Snort – Intrusion Detection

By David , CISSP

Dynasafe

Agenda

- Introduction to Intrusion Protection (40 min)
  - Intro
  - HIDS / NIDS
  - Deployment
  - Forensics / Sniffer / TCPDump
- The software – Snort (50 min)
  - Installation (Software Dependency)
  - Configuration File
  - Test Running
- Management (80 min)
  - Rules Selecting
  - Snort Rules Base
  - Alerts & Logs

Dynasafe

Agenda (2)

- Snort Rules (60 min)
  - Logic of Rules
  - Rule Bases on the Internet
  - Writing My Own Rules
- Snort Rules (30 min)
  - False Alarms Problem
  - ACID Integration
  - Integration (to Log Management or SIEM)

Dynasafe

入侵防禦與資安管理

Dynasafe

4

入侵防禦 – Introduction

- 利用軟體達到自動化偵測
- 偵測網路及本機惡意行為
  - NIDS vs HIDS
- 入侵防禦的告警(Alert)與日誌(Log)
  - False Alarm 的問題
  - 人工判讀 vs 自動化判讀 -> 集中化管理

Dynasafe

5

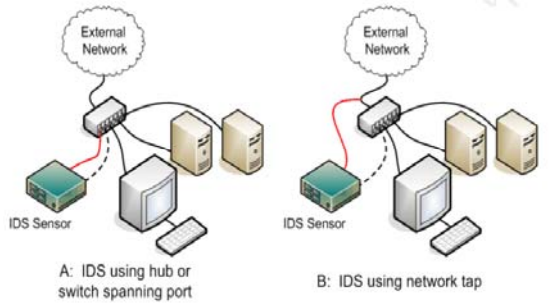
入侵防禦之目的

- 協助單位符合安全性稽核
- 發生大規模惡意入侵時，能查出源頭
- 偵測違反資安政策的設定
- 保護資產，自動化資安管理流程

Dynasafe

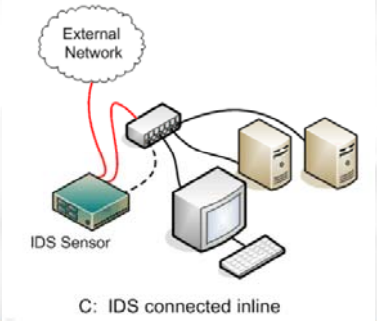
6

入侵防禦部署



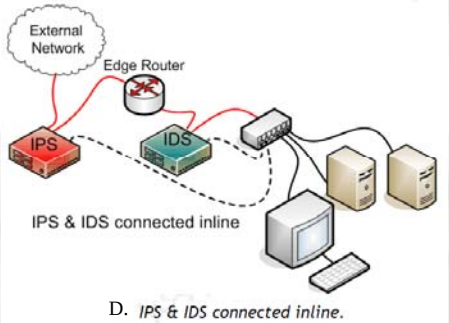
Dynasafe

入侵防禦部署 (2)



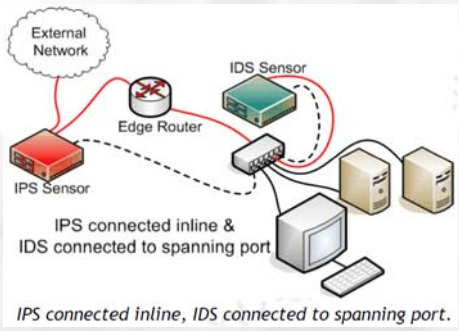
Dynasafe

入侵防禦部署 (3)



Dynasafe

入侵防禦部署 (4)



Dynasafe

10

入侵防禦與資安管理

Dynasafe

11

入侵防禦 vs ...

- Sniffer / Network Dump
- Network Forensics
  - Sniffer + Hash Protection (Non Repudiation)
- Vulnerability Scanner
- Log Management & SIEM

Dynasafe

12

Snort安裝與測試

Snort Installation (Mandriva 2011)

```
[root@mdva109 ~]# urpmi snort
To satisfy dependencies, the following packages are going to be installed:
 Package                  Version      Release       Dist  DEpoch Arch
(medium "Main")
 libdaq0                  0.6.1        1             mdv   2011.0 i586
 libdnet1                 1.12         11            mdv   2011.0 i586
 pcre                     8.12         2             mdv   2011.0 i586
 perl-Archive-Tar         1.680.0      2             mdv   2011.0 noarch (suggested)
 snort                    2.9.1        1             mdv   2011.0 i586
 snort-rules              2.4          6             mdv   2011.0 noarch
(medium "Contrib")
 oinkmaster               2.1          2.20080218.2>             noarch (suggested)
 snortsam                 2.70         1             mdv   2011.0 i586 (suggested)
13MB of additional disk space will be used.
Proceed with the installation of the 8 packages? (Y/n)
```


Installation Confirm

```
# snort --version

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.1 IPv6 GRE (Build 71)
   ''''    The Snort Team: www.snort.org/snort/snort-team
           Copyright (C) 1998-2011 Sourcefire, Inc., et al.
           Using libpcap version 1.1.1
           Using PCRE version: 8.12 2011-01-15
           Using ZLIB version: 1.2.5
```

Installation – Test Running (1)

```
# cat snort-test.conf
config daq: pcap
config daq_dir: /usr/lib/daq
include /root/icmp-basic.rules

# cat /root/icmp-basic.rules
alert icmp any any -> any any (msg:"ICMP Packet"; sid:477; rev:3;)

# snort -c /root/snort-test.conf -l /var/log/snort/
```


Installation – Test Running (2)

```
# find /var/log/snort
/var/log/snort
/var/log/snort/empty
/var/log/snort/snort.log.1345774981
/var/log/snort/alert

# file /var/log/snort/snort.log.1345774981
snort.log.1345774981: tcpdump capture file (little-endian) –
 version 2.4 (Ethernet, capture length 1514)
```


Installation – Test Running (3)

```
# cat /var/log/snort/alert
[**] [1:477:3] ICMP Packet [**]
[Priority: 0]
08/24-10:30:10.680421 10.0.0.1 -> 10.0.0.161
ICMP TTL:128 TOS:0x0 ID:3901 IpLen:20 DgmLen:60
Type:8  Code:0  ID:3   Seq:51944  ECHO

[**] [1:477:3] ICMP Packet [**]
[Priority: 0]
08/24-10:30:10.680589 10.0.0.161 -> 10.0.0.1
ICMP TTL:64 TOS:0x0 ID:10863 IpLen:20 DgmLen:60
Type:0  Code:0  ID:3  Seq:51944  ECHO REPLY
```

Installation – Test Running (4)

```
# snort -A fast -c /root/snort-test.conf -l /var/log/snort/

# cat /var/log/snort/alert

08/24-10:52:53.643931  [**] [1:477:3] ICMP Packet [**] [Priority:
  0] {ICMP} 10.0.0.1 -> 10.0.0.161

08/24-10:52:53.644113  [**] [1:477:3] ICMP Packet [**] [Priority:
  0] {ICMP} 10.0.0.161 -> 10.0.0.1
```

Dynasafe

Snort Configuration & Rules

Dynasafe

Snort Configuration

- Snort configuration consists of:
  - Global configuration (snort.conf)
  - Optional *.rules file(s)
  - Additional files

Dynasafe

Snort Rules

- Default (Sourcefire) Rules
  - Auto Update
- Breeding Edge Rules
  - Oinkmaster Tool
- Customized Rules

Dynasafe

Snort Rules

- Default (Sourcefire VRT) Rules
  - Auto Update
  - Free for use after **7 days**
- Breeding Edge Rules
  - Use Oinkmaster Tool
- Customized Rules

Dynasafe

Snort Rules - Sources

- Default (Sourcefire) Rules
  - Auto Update
- Breeding Edge Rules
  - Oinkmaster Tool
- Customized Rules

Dynasafe

File: */etc/snort/snort.conf*

```
ipvar HOME_NET 192.168.3.0/24
ipvar EXTERNAL_NET !$HOME_NET
portvar HTTP_PORTS 80
var RULE_PATH rules
…
config decoder
config logdir
…
preprocessor http_inspect
preprocessor rpc_decode
preprocessor smtp: ports { 25 465 587 691 }
```

Dynasafe

File: */etc/snort/snort.conf* (2)

```
include $RULE_PATH/attack-responses.rules
include $RULE_PATH/backdoor.rules
include $RULE_PATH/bad-traffic.rules
include $RULE_PATH/blacklist.rules
include $RULE_PATH/botnet-cnc.rules
include $RULE_PATH/dos.rules
include $RULE_PATH/exploit.rules
include $RULE_PATH/ftp.rules
include $RULE_PATH/icmp.rules
……
```

Dynasafe

The Rule Header

```
alert tcp any any -> any any (msg:"Sample alert";)
```

- Header contains the following fields
  - Action (log, alert)
  - Protocol (ip, tcp, udp, icmp, any)
  - Src IP & Port
  - Dst IP & Port
  - Direction operator ("->", "<>")

Dynasafe

The Rule Header (2)

```
alert tcp $EXT_NET any -> 192.168.3.0/24 80 (msg:"Alert A";)
```

- Src or dst IP addresses can be:
  - Variables (like $EXT_NET)
  - Individual IP addresses
  - CIDR IP Subnets
  - Lists of the above
    - ("[192.168.3.12,192.168.3.9]")
- Ports can be:
  - Individual ports
  - Port ranges ("80:85", ":1024", "1025:")

Dynasafe

The Rule Body

```
alert tcp $EXT_NET any -> 192.168.3.0/24 80 (msg:"Alert A";)
```

- The body is usually the complex part
  - Begins and ends with "()"
  - Series of "rule options" (<關鍵字>:<內涵> ， 並以分號區隔)

Dynasafe

Rule Options

- Five types of options
  - Metadata
  - Payload detection
  - Non-payload detection
  - Post-detection
  - Thresholding and suppression

Dynasafe

Metadata options

- "msg" specifies the human-readable alert message
- "reference" includes a URL for more info
- "classtype" and "priority" give some idea about the type of attack and the severity of the event
- "sid" and "rev" uniquely identify the rule (including revisions & edits)

Dynasafe

Payload Detection Options

- Start with the basics
  - "content" looks for a string of bytes
  - "nocase" modified content, makes it case insensitive
  - "offset" skips a certain number of bytes before searching
  - "pcre" allows the use of Perl-compatible regular expressions (support must be compiled in)

Dynasafe

Real Rule Sample (1)

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 110
(msg:"POP3 APOP USER overflow attempt";

 flow:to_server,established;

 content:"APOP"; nocase;

 isdataat:256,relative;

 pcre:"/^APOP\s+USER\s[^\n]{256}/smi";

 reference:bugtraq,9794; classtype:attempted-admin;
 sid:2409; rev:1;
)
```

Dynasafe

Real Rule Sample (2)

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 995
(msg:"POP3 PCT Client_Hello overflow attempt";
 flow:to_server,established;
 flowbits:isnotset,sslv2.server_hello.request;
 flowbits:isnotset,sslv3.server_hello.request;
 flowbits:isnotset,tlsv1.server_hello.request;
 content:"|01|"; depth:1; offset:2; byte_test:2,>,0,5;
 byte_test:2,!,0,7; byte_test:2,!,16,7;
 byte_test:2,>,20,9; content:"|8F|"; depth:1; offset:11;
 byte_test:2,>,32768,0,relative; reference:bugtraq,10116;
 reference:cve,2003-0719;
 reference:url,www.microsoft.com/technet/security/bulleti
 n/MS04-011.mspx; classtype:attempted-admin; sid:2518;
 rev:13;)
```

Dynasafe

34

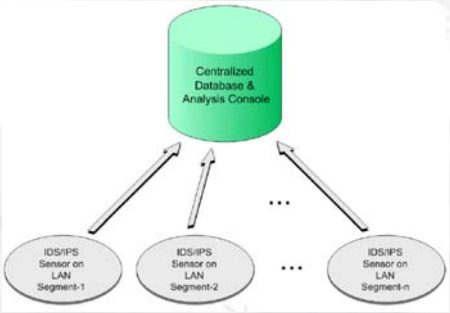
Situations

- IDS vs IPS
- False Positives
- Log Management
- Benefits of Security Management ?

Dynasafe

36

日誌集中化管理、分析



Dynasafe

37

Snort 簡介、安裝與使用

建議：安裝 Snort 之前，先安裝 tcpdump。（即安裝 pcap library，有利於 Troubleshooting）

Snort 簡介

Snort 是一個基於 libpcap 的資料封包嗅探器，可以作為一個輕量級的網路入侵偵測系統（NIDS）。Snort 可運行在多種作業系統平臺，例如 UNIX 系列和 Windows.（需要 libpcap for Win32 的支援），與很多商業產品相比，它對作業系統的依賴性比較低。其次使用者可以根據自己的需要及時在短時間內調整檢測策略。

Snort 整合了多種告警機制來提供即時告警功能，包括：syslog、用戶指定檔、UNIX Socket、通過 SMB Client 使用 WinPopup 對 Windows 用戶端告警。 Snort 的現實意義維作為開源軟體填補了只有商業入侵偵測系統的空白，可以幫助中小網路的系統管理員有效地監視網路流量和檢測入侵行為。

FAQ 01 - 配置 Snort Senor 的地方哪兒比較好？

答：
必須根據你公司或單位的政策而定，而且你想要保護哪些。
有一個方面來看，你要把它擺在防火牆之前或之後。

把入侵偵測系統(IDS)放在防火牆之前，會讓你可以監測所有對你網路的攻擊，不管它們會不會被防火牆所阻止。這幾乎就是說，入侵偵測系統會比放在防火牆之後的入侵偵測系統看到更多的事件，更多的紀錄會被產生。

如果只對監測穿越過防火牆的流量有興趣，就把入侵偵測系統擺在防火牆之後。如果資源許可的話，最好是放在一台在防火牆之前，一台在防火牆之後。這樣你就可以看到所以要到你的網路的流量，及任何真正進入你網路的流量。

Mandriva 2011 安裝過程：

```
[root@mdva109 ~]# urpmi snort
To satisfy dependencies, the following packages are going to be installed:
 Package                    Version      Release      Dist  DEpoch Arch
```

```
(medium "Main")
 libdaq0                       0.6.1       1            mdv   2011.0 i586
 libdnet1                      1.12        11           mdv   2011.0 i586
 pcre                          8.12        2            mdv   2011.0 i586
 perl-Archive-Tar              1.680.0      2            mdv   2011.0 noarch (suggested)
 snort                         2.9.1       1            mdv   2011.0 i586
 snort-rules                    2.4         6            mdv   2011.0 noarch
(medium "Contrib")
 oinkmaster                    2.1         2.20080218.2>             noarch (suggested)
 snortsam                      2.70        1            mdv   2011.0 i586 (suggested)
13MB of additional disk space will be used.
Proceed with the installation of the 8 packages? (Y/n) Y

http://ftp.kddilabs.jp/Linux/Mandriva/official/2011/i586/media/main/release/snort-2.9.1-1-mdv2011.0.i586.rpm

installing snort-2.9.1-1-mdv2011.0.i586.rpm pcre-8.12-2-mdv2011.0.i586.rpm
perl-Archive-Tar-1.680.0-2-mdv2011.0.noarch.rpm libdnet1-1.12-11-mdv2011.0.i586.rpm
snort-rules-2.4-6-mdv2011.0.noarch.rpm oinkