože tyto útoky přicházejí zvenčí, tvoří hlavní ochranu před nimi firewall, umístěný mezi celou vnitřní sítí organizace a internetem. Právě na firewallu je tedy možné detekovat a zastavovat tento typ útoků.

2.1.4 Útok zevnitř

Útoky zevnitř sítě jsou namířeny proti jiným strojům, které jsou fyzicky propojeny se strojem útočníka. Obvykle je takový útok spojen se sociálním inženýrstvím, které slouží k získání spojence v osobě uvnitř cílové organizace nebo přístupu ke stroji, například pomocí trojského koně, z nějž je následný útok veden. Protože pokus o průnik nepochází z internetu, nehraje tu obvykle roli kvalita a konfigurace firewallu. Obranu proti těmto útokům je třeba řešit na úrovni routerů, switchů a operačních systémů koncových strojů.

2.2 Typy síťových útoků

Síťové útoky zahrnují vnější a vnitřní pokusy o omezení provozu, zcizení a poškození dat cílové organizace. Tyto akce obvykle zanechávají určitou stopu na zařízeních, přes která je útok veden a je tak možné je při vhodné konfiguraci síťových prvků zastavit již v přípravné fázi. Níže jsou stručně uvedeny základní typy útoků. Hlavním zdrojem informací je článek [10]. Podrobnější informace lze získat například na stránkách *www.computernetworkingnotes.com*.

Port Scan

Sken portů slouží útočníkovi ke zjištění, jaké typy služeb běží na cílovém stroji, respektive které porty jsou otevřené, aby mohl přesněji směrovat následující útoky. Lze mu zabránit včasnou detekcí příchozích dotazů na mnoho různých portů z jedné IP adresy.

Network Traffic Flood

Zahlcení síťového provozu je útok typu DoS (Denial of service), který zahltí lokální síť extrémním množstvím paketů směrovaných a širokou škálu různých portů. Před tímto typem útoku je typickou metodou obrany omezení frekvence, s jakou pakety proudí v síti. Jedná se o typ útoku, který značí, že útočník se již dostal přes firewall, protože je veden zevnitř sítě.

Malformed Network Packets

Deformované pakety využívají skutečnosti, že většina služeb má dlouhou čekací dobu na přijímané pakety, případně se snaží poškozené pakety rekonstruovat. Čekací doba umožňuje útoky typu DoS při úmyslném vysílání poškozených paketů, které způsobí nedostupnost dané služby či stroje. Rekonstrukce paketů je pak způsobem, jak dostat jinak snadno odhalitelný škodlivý kód přes firewall nebo jiný bezpečnostní prvek. Obranou je porovnávání síťového provozu s množinou otisků známých útoků na firewallu, které zajistí, že případný závadný paket nebude doručen cílové službě.

Fragmentation attacks

Fragmentační útoky spočívají v rozdělení paketů se škodlivým kódem na menší celky, které snáze projdou bezpečnostními prvky cíle a do původní podoby se spojí až na konci své cesty. Proti tomuto typu útoku chrání IDS nebo firewall defragmentací paketů před jejich přeposláním cílové stanici.

IP Spoofing

IP Spoofing využívá skutečnosti, že firewally obvykle mají odlišnou politiku pro pakety, které přicházejí z vnějšku sítě a pakety, které proudí ve vnitrosíťové komunikaci. Podvržením zdrojové IP adresy se paket přestrojí za interní paket, který tak snáze projde ochranou. Obranou je filtrování všech paketů z vnější sítě, které mají cílovou adresu uvnitř sítě.

MAC útoky

CAM tabulka, ve které se MAC adresy mapují na porty switche, může být zahlcena útokem např. pomocí nástroje dsniff, který toto dokáže pomocí sofistikované hashovací funkce, schopné během minuty kompletně zahltit cílový switch. Síťový provoz, který by jinak vyvolal další zápis do tabulky pak prosakuje do celé VLAN. Proti útoku tohoto typu se volí obrana založená na apriorním přiřazení MAC adres všem portům.

VLAN Hopping

Jednou z metod VLAN Hoppingu je tzv. double tagging, který využívá zdvojeného identifikátoru VLAN. První switch odebere svůj identifikátor a přesměruje paket na další switch, který již vidí falešný tag a přepošle paket cílovému počítači, jako kdyby se jednalo o paket zaslaný uvnitř sítě. Druhou metodou je switch spoofing, což je útok využívající porty, které standardně slouží pro komunikaci mezi switchi (trunk porty). Útočník se na takový port připojí a může pak odposlouchávat veškerou komunikaci mezi switchi a získat tak například uživatelská jmén a hesla. Double taggingu je možné zabránit prostým zákazem defaultní VLAN, tedy VLAN 1. Proti switch spoofingu existuje obrana v podobě zakázání automatického přidělování trunk portů.

ARP útoky

Jedná se zejména o ARP Spoofing, což je v zásadě útok typu man-in-the-middle. Dvěma komunikujícím strojům se podstrčí MAC adresa útočníka a veškerá komunikace je pak směrována přes něj. Alternativou je ARP Poisoning, při kterém je veškerá komunikace rovnou zahazována bez odposlechu. Obranou je použití sítí Private VLAN, které síť rozdělí do virtuálně oddělených bloků, mezi kterými není možné ARP Spoofing použít, protože nesdílejí ARP tabulku.

Spanning Tree útoky

Při těchto útocích se využívá způsobu, jakým Spanning Tree algorytmus směruje komunikaci. Útočník se při něm vydává za kořenový most a veškerá komunikace pak proudí přes něj. Toho se dá využít pro širokou škálu útoků, jako MITM, DoS a dalších. Tento útok vyžaduje propojení útočníka s minimálně dvěma switchi, jejichž komunikaci pak odposlouchává, případně modifikuje. Obrana spočívá v omezení Spanning Tree algorytmu, který pak nemůže komunikaci směrovat úplně volně.

DHCP Starvation

Tento útok spočívá v konstantním dotazování se na dostupné DHCP adresy, což přetíží DHCP server. V této fázi se jedná o DoS útok, protože další stroje se nemají přístup k síťové komunikaci. Druhým krokem je podstrčení druhého DHCP serveru, který bude přidělovat nové adresy. Protože DHCP server poskytuje i adresy DNS serverů a výchozích bran, otevírá se tak cesta k libovolnému přesměrování síťového provozu. Obranou je, podobně jako u CAM útoku, pevné namapování MAC adres na porty DHCP serveru.

CDP útoky

V tomto případě je zneužit proprietární protokol společnosti Cisco, který slouží k automatické konfiguraci propojených zařízení a není šifrován. Díky absenci šifrování a broadcastingu konfigurace připojených zařízení je možné jednoduše získat informace o architektuře celé sítě a ty pak využít pro nalezení bezpečnostních chyb. Obranou je deaktivace tohoto protokolu na všech síťových prvcích.

2.3 Systémy pro detekci vniknutí

Cílem systémů pro detekci vniknutí (IDS) je monitoring síťového provozu za účelem detekce nesprávného užití nebo anomálního chování. V odborné literatuře bylo postupně definováno několik typů IDS, které mohou být rozděleny do dvou hlavních kategorií : network based IDS (NIDS) a host based IDS (HIDS). NIDS se snaží detekovat jakýkoliv pokus o podvržení normálního chování systému analýzou síťového provozu, zatímco HIDS je míněn jako poslední linie obrany. HIDS se pokouší o detekci vniknutí na základě analýzy událostí v lokálním systému, kde IDS běží. HIDS se dálě dělí na dvě skupiny. Jde o detekci anomálií (anomaly detection) a detekci nesprávného užití (misuse detection). Detektor nesprávného užití se pokouší identifikovat vzorce chování, které jsou charakteristické pro pokusy o průnik do systému, což může být zkoplikováno, pokud útočník používá dosud nerozšířený postup. Detektor anomálií se naproti tomu snaží charakterizovat normální chování systému a odchylky od něj následně označí jako možné pokusy o průnik. Detekce anomálií předpokládá, že nesprávné užití a pokusy o průnik do systému jsou silně korelovány s abnormálním chováním, které projevuje buď uživatel nebo samotný systém. Přístup pomocí detekce anomálií musí nejprve určit, jaké chování uživatelů a monitorovaného systému je normální, aby mohl identifikovat odchylky coby základ detekce možných pokusů o průnik [12].

Všechny typy IDS se potýkají s problémem, jak odhalit co největší část pokusů o průnik a zároveň minimalizovat počet falešně pozitivních výsledků. Při tom je také třeba nepoužívat příliš složité algoritmy, které by příliš zatěžovaly výpočetní nebo datovou kapacitu monitorovaných sítí.

2.4 Analýza logů

Analýzou logů se rozumí umění či věda, mající za cíl dát smysl počítačově generovaným záznamům (také zvaným logy nebo audit trails). Proces tvorby takových záznamů se nazývá logování dat (data logging).

Typickými důvody k provádění analýzy logů jsou:

- Dodržení bezpečnostních norem
- Splnění vládních nařízení a auditových norem
- Řešení problémů
- Forenzní účely (přímo při vyšetřování nebo jako reakce na soudní příkaz)
- Reakce na bezpečnostní incidenty

Tyto záznamy ale často mají formu, která je pro lidské operátory nečitelná. Objem takových záznamů je navíc enormní. Právě proto je zapotřebí nástroj, který umožní automatické překládání a třídění takových záznamů do čitelné a srozumitelné podoby. OSSEC je právě takovým nástrojem. Původně nesrozumitelná a objemná data umí filtrovat a přetvořit na informace podle nastavitelných pravidel a tyto informace buď někde ukládat, někam odesílat nebo jiným způsobem na jejich základě reagovat [7] [6]. Bohužel i nástroje pro analýzu logů se mohou stát cíli útoku. Ty nejčastěji operují s nějakou formou zahlcení logovacího systému [9].

Při zpracování logů se používá několik postupů a technologií [11].

- **Rozpoznávání vzorů** (pattern recognition) je funkcí, která identifikuje došlé zprávy podle databáze vzorů za účelem filtrování nebo specifického zacházení
- **Normalizace** konvertuje části zprávy při zachování formátu (například převede časové značky do bežného vyjádření)
- **Klasifikace a značkování** (classification and tagging) je třídění zpráv do různých kategorií a značkování rozličnými klíčovými slovy pro pozdější využití (např. filtrování nebo zobrazení)
- **Korelační analýza** (correlation analysis) je technologie sběru zpráv z různých systémů, patřících jediné události (např. zprávy generované nepřátelskou aktivitou na různých platformách: síťových prvcích, firewallech, serverech, atd.). Obvykle je korelační analýza napojena na varovný systém.

- **Umělá ignorace** (artificial ignorance) je proces zahazování záznamů, o kterých se předem ví, že jsou nezajímavé. Uměla ignorace je metoda detekce anomálií v produkčním prostředí. V analýze logů jde o rozpoznání a ignorování běžných hlášení, která jsou výsledkem standardní činnosti systému, a proto nezajímavá. Nové zprávy, které se dříve v lozích neobjevily, mohou ale signalizovat nějaké důležité události, takže by měly být prozkoumány.

2.5 OSSEC

OSSEC začal vyvíjet Daniel B. Cid, který publikoval zdrojový kód v roce 2004. O čtyři roky později, v roce 2008, byl celý projekt se všemi právy získán společností Third Brigade, která se zavázala k pomoci přidružené open-source komunitě, rozšíření komerční technické podpory a tréninkových programů. V dalším roce byla společnost Third Brigade pohlcena velkou firmu ve světě IT bezpečnosti Trend Micro se stejným příslibem zachování podpory.

Steve Chang, předseda Trend Micro, vydal v roce 2011 prohlášení, že Android, coby operační systém, je díky svému otevřenému zdrojovému kódu méně bezpečný, než jeho uzavřený konkurent iOS. To vyvolalo obavy o budoucnost projektu OSSEC, ale vzhledem k tomu, že vývoj nadále probíhá, jednalo se z jeho strany nejspíše jen o pokrytectví nebo propagaci svého bezpečnostního produktu.

OSSEC nabízí mnoho možností analýzy různých typů logů: operační systémy, HW zařízení i systémové a uživatelské služby [5]. Navíc oplývá funkcionalitou podporující připojení libovolných pevně specifikovaných logů s pomocí regulárních výrazů.

