# *IDS Snort*

Semestrální projekt do předmětu Směrované a přepínané sítě

Radomír Orkáč 19.6.2006

Tento text slouží k obecnému seznámení čtenáře s pojmem systém detekce narušení (IDS - Intrusion Detection Systems) a podrobně se zabývá popisem programu Snort. Po přečtení by měl být čtenář schopen základní instalace, nastavení a užívání tohoto programu. V závěru článku se prakticky věnuji možnostem generování falešných poplachů a v dodatku uvádím popisné schéma uložení dat v databázi MySQL.

# IDS

## Co je systém detekce průniku (IDS)?

Intrusion Detection Systems (systémy detekce průniku) patří mezi bezpečnostní nástroje počítačových sítí a začaly se používat v druhé polovině devadesátých let minulého století. IDS může být definován jako soubor zdrojů, metod a nástrojů, které nám pomáhají identifikovat a hlásit neschválené či neautorizované aktivity. Podstata těchto nástrojů vychází z předpokladu, že činnost útočníka (narušitele) bude odlišitelná na základě identifikátorů od běžné činnosti uživatelů.

IDS se dá přirovnat k alarmovému systému v rodinném domě. Zabezpečení dveří a oken by zde zajišťoval firewall, nicméně sám o sobě by nijak nevaroval majitele domu o narušení. Alarmový systém nezabrání narušení, ale dovede varovat před potencionálním narušením.

## Kategorie IDS

Systémy detekce průniku řadíme do dvou kategorií a to do tzv. hostitelských (host-based intrusion-detection systems, HIDS) a síťových (network-based intrusion-detection systems, NIDS). Hostitelské IDS vyžadují určitý software, který je umístěn na tomto systému. Využívá se například logovacích souborů (systémových a událostních) a sleduje se v nich, zda nedochází k nějakým významným a podezřelým změnám. HIDS mají výhodu v tom, že se provádí pouze kontrola událostí na vlastním systému, a také, že umí detekovat útok vedený přes šifrovací kanál. Slabinou těchto systémů je možnost jejich napadení v rámci útoku na hostitelský systém a jejich možným vyřazení z činnosti při DoS útoku. Síťové IDS se nasazují do jednoho ze segmentů sítě a zde analyzují síťové pakety, z čehož se pak vyhodnocuje narušení. NIDS nijak do komunikace nezasahují a data nechávají procházet dále, dojde-li však k vyhodnocení napadení, vyhlásí IDS poplach. Hlavním problémem těchto IDS je objem analyzovaných dat a vysoká míra falešných poplachů.

# IDS Snort

Snort je program, patřící do kategorie síťových IDS, založených na pravidlech. Jak je z předchozích odstavců patrné, Snort plní funkci detekce útoků a odposlechu v síti. Hledá vzorky známých útoků a v případě nalezení je schopen provádět různé akce, probíhající provoz však nepřerušuje. Snort je vyvíjen tak, aby pracoval na velkém spektru operačních systémů. Neměl by být tedy problém jej nainstalovat na Linux, Windows NT/2000/XP, Unix (Solaris, *BSD, a jiných). Určitě potěšující skutečností je i fakt, že Snort je vyvíjen jako open source.

## Režimy Snortu

Snort může běžet ve třech různých módech: sniffer mode (režim slídiče), packet logger mode (režim záznamníku) a network intrustion detection system mode (režim detekce narušení).

- **Sniffer mode** – tento mód zachytává pakety (sniffuje) procházející sítí a zobrazuje je uživateli na obrazovku,
- **Packet logger mode** – logger je v podstatě rozšířený sniffer mode, rozdíl je v tom, že se paketová data nebo hlavičky zaznamenávají do log souborů na pevný disk,
- **Network intrustion detection system mode** - nejvýznamnější režim programu, Snort zde odchytává síťová data a analyzuje je v kontextu s uživatelem definovanými pravidly a provádí akce podle nálezu,
- **Inline** - Získává pakety z iptables. Na základě pravidel rozhoduje o zahození nebo povolení paketů. V tomto módě pracuje jako HIDS.

### *Komponenty Snortu*

Snort je logicky rozdělený do několika komponent. Tyto komponenty pracují společně, detekují tak jednotlivé útoky a generují výstup v požadovaném formátu. IDS Snort se skládá z následujících hlavních komponent:

- Jednotka paketového záchytu
- Zásuvné moduly preprocesoru
- Detekční jednotka
- Systém logování a výstrah
- Výstupní zásuvné moduly

Jednotka paketového záchytu:

Komponenta bere pakety z různých síťových rozhraní počítače a před zpracovává pakety pro zaslání detekční jednotce. Rozhraním může být Ethernet, SLIP, PPP a tak dále.

Zásuvné moduly preprocesoru:

Preprocesory jsou komponenty, které lze použít ve Snortu k uspořádání nebo úpravě datových paketů, předtím než detekční jednotka provede operace, aby zjistila, jestli je paket generován vetřelcem. Některé preprocesory také hledají neobvyklosti v hlavičce paketu. Preprocesory jsou velmi důležité pro NIDS k přípravě datových paketů pro analýzu v detekční jednotce. Hackeři užívají různou techniku a různé způsoby k oklamání IDS. Například, můžete vytvořit pravidlo k nalezení signatury „`scripts/iisadmin`“ v HTTP paketech. Jestliže se paket testuje na výskyt tohoto řetězce, může být IDS snadněji hackerem oklamán, ten totiž může udělat nepatrné úpravy tohoto řetězce, např.:

- „`scripts/./iisadmin`“
- „`scripts/examples/../iisadmin`
- „`scripts\iisadmin`“
- „`scripts/.\issadmin`“

Komplikovanější situací je, že hacker může také vkládat do URI hexadecimální nebo unicode znaky, které jsou zcela legální. Všimněte si, že webové servery rozumějí všem těmto řetězcům a jsou je schopny přijmout jako stejný požadavek „`scripts/iisadmin`“. Nicméně jestliže IDS nepozná takové změny řetězce, není pak schopno odhalit útok. Preprocesor může tedy uspořádat či seskupit řetězec tak, že bude pro IDS jednoznačně detekovatelný.

Preprocesory se také používají pro paketovou defragmentaci. Když se přenášejí velká data, jsou vetšinou pakety rozděleny. Např. implicitní velikost paketu v Ethernetové síti je obvykle 1500 bajtů. Tato hodnota je řízena MTU hodnotou. MTU (Maximum transfer unit) je hodnota udávající maximální velikost přenosové jednotky na úrovni vrstvy síťového rozhraní (nejnižší vrstvy) komunikačního modelu TCP/IP. Jestliže pošleme data, která jsou větší než 1500 bajtů, ty se rozčlení do mnohočetných datových paketů tak, aby délka byla menší nebo rovna 1500 bajtům. Přijímací systémy jsou schopné zkompletovat rozčleněné jednotky, a tak vytvořit originální datový paket. Předtím, než můžeme použít některé pravidlo nebo vyzkoušet najít signaturu, musí dojít ke kompletaci paketu, protože například polovina signatury může být přítomna v jedné části (fragmentu) a druhá polovina v další části. K správnému detekování signatury je tedy třeba spojení všech paketových fragmentů. Je tedy jasné, že právě hackeři používají fragmentaci k oklamání IDS.

Detekční jednotka:

Detekční jednotka je nejdůležitější část Snortu. Její úkolem je systematicky porovnávat data uvnitř každého paketu, zda obsahuje zvláštní řetězec nebo hodnotu sdruženou s nějakým pravidlem. Detekční jednotka pro tento účel využívá Snortovské pravidla. Pravidla jsou načítána do vnitřních datových struktur nebo řetězců a jsou porovnávána proti všem paketům. Jestliže paket odpovídá některému pravidlu, vyvolá se příslušná akce. Příslušnou akcí zde může být zápis do logu nebo vytvoření výstrahy.

Detekční jednotka je časově náročný modul Snortu. Hodně proto záleží na tom, jak je počítač výkonný a kolik pravidel má definovaných. Jestliže je zatížení Vaší sítě příliš vysoké a Snort pracuje v NIDS režimu, může docházet k zahazování některých paketů, což vede k nesprávné odezvě v reálném čase. Zatížení detekční jednotky je ovlivněno následujícími faktory:

- Počet pravidel,
- výkon počítače, na kterém Snort právě běží,
- rychlost vnitřní sběrnice, používané v počítači kde běží Snort,
- zatížení sítě.

Při navrhování Intrusion Detection Systém bychom měli tyto všechny faktory zohlednit. Detekční jednotka pracuje různými způsoby pro jednotlivé verze Snortu. V 1.x verzích Snortu, se zastaví další zpracovávání paketu, když vyhovuje některému z pravidel. To znamená, že jestliže paket odpovídá kriteriím definovaných ve více pravidlech, je aplikováno pouze první pravidlo. Dochází zde k jednomu problému. Uvědomme si, že nízká priorita pravidla implikuje nízkou prioritu výstrahy, a že vysoká priorita pravidla si zasluhuje vysokou prioritu výstrahy. Pokud se tedy zpracovávání zastaví díky pravidlu s nízkou prioritou, přicházíme o potenciální porovnávání s pravidlem, který má vyšší prioritu. Tento problém je napraven ve Snortu verzích 2.x, kde všechny pravidla jsou porovnávána proti paketu před tvořením výstrahy. Po porovnání všech pravidel, se vybere vyhovující pravidlo s nejvyšší prioritou a vytvoří se příslušná výstraha.

Detekční jednotka ve Snortu verze 2.0 byla kompletně přepsána, takže je o mnoho rychlejší v porovnávání s dřívějšími verzemi Snortu. Analýzy ukazují, že rychlost nových detekčních jednotek se může zvýšit až 18 krát oproti detekční jednotce používané ve Snortu verze 1.x .

Systém logování a výstrah

Tato část Snortu závisí na tom, co detekční jednotka najde podezřelého uvnitř paketu, což je následně použito k záznamu do logu nebo k vytvoření výstrahy. Logy mají strukturu jednoduchého textovém souboru v tcp-dump tvaru nebo v jiných formách. Všechny logovací soubory se implicitně ukládají do adresáře /var/log/snort.

Výstupní zásuvné moduly

Výstupní jednotka nebo zásuvný modul provádí operace podle toho, jak chceme mít uleženy výstupy vytvořené systémem logování a výstrah. S ohledem na nastavení, mohou výstupní moduly dělat následující věci:

- Zaznamenávat (pouze) do /var/log/snort/alert souboru (nebo nějakého jiného),
- zasílání SNMP trapů,
- zasílání zprávy do syslogu,
- zapisovat do databáze jako MySQL nebo Oracle,
- generovat XML výstup,
- modifikovat konfigurace routerů a firewallů.

Další nástroje mohou také zasílat výstrahy v jiných formátech jako je e-mail, nebo nám umožňují sledovat výstrahy prostřednictvím webu, atd.. .

### *Pravidla Snortu*

Jednou z nejlepších vlastností Snortu je možnost snadného vytváření a přidávání (aktualizace) pravidel. Jednotka pravidel programu Snort poskytuje rozšířený jazyk, který nám umožňuje napsat si vlastní pravidla. To vede k tomu, že si vše můžeme přizpůsobit potřebám naší sítě.

Každé pravidlo se skládá ze dvou části: **hlavičky** a **volby** (options). Hlavička pravidla obsahuje akci pro vykonání, protokol ke kterému se pravidlo vztahuje, zdrojové a cílové adresy s porty. Options pravidla nám dovolují vytvořit popisnou zprávu spojenou s pravidlem, a taky kontrolovat různé druhy dalších atributů paketu, které Snort používá v rozsáhlé knihovně zásuvných modulů.

Takto vypadá obecný tvar pravidla Snortu:

```
action protokol zdroj_ip zdroj_port směr cíl_ip cil_port (options)
```

Když přijde paket, porovná se jeho zdrojová i cílová IP adresa a porty s pravidly v množině pravidel. Jestliže některé pravidlo odpovídá paketu, pak se vyhodnotí options. Jestliže souhlasí všechna porovnání, vykoná se příslušná akce.

Snort poskytuje několik vestavěných akcí, které můžeme použít, když pravidla vytváříme:

1. **Pass** (předání) – ignoruje paket,
2. **log** (zaznamenání) – zaznamená paket,
3. **alert** (výstraha) – vygeneruje výstrahu a paket zaznamená,
4. **activate** (aktivace) – vygeneruje výstrahu a vyvolá k otestování další pravidlo,
5. **dynamic** (dynamika) – akce „dynamic“ jsou vyvolávány pouze dalšími pravidly užitím akce „activate“. Za normálních okolností nejsou používány na paket,
6. **drop** (zahození) – přidá do iptables pravidlo pro zahození paketu a paket zaznamená,
7. **reject** (odmítnutí)– přidá do iptables pravidlo pro zahození paketu, paket zaznamená a pošle TCP reset, jestliže je protokolem TCP nebo ICMP zprávu o nedostupnosti portu, jestliže je protokolem UDP,
8. **sdrop** - přidá do iptables pravidlo pro zahození paketu, ale paket nezaznamená.

Implicitně se jako první testují pravidla Aktivace, poté pravidla Dynamiky, následují pravidla Výstrahy, poté přijdou pravidla Předání a končí pravidly Zaznamenání. Nicméně se dá pořadí měnit.

Hlavička pravidla:

**Akce:**

Hlavička pravidla obsahuje informaci, která lze popsat slovy "kdo, kde a co". První položka v pravidle je akce. Vestavěné akce jsem popsal v předchozích dvou odstavcích. Základní tři akce jsou výstraha (alert), zaznamenání (log) a předání (pass).

**Protokoly:**

Další položka v pravidle je protokol. Snort je schopen analyzovat tři IP protokoly tcp, udp a icmp.

**IP adresy:**

Další částí z hlavičky pravidla jsou IP adresy a porty. Slovo „any“ definuje jakoukoliv adresu. Snort nedisponuje žádným mechanismem na zpracování jmenných adres pro práci s pravidly. Adresy jsou tvořené přímo číselnou IP adresou v CIDR notaci.

V následující ukázce je použita libovolná zdrojová IP adresa a za cílovou IP adresu je brána 81.31.46.0 třídy C.

```
alert tcp any 3389 -> 81.31.46.0/24 3389 (msg: "RDP spojeni";
content: "|03|"; offset: 0; depth: 1; content: "|D0|";
offset: 5; depth: 1; flags: A+;)
```

Je možno taky použít operátor „!“, který značí „všechno mimo této adresy“. Pokud tedy chceme například rozlišit vnitřní síť od vnější, použijeme operátor přibližně takto:

vnitřní: 10.154.210.0/24

vnější: !10.154.210.0/24

**Čísla portu:**

Porty můžeme definovat více způsoby, a to buď klasicky číselně, použitím neurčitého „any“, definicí statického portu, rozsahem a nebo negací. Statické porty jsou definované jedinečným číslem portu, jako například 23 pro telnet nebo 80 pro http. Rozsahy jsou zapsány operátorem „:“.

Záznam udp paketu, přicházejícího z neurčité adresy (sítě) a neurčitého portu na cílový port (v rozsahu 1 až 80) na adresu sítě 10.154.210.0/24 – si vyžádámě tímto pravidlem:

```
log udp any any -> 10.154.210.0/24 1:80
```

Záznam udp paketu, přicházejícího z neurčitého portu na cílový port menší nebo roven 10000:

```
log tcp any any -> 10.154.210.0/24 :10000
```

Záznam udp paketu, přicházejícího z portu menší nebo rovno 80, jdoucí na cílový port větší rovno 10000:

```
log tcp any :80 -> 192.168.1.0/24 10000:
```

**Operátor určující směr:**

Operátor určující směr provozu (značí se „->“) říká, že se pravidlo bude aplikovat pouze na pakety proudící daným směrem. V pravidlech také můžeme používat operátor, který se značí symbolem ostrých závorek „<>“. Tento operátor využijeme v situacích, když potřebujeme použít pravidlo v obou směrech proudících dat, jako například pro telnet či pop3.

log !10.154.210.0/24 any <> 10.154.210.0/24 23

Volby pravidla (options):

Volby pravidla tvoří srdce detekční jednotky Snortu. Všechny volby pravidla jsou od sebe odděleny středníkem „;“. Za jednotlivá klíčová slova následuje dvojtečka, která oddělí jméno od argumentu.

K dispozici jsou tyto volby pravidla:

- **msg: "*text zprávy*";**
  hlášení pro výstrahu a log paketu
- **reference: *id_systému,id*;**
  popisuje, kde lze nalézt informace o dané signatuře, viz reference.config
- **sid: *id_pravidla_snortu*;**
  jedinečná identifikace pravidla
- **rev: *číslo*;**
  číslo revize pravidla
- **classtype: *jméno*;**
  označuje zařazení (klasifikaci) události (viz. tabulka níže)
- **priority: *číslo*;**
  číslo definuje prioritu (vážnost) útoku (viz. tabulka níže)
- **logto: "*jméno souboru*";**
  zaznamená paket do uživatelem definovaného souboru
- **ttl: "*číslo*";**
  parametr životnosti paketu (time-to-live), testuje hodnotu pole ttl havičky IP
- **id: "*číslo"*;**
  testuje ID pole z hlavičky IP fragmentu, na specifickou hodnotu
- **dsize: [>|<] *číslo*;**
  velikost dat paketu
- **content: "*řetězec*";**
  pátrá po vzoru v paketu, binární data se uzavírají mezi znaky roury „|“
- **nocase;**
  nebude rozlišovat malá a velká písmena
- **offset: *číslo*;**
  posunutí počáteční pozice pro hledání vzoru
- **depth: *číslo*;**
  jak daleko se bude hledat datové části paketu
- **flags: *příznaky*;**
  položka flags v TCP hlavičce. Testuje se TCP flags na jisté hodnoty (F – FIN, S – SYN, R – RST, P – PSH, A – ACK, U – URG, 2 – rezervovaný bit 2, 1 – rezervovaný bit 1)
- **seq: *číslo*;**
  testuje TCP číselné sekvence na specifickou hodnotu
- **itype: *číslo*;**
  testuje ICMP typ pole na specifickou hodnotu

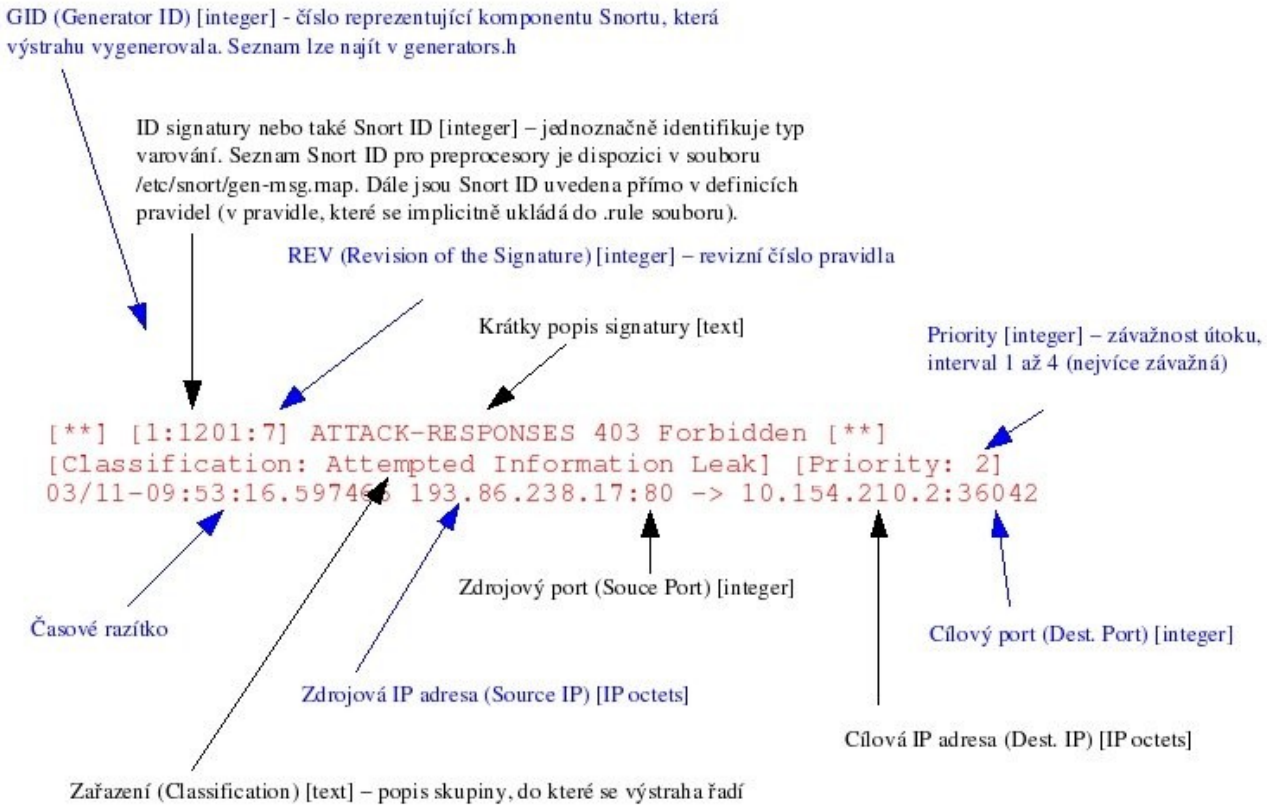
- **icode:** ***číslo*****;**
  testuje ICMP kódové pole na specifickou hodnotu
- **session:** ***[printable|all]*****;**
  výpis informací o sezení z aplikační vrstvy

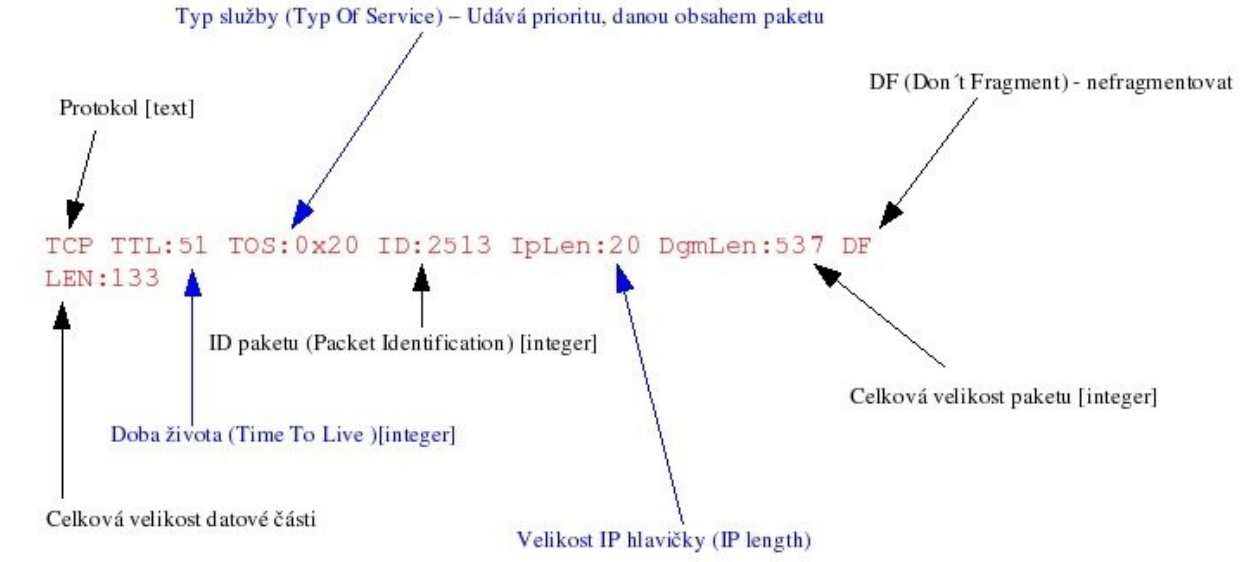
Ještě uvedu tabulku zařazení výstrah (využito pro class type a priority):

| Zařazení | Popis | Priorita |
|---|---|---|
| attempted-admin | pokus o získání administratorských práv | vysoká |
| attempted-user | pokus o získání uživatelských práv | vysoká |
| shellcode-detect | detekovaný spustitelný kód | vysoká |
| successful-admin | úspěšné získání administratorských práv | vysoká |
| successful-user | úspěšné získání uživatelských práv | vysoká |
| trojan-activity | detekovaný síťový trojský kůň | vysoká |
| unsuccessful-user | neúspěšné získání uživatelských práv | vysoká |
| web-application-attack | útok na webovou aplikaci | vysoká |
| attempted-dos | pokus o DOS | střední |
| attempted-recon | pokus získat informace o dírách | střední |
| bad-unknown | potenciálně špatný provoz na síti | střední |
| denial-of-service | odhalení DOS | střední |
| misc-attack | misc attack | střední |
| non-standard-protocol | detekce nestandardního protokolu nebo události | střední |
| rpc-portmap-decode | dekódovaný RPC dotaz | střední |
| successful-dos | DOS | střední |
| successful-recon-largescale | velká škála bezpečnostních děr | střední |
| successful-recon-limited | bezpečnostní díra | střední |
| suspicious-filename-detect | detekováno podezřelé jméno souboru | střední |
| suspicious-login | pokus o přihlášení s podezřelým jménem | střední |
| system-call-detect | detekováno systémové volání | střední |
| unusual-client-port-connection | klient užil neobvyklý port | střední |
| web-application-activity | přístup k potenciálně zranitelné webové aplikaci | střední |
| icmp-event | obecná ICMP událost | nízká |
| misc-activity | misc activity | nízká |
| network-scan | detekce síťového skenu | nízká |
| not-suspicious | nepodezřelý datový provoz | nízká |
| protocol-command-decode | dekódován běžný protokolový příkaz | nízká |
| string-detect | detekován podezřelý řetězec | nízká |
| unknown | neznámý datový provoz | nízká |

### *Formát výstrahy z logu*

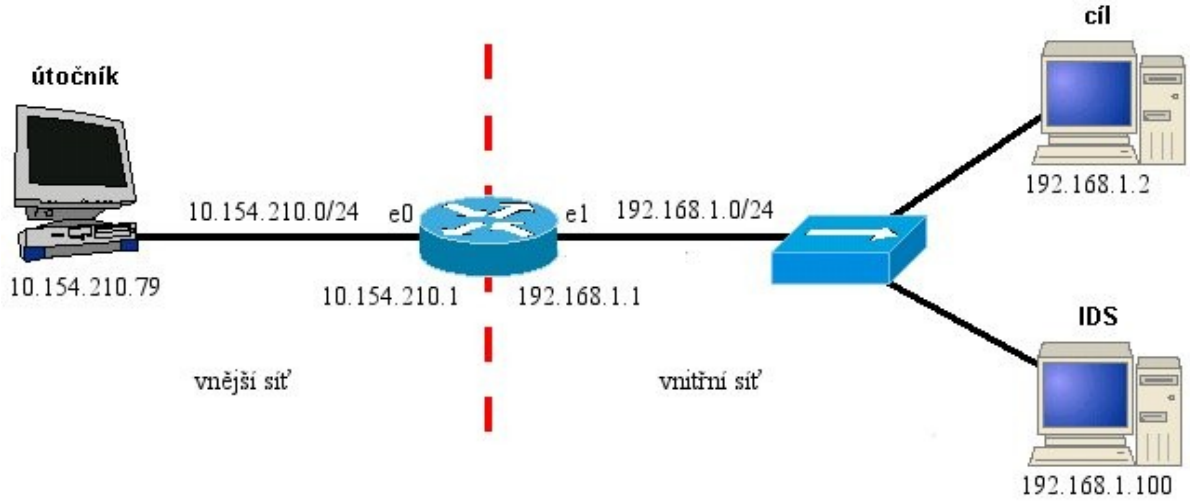
Pro popis formátu zaznamenané výstrahy jsem si náhodně jednu výstrahu vybral ze souboru „`/var/log/snort/alert`“ a na následujícím obrázku ji celou vysvětlím.


GID (Generator ID) [integer] - číslo reprezentující komponentu Snortu, která výstrahu vygenerovala. Seznam lze najít v generators.h
ID signatury nebo také Snort ID [integer] – jednoznačně identifikuje typ varování. Seznam Snort ID pro preprocesory je dispozici v souboru /etc/snort/gen-msg.map. Dále jsou Snort ID uvedena přímo v definicích pravidel (v pravidle, které se implicitně ukládá do .rule souboru).
REV (Revision of the Signature) [integer] – revizní číslo pravidla
Krátky popis signatury [text]
Priority [integer] – závažnost útoku, interval 1 až 4 (nejvíce závažná)
[**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**]
[Classification: Attempted Information Leak] [Priority: 2]
03/11-09:53:16.597467 193.86.238.17:80 -> 10.154.210.2:36042
Časové razítko
Zdrojový port (Souce Port) [integer]
Cílový port (Dest. Port) [integer]
Zdrojová IP adresa (Source IP) [IP octets]
Cílová IP adresa (Dest. IP) [IP octets]
Zařazení (Classification) [text] – popis skupiny, do které se výstraha řadí


Typ služby (Typ Of Service) – Udává prioritu, danou obsahem paketu
DF (Don´t Fragment) - nefragmentovat
Protokol [text]
TCP TTL:51 TOS:0x20 ID:2513 IpLen:20 DgmLen:537 DF
LEN:133
ID paketu (Packet Identification) [integer]
Celková velikost paketu [integer]
Doba života (Time To Live )[integer]
Celková velikost datové části
Velikost IP hlavičky (IP length)

# *Praktická část*

## *Síťová topologie*




Na obrázku je uvedeno schéma zapojení síťové topologie. Počítač útočníka se nachází ve vnější síti a k routeru je připojen přes rozhraní „e0“. Do vnitřní sítě patří rozbočovač, který je k routeru připojený přes rozhraní „e1“. K rozbočovači jsou připojené dva počítače - „cíl“ (může např. poskytovat služby SMTP, WWW, DNS atd.) a „IDS“ (zde poběží náš IDS systém Snort). Místo rozbočovače lze použít přepínač, ale počítač s IDS systémem musí být připojen na monitorovací port (v Cisco terminologii SPAN port), který lze nakonfigurovat tak, aby na něj byl kopírován provoz z ostatních portů.

## *Instalace Snortu*

IDS snort můžeme nainstalovat dvěma způsoby. Pokud Vaše linuxová distribuce disponuje příslušným balíčkem a spokojíte se s danou verzí, zvolte tento způsob.

**(Debian, Ubuntu):**

- apt-get install snort

**Druhou možností je instalace ze zdrojových kódů:**

- přejdeme do adresáře
  cd /usr/local/src
- stáhneme si zdrojové kódy
  `wget http://www.snort.org/dl/current/snort-2.4.4.tar.gz`
- příslušný archiv rozbalíme
  `tar -zxf snort-2.4.4.tar.gz`
- přejdeme do adresáře
  `cd snort-2.4.4`
- použijeme klasickou trojkombinaci pro přeložení a nainstalování
  `./configure && make && make install`

  Musím poznamenat, že jsem ještě doinstaloval některé balíčky, bez kterých by se mi Snort nepodařilo přeložit. V distribuci Ubuntu nebyly nainstalované tyto balíčky:
  libpcre3-dev - Perl 5 Compatible Regular Expression Library - development files

libpcap0.8-dev - Development library and header files for libpcap 0.8

- vytvoříme adresář pro zápis logu
  `mkdir /var/log/snort`
- Vytvoříme adresář pro konfigurační soubory a soubory se signaturami
  `mkdir /usr/local/snort`
- Překopírujeme konfigurační soubory
  `cp -r etc /usr/local/snort`
- Stáhněte si soubor se signaturami, rozbalte a překopírujte
  ```
  wget http://www.snort.org/pub-
  bin/downloads.cgi/Download/vrt_pr/snortrules-pr-2.4.tar.gz
  tar -zxf snortrules-pr-2.4.tar.gz
  cp -r rules /usr/local/snort
  ```

  Soubor se signaturami se na webových stránkách aktualizuje pouze v případě vydání nové verze Snortu.

### *Spuštění snortu*

#### Sniffer mode

Zobrazení IP a TCP/UDP/ICMP hlaviček: `snort -v`

Pro vyzkoušení tohoto modu nám stačí spustit program ping z jiného počítače.

Například: `ping 192.168.1.2`
Odchycené pakety:

```
04/28-16:26:34.733730 10.154.210.79 -> 192.168.1.2
ICMP TTL:58 TOS:0x0 ID:8368 IpLen:20 DgmLen:28
Type:8  Code:0  ID:45841   Seq:34552  ECHO
```

Zobrazení hlaviček spojové vrstvy: `snort -ve`

```
04/28-15:41:03.481718 0:0:C:46:3E:8B -> 0:2:B3:2B:6B:25 type:0x800 len:0x62
10.154.210.79 -> 192.168.1.2 ICMP TTL:63 TOS:0x0 ID:0 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:27395   Seq:1  ECHO
```

Zobrazení včetně datové části paketu: `snort -ved`

Spustíme příkaz z útočníkova počítače: `telnet 192.168.1.2`

Když se naváže TCP spojení (první tři pakety), dochází pak už jen k přenosu dat:

```
04/28-15:46:16.135204 0:0:C:46:3E:8B -> 0:2:B3:2B:6B:25 type:0x800 len:0x4A
10.154.210.79:1035 -> 192.168.1.2:23 TCP TTL:63 TOS:0x10 ID:41626 IpLen:20 DgmLen:60
DF
******S* Seq: 0x65AA3326  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 233916 0 NOP WS: 0
```

```
04/28-15:46:16.135376 0:2:B3:2B:6B:25 -> 0:0:C:46:3E:8B type:0x800 len:0x4A
192.168.1.2:23 -> 10.154.210.79:1035 TCP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:60 DF
***A**S* Seq: 0x62DE8CEA  Ack: 0x65AA3327  Win: 0x16A0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 133491 233916 NOP WS: 0


04/28-15:46:16.136541 0:0:C:46:3E:8B -> 0:2:B3:2B:6B:25 type:0x800 len:0x42
10.154.210.79:1035 -> 192.168.1.2:23 TCP TTL:63 TOS:0x10 ID:41627 IpLen:20 DgmLen:52
DF
***A**** Seq: 0x65AA3327  Ack: 0x62DE8CEB  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 233916 133491


04/28-15:46:16.168213 0:0:C:46:3E:8B -> 0:2:B3:2B:6B:25 type:0x800 len:0x5D
10.154.210.79:1035 -> 192.168.1.2:23 TCP TTL:63 TOS:0x10 ID:41628 IpLen:20 DgmLen:79
DF
***AP*** Seq: 0x65AA3327  Ack: 0x62DE8CEB  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 233919 133491
FF FD 03 FF FB 18 FF FB 1F FF FB 20 FF FB 21 FF  ........... ..!.
FB 22 FF FB 27 FF FD 05 FF FB 23                 ."..'.....#
```

**Paket logger mode**

Při specifikování adresáře pro logování Snort automaticky přejde do módu logování:

```
snort -vde -l /var/log/snort
```

V sítích s velkým provozem je výhodnější logovat do binárního souboru:

```
snort -vde -l /var/log/snort -b
```

Formát binárního souboru je stejný jako u programu tcpdump. Proto může být přečten programy tcpdump, ethereal a také Snortem:

```
snort -vd -r packet.log
```

Význam použitých parametrů:

- „`-r`“ - snort čte a zpracovává tcpdump soubor
- „`-l`“ - loguje do (uvedeného) adresáře

**NIDS mode**

Pro zapnutí NIDS módu je třeba specifikovat konfigurační soubor s definovanými pravidly.

```
snort -c /usr/local/snort/etc/snort.conf -l /var/log/snort
```

## Konfigurační soubor

Snort načítá konfigurační soubor při startu. Vzorový konfigurační soubor je obsažený v distribuci Snortu. Název tohoto souboru si můžeme zvolit libovolný, nicméně implicitně má jméno snort.conf. Pro načtení konfiguračního souboru se v příkazovém řádku využívá direktiva `-c` , pomocí které se dá specifikovat jméno konfiguračního souboru.

***Proměnné***

V konfiguračním souboru můžeme vytvářet proměnné, což je velice výhodné pro tvorbu

pravidel. Například si můžeme v konfiguračním souboru definovat proměnnou HOME_NET (jako reprezentaci vnitřní sítě) a EXTERNAL_NET (vnější síť):

```
var HOME_NET 192.168.1.0/24
var EXTERNAL_NET any
```

Později můžeme tuto proměnnou využít při tvorbě pravidel například takto:

```
alert ip any any -> $HOME_NET any (…)
```

Pokud totiž uvedeme například vnitřní síť v proměnné a v pravidlech tuto proměnnou budeme využívat, stačí nám pro změnu upravit danou síť na jednom místě (čili upravíme pouze HOME_NET). Nemusí se tak procházet všechna pravidla a upravovat příslušné záznamy ručně.

Seznam serverů - explicitní konfigurace seznamu serverů umožní Snortovi hledat útoky na systémy, na kterých služby opravdu běží. Je zbytečné hledat např. HTTP útoky, pokud nikde neběží web server. Takto nadefinované proměnné nejsou však implicitně použity v pravidlech.

```
var DNS_SERVERS 192.168.1.2
var SMTP_SERVERS 192.168.1.2
var HTTP_SERVERS $HOME_NET
```

Dalším vhodným způsobem použití proměnných je nastavení cesty k pravidlům:

```
var RULE_PATH ../rules/
```

***Nastavení direktiv***

Použitím direktiv v souboru snort.conf můžeme konfigurovat mnoho nastavení Snortu. Např. umístění logovacího souboru, dále pak možnost nastavit Snort, aby běžel jako démon (config daemon) nebo vypnout generování varování.

Konfigurace voleb dekodéru se provádí následovně:

```
config direktiva [: hodnota]
```

***Preprocesory***

Preprocesory předzpracovávají přijaté pakety před aplikováním pravidel Snortu. Nastavování preprocesoru je druhá nejvýznamnější část konfiguračního souboru. Tato část poskytuje základní informaci o přidávání nebo odstranění preprocesorů Snortu.

Příkazy pro konfiguraci preprocesoru:

```
preprocessor <preprocesor_jmeno>[: <konfiguracni_volby>]
```

První část řádku je klíčové slovo **preprocessor**, následované jménem preprocesoru. Jestliže preprocesor může přijímat nějaké volby nebo argumenty, můžete je uvést po dvojtečce za koncem jména preprocesoru. Toto však již není povinné.

Preprocesory jsou už v standardní konfiguraci vhodně nakonfigurované.

***Výstupní moduly***

Vstupní moduly pracují s výstupem z pravidel Snortu. Pokud chceme zaznamenávat výstup do databáze, logu či je posílat na určitý port, musíme zavést příslušný modul. Například chceme-li zaznamenávat výstrahy do MySQL databáze, přidáme do konfiguračního souboru tento řádek:

```
output database: alert, mysql, user=snort
```

```
password=tajneheslo dbname=snort host=localhost
```

Vysvětlení uvedených parametrů:

- „`output database:`“ - zaznamenávání do databáze
- „`alert`“ - zaznamenávat výstrahy
- „`mysql`“ - výstup do databáze mysql
- „`user=snort`“ - do databáze má přístup uživatel „snort“
- „`password=tajneheslo`“ - heslo uživatele je „tajneheslo“
- „`dbname=snort`“ - jméno databáze je „snort“
- „`host=localhost`“ - hostname (nebo IP) serveru, na kterém běží MySQL

Obecný formát specifikace výstupního modulu má tento tvar:

```
output <output_module_name>[: <configuration_options>]
```

***Pravidla***

Předkonfigurovaná pravidla jsou rozdělena do jednotlivých souborů, podle zaměření. Pokud chceme určitá pravidla vložit/odstranit, stačí odkomentovat/zakomentovat/přidat řádek s příslušným souborem.

V případě, že chceme přidat do konfiguračního souboru soubor s dalšími pravidly, učiníme tak příkazem

```
include mojepravidla.rules
```

***Ukázkový soubor snort.conf:***

```
###### Definice proměnných, které mají platnost i pro soubory s pravidly
# definice proměnné s vnitřní sítí
var HOME_NET 192.168.1.0/24
# definice proměnné s vnější sítí
var EXTERNAL_NET any
# definice proměnné s IP adresou počítače, kde běží http server
var HTTP_SERVERS 192.168.1.2
# definice proměnné s IP adresou počítače, kde běží DNS server
var DNS_SERVERS 192.168.1.2
# proměnná s relativní cestou k adresáři s pravidly
var RULE_PATH ../rules/

###### nastavení preprocesoru
# S klíčovým slovem frag2 nastavíme časový a paměťový limit potřebný pro
# paketovou defragmentaci. Pokud neuvedeme jinak, použijí se implicitní
# hodnoty – 4 MB (paměťový limit) a 60 sekund (časový limit). Jestliže se v
# uvedeném časovém limitu nepodaří paketová kompletace, dojde k zahození
# nasbíraných fragmentů.
preprocessor frag2

# preprocesor stream4 bude odhalovat neviditelné portscany a generovat pro ně
# výstrahy
preprocessor stream4: detect_scans
```

```
# skládání TCP streamů ze segmentů nesoucích části signatur
preprocessor stream4_reassemble

# aktivace preprocesoru bo, který detekuje zneužití zadních vrátek
# argument -nobrute nastavuje preprocesor tak, že se nebude využívat k detekci
# hrubé síly
preprocessor bo: -nobrute

# tento preprocesor detekuje UDP nebo TCP SYN pakety jdoucí na čtyři různé
# porty za dobu kratší tří sekund
preprocessor portscan: $HOME_NET 4 3 portscan.log

# preprocesor k detekci arp útoku
preprocessor arpspoof

###### výstupní moduly
# záznam paketů v binárním tcpdump formátu
output log_tcpdump: snort.log

# záznam výstrah do databáze
# output database: - zaznamenávání do databáze
# alert - zaznamenávat výstrahy
# mysql - výstup do databáze mysql
# user=snort - do databáze má přístup uživatel „snort"
# password=tajneheslo - heslo uživatele je „tajneheslo"
# dbname=snort - jméno databáze je „snort"
# host=localhost - hostname (nebo IP) serveru, na kterém běží MySQL
output database: alert, mysql, user=snort password=tajneheslo dbname=snort
host=localhost

# záznam výstrah do xml souboru
output xml: alert, file=/var/log/snortxml

###### Pravidla
# každý řádek přidá soubor s pravidly
# $RULE_PATH je proměnná s relativní cestou k pravidlům, definovaná na
# začátku konfiguračního souboru
include $RULE_PATH/bad-traffic.rules
include $RULE_PATH/exploit.rules
include $RULE_PATH/scan.rules
include $RULE_PATH/finger.rules
include $RULE_PATH/ftp.rules
include $RULE_PATH/telnet.rules
include $RULE_PATH/smtp.rules
include $RULE_PATH/rpc.rules
include $RULE_PATH/dos.rules
include $RULE_PATH/ddos.rules
include $RULE_PATH/dns.rules
include $RULE_PATH/tftp.rules
include $RULE_PATH/web-cgi.rules
include $RULE_PATH/web-coldfusion.rules
include $RULE_PATH/web-iis.rules
include $RULE_PATH/web-frontpage.rules
include $RULE_PATH/web-misc.rules
include $RULE_PATH/web-attacks.rules
include $RULE_PATH/sql.rules
include $RULE_PATH/x11.rules
```

```
include $RULE_PATH/icmp.rules
include $RULE_PATH/netbios.rules
include $RULE_PATH/misc.rules
include $RULE_PATH/attack-responses.rules
include $RULE_PATH/myrules.rules
```

### *Praktická ukázka z tvorby pravidel*

Pro ukázku jsem opsal jedno z pravidel ze souboru /etc/snort/rules/icmp.rules

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP
Large ICMP Packet"; dsize: >800; reference:arachnids,246;
classtype:bad-unknown; sid:499; rev:3;)
```

Tímto pravidlem jsme schopni poznat *icmp echo request* paket, který bude mít podezřele velkou datovou část. Jednotlivé aplikace sice vytvářejí tyto pakety o různých délkách a s nestejným obsahem, délka s obsahem větším než 128 bytů je však podezřelá. Klíčovým slovem **alert** říkáme, že naše pravidlo vygeneruje zvolenou metodou výstrahu a paket zaznamená. Slovo **ICMP** specifikuje protokol, nad kterým má být prováděna analýza. $EXTERNAL_NET specifikuje zdrojovou adresu a $HOME_NET adresu cílovou (jde o proměnné, které se definují v konfiguračním souboru `snort.conf`). V tomto případě nám obě „any“ určují, že nezáleží na zdrojovém a cílovém portu. V závorce je popsáno vlastní pravidlo, jednotlivé hodnoty jsou od sebe odděleny středníkem a jsou ve formátu *název_pole: hodnota;* Co jednotlivá slova znamenají, jsem již vysvětlil.

Nyní si pravidlo otestujeme tak, že zadáme na některém počítači z vnější sítě příkaz ping:

```
ping 192.168.1.2 -c 1 -s 1000
```

Použité volby příkazu ping:

- „`-c 1`“ - příkaz ping se provede jedenkrát
- „`-s 1000`“ - velikost zaslaného paketu (1000 bytů)

V souboru /var/log/snort/alert se nám zaznamenala tato výstraha:

```
[**] [1:499:4] ICMP Large ICMP Packet [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
05/21-20:48:16.659920 10.154.210.79 -> 192.168.1.2
ICMP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:1228 DF
Type:8  Code:0  ID:25920   Seq:1  ECHO
[Xref => http://www.whitehats.com/info/IDS246]
```

Druhou malou ukázkou je zachycení odeslané emailové zprávy s určitým obsahem. Jelikož spousta útoků obsahuje statický text, není těžké takový útok odhalit. Ukázkové pravidlo bude generovat výstrahu s prioritou 3 a zprávou „Zachycení mailu“, pokud bude detekován paket s následujícími podmínkami (paket musí):

- Obsahovat řetězec „vas@email.com“ (nocase = nezáleží na velikosti písmen),
- přicházet z libovolného počítače a libovolného portu,
- směřovat na libovolný počítač a port s číslem 25 (služba SMTP).

Definice pravidla:

```
alert tcp any any -> any 25 (msg: "Zachyceni mailu";
content: "vas@email.com"; priority: 3; nocase;)
```

Výstraha:

```
[**] [1:0:0] Zachyceni mailu [**]
[Priority: 3]
05/21-23:00:59.607038 10.154.210.79:60298 -> 192.168.1.2:25
TCP TTL:64 TOS:0x0 ID:54497 IpLen:20 DgmLen:850 DF
***AP*** Seq: 0x4DE99F49  Ack: 0x83CFBF47  Win: 0x16D0 TcpLen: 20
```

### *Generování útoku*

V poslední části mého textu si ukážeme tři nástroje, pomocí kterých si otestujeme IDS Snort. Je dobré vědět, jestli dokáže Snort opravdu zachytit útoky a zda-li dokáže na ně správně reagovat (výstraha, zápis do logu, zahození..).

**IDSwakeup:**

Idswakeup je nástroj k otestování NIDS. Jde o shellový skript, který k vyvolání falešných útoků používá nástroje hping2 (vyžadován) a iwu (který je součásti tohoto balíčku). Nástroj v sobě zahrnuje mnoho simulací útoku a před spuštěním nepotřebuje žádnou konfiguraci.

Instalace:

- (debian, ubuntu) instalace – apt-get install idswakeup
- pro stažení zdrojových kódů idswakeup můžete použít tento odkaz: `http://www.hsc.fr/ressources/outils/idswakeup/download/`
- pro stažení zdrojových kódů hping2 můžete použít tento odkaz: `http://www.hping.org/hping2.0.0-rc1.tar.gz`

Spuštění:

- `idswakeup <zdroj.adr> <cil.adr> [pocetopakovani] [ttl]`

Ukázka práce:

- Z útočníkova počítače (10.154.210.79) spustíme skript pomocí příkazu `idswakeup`, uvedeme ještě adresu počítače „cíl“ (192.168.1.2), počet opakování skriptu (1) a dobu života (10).
- „idswakeup 10.154.210.79 192.168.1.2 1 10“.

První dvě vygenerované výstrahy (v programu Snort):

```
[**] [116:1:1] (snort_decoder) WARNING: Not IPv4 datagram! [**]
06/19-01:21:36.005166

[**] [1:1142:5] WEB-MISC /.... access [**]
[Classification: Attempted Information Leak] [Priority: 2]
06/19-01:21:36.102435 10.154.210.79:2204 -> 192.168.1.2:80
TCP TTL:10 TOS:0x0 ID:92 IpLen:20 DgmLen:89
***AP*** Seq: 0x81B75BC  Ack: 0x724EE835  Win: 0x200  TcpLen: 20
```

**Nessus:**

Nessus je program, který hledá na vzdálených počítačích bezpečnostní chyby, což se dá využít i pro otestování výstrah Snortu, který má za úkol varovat před takovou „nekalou“ činností. Nessus se skládá ze dvou částí. Tou první je démon nessusd, který realizuje všechny bezpečnostní testy, druhou je pak X11 rozhraní, pomocí kterého se Nessus konfiguruje, spouští se scan a také slouží k prohlížení výsledků. Výhoda takového oddělení je např. taková, že můžeme Nessus spouštět z počítače, který chceme otestovat, přičemž nessusd bude nainstalován jinde(nessusd poběží na jiném počítači).

Instalace:

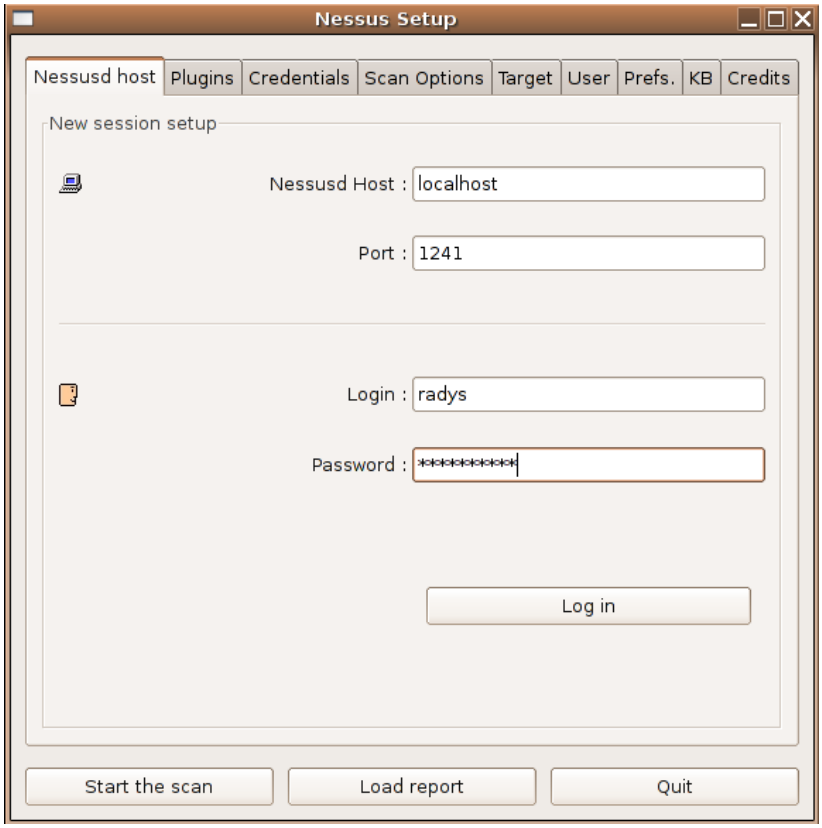
- (Debian, Ubuntu) apt-get install nessus nessusd
- při instalaci budete dotázáni na potřebné údaje pro certifikát (platnost CA certifikátu, platnost serverového certifikátu, stát, území, místo, organizace), který nessusd použije při SSL komunikaci.

Spuštění:

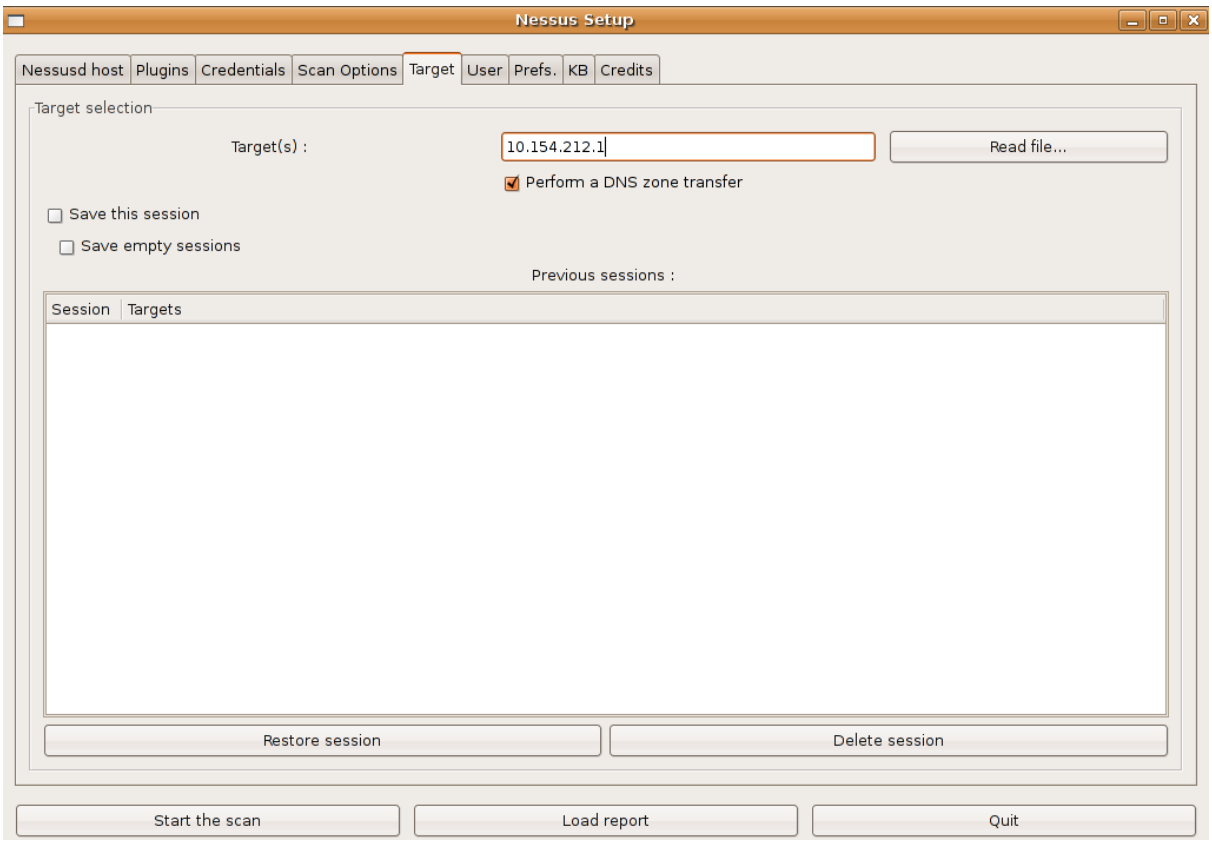
- nejprve je třeba spustit nessusd příkazem se stejným názvem
  ```
  sudo nessusd
  ```

- nyní spustíme program s X11 rozhraním (v případě úplně prvního spuštění si program vyžádá registraci uživatele)
  ```
  sudo nessus
  ```

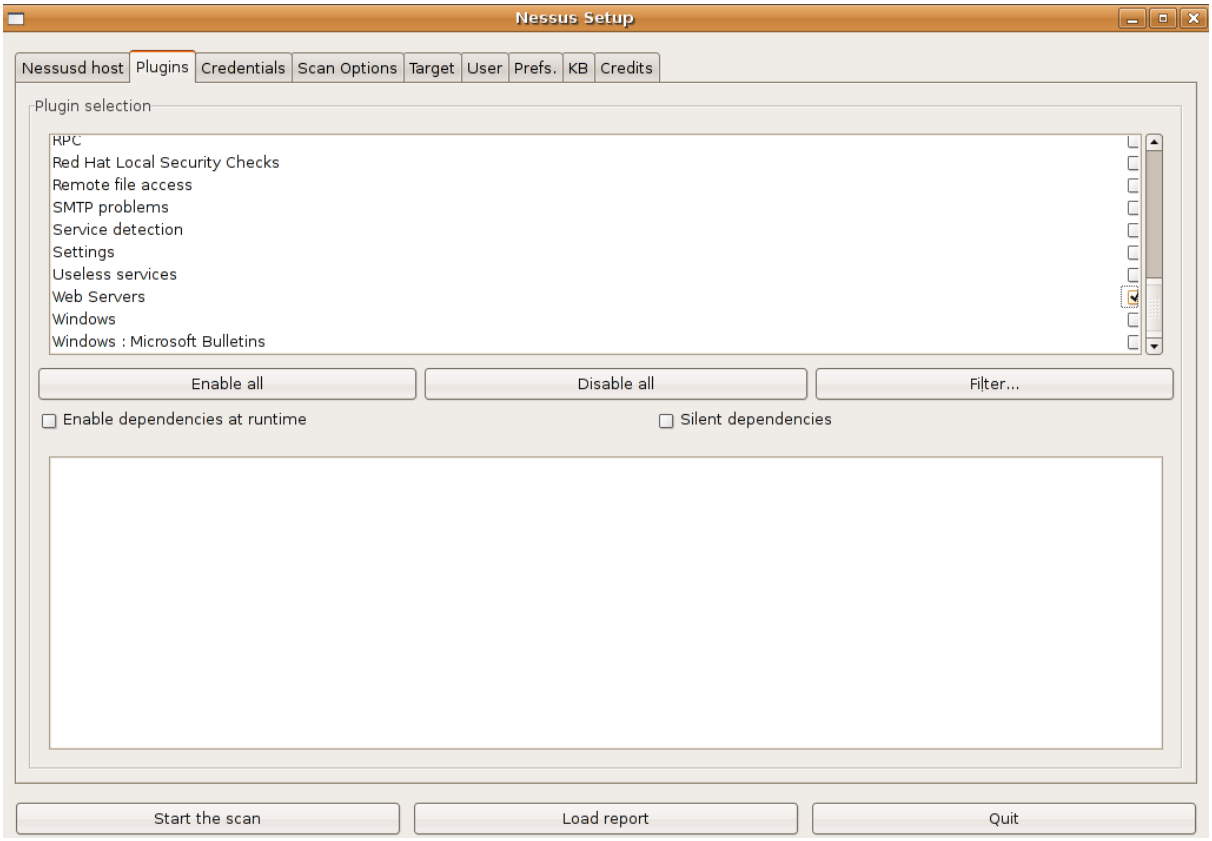
Ukázka práce:




přihlášení k nessusd(program nessus)




cílový počítač pro otestování




Výběr pluginů pro otestování (v mém případě webové servery)

První dvě vygenerované výstrahy (v programu Snort):

```
[**] [1:1242:10] WEB-IIS ISAPI .ida access [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
06/19-00:41:13.897561 10.154.210.79:39708 -> 192.168.1.2:80
TCP TTL:64 TOS:0x0 ID:39023 IpLen:20 DgmLen:313 DF
***AP*** Seq: 0xD25A9747  Ack: 0xFCC8E7EF  Win: 0x16D0  TcpLen: 20
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2000-0071][Xref =>
http://www.securityfocus.com/bid/1065][Xref =>
http://www.whitehats.com/info/IDS552]

[**] [1:971:9] WEB-IIS ISAPI .printer access [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
06/19-00:41:13.911618 10.154.210.79:39712 -> 192.168.1.2:80
TCP TTL:64 TOS:0x0 ID:64013 IpLen:20 DgmLen:322 DF
***AP*** Seq: 0xD206F53E  Ack: 0xFC54D43A  Win: 0x16D0  TcpLen: 20
[Xref => http://cgi.nessus.org/plugins/dump.php3?id=10661][Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=2001-0241][Xref =>
http://www.securityfocus.com/bid/2674][Xref =>
http://www.whitehats.com/info/IDS533]

...
```

**Sneeze:**

Sneeze je generátor falešných útoků, speciálně napsaný pro Snort. Jedná se o perlovský skript, který čte soubory s pravidly, analyzuje tyto soubory a generuje pakety, které budou vyhovovat daným pravidlům. Tento skript byl testovaný se Snortem 1.8 a jeho pravidly. Vývoj tohoto generátoru se na čas zastavil, nicméně to vypadá, že se opět na něm začalo pracovat.

Instalace:

- `wget http://snort.sourceforge.net/sneeze-1.0.tar`
- (nutno: libnet-rawip-perl - Perl interface to lowlevel TCP/IP)
  `apt-get install libnet-rawip-perl`
- archiv stačí rozbalit

Spuštění:

- `./sneeze.pl -d 192.168.1.2 -f /etc/snort/rules/exploit.rules`
- direktiva „`-d`“ nám určuje cílový stroj a „`-f`“ specifikuje soubor s pravidly

### *Ostatní nástroje:*

**Perfmon-graph:**

Perfmonitor-graph (nebo také pmgraph) je jednoduchý perlovský skript, který generuje HTML stránky, s grafem výstupních hodnot Snortu (kolik bylo vygenerováno výstrah za danou časovou periodu, kolik a kdy Snort zahodil paketů, zatížení procesoru atd.), k tomu využívá RRDTool. Současná verze pracuje s perfmonitor preprocesorem, který je již součástí Snortu verze 2.4.0, 2.4.1 a 2.4.21, ale ne ve starších verzích.

## *Literatura:*


IDS - Systémy detekce průniku a jejich význam pro bezpečnost IS
[Dagmar BRECHLEROVÁ - Arnošt VESELÝ]

O'Reilly - Network Security Hacks
[Andrew Lockhart]

Intrusion Detection Systems with Snort
[http://www.phptr.com/content/images/0131407333/downloads/0131407333.pdf]

Writing Snort Rules
[Martin Roesch]

Časopis Computer World *4/2004*

Systémy detekce průniku
[Ing. Roman Aprias]
http://www.cs.vsb.cz/grygarek/SPS/projekty0405/IDS/ids.html