2.5.1 Logy operačních systémů

OSSEC Agent podporuje následující operační systémy :

- GNU/Linux (všechny distribuce včetně RHEL, Ubuntu, Slackware, Debian, atd.)
- Windows XP, Windows 2000, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7
- VMWare ESX 3.0, 3.5 (včetně kontroly CIS)
- FreeBSD (všechny současné verze)
- OpenBSD (všechny současné verze)
- NetBSD (všechny současné verze)
- Solaris 2.7, 2.8, 2.9, 2.10
- AIX 5.2 a 5.3
- Mac OS X 10.x
- HP-UX 11

2.5.2 Zařízení podporovaná přes vzdálený syslog

Na následující zařízení nemůže být OSSEC přímo nainstalován ani ve formě agenta, takže je třeba jejich správu řešit odlišně. Zde přichází na řadu tzv. vzdálený syslog, který umožňuje agregovat zprávy z mnoha různých zařízení. Je jen zapotřebí jejich výstupy přesměrovat někam, kde na ně OSSEC "uvidí", tedy do síťově přístupného souboru.

Podporována jsou následující zařízení:

- Cisco PIX, ASA a FWSM (všechny verze)
- Cisco IOS routery (všechny verze)
- Juniper Netscreen (všechny verze)
- SonicWall firewall (všechny verze)
- Checkpoint firewall (všechny verze)
- Cisco IOS IDS/IPS modul (všechny verze)
- Sourcefire (Snort) IDS/IPS (všechny verze)
- Dragon NIDS (všechny verze)
- Checkpoint Smart Defense (všechny verze)
- McAfee VirusScan Enterprise (verze 8 a 8.5)
- Bluecoat proxy (všechny verze)
- Cisco VPN koncentrátory (všechny verze)
- VMWare ESXi 4.x

2.5.3 Podpora pro bezagentní monitoring

OSSEC také nabízí pro některá zařízení tzv. bezagentní monitoring. V tomto případě jsou všechny potřebné informace předávány prostřednictvím síťového rozhraní. Bezagentní monitoring nabízí kontrolu integrity na následujících platformách:

- Cisco PIX, ASA a FWSM (všechny verze)
- Cisco IOS routery (všechny verze)
- Juniper Netscreen (všechny verze)
- SonicWall firewall (všechny verze)
- Checkpoint firewall (všechny verze)
- Všechny operační systémy typu Unix nebo Linux

2.5.4 Další podobné nástroje

Na trhu je mnoho alternativ k OSSECu od jiných opensource nástrojů až po velice drahá profesionální řešení. Pro přehled jsem vybral několik programů, které funkcionalitu OSSECu částečně nebo plně nahrazují.

Snort

Snort je pravděpodobně nejpoužívanějším opensource IDS/IPS systémem a je založen na analýze síťového provozu. Jedná se o multiplatformní opensource nástroj, který nabízí to nejlepší, co lze v současnosti u takových systémů nalézt. V reálném čase provádí analýzu síťového protokolu a sledování obsahu komunikace. Dokáže detekovat sondy a útoky pomocí otisků známých vzorců vniknutí, přetečení bufferů i skryté skeny portů. Snort pracuje v módech sniffer, packet logger a intrusion detection. Sniffer mód čte síťové pakety a zobrazuje je do konzole. Packet logger mód zaznamenává pakety do logovacího souboru na disk a intrusion detection mód aplikuje na zachycený provoz sadu pravidel, která slouží k odhalení pokusů o průnik a na jejich základě může vykonávat různá protiopatření.

Snare

Snare je robustní nástroj, který slouží k analýze logů, provádění auditů systému a spouštění odpovídající reakcí. Multiplatformní agenty, které běží na sledovaných strojích, jsou opensource a zajišťují shromažďování všech potřebných dat. Výstup z těchto agentů je v reálném čase přeposílán serveru ke zpracování. Serverem může být takřka jakýkoliv nástroj, schopný analýzy logů, ale existuje i placený Snare Server, u kterého je zaručeno, že se všechna data zpracují přesně podle požadavků. Součástí licence je i profesionální podpora. Oficiální Snare server nabízí bohatou funkcionalitu, která zahrnuje správu uživatelských skupin, emailová upozornění,

kontrolu konfigurace strojů, na nichž agenty běží, sledování aktivních i pasivních síťových prvků a další. K uživatelům Snare patří velké společnosti, jako Allianz, BAE Systems, AXA nebo australská vláda.

EventLog Analyzer

EventLog Analyzer je komerční řešení pro administraci logů z heterogenních sítí. Umožňuje shromažďování logů operačních systémů, aplikací i síťových prvků a jejich centralizovanou správu. Na rozdíl od ostatních zde uvedených nástrojů se jedná o plně komerční řešení určené pro větší společnosti. Funkcionalita samozřejmě zahrnuje podporu bezagentního monitoringu, zpracování nepřeberného množství různých vstupních logů, obsáhlou sadu pravidel pro práci s logy, generování standardizovaných výpisů ve formátu syslog, emailovou notifikaci, automatizované reakce na bezpečnostní hrozby, upozornění na porušení interních pravidel, sledování trendů, sledování konzistence souborů a přehledné grafické rozhraní. EventLog Analyzer je používán těmi největšími společnostmi, jako jsou IBM, Ernst and Young, US Air Force, US Army nebo Kellogg.

Syslog-ng

Syslog-ng je linuxový opensource nástroj bez grafického rozhraní, který umožňuje velice bohatým způsobem třídit a filtrovat zprávy formátu syslog z libovolných zdrojů a odesílat výsledky do souboru nebo na cílový port. Velice užitečná vlastnost je použití proměnných při zadávání zdrojových souborů, takže různé logy mohou být přijímány jako jeden zdroj, pokud mají daný formát názvu.

Existuje i komerční varianta, která profesionální podporu a několik dalších utilit, jako například výstup do databáze, šifrování uložených logů nebo zaručení bezztrátového přenosu zpráv. Tento systém využívají například ve společnosti Airfrance.

LogExpert

LogExpert je opensource nástroj k práci s logy pro systémy MS Windows. Nabízí jednoduché grafické rozhraní, vyhledávání pomocí regulárních výrazů, bohatá nastavení filtrů, záložky, vyhledávání v reálném čase, zobrazení logů ve formě tabulek se sloupečky, podporu pro rozšíření a další funkce. Nenabízí ale žádnou funkcionalitu pro automatické reakce na zaznamenané události.

EventViewer

Windows Event Viewer neumí logy zpracovávat, pouze v nich dokáže vyhledávat a filtrovat. Uvádím ho tu jen pro úplnost, protože je standardní součástí všech instalací Windows řady NT a jejích následovníků. Pracuje pouze s nativními soubory typu .evt, které Windows generují. Nativní logy dokáže prohlížet podle uživatele, stroje, řetězce, typu události, času a dalších kritérií. Jako většina nástrojů od Microsoftu, je i Event Viewer closedsource projekt.

MS LogParser

MS LogParser je nástroj s uzavřeným zdrojovým kódem pro zpracování logů a jiných řetězců od firmy Microsoft. Standardně nemá grafické rozhraní a funguje jen v příkazové řádce. Umožňuje prohledávat libovolný typ dat, včetně textových souborů, logů, registrů a dalších, s kterými dokáže manipulovat pomocí syntaxe, podobné SQL. Výstup pak může mít podobu libovolného textového souboru, SQL databáze, tabulky, syslogu nebo výstupu na obrazovce. Nabízí i výstup do grafu, který je ale navázán na balík MS Office, bez nějž grafový výstup zahlásí chybu.

2.5.5 Shrnutí dalších nástrojů

Pro lepší přehlednost jsou zde uvedeny důležité vlastnosti jednotlivých zmíněných programů.

	OSSEC	**Snort**	**Snare**	**EventLog Analyzer**
Windows	jen agenty	✓	✓	✓
Linux	✓	✓	✓	✓
Filtrování logů	✓	ne	✓	✓
Poplachy	✓	✓	✓	✓
Reakce	✓	✓	✓	✓
Zdarma	✓	✓	✓	✓
Opensource	✓	✓	jen agenty	ne

Tabulka 2.1: Další nástroje 1

	Syslog-ng	**LogExpert**	**EventViewer**	**MS LogParser**
Windows	✓	✓	✓	✓
Linux	✓	ne	ne	ne
Filtrování logů	✓	✓	✓	✓
Poplachy	✓	ne	ne	ne
Reakce	ne	ne	ne	ne
Zdarma	✓	✓	✓	✓
Opensource	✓	✓	ne	ne

Tabulka 2.2: Další nástroje 2

Z tohoto srovnání vychází OSSEC jako nenákladný, modifikovatelný nástroj s bohatou funkcionalitou, která je jinak běžná spíše u komerčních řešení.

Kapitola 3

Nastavení OSSEC

Samotná instalace OSSEC není zvlášť komplikovaná. Nastavení notifikací, propojení se sledovanými stanicemi nebo instalace webového rozhraní jsou pak o něco složitější. Pro efektivní využívání OSSECu je pak třeba mít širokou škálu znalostí o něm samotném, o zpracování logů a práci v prostředí linuxového terminálu.

3.1 Instalace pod Linuxem

Protože v různých distribucích Linuxu se budou konkrétní příkazy lišit, uvedu zde ty, které se použijí při instalaci v distribuci CentOS[1], která se běžně používá na serverech menších a středních podniků.

3.1.1 Instalace v CentOS

CentOS neobsahuje mnoho jinak běžných knihoven, takže na rozdíl například od Ubuntu je potřeba připojit repozitáře pro vyhledání a instalaci závislostí. Nejprve je zapotřebí nainstalovat příkaz *wget*.

```
$ sudo yum install wget
```

Nyní potřebujeme připojit repozitáře *EPEL* a *Atomic*. Repozitář *EPEL* získáme následujícími příkazy:

Pro CentOS 5.x

```
$ wget http://dl.fedoraproject.org/pub/epel/5/x86_64/epel-release-5-4.noarch.rpm
$ wget http://rpms.famillecollet.com/enterprise/remi-release-5.rpm
$ sudo rpm -Uvh remi-release-5*.rpm epel-release-5*.rpm
```

Pro CentOS 6.x

```
$ wget http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
$ wget http://rpms.famillecollet.com/enterprise/remi-release-6.rpm
$ sudo rpm -Uvh remi-release-6*.rpm epel-release-6*.rpm
```

[1] www.centos.org

Zbývá už jen stáhnout a nainstalovat repozitář *Atomic*.

```
$ wget https://www.atomicorp.com/installers/atomic
&& sudo chmod +x atomic && sudo ./atomic
```

Nyní můžeme stáhnout OSSEC včetně všech závislostí příkazem

```
$ sudo yum install ossec-hids ossec-hids-server
```

a spustit ho

```
$ sudo service ossec-hids start
```

3.1.2 Ruční instalace

Ruční instalace může být zapotřebí, pokud chceme používat více, než 256 agentů. V takovém případě doporučuji nejprve provést instalaci s pomocí repozitářů kvůli závislostem. Poté je třeba stáhnout instalátor OSSEC a rozbalit ho.

```
$ wget http://www.ossec.net/files/ossec-hids-latest.tar.gz
$ tar -zxvf ossec-hids-*.tar.gz
```

OSSEC v základním sestavení umožňuje připojit maximálně 256 agentů k jednomu serveru. Toto omezení se dá obejít, pokud ještě před instalací přeložíme zdrojové soubory s dodatečným parametrem. Ve složce s rozbaleným OSSEC zadáme následující kód:

```
$ cd src
# make setmaxagents
```

Skript nás vyzve k zadání cílového počtu agentů. Pouze je třeba dát pozor na omezení maximálního počtu otevřených souborů na jednoho uživatele v rámci operačního systému. Standardní limit 256 agentů je zvolen právě tak, aby všechny podporované systémy takový počet otevřených souborů umožňovaly.

Po případných úpravách sestavení vstoupíme opět do složky s rozbaleným OSSEC a spustíme instalační skript.

```
# ./install.sh
```

Instalační skript se nejdříve zeptá na typ instalace. Na výběr je mezi serverovou[2] instalací, agentem[3] a lokální[4] instalací. Instalační adresář doporučuji zachovat beze změn kvůli návaznosti na zbytek mojí práce.

[2] Serverová instalace zpracovává logy z připojených zařízení

[3] Agentní instalace sama nic nevyhodnocuje, pouze odesílá data na server

[4] Lokální instalace slouží jen pro počítač, na kterém běží. Chová se tedy jako server i agent zároveň, pro účely nasazení v ČTK nás ale nezajímá.

Skript se dále ptá na aktivaci různých součástí OSSEC, jmenovitě jde o e-mailové notifikace, kontrolu integrity, detekci rootkitů, aktivní reakce a vzdálený syslog. Doporučuji vše přijmout, protože pak jsou vytvořeny dané sekce v konfiguračním souboru a nastavení je tím jednodušší.

3.2 Konfigurace serveru

OSSEC v základním nastavení (ihned po instalaci) zpracovává logy ze souborů */var/log/auth.log*, */var/log/syslog* a */var/log/dpkg.log*, přičemž je schopen sám najít i další logy ve složce */var/log* a správným způsobem je připojit. Jistější je ale udělat to ručně nebo přinejmenším zkontrolovat, jestli je všechno tak, jak má. Výstup probíhá jen logovacího souboru OSSEC, s umístěním */var/ossec/logs/ossec.log*. Všechna vygenerovaná upozornění se ukládají v souboru */var/ossec/logs/alerts/alerts.log* a denně jsou rotovány. V případě serverové instalace jsou ve složce */var/ossec/logs/alerts/* uloženy i logy připojených agentů.

OSSEC vyhodnocuje závažnost situace na základě hodnoty *level*, která určuje míru ohrožení a má 16 úrovní (0-15)[5]. Podle hodnoty *level* můžeme například odesílat e-maily různým administrátorům, spouštět kontroly konzistence nebo blokovat uživatele.

Jiné vstupní logy, notifikace nebo další funkcionalitu je třeba nastavit v souboru */var/ossec/etc/ossec.conf*, který je hlavním konfiguračním souborem celého OSSEC. Protože se jedná o .xml soubor, musí být všechna dodatečná nastavení vložena před koncovým tagem *</ossec_config>*.

3.2.1 Přidání sledovaných souborů

V OSSEC se na logovací soubory jednotlivých zařízení a služeb odkazujeme ve výše uvedeném konfiguračním souboru. Formát zápisu je následující:

```
<localfile>
  <log_format>apache</log_format>
  <location>/var/log/apache.log</location>
</localfile>
```

Tag **log_format** odkazuje na použitý typ dekodéru. K dispozici jsou kromě standardizovaného formátu *syslog* ještě následující volby pro stejnojmenné služby:
snort-full, *snort-fast*, *squid*, *iis*, *eventlog*, *mysql_log*, *postgresql_log*, *nmapg* a *apache*

V tagu **location** je pak cesta k souboru (případně souborům), do kterého se dané logy ukládají.

[5] http://www.ossec.net/doc/manual/rules-decoders/rule-levels.html

Základní dekodéry se nachází v souboru */var/ossec/etc/decoders.xml*. Jak již bylo zmíněno, OSSEC umožňuje definici vlastních pravidel pro zpracování logů ze zařízení či služeb, které nejsou obsaženy v základním sestavení (základní dekodéry viz přiložený optický disk). Tyto dekodéry je vhodné ukládat do separátního souboru */var/ossec/etc/local_decoder.xml*, protože pokud by se ukládaly do původního, hrozilo by jejich přepsání při upgradu OSSEC.

3.2.2 Nastavení e-mailové notifikace

Základní syntaxe nastavení e-mailů je následující:

```
<global>
    <email_notification>yes</email_notification>
    <email_to>admin@firma.cz</email_to>
    <smtp_server>smtp.firma.cz</smtp_server>
    <email_from>poplach@ossec.firma.cz</email_from>
</global>
```

email_notification zapíná nebo vypíná e-mailovou notifikaci. Jediné povolené hodnoty jsou *yes* a *no*.

email_to udává cílovou adresu, kam se budou zprávy zasílat.

smtp_server je server pro odesílanou poštu.

email_from umožňuje zadat, jaká adresa bude v e-mailech uvedena jako zdroj.

Zároveň je třeba, aby v konfiguračním souboru mezi tagy *alerts* byla nastavena úroveň závažnosti.

```
<alerts>
    <email_alert_level>8</email_alert_level>
</alerts>
```

Je možné dále specifikovat, komu a za jaké situace se má posílat další upozornění. V tagu *email_alerts* můžeme detailněji určit podmínky. Podmínek i jednotlivých poplachů může být více.

```
<email_alerts>
    <email_to>master_admin@firma.cz</email_to>
    <group>sshd</group>
    <level>9</level>
    <event_location>server_1|server_2</event_location>
</email_alerts>
```

group umožňuje určit skupinu pravidel, jejíž poplachy se budou zasílat.

level určuje stupeň závažnosti, při kterém se e-mail pošle.

event_location nám umožňuje spouštět poplach jen pro určité vybrané stroje. Je možné zadat název stroje nebo IP adresu.

3.2.3 Nastavení SMS notifikace

SMS notifikace se nastavuje obdobně jako e-mailová. Jen je třeba přidat tag *format*, který umožní změnu formátu zprávy na takový, s kterým si poradí SMS standard. Nejjednodušší způsob, jak notifikační SMS zasílat, je využít e-mailové brány operátora.

```
<email_alerts>
    <email_to>605232766@sms.o2.cz</email_to>
    <level>10</level>
    <format>sms</format>
</email_alerts>
```

3.3 Regulární výrazy

Tato sekce je překladem zdroje s pouze mírnými úpravami. Protože se ale tyto regulární výrazy používají napříč celým systémem OSSEC, považoval jsem za nutné je zde uvést.

OSSEC používá knihovny regulárních výrazů *OS_Regex* a *OS_Match*, které jsou napsány hlavně s ohledem na rychlost zpracování. Proto je syntaxe spíše jednodušší a neumožňuje konstrukci skutečně komplexních regulárních výrazů.

3.4 Použitelné výrazy

Jedná se o sadu zástupných znaků, které tvoří základ regulárních výrazů. Všechny tyto znaky se píšou včetně zpětného lomítka.

\w zastupuje velká a malá písmena anglické abecedy bez diakritiky a číslice

\d zastupuje jednotlivé číslice

\s zastupuje mezeru

\t zastupuje tabulátor

\p zastupuje interpunkční znaménka, přesněji ( ) * + , - . : ; <= >? [ ]

\W zastupuje cokoliv, co není velkým nebo malým písmenem anglické abecedy, ani číslicí

\D zastupuje cokoliv, co není číslicí

\S zastupuje cokoliv, co není mezera

\. zastupuje libovolný znak

Tzv. **modifikátory** upravují význam použitelných výrazů

+ říká, že znak je přítomen jednou nebo vícekrát

***** říká, že znak je přítomen nulakrát nebo vícekrát

3.4.1 Speciální znaky

Tyto tři symboly se dají používat v kombinaci se standardními regulárními výrazy a modifikátory. Lze je ale použít i tam, kde komplexní výrazy nejsou dostupné. V předchozím textu jsem se na ně odkazoval také jako na *zjednodušené regulární výrazy.*

^ znamená začátek textu

$ znamená konec textu

— znamená logické nebo (OR) pro použití více regulárních výrazů

3.4.2 Příklad regulárního výrazu

Dejme tomu, že chceme zkonstruovat regulární výraz, který bude rozpoznávat typický log HTTP serveru Apache. Příklad :

```
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700]

"GET /apache_pb.gif HTTP/1.0" 200 2326
```

Odpovídající regulární výraz může vypadat například takto :

```
^\d+.\d+.\d+.\d+ - \w+ [\d+/\w+/\d+:\d+:\d+:\d+ -\d+]

"\w+ \.+ \.+" \d+ \d+
```

Poznámka :
I OSSEC provádí escapování řídících znaků. Pokud chceme v regulárním výrazu použít znaky *$*, *(*, *)*, \ nebo —, je třeba je escapovat jako *\$*, *\(*, *\)*, \\ nebo \— [4]

3.4.3 Aktivní reakce

Aktivní reakce znamenají použití předdefinovaného skriptu za určité situace. Takovou situací může být spuštění určitého pravidla nebo spuštění pravidla dané úrovně. V základním sestavení jsou takto přítomny například skripty pro odepření síťového přístupu nebo restart systému OSSEC. Aktivní reakce podporují zasílání vnitřních proměnných OSSECu v argumentu.

Definování příkazu

Aktivní reakce se konfigurují na dvou místech souboru */var/ossec/etc/ossec.conf*. Nejprve je třeba nadefinovat příkaz, který bude spuštěn podle následující syntaxe, přičemž každý je třeba definovat zvlášť:

```
<command>
    <name>nazev_prikazu</name>
    <executable>nazev_skriptu.sh</executable>
    <timeout_allowed>yes</timeout_allowed>
    <expect>srcip, username</expect>
</command>
```

name je název příkazu, kterým se chceme na skript odkazovat z aktivních reakcí a pravidel.

executable je název spustitelného souboru, který musí být umístěn ve složce */var/ossec/active-response/bin/* a musí mít nastavené oprávnění *executable*.

timeout_allowed umožňuje použití odpočtu aktivním reakcím, které to vyžadují. Pokud aktivní reakce nevyžaduje odpočet, nastavuje se na hodnotu *no*.

expect pokud skript vyžaduje použití vstupních proměnných, zde se zadává, které dostane. Lze použít pouze *srcip* a *username*. Tag je povinný, ale nemusí obsahovat žádnou hodnotu.

Definování aktivní reakce

Dále je třeba nadefinovat aktivní reakci jako takovou, přičemž každá je definována zvlášť [8]:

```
<active-response>
<disabled>no</disabled>
    <command>nazev_prikazu</command>
    <location>windows_server_03, windows_server_08</location>
    <rules_id>2100, 2101, 2102</rules_id>
    <level>10</level>
    <timeout>3600</timeout>
    <repeated_offenders>3600, 7200, 86400</repeated_offenders>
</active-response>
```

disabled umožňuje deaktivovat danou aktivní reakci. Tato možnost se využije například během testování, kdy by nám reakce překážely.

command obsahuje název příkazu.

location může být *server* nebo *local* nebo *all* pro spuštění příkazu na všech připojených strojích. Je možné také zadat hodnotu *defined-agent*, která spustí skript jen na vyjmenovaných agentech. Ty je potřeba vyjmenovat v tagu *agent_id*.

agent_id obsahuje čísla agentů, kde se má skript spuštět. Je potřeba mít zároveň správně definován tag *location*.

rules_group obsahuje názvy skupin pravidel, pro která se skript bude spouštět.

rules_id určuje, pro jaká pravidla se má aktivní reakce spustit.

level určuje, pro jakou závažnost pravidel se má aktivní reakce spustit. Obvykle se používá buď tag *rules_id* nebo *level*. Jejich kombinace je zbytečná.

timeout určuje, za jak dlouho po spuštění skriptu se aplikuje reverzní skript, například znovupovolení přístupu uživateli.

repeated_offenders umožňuje timeout postupně zvyšovat v případě, že k aplikaci stejné aktivní reakce došlo pro stejnou *srcip* nebo stejné *username*. Tento tag je, na rozdíl od ostatních, zapotřebí nadefinovat i na straně klienta. Příklad:

```
<active-response>
    <repeated_offenders>3600, 7200, 86400</repeated_offenders>
</active-response>
```

3.4.4 Kontrola konzistence

Kontrola konzistence je činností při které se prověřuje, jestli různé důležité soubory (operační systém, data) zůstaly nezměněny. Při změně ve sledovaných souborech je administrátor upozorněn [14].

Nastavení kontroly konzistence se na rozdíl od předchozích nastavení děje v agentech, nikoliv na serveru. Jednotlivé kontroly jsou uvedeny tagem *syscheck*, kterých je opět možné mít libovolný počet.

```
<syscheck>
    <frequency>10800</frequency>
    <alert_new_files>yes</alert_new_files>
    <directories check_all="yes" report_changes="yes" realtime="yes">/etc</directori
    <ignore>/etc/ossec</ignore>
    <directories check_perm="yes">/bin, /sbin</directories>
</syscheck>
```

Ve Windows jsou k dispozici dva tagy navíc pro práci s registry.

```
<syscheck>
    <windows_registry>HKEY_LOCAL_MACHINE</windows_registry>
    <ignore>HKEY_LOCAL_MACHINE\Software</windows_registry>
</syscheck>
```

frequency udává periodu kontrol v sekundách.

alert_new_files nastavuje upozornění i na nově vzniklé soubory, což se bez tohoto tagu neděje.

directories obsahuje složky, které chceme kontrolovat oddělené čárkou. Je možné dodatečně specifikovat parametrem *check_*, jaký typ kontroly chceme provádět. Za parametrem bude vždy hodnota *yes*.

- **check_sum** kontroluje MD5 kontrolní součty souborů
- **check_size** kontroluje změny velikosti souborů
- **check_owner** kontroluje změny vlastnictví
- **check_group** kontroluje změny skupinového vlastnictví
- **check_perm** kontroluje změny oprávnění
- **check_all** provede všechny dostupné kontroly

Dalším dodatečným parametrem je *report_changes*, který umožňuje nejen nahlásit změny v souborech, ale i oznámit, co přesně se změnilo.

Posledním dodatečným parametrem je *realtime*. OSSEC v základu hlásí změny v souborech pouze v definované frekvenci. Tento parametr způsobí, že změny jsou hlášeny v reálném čase.

ignore obsahuje složky, které jsou obsaženy v *directories*, ale chceme je z kontroly vyjmout. Umožňuje také použití *OS_Match* regulárních výrazů, pokud ho doplníme o parametr *type="sregex"*.

windows_registry slouží k zadání adresářů registru ke kontrole.

3.5 Konfigurace klientů

OSSEC nabízí pro správu klientů spustitelný soubor */var/ossec/bin/manage_agents*. Spuštění bez parametru nás přesune do jednoduchého menu, ve kterém můžeme přidávat agenty, vytvářet pro ně klíče, vypsat již přidané agenty, nebo je odstraňovat. Při spuštění s parametrem *-h* dostaneme informace o použití ostatních parametrů, které ale nepřinášejí novou funkcionalitu, jen umožňují provádět jednotlivé akce bez použití menu přímo z terminálu.

3.5.1 Bezagentní monitoring

Agentní monitoring umožňuje na cílové stanici provádět vše, co zvládne OSSEC lokálně, tedy zpracování logů, kontrolu integrity, atd.[13]

Bezagentní monitoring naproti tomu umožňuje pouze kontrolu integrity na základě kontrolních součtů. Komunikace probíhá výhradně protokolem SSH. Pro jeho použití je nutné ho na serveru explicitně povolit příkazem:

```
# /var/ossec/bin/ossec-control enable agentless
```

Stroje pro bezagentní monitorování se přidávají na dvou úrovních.
1) OSSEC si musí vygenerovat skripty pro SSH připojení ke vzdáleným terminálům.
2) Je třeba každý připojený stroj definovat v konfiguračním souboru.

Stroje se přidávají následujícím příkazem:

```
# /var/ossec/agentless/register_host.sh add user@host user_password
```

Pro síťová zařízení společnosti Cisco je třeba dodat další parametr *enablepass*, který umožní přihlašování s pomocí hesla:

```
# /var/ossec/agentless/register_host.sh add user@host user_password enablepass
```

Bezagentní monitoring se nastavuje v konfiguračním souboru */var/ossec/etc/ossec.conf*. Základní syntaxe je následující:

```
<agentless>
    <type>ssh_integrity_check_linux</type>
    <frequency>86400</frequency>
    <host>root@192.168.42.10</host>
    <state>periodic</state>
    <arguments>/bin /sbin /etc</arguments>
</agentless>
```

type v současnosti poskytuje čtyři možnosti nastavení.

- **ssh_integrity_check_bsd** je skript pro BSD systémy, který prohlíží všechny specifikované adresáře a podle kontrolních součtů hledá změny.
- **ssh_integrity_check_linux** je prakticky stejný skript, jen je určen pro linuxové distribuce.
- **ssh_generic_diff** slouží k zadání příkazu, jehož výstup se bude při každém opakování porovnávat a v případě změny dojde k upozornění.
- **ssh_pixconfig_diff** hlídá změny v nastavení síťových zařízení společnosti Cisco.

frequency je čas v sekundách, po němž se opakuje daná akce.

host je přihlašovací jméno uživatele a síťová adresa cílové stanice.

state se pro typy *ssh_integrity_check* nastavuje standardně na hodnotu *periodic* a pro zbylé dva typy na *periodic_diff*.

arguments obsahuje další argumenty skriptu. U *ssh_integrity_check* jsou to adresáře, které chceme sledovat, oddělené mezerou. U *ssh_generic_diff* je zde umístěn příkaz, u nějž chceme sledovat změnu výstupu a *ssh_pixconfig_diff* se používá bez argumentu.

3.5.2 Připojování agentů k serveru

Agenty je k serveru potřeba připojovat po jednom. Když máme agenty na všech strojích nainstalovány, je třeba je připojit k serveru a vygenerovat klíče pro bezpečnou komunikaci. Spustíme nástroj *manage_agents* (viz *Konfigurace klientů*). Zde postupně přidáme jednotlivé agenty (volba *(A)dd an agent*) a následně pro ně získáme klíče (volba *(E)xtract key for an agent*).

Pod Linuxem a Unixovými systémy spustíme v nástroj *manage_agents* v klientských aplikacích, zvolíme možnost *(I)mport key from the server* a klíč zadáme. Aby byly logy z nově připojeného agenta zpracovávány, je třeba serverovou instalaci OSSEC restartovat.

Agenty standardně používají port UDP 1514, který je zároveň jediným portem, jež OSSEC využívá pro své datové přenosy. Pokud je mezi sledovanými stroji a OSSEC serverem firewall, je třeba tento port otevřít.

3.5.3 Nastavení jednotlivých platforem

V OSSEC není mezi instalacemi na jednotlivých platformách zásadní rozdíl. Liší se jen posloupnost příkazů pro samotnou instalaci. Adresářová struktura i mechanismus ovládání je zachován na všech Linuxových a Unixových systémech.

Vyjímkou jsou pouze systémy z dílny společnosti Microsoft, kde nelze zprovoznit OSSEC server. Pro jejich monitorování je zapotřebí nainstalovat speciální klient, který lze nalézt na adrese *http://www.ossec.net/files/ossec-agent-win32-2.7.exe*. Ten je oproti ovládání klientské aplikace pod Unixem či Linuxem značně zjednodušen a je zapotřebí jen zadat IP adresu serveru a vygenerovaný klíč pro zabezpečenou komunikaci.

Obrázek 3.1: Vzhled agentní aplikace pro Windows

3.5.4 Konfigurační soubory specifické pro Windows

Konfigurační soubory pro audit pod OS Windows mají speciální syntaxi, která používá odlišné regulární výrazy. Tyto soubory se ve výchozím nastavení nacházejí ve složce *C:\Program Files\ossec-agent\shared*[6].

Soubory umožňují specifikovat, jestli se díváme po procesu, souboru nebo do registrů. Syntaxe záznamu je následující:

```
[Localhost found][any][]
f:%WINDIR%\System32\Drivers\etc\HOSTS
-> r:^127.0.0.1;
```

První řádek má tři atributy. První je název záznamu, druhý je *all* nebo *any* podle toho, jestli chceme u záznamu dostat hlášení pro jakýkoliv řádek nebo jen při spuštění ve všech, třetí pole slouží pro reference formou odkazů na web.

Následuje libovolný počet řádek, které jsou vždy uvedeny *f:*, *r:* nebo *p:* a končí středníkem.

- **f:** slouží k procházení složek nebo souborů
- **r:** slouží k procházení registrů
- **p:** slouží k procházení běžících procesů

Bez použití dalších složek výrazu monitoruje agent všechny změny v zadaném umístění.

Šipka -> se chová odlišně při použití u registrů a při použití u souborů/složek. U registrů její první použití ukazuje na konkrétní hledaný záznam, druhé použití ukazuje na konkrétní hledanou hodnotu. U souborů a složek šipka ukazuje na konkrétní hledanou hodnotu v nějakém ze souborů.

Hodnoty jsou bez použití dalších operátorů vyhledávány doslovně. Operátory jsou *=:* pro doslovné vyhledávání (defaultní operátor), *r:* pro použití regulárního výrazu, *<:* pro menší než a *>:* pro větší než. Více vzorů lze hledat s použitím výrazu *&&* a v takovém případě musí být splněny všechny, aby řádka byla označena za aktivovanou.

[6] Všechny uživatelské úpravy souborů ve složce *C:\Program Files\ossec-agent\shared* je třeba před updatem zálohovat, jinak budou smazány.

3.6 Webové uživatelské rozhraní

3.6.1 Instalace webového rozhraní

Nejprve je potřeba, aby na stroji, kde je umístěna serverová instalace OSSEC, byl nainstalovaný a funkční server apache. Poté je možno nainstalovat aktuální verzi samotného webového rozhraní. Pokud bude adresář pojmenován jinak, než *wui*, bude se tak jmenovat i webová stránka.

```
$ cd /var/www
# wget http://www.ossec.net/files/ossec-wui-0.8.tar.gz
# tar -zxvf ossec-wui-0.8.tar.gz
# mv ossec-wui-0.8 wui
$ cd wui
# ./setup.sh
```

Instalační skript se zeptá na přihlašovací jméno a heslo pro přístup do webového rozhraní a následně požádá o zadání jména uživatele, který spouští webový server (obvykle www nebo www-data). Následně je třeba přiřadit přístupová práva pro složku */tmp* uživateli webového serveru, umožnit v ní zápis a restartovat apache.

```
# chmod 770 tmp/
# chgrp username tmp/
# apache2ctl restart
```

Pokud vše proběhlo správně, lze nyní otevřít webové rozhraní na adrese *http://127.0.0.1/wui/*, případně na jiné, pokud je odlišný název příslušného adresáře.

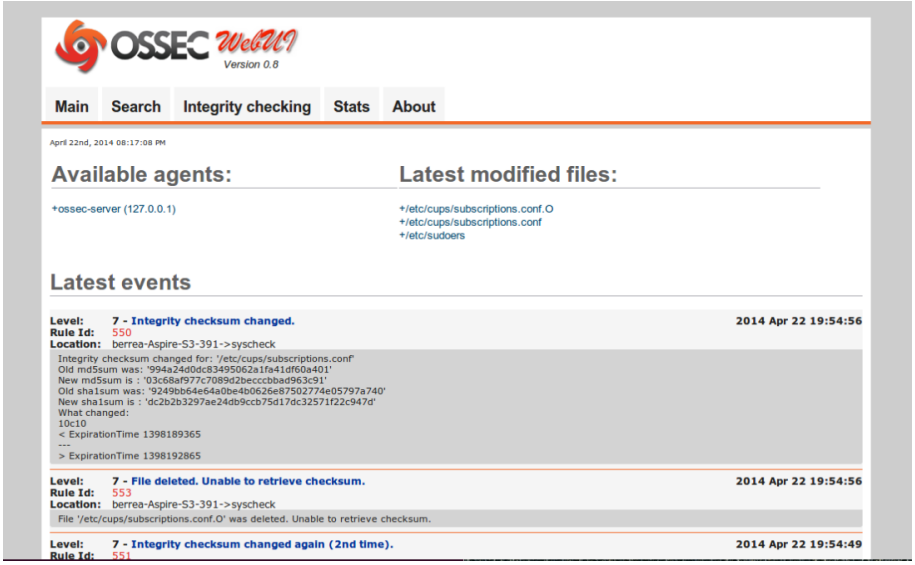
3.6.2 Možnosti webového rozhraní

Na domovské stránce WUI (viz obrázek 3.2) je k dispozici seznam připojených agentů, přičemž jednotlivé položky lze rozbalit pro detailnější informace o hostitelských strojích. Dále se zde nachází seznam posledních modifikovaných souborů, tedy výstup z kontroly integrity a nakonec seznam posledních událostí včetně závažnosti spuštěného pravidla, jeho názvu, čísla a agenta, na kterém k incidentu došlo.

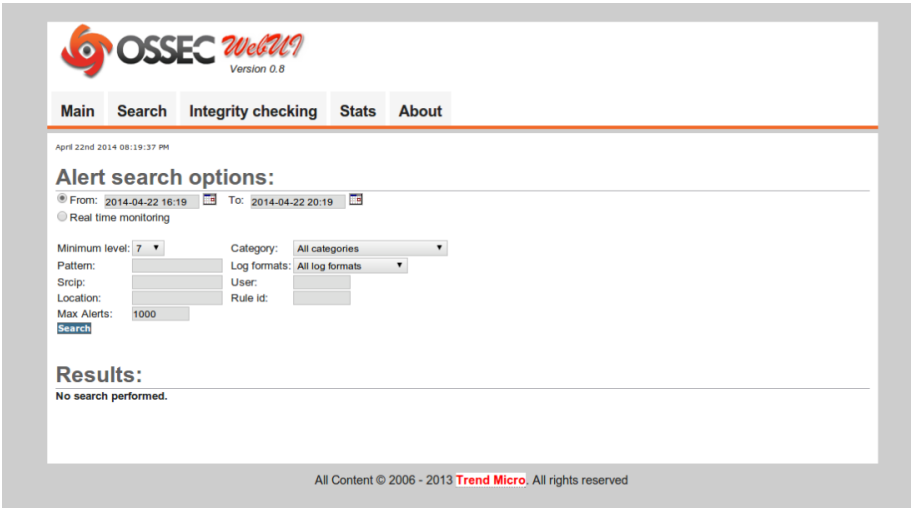
Záložka *Search* (viz obrázek 3.3) umožňuje komplexní vyhledávání v zaznamenaných incidentech. Vyhledávat lze podle data (od - do nebo aktuální), úrovně závažnosti, kategorie (typu incidentu), regulárního výrazu, formátu vstupního logu (podle použitého dekodéru), zdrojové IP, jména uživatele, názvu stroje a čísla pravidla. Je také možné specifikovat, kolik výsledků se má zobrazit.

Záložka *Integrity checking* (viz obrázek 3.4) nabízí informace o výstupech z kontroly integrity. Oproti hlavní stránce zde nejsou detailnější informace, ale je možné přepínat mezi výstupy z jednotlivých připojených zařízení a zobrazuje se celá historie.

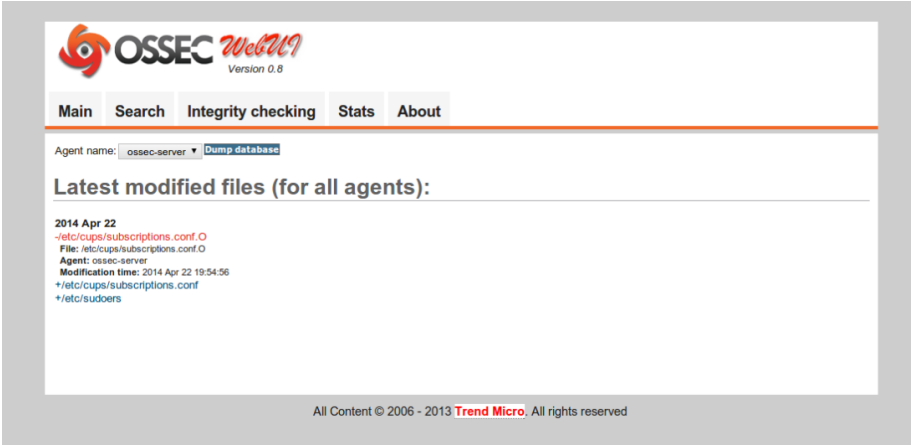
Záložka *Stats* (viz obrázek 3.5) nabízí souhrnný přehled o zaznamenaných incidentech. Je zde vidět, která pravidla se spouštěla nejčastěji, četnost jednotlivých úrovní závažnosti i celkový počet incidentů a jejich průměrná frekvence.



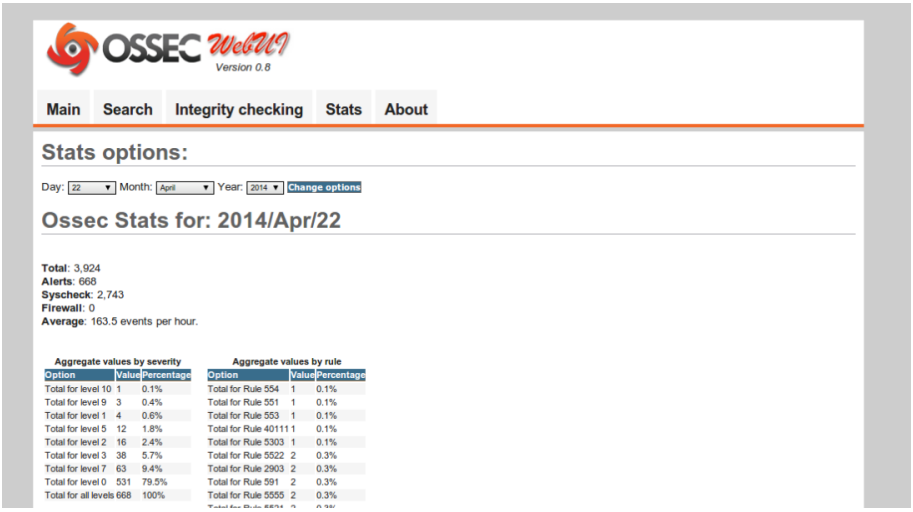
Obrázek 3.2: Domovská stránka WUI



Obrázek 3.3: Vyhledávání záznamů



Obrázek 3.4: Záznamy o kontrole integrity



Obrázek 3.5: Statistiky

Kapitola 4

Dekodéry a pravidla

OSSEC umožňuje tvorbu libovolně rozsáhlých stromů dekodérů a pravidel pro reakce na určité události, o jejichž existenci se dozví na základě záznamů v lozích. Uživatelské dekodéry se zapisují do souboru */var/ossec/etc/local_decoder.xml*, uživatelská pravidla se zapisují do souboru */var/ossec/rules/local_rules.xml*. Protože takové soubory existují i u klientských instalací, je třeba dbát na to, aby se dekodéry a pravidla přidávaly na stroji, na kterém je OSSEC server, protože právě tam se logy vyhodnocují.

4.1 Dekodéry

Dekodéry umožňují předzpracovat (parsovat) logy pro pravidla, která s nimi pak pracují. Obvykle je cílem extrahovat důležité informace jako čas, zdrojová/cílová IP adresa nebo vlastní obsah zprávy. Pro naprostou většinu běžně používaných aplikací a služeb jsou k dispozici předdefinované dekodéry. V případě, že chceme zpracovávat logy z aplikace či služby, která není podporována v základním sestavení OSSEC, můžeme si vytvořit vlastní dekodér.

Dekodéry pracují s omezenou množinou formátů logů. Těmi jsou *syslog*, *snort-full*, *snort-fast*, *squid*, *iis*, *eventlog*, *mysql_log*, *postgresql_log*, *nmapg*, *apache*, *djb-multilog* a *multi-line*. Pro jednořádkové uživatelské logy (nejběžnější) se doporučuje jednořádkový formát *syslog*. Pokud nutně potřebujeme zpracovávat víceřádkový log, použijeme formát *multi-line*.

Pozor je třeba dát na skutečnost, že před samotným dekódováním si OSSEC daný log upraví. Cílem je, aby do dekodéru vstupoval skutečně jen záznam z aplikace/služby, takže jsou odstraněna počáteční metadata. Protože pro uživatelské dekodéry používáme formát *syslog*, je každý záznam uveden datem a časem, jménem stroje, názvem procesu a PID. To všechno je zpracováno automaticky ještě před vstupem do dekodéru. Název procesu je uchován pro použití uvnitř rodičovského dekodéru. Více viz tag *program_name*.

Aplikace či služba obvykle generuje několik typů logů a struktura zápisu dekodérů toto větvení zohledňuje. Každý zdroj logů tak může mít libovolný počet potomků. Za příklad zde poslouží dekodér, kterým OSSEC zpracovává logy virtualizačního prostředí VMware[1]. To je vhodné i z toho důvodu, že v ČTK se toto prostředí používá. V příkladu je dekodér s názvem *vmware-syslog* rodičem, zatímco dekodéry *vmware-success* a *vmware-login* jsou potomky. Pro kompletní seznam a kód zabudovaných dekodérů viz přiložený optický disk.

```
<decoder name="vmware-syslog">
  <program_name>vmware</program_name>
</decoder>

<decoder name="vmware-success">
  <parent>vmware-syslog</parent>
  <prematch>^Accepted|^Rejected</prematch>
  <regex offset="after_prematch">^ \S+ for user (\S+) from (\S+)$</regex>
  <order>user, srcip</order>
</decoder>

<decoder name="vmware-login">
  <parent>vmware-syslog</parent>
  <prematch>^login from </prematch>
  <regex offset="after_prematch">^(\S+) as</regex>
  <order>srcip</order>
</decoder>
```

4.1.1 Parametry tagu decoder

- **name** je jediným povinným parametrem dekodéru. Ostatní parametry se běžně nevyužívají.
- **id** umožňuje použití číselného identifikátoru dekodéru.
- **type** definuje druh pravidla, standardně je ale tato definice umístěna v tagu, nikoliv v parametru. Protože se běžně nepoužívá, není ani v příkladu.

[1] VMware tvoří i logy v proprietárním formátu, které se zpracovávají odlišně, nás pro účely názorného vysvětlení nezajímají.

4.1.2 Tagy v definicích dekodérů

parent slouží ke spojení potomka s rodičem. Pokud je log zpracován dekodérem s daným jménem, vyzkouší se i všechny dekodéry, které mají tento dekodér jako rodiče. Pokud neprojde rodič, netestují se ani potomci. Příklad:

```
<parent>vmware-syslog</parent>
```

program_name porovnává název programu s názvem extrahovaným z logu ve formátu *syslog*. Umožňuje použití zjednodušených regulárních výrazů. Příklad:

```
<program_name>^vmware</program_name>
```

type slouží k detailnější organizaci dekodérů. Obvykle se nepoužívá.

```
<type>firewall</type>
```

prematch zjišťuje, jestli analyzovaný log odpovídá uvedenému regulárnímu výrazu. Je možné tento tag doplnit parametrem *offset*, který způsobí že k porovnávání nedochází v celém logu. Lze zadat hodnotu *after_parent*, která odsadí začátek logu o část, která se již prošla tagem *prematch* u rodičovského dekodéru. To je užitečné pro zkrácení reakční doby při zpracování logů. Příklad:

```
<prematch>^\w+ login from </prematch>
<prematch offset="after_parent">^login from </prematch>
```

regex je nejdůležitějším tagem dekodéru. Jedná se o vlastní regulární výraz, který nám umožňuje extrahovat různé proměnné. Stejně jako *prematch* umožňuje odsazení, je zde ale více možností. Těmi jsou:

- **after_parent** je vysvětlen v předchozím odstavci.
- **after_prematch** funguje obdobně jako *after_parent*, jen odsazení je posunuto tam, kde končí regulární výraz z *prematch* v dekodéru.
- **after_regex** se využívá, pokud konstruujeme subdekodéry, které mají stejný název jako hlavní dekodér. V takovém případě se log odsadí část, která již prošla jiným dekodérem. Tím se šetří výpočetní čas, protože není třeba porovnávat prakticky shodné regulární výrazy pro každý subdekodér.

V tagu *regex* také specifikujeme, které části logu chceme uchovat pro další zpracování. To děláme tak, že danou část regulárního výrazu umístíme do závorky. Příklad:

```
<regex>^Accepted (\S+)|^Rejected (\S+)|^ login from (\S+)</regex>
<regex offset="after_parent">^(\S+) as (\w+)</regex>
<regex offset="after_prematch">^(\S+) as (\w+)</regex>
<regex offset="after_regex">^(\w+)</regex>
```

U logů typu *multi-line* je syntaxe rozdílná pouze použitím více tagů *regex*, kdy každý zpracovává jednu řádku logu.

order souvisí přímo s tagem *regex*. Zde určíme, do jakých proměnných se uloží řetězce ze závorek. Jednotlivé proměnné oddělujeme čárkou a mezerou. OSSEC má omezenou množinu proměnných, které používá. K dispozici jsou tyto proměnné s přesně definovaným významem, který vyplývá z názvu: *user*, *srcuser*, *dstuser*, *srcip*, *dstip*, *srcport*, *dstport*, *port*, *status*, *protocol*, *url*, *id*, *action*. Navíc je tu proměnná *extra_data*, která slouží pro uložení dat, která nezapadají ani do jedné z uvedených kategorií. Příklad:

```
<order>user, srcip</order>
```

fts je zkratka pro "first time seen". Používá se, pokud chceme následně v pravidlech nastavit reakci jen při prvním výskytu uložené hodnoty[2] [2]. Příklad:

```
<fts>user, location, srcip</fts>
```

[2]více viz sekci 4.3.2 Tagy v definicích pravidel

4.2 Pravidla

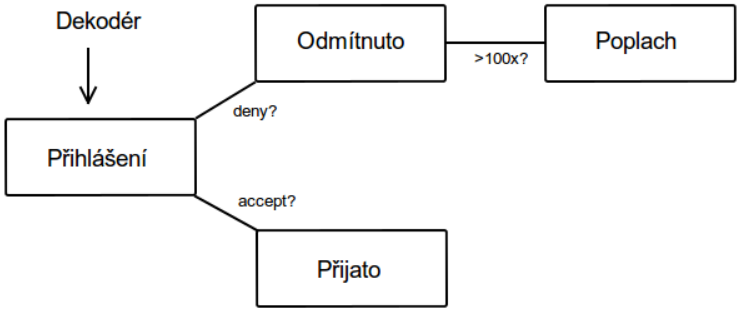
4.2.1 Pravidlo jako trigger

Každé jednotlivé pravidlo se chová jako trigger (spoušť), takže jakmile nějaká událost tuto spoušť aktivuje, spustí se kaskáda přidružených pravidel, která dále mohou, ale nemusí být také aktivována. Protože každé pravidlo má své jedinečné identifikační číslo, můžeme se na ně snadno odkázat v jiných pravidlech a na základě jejich spuštění/nespuštění se dále rozhodovat.

Každé pravidlo má také svoji úroveň závažnosti. Podle ní se rozhoduje jestli, případně jak bude OSSEC reagovat na jeho aktivaci. Jak jsem rozebíral v kapitole *Nastavení OSSEC*, lze potřebnou závažnost pro různé akce nastavit ve standardním konfiguračním souboru.

4.2.2 Stromy pravidel

Díky výše popsanému chování můžeme pravidla řetězit dle libosti na základě vzájemné provázanosti. Vytváříme tak stromy pravidel pro události z různých zdrojů podle toho, jak byly na počátku dekódovány vstupní logy. Pro vhodnou představu je lepší zde nepoužít reálný soubor pravidel.



Obrázek 4.1: Jak mohou být pravidla řezena ve stromu

Na obrázku je situace, kdy první pravidlo podle typu dekodéru pozná, že se jedná o přihlašování k nějaké službě. Podle výskytu slov *deny* nebo *accept* pozná, jestli bylo přihlášení přijato nebo odmítnuto a pokud se přihlášení nezdaří po více, než 100 pokusech, bude vygenerováno upozornění.

4.3 Definice vlastních pravidel

Protože může nastat situace, kdy nám základní množina pravidel nestačí, je třeba vědět, jak vytvořit vlastní. Standardní soubor s uživatelskými pravidly *local_rules.xml* je při inicializaci systému vždy zahrnut. Pokud si vytvoříme vlastní soubor, musíme ho přidat v */var/ossec/etc/ossec.conf*. To uděláme tak, že do tagu *rules* doplníme takto náš soubor:

```
<rules>
    <include>nase_pravidla.xml</include>
</rules>
```

Vlastní pravidla vždy patří do nějaké skupiny. Základní syntaxe pro uživatelskou skupinu pravidel vypadá následovně:

```
<group name="process_1">
    <rule id="10101" level="0">
        <decoded_as>process_1</decoded_as>
        <description>Process_1 group first rule</description>
    </rule>
</group>
```

Tag **group** s parametrem **name** je pouze organizační a slouží k pojmenování daného stromu pravidel. Jeho obsahem jsou všechna pravidla dané skupiny.

Následující tag **rule** slouží k uvedení konkrétního pravidla a nabízí dostatek parametrů k bližší specifikaci. Obsahuje pak konkrétní podmínky, za jakých se pravidlo spustí.

4.3.1 Parametry tagu rule

- **level** slouží ke specifikaci závažnosti pravidla. Tato hodnota se využívá při rozhodnutí, jak reagovat na jeho spuštění, tedy například zaslání výstražného e-mailu. Povolené hodnoty jsou 0 - 16.
- **id** je jednoznačné identifikační číslo pravidla. Je zapotřebí pro podmínkovou aplikaci jiných pravidel. Povolené hodnoty jsou 100 - 99999.
- **ignore** umožňuje zabránit zahlcení, pokud se nějaké pravidlo typicky spouští mnohokrát po sobě, ale nás zajímá jen jeho první výskyt. Hodnota je v sekundách a povolený rozsah je 1 - 9999.
- **overwrite** se využívá, pokud chceme nahradit nějaké pravidlo ze standardní výbavy vlastním. Jediná povolená hodnota je "yes".

Následující dva parametry se používají ve spojení se zvláštní kategorií tagů v definici pravidel (viz další subsekce):

- **frequency** udává, po kolika splněních podmínek v daném pravidle je toto pravidlo spuštěno. Používá se výhradně s parametrem *timeframe*. Hodnota frekvence *n* říká, že chceme být upozorněni, pokud se pravidlo spustilo *n+2*-krát. Pokud tedy napíšeme *frequency="3"*, znamená to, že podmínka musí být splněna 5 krát. Povolené hodnoty jsou 1 - 999.
- **timeframe** specifikuje, v jakém časovém intervalu musí k opakováním dojít, aby bylo pravidlo spuštěno. Používá se výhradně ve spojení s parametrem *frequency*. Hodnota je v sekundách a povolený rozsah je 1 - 9999.

4.3.2 Tagy v definicích pravidel

V pravidlech lze dále nastavit mnoho dalších funkcí s pomocí nepovinných tagů.

decoded_as nám umožňuje spustit pravidlo pouze, pokud má určený původ. Obsahem je název dekodéru.

description je druhým tagem z příkladu a slouží ke stručnému popisu pravidla. Obsahem může být libovolný řetězec znaků.

Ostatní tagy lze rozdělit do několika hlavních skupin podle toho, s jakými informacemi pracují. Těmi jsou:

- Tagy pro práci se vstupním logem
- Tagy pro práci s proměnnými z dekodéru
- Tagy s doplňující podmínkou spuštění
- Tagy typu if
- Tagy typu if_matched a same
- Ostatní tagy

Tagy pro práci se vstupním logem

match umožňuje porovnávat zadaný text s textem zkoumaného logu. Umožňuje použití zjednodušených regulárních výrazů. Konkrétní syntaxe regulárních výrazů se nechází v příloze. Příklad:

```
<match>attack$</match>
```

regex je tag, který se chová podobně jako *match*, ale umožňuje konstrukci komplexnějších regulárních výrazů. Příklad:

```
<regex>\d+.\d+.\d+.\d+:\d+</regex>
```

Toto pravidlo se spustí, pokud v logu najde nějaká čísla ve formátu IP adresy s portem, tedy něco jako *192.168.0.2:3015*. Důvodem, proč jsou k dispozici tagy *match* i *regex* je ten, že *match* je výrazně rychlejší na zpracování, protože překládání regulárních výrazů je výpočetně náročnější. Více o regulárních výrazech v sekci *Regulární výrazy*.

category umožňuje spustit pravidlo pouze, pokud patří do jedné z předdefinovaných kategorií. Příklad:

```
<category>firewall</category>
```

Dalšími povolenými hodnotami jsou: *ids*, *syslog*, *web-log*, *squid* a *windows*.

Tagy pro práci s proměnnými z dekodéru

srcip porovnává zadanou hodnotu adresy IP s IP adresou zdroje *srcip*, získanou z dekodéru. Umožňuje použití vykřičníku pro negaci. Příklad:

```
<srcip>!192.168.0.2</srcip>
```

dstip porovnává zadanou hodnotu adresy IP s IP adresou cíle *dstip*, získanou z dekodéru. Umožňuje použití vykřičníku pro negaci. Příklad:

```
<destip>192.168.0.10</destip>
```

user porovnává zadaný výraz s hodnotou, která byla dekodérem rozpoznána jako uživatelské jméno. Lze zde použít zjednodušených regulárních výrazů, podobně jako u tagu *match*. Příklad:

```
<user>^A</user>
```

program_name je tag, který umožňuje porovnávat název aplikace, která vygenerovala zkoumaný log, pokud je tento ve standardizovaném formátu *syslog*. Opět lze použít zjednodušený regulární výraz. Příklad:

```
<program_name>gparted</program_name>
```

id slouží k porovnání s *id* z příslušného dekodéru. Je možno opět použít zjednodušené regulární výrazy. Příklad:

```
<id>100013</id>
```

url slouží k porovnání s *url* z příslušného dekodéru. Je možno opět použít zjednodušené regulární výrazy. Příklad:

```
<url>www.google.com | www.seznam.cz</url>
```

Tagy s doplňující podmínkou spuštění

time umožňuje porovnávat, kdy došlo k dané události. Povolený je jakýkoliv rozsah času ve formátu HH:MM - HH:MM. Příklad:

```
<time>12:40 - 16:40</time>
```

weekday má obdobnou funkcionalitu, jen odkazuje na dny v týdnu. Povolenými hodnotami jsou dny v týdnu, jejich rozpětí, víkend nebo pracovní týden. Příklady:

```
<weekday>monday</weekday>
<weekday>monday - thursday</weekday>
<weekday>weekend</weekday>
<weekday>weekday</weekday>
```

Tagy typu if

if_sid slouží k řetězení pravidel. Pokud bylo při analýze daného logu aktivováno určené pravidlo, může být aktivováno (při splnění ostatních podmínek) i to aktuální. Příklad:

```
<if_sid>10101</if_sid>
```

if_group funguje velice obdobně jako *if_sid*, ale vztahuje se k celé skupině pravidel. Pokud tedy bylo v dané skupině aktivováno alespoň jedno pravidlo, může být aktivováno i to aktuální. Příklad:

```
<if_group>firewall</if_group>
```

if_level opět funguje podobně jako předchozí dvě *if_* pravidla. Na rozdíl od nich zkoumá, jestli už bylo spuštěno nějaké pravidlo dané úrovně. Příklad:

```
<if_level>10</if_level>
```

if_fts se používá, pokud chceme spustit reakci jen při prvním spuštění pravidla pro určitého uživatele, určitou udrojovou ip a podobně. Jakákoliv proměnná, na jejímž základě chceme unikátnost posuzovat, musí být přítomna v *fts cache*[3]. Příklad:

```
<if_fts>srcip</if_fts>
```

[3]viz Sekce 4.1.2 Tagy v definicích dekodérů

Tagy typu if_matched a same

Tyto tagy se využívají ve spojení s *frequency* a *timeframe*, což jsou parametry nadřazeného tagu rule. Týkají se tedy jen vymezeného časového intervalu. Na tyto tagy se dá nahlížet jako na počítadla. Při každém kladném vyhodnocení pravidla se interní hodnota zvedne o 1. Pokud přesáhne v určeném čase počítadlo zadanou hodnotu, pak je teprv pro OSSEC označeno jako spuštěné. Konkrétní časový interval a potřebný počet kladných vyhodnocení se uvádějí v parametrech tagu *rule*.

Tagy typu *if_matched_* ignorují pravidla úrovně *level="0"*. Pokud mají být pravidla brána pro účely *if_matched_* tagů jako spuštěná, jejich úroveň musí být minimálně *level="1"*.

if_matched_sid se spustí, pokud ve vymezeném čase došlo k danému počtu spuštění zadaného pravidla. Příklad:

```
<if_matched_sid>10101</if_matched_sid>
```

if_matched_group funguje obdobně, ale týká se celé skupiny pravidel. Pokud se tedy v dané skupině spustí nějaké pravidlo, počítadlo se zvedne o 1. Příklad:

```
<if_matched_group>Process_1<if_matched_group/>
```

if_matched_level zvedá počítadlo, pokud bylo spuštěno nějaké pravidlo dané úrovně. Příklad:

```
<if_matched_level>10</if_matched_level>
```

same_source_ip specifikuje, že počítadlo běží pro každou zdrojovou IP zvlášť. Příklad:

```
<same_source_ip />
```

Úplně stejně fungují tagy **same_source_port** a **same_dst_port**, pouze rozdělují počítadla podle zdrojového, respektive cílového portu. Příklady:

```
<same_source_port />
<same_dst_port />
```

Ostatní tagy

info je tag s dodatečným atributem *type*, který slouží k výpisu dodatečných informací o pravidle. Hodnoty atributu mohou být *link*, *cve*[4], *osvdb*[5] nebo *text*. V případě nepoužití atributu *type* je automaticky přiřazena hodnota *text*. Příklady:

```
<info type="link">http://www.ossec.net</info>
<info type="cve">20013-1006</info>
<info type="osvdb">54067</info>
<info type="text">Jakýkoliv řetězec</info>
<info>Stejné jako při použití type="text"</info>
```

options se používá k dodatečné specifikaci reakce na spuštění pravidla. Umožňuje nastavit, že se má vždy odeslat upozorňující e-mail nezávisle na dalších okolnostech. Také umožňuje naopak odeslání e-mailu za jakékoliv situace zakázat, případně pravidlo úplně vynechat z logu OSSEC. Příklady:

```
<options>alert_by_email</options>
<options>no_email_alert</options>
<options>no_log</options>
```

check_diff umožňuje při opakovaném spouštění pravidla porovnávat výstup proti předchozímu. To se hodí například, když chceme pravidelně volat systémovou službu s informacemi o provozu na síti typu *netstat*. Příklad:

```
<check_diff />
```

group slouží pro zařazení pravidla do dalších skupin, například pro účely hromadného výpisu. Příklad:

```
<group>suspicious_activities</group>
```

list je tagem, který zde uvádím jen pro úplnost, protože jeho implementace není dosud plně dokončena, takže jeho užití bych zatím nedoporučoval. Slouží k procházení seznamů, například pro tvorbu obsáhlého blacklistu IP a podobně [3].

[4]Common Vulnerabilities and Exposures je databáze známých bezpečnostních problémů, takže teoreticky můžeme většině pravidel přiřadit nějaké číslo z jejich databáze. Viz cve.mitre.org

[5]OSVBD je Open source Vulnerability Database, která funguje na stejném principu jako CVE. Viz www.osvdb.org

Kapitola 5

Podpora OSSEC

V případě, že se někdo rozhodne nasadit OSSEC v produkčním prostředí, setká se s řadou nejasností a nedokonalými formulacemi, které občas vyvolávají zmatek. I z toho důvodu byly zřízeny mailing listy, které pomáhají novým i zkušenějším uživatelům orientovat se v tak organicky rostoucím nástroji, jakým je OSSEC.

5.1 Mailing listy

Prvním a pravděpodobně nejlepším způsobem, jak získat podporu k OSSEC, jsou mailing listy. Všechny fungují pod službou Google Groups[1] od firmy Google. K dispozici je hned několik typů těchto maling listů. Jde o uživatelský[2], vývojářský[3] a CVS (Concurrent Version System)[4] list. Nás bude nejvíce zajímat ten uživatelský.

5.1.1 Users mailing list

Uživatelské mailing listy slouží k dotazování se komunity kolem OSSEC na zabudovanou funkcionalitu. To zahrnuje například otázky na instalaci systému, jeho správu a nastavení, napojování na jiné služby nebo způsob psaní konkrétních pravidel.

Reakční doba je obvykle velmi nízká. Odpověď na typický dotaz se objeví ještě v den, kdy byl dotaz zadán. V některých případech může samozřejmě odpověď trvat déle, protože uživatelský mailing list spravují zejména dobrovolníci, ačkoliv poměrně často odpovídají přímo vývojáři OSSEC.

K odběru OSSEC user mailing listu je třeba zaslat e-mail na adresu *ossec-list+subscribe@googlegroups.com* s předmětem zprávy *Subscribe ossec-list*. Pro odhlášení

[1] www.groups.google.com
[2] www.groups.google.com/forum/# !forum/ossec-list
[3] www.groups.google.com/forum/# !forum/ossec-dev
[4] www.groups.google.com/forum/# !forum/ossec-cvs

odběru je třeba zaslat na adresu *ossec-list+unsubscribe@googlegroups.com* e-mail s předmětem *Unsubscribe ossec-list*.

5.1.2 Development mailing list

Vývojářský mailing list slouží především k práci se zdrojovým kódem OSSEC. Sem posílají dobrovolníci moduly s novou funkcionalitou, hovoří se zde o možnostech dalšího vývoje a kompatibilitě vyvíjených rozšíření se stávající funkcionalitou. Pro běžné uživatele tento list nemá význam, protože je velmi technický.

K odběru OSSEC development mailing listu je třeba zaslat e-mail na adresu *ossec-dev+subscribe@googlegroups.com* s předmětem zprávy *Subscribe ossec-dev*. Pro odhlášení odběru je třeba zaslat na adresu *ossec-dev+unsubscribe@googlegroups.com* e-mail s předmětem *Unsubscribe ossec-dev* [1].

5.2 Komunita kolem OSSEC

Ačkoliv hlavním komunikačním kanálem s OSSEC komunitou jsou mailing listy, je možné sehnat například informace o chování pod konkrétními systémy nebo o komunikaci s jinými službami na specializovaných fórech. OSSEC je poměrně rozšířen, takže pokud existuje nějaký známý problém, potřebné informace jsou často dostupné bez dalšího dotazování jen s použitím vyhledávání společnosti Google.

V případě, že problém, se kterým se právě potýkáme, nikdo před námi veřejně neřešil, je nejefektivnější zadat dotaz simultánně do mailing listu i na fórum nástroje nebo systému, s kterým je problém.

5.3 Současný a budoucí vývoj

Vývoj OSSEC stále probíhá a to na úrovni profesionálních vývojářů pod společností Trend Micro i díky dobrovolníkům, kteří si obvykle naprogramují nějakou specifickou funkcionalitu sami pro sebe, aby ji pak zveřejnili. Po otestování bývá tato dodatečná funkcionalita zařazena do další verze hlavní distribuce OSSEC a stane se tak dostupná všem uživatelům.

Hlavní vývojářský tým tvoří šest programátorů včetně původní tvůrce OSSEC, Daniela B. Cida, který je velmi aktivní i v mailing listech, včetně uživatelských. Dále na vývoji stabilně pracuje dalších devět programátorů, kteří nespadají přímo pod Trend Micro, a mezi kterými je i Cidova žena Liliane.

Velké verze systému vychází přibližně jednou za rok. Poslední velká verze (2.7) byla vydána 19. listopadu 2012, předchozí (2.6) je z července 2011. Je tak vidět, že vývoj stále probíhá. Každá tato velká verze přináší novou funkcionalitu, opravené chyby a vylepšenou podporu stávajících nebo nových systémů.

Update na novou verzi je velmi jednoduchý. Po stažení instalačního balíčku je tento sám schopen detekovat předchozí instalaci OSSEC a nabídnout update, přičemž na každou jednotlivou složku systému (jádro, pravidla, ...) se instalátor dotazuje zvlášť. Lokální pravidla a nastavení update nezmění, což je i důvodem, proč jsem doporučoval[5] psaní vlastních pravidel právě do těchto uživatelských souborů.

5.4 Placená podpora

K dispozici je i oficiální placená podpora od Trend Micro, ale její cena je mírně řečeno nadsazená. Trend Micro sice nabízí pasivní i aktivní podporu zavádění a správy systému, stejně jako školení uživatelů, ale ceny nejsou veřejně dostupné. Z dotazování nezávislých odborníků na počítačovou bezpečnost vyplynulo, že řádově se ceny pohybují na úrovni desítek tisíc dolarů za instalace pro středně velké podniky. Na druhou stranu lze, vzhledem k relativní rozšířenosti systému, sehnat i jednotlivce mimo Trend Micro, kteří jsou schopni udělat stejnou práci s menšími náklady.

Oficiální podporu lze zkontaktovat prostřednictvím e-mailu *ossec@trendmicro.com*.

[5] sekce 4.3 Definice vlastních pravidel

Kapitola 6

Testování

6.1 Testovací prostředí

Pro účelu testování bylo zvoleno řešení v podobě virtualizovaného prostředí nástroje GNS3 verze 0.8.2, které plně simuluje reálný síťový provoz a umožňuje použití řady aktivních i pasivních síťových prvků včetně připojení virtualizovaných systémů, vytvořených v Oracle VM VirtualBox verze 4.3.10.

Emulované systémy :

- Kali Linux 1.0.6
- CentOS 6.4
- Ubuntu 13.04
- Windows XP SP3
- Windows 7 SP1
- Windows Server 2008 R2 SP1

Na stanici s OS CentOS byl nainstalován OSSEC server, stanice s Kali Linuxem sloužila k provádění penetračních testů. Později byl CentOS naklonován a ve shodné instalaci použit jako sniffer se Snortem. Na ostatních stanicích byly nainstalovány příslušné OSSEC klienty. Pro testování byla použita základní sada pravidel spolu s dále uvedenými úpravami.

6.2 Konfigurace

Zařízení byla rozdělena do dvou sítí, mezi nimiž router fungoval jako brána. Všechny stanice kromě Kali Linuxu byly umístěny do sítě 10.0.0.0, Kali Linux se nacházel v síti 10.0.1.0. Na routeru bylo nastaveno logování ve formátu syslog na stanici s OSSEC serverem. Syslog byl zvolen pro svou jednoduchost a nejdůkladnější popis v dokumentaci. Druhý router funguje v síti pouze jako aktivní switch za použití přídavného modulu, který má za úkol zrcadlit všechen síťový provoz na stanici s CentOSem, kde je provoz analyzován nástrojem Snort, jenž zachytává možné útoky.

6.3 Provedené testy

Dalším úkolem této práce bylo na testovací síti provést vybrané penetrační testy a sledovat, jak na ně bude OSSEC reagovat. V příloze B jsou uvedeny e-mailové výstupy z jednotlivých testů. Zvoleno bylo sedm bezpečnostních incidentů z různých oblastí bezpečnosti, které OSSEC pokrývá.

- Připojení externího úložiště
- Lámání přihlašovacího hesla hrubou silou
- Změny systémových souborů
- Sken portů
- Útok typu Denial of Service
- ARP Spoofing
- Výpadek rozhraní síťového prvku

První tři bezpečnostní incidenty byly monitorovány přímo na stanicích agenty. Připojení externího úložiště bylo ve Windows sledováno na změnách v systémovém registru, na Linuxu objevením se příslušného souboru v mountovacím adresáři. Cílem přihlašování byly samotné systémy.

Další tři incidenty byly monitorovány nástrojem *snort*, běžícím na stanici s CentOSem. Sken portů byl proveden nástrojem *nmap* bez dodatečných parametrů, DoS útok byl realizován nástrojem *hping3* s parametry *–rand-source 10.0.0.1 -p 53 -S -L 0 –udp –flood*. Nakonec ARP Spoofing byl proveden nástrojem *arpspoof* z balíku *dsniff* s obvyklými parametry.

Poslední incident byl sledován přes zprávy, které aktivní síťový prvek generoval ve formátu syslog.

Níže uvedené tabulky ukazují, jak se podařilo získat hlášení o jednotlivých incidentech na různých platformách. Je třeba zmínit, že PortScan, DoS útok a ARPSpoofing zde nemají roli v rámci jednotlivých platforem, protože byly vyhodnocovány nástrojem *snort* přímo ze síťového provozu.

	CentOS	**Ubuntu**	**Win XP**	**Win 7**	**WinServer 08**
Externí úložiště	✓	✓	neúspěšné	problematické	problematické
Hrubá síla	✓	✓	✓	✓	✓
Změna souborů	✓	✓	✓	✓	✓

Tabulka 6.1: Incidenty sledované na stanicích

PortScan	✓
DoS	✓
ARPSpoof	✓
Výpadek rozhraní	✓

Tabulka 6.2: Incidenty sledované na serveru

Jak je z tabulky vidět, podařilo se pod Linuxem otestovat všechny vybrané hrozby. Pro detekci připojeného úložiště bylo třeba pod Linuxem vytvořit krátký skript v Pythonu a pod Windows upravit lokální sadu pravidel. Úprava lokální sady pravidel ve Windows byla mírně problematická, protože detekci úložiště bylo třeba řešit jako pravidelně se opakující skript v PowerShellu, což není optimální. Detekce výpadku rozhraní a DoS útoku vyžadovala doplnění pravidel na straně OSSEC serveru. K zachycení DoS útoku bylo potřeba dopsat pravidlo pro snort. Veškerá doplňková konfigurace se nachází v příloze C.

Kapitola 7

Závěr

Cílem mojí práce bylo otestovat systém OSSEC na běžně využívaných platformách a typické síťové konfiguraci a zjistit, jestli skutečně dokáže zasílat všechna potřebná upozornění.

OSSEC se ukázal jako velmi vhodný nástroj pro agregaci logů z heterogenního síťového prostředí, který si ale s sebou nese některé neduhy. Jedním z nich je chybějící sada pravidel pro nástroj *snort*. Události z něj tak byly dekódovány jako obecné hlášení z IDS bez možnosti detailnějšího rozlišení závažnosti hrozby. Toto by bylo možné napravit vytvořením vlastní sady pravidel pro zpracování těchto logů, ale v současnosti žádná taková neexistuje. Dalším neduhem je skutečnost, že nástroj *syscheck*, který OSSEC využívá pro kontrolu integrity, se chová mírně nedeterministicky. Ke kontrole integrity sice dojde, ale někdy je zpožděna i o celé minuty, a to i v případě, kdy je u sledovaného souboru nastavena kontrola v reálném čase. Ke stejnému jevu dochází i při odesílání poplašných e-mailů, které jsou často odeslány i s desetiminutovým zpožděním. Tento problém se mi nepodařilo nijak vyřešit a pravděpodobně by šlo o vážnou překážku nasazení OSSECu na úrovni středních a větších podniků.

OSSEC odpovídá svojí funkčností skutečnosti, že je zcela zdarma. Je třeba zdůraznit, že OSSEC není plnohodnotným IDS, ale jen agregátorem logů, který je schopen aktivních reakcí, ale pro rozpoznání většiny bezpečnostních incidentů potřebuje nástroje třetích stran. Pro menší podniky, kde nevadí několikaminutové zpoždění reakce, se jedná o výborný způsob, jak vzdáleně sledovat stav sítě a agregovat systémové zprávy na jednom místě. Díky bohatým možnostem konfigurace je možné nastavit upozornění prakticky na jakoukoliv událost, což ale vyžaduje velkou trpělivost, protože dokumentace OSSECu je místy děravá a potřebné informace je nutno zjišťovat jinde. Nasazení OSSECu bych tedy doporučil jen tam, kde jedná spíše o doplňkovou, než kritickou funkcionalitu.

Příloha A

Obsah přiloženého optického disku

Bakalářská práce se zdrojovými kódy LaTeXu.

`\doc\bp.pdf`

`\doc\tex\`

Obrazy použitých systémů CentOS a Ubuntu.

`\images\`

Topologie sítě a konfigurace routerů pro GNS3.

`\topology\topology.net`

`\topology\config\`

Soubor s tímto obsahem disku.

`\readme.pdf`

Příloha B

Výstupy z testování

B.1 Připojení externího úložiště

B.1.1 Windows

```
** Alert 1403727132.192992272: mail  - local,syslog,
2014 Jun 25 22:12:12 (windows7) 10.0.0.4->hkeyusbcheck
Rule: 503002 (level 7) -> 'Mounted Device change detected'
ossec: output: 'hkeyusbcheck':
Model                  : VBOX HARDDISK ATA Device
InterfaceType          : IDE
serialnumber           : 42566338373162303663372d6631623830342066
Size                   : 26839088640
MediaType              : Fixed hard disk media
CapabilityDescriptions : {Random Access, Supports Writing, SMART Notification}

Previous output:
ossec: output: 'hkeyusbcheck':
Model                  : VBOX HARDDISK ATA Device
InterfaceType          : IDE
serialnumber           : 42566338373162303663372d6631623830342066
Size                   : 26839088640
MediaType              : Fixed hard disk media
CapabilityDescriptions : {Random Access, Supports Writing, SMART Notification}
Model                  : ADATA USB Flash Drive USB Device
InterfaceType          : USB
serialnumber           : AA00000000000485
Size                   : 4005711360
MediaType              : Removable Media
CapabilityDescriptions : {Random Access, Supports Writing,
Supports Removable Media}
```

B.1.2 Linux

```
** Alert 1403728120.195245024: mail  - local,syslog,
2014 Jun 25 22:28:40 (ubuntu) 10.0.0.3->/var/log/syslog
Rule: 503003 (level 7) -> 'Mounted Device change detected'
Jun 25 16:28:41 ubuntu-VirtualBox usb_device_watcher[4866]:
ALERT - USB device connected
```

B.2 Lámání přihlašovacího hesla hrubou silou

B.2.1 Windows

```
** Alert 1403719114.101774: - windows,win_authentication_failed,
2014 Jun 25 19:58:34 (windows7) 10.0.0.4->WinEvtLog
Rule: 18106 (level 5) -> 'Windows Logon Failure.'
User: (no user)
2014 Jun 25 19:58:33 WinEvtLog: Security: AUDIT_FAILURE(4625):
Microsoft-Windows-Security-Auditing: (no user): no domain:
user-PC: An account failed to log on. Subject:  Security ID:
S-1-5-18  Account Name:  USER-PC$  Account Domain:  WORKGROUP
Logon ID:  0x3e7  Logon Type:   2  Account For Which Logon Failed:
Security ID:  S-1-0-0  Account Name:  user  Account Domain:  USER-PC
Failure Information:  Failure Reason:  %%2313  Status:   0xc000006d
Sub Status:  0xc000006a  Process Information:  Caller Process ID: 0x85c
Caller Process Name: C:\Windows\System32\winlogon.exe
Network Information:  Workstation Name: USER-PC
Source Network Address: 127.0.0.1  Source Port:  0
Detailed Authentication Information:  Logon Process:  User32
Authentication Package: Negotiate  Transited Services: -
Package Name (NTLM only): -  Key Length:  0
This event is generated when a logon request fails.
It is generated on the computer where access was attempted.
```

B.2.2 Linux

```
** Alert 1403718798.99250: mail  - syslog,access_control,authentication_failed,
2014 Jun 25 19:53:18 (ubuntu) 10.0.0.3->/var/log/auth.log
Rule: 2502 (level 10) -> 'User missed the password more than one time'
Jun 25 13:53:19 ubuntu-VirtualBox sshd[2451]:
PAM 2 more authentication failures;
logname= uid=0 euid=0 tty=ssh ruser= rhost=10.0.3.2  user=root
```

B.3 Změny systémových souborů

```
** Alert 1395311102.2865: mail  - ossec,syscheck,
2014 Mar 20 11:25:02 localhost->syscheck
Rule: 550 (level 7) -> 'Integrity checksum changed.'
Integrity checksum changed for: '/etc/ossec-init.conf'
Size changed from '87' to '89'
Old md5sum was: '6ebb2d0c487b9094e1d89a0750f6d6e3'
New md5sum is : '12c903211be94dedf826c407f71f716f'
Old sha1sum was: '4d951dec0d2cd251bdba481b1fd092bd3176e7ed'
New sha1sum is : '8cdaa6bf2554eb1718e7251d1e3d9c519ef844b3'
```

B.4 Sken portů

```
** Alert 1403729364.195255843: - ids,
2014 Jun 25 22:49:24 snort->/var/log/messages
Rule: 20101 (level 6) -> 'IDS event.'
Src IP: 10.0.1.2
Dst IP: 10.0.0.1
Jun 25 22:49:23 snort snort[2447]: [122:1:1] (portscan) TCP Portscan
[Classification: Attempted Information Leak] [Priority: 2]
{PROTO:255} 10.0.1.2 -> 10.0.0.1
```

B.5 Útok typu Denial of Service

```
** Alert 1403722861.184314147: mail  - local,syslog,fts,
2014 Jun 25 21:01:01 localhost->/var/log/messages
Rule: 90101 (level 8) -> 'Multiple DOS Events detected'
Src IP: 57.161.128.172
Dst IP: 10.0.0.1
Jun 25 21:00:59 localhost snort[1422]: [1:25101:1]
DOS flood denial of service attempt
[Classification: Attempted Denial of Service] [Priority: 2]
{UDP} 57.161.128.172:58337 -> 10.0.0.1:53
Jun 25 21:00:59 localhost snort[1422]: [1:25101:1]
DOS flood denial of service attempt
[Classification: Attempted Denial of Service] [Priority: 2]
{UDP} 78.154.108.205:57427 -> 10.0.0.1:53
Jun 25 21:00:59 localhost snort[1422]: [1:25101:1]
DOS flood denial of service attempt
[Classification: Attempted Denial of Service] [Priority: 2]
{UDP} 110.124.196.196:55904 -> 10.0.0.1:53
Jun 25 21:00:58 localhost snort[1422]: [1:25101:1]
DOS flood denial of service attempt
[Classification: Attempted Denial of Service] [Priority: 2]
{UDP} 173.159.181.24:52473 -> 10.0.0.1:53
Jun 25 21:00:58 localhost snort[1422]: [1:25101:1]
DOS flood denial of service attempt
[Classification: Attempted Denial of Service] [Priority: 2]
{UDP} 90.172.95.22:52214 -> 10.0.0.1:53
```

B.6 ARP Spoofing

```
** Alert 1403716481.31973: - ids,
2014 Jun 25 19:14:41 localhost->/var/log/snort/eth0/alert
Rule: 20101 (level 6) -> 'IDS event.'
06/25-19:14:40.922119  [**] [112:4:1] (spp_arpspoof)
Attempted ARP cache overwrite attack [**]
```

B.7 Výpadek rozhraní síťového prvku

```
** Alert 1403730473.195331070: mail  - local,syslog,
2014 Jun 25 23:07:53 10.0.0.1->/var/log/messages
Rule: 503004 (level 7) -> 'FastEthernet0/1 is up'
Jun 25 23:07:53 10.0.0.1 258: *Mar  1 03:55:42.483:
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/1,
changed state to up

** Alert 1403730485.195331360: mail  - local,syslog,
2014 Jun 25 23:08:05 10.0.0.1->/var/log/messages
Rule: 503005 (level 7) -> 'FastEthernet0/1 is down'
Jun 25 23:08:04 10.0.0.1 261: *Mar  1 03:55:53.651:
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/1,
changed state to down
```

Příloha C

Vlastní kód

C.1 Připojení externího úložiště

C.1.1 Windows

Lokální pravidlo

```
<localfile>
<log_format>full_command</log_format>
<command>powershell.exe -command "gwmi win32_diskdrive |
select Model, InterfaceType, serialnumber, Size, MediaType,
CapabilityDescriptions"</command>
<frequency>10</frequency>
<alias>hkeyusbcheck</alias>
</localfile>
```

C.1.2 Linux

Pravidlo pro udev

```
KERNEL=="sd*", ACTION=="add", SUBSYSTEMS=="usb",
RUN+="/usr/local/bin/device_add_alert.py"
```

Skript pro nahlášení úložiště

```
import logging
import logging.handlers
import os
my_logger = logging.getLogger('MyLogger')
my_logger.setLevel(logging.DEBUG)
handler = logging.handlers.SysLogHandler(address = '/dev/log')
my_logger.addHandler(handler)
my_logger.critical("usb_device_watcher[%s]:
ALERT - USB device connected" % (os.getpid()))
```

C.1.3 Pravidla na OSSEC serveru

```
<rule id="503002" level="7">
<if_sid>530</if_sid>
<match>ossec: output: 'hkeyusbcheck'</match>
<check_diff />
<description>Mounted Device change detected</description>
</rule>

<rule id="503003" level="7">
<match>^ALERT - USB device connected</match>
<description>Mounted Device change detected</description>
</rule>
```

C.2 Útok typu Denial of Service

C.2.1 Pravidlo pro Snort

```
alert udp any any -> $HOME_NET 53
(msg:"DOS flood denial of service attempt";flow:to_server;
detection_filter:track by_dst, count 50, seconds 1;
metadata:service syslog; classtype:attempted-dos; sid:25101; rev:1;)
```

C.2.2 Pravidla na OSSEC serveru

```
<rule id="90100" level="8">
<category>ids</category>
<if_fts></if_fts>
<match>DOS</match>
<group>fts,</group>
</rule>

<rule id="90101" level="8" frequency="3" timeframe="10">
<if_matched_sid>90100</if_matched_sid>
<description>Multiple DOS Events detected</description>
<group>fts,</group>
</rule>
```

C.3 Výpadek rozhraní síťového prvku

C.3.1 Pravidla na OSSEC serveru

```
<rule id="503004" level="7">
<match>Line protocol on Interface FastEthernet0/1, changed state to up</match>
<description>FastEthernet0/1 is up</description>
</rule>

<rule id="503005" level="7">
<match>Line protocol on Interface FastEthernet0/1, changed state to down</match>
<description>FastEthernet0/1 is down</description>
</rule>
```

Literatura

[1] Podpora.
http://www.ossec.net/?page_id=21, 2010.

[2] Syntaxe dekodérů.
http://www.ossec.net/doc/syntax/head_decoders.html,
2009.

[3] Syntaxe pravidel.
http://www.ossec.net/doc/syntax/head_rules.html,
2009.

[4] Syntaxe regulárních výrazů.
http://www.ossec.net/doc/syntax/regex.html,
2009.

[5] Podporované platformy.
http://www.ossec.net/doc/manual/supported-systems.html,
2010.

[6] T. S. Adams. What Is Log Analysis?
http://www.wisegeek.com/what-is-log-analysis.htm,
2013.

[7] Martin Brown. Log Analysis Basics.
http://www.serverwatch.com/tutorials/article.php/3366531/Log-Analysis-Basics.htm,
2004.

[8] Daniel B. Cid. Blocking repeated offenders with OSSEC.
http://www.dcid.me/texts/blocking-repeated-offenders-with-ossec.html?highlight=ossec,
2011.

[9] Daniel B. Cid. Attacking Log Analysis Tools.
http://www.dcid.me/texts/attacking-log-analysis-tools.html,
2013.

[10] Roger Grimes. External firewall attacks.
http://www.windowsitpro.com/networking/external-firewall-attacks,
2002.

[11] Anita Kesavan Neil Daswani, Christoph Kern. *Foundations of security*. Apress, 2002.

[12] Luigi V. Mancini Roberto Di Pietro. *Intrusion Detection Systems*. Springer Science+Business Media, 2008.

[13] Jeremy Rossi. OSSEC: Agentless to save the day.
http://www.praetorianprefect.com/archives/2009/11/ossec-agentless-to-save-the-day/,
2009.

[14] "Xavier". Improving File Integrity Monitoring with OSSEC.
http://www.blog.rootshell.be/2013/05/13/improving-file-integrity-monitoring-with-ossec/,
2013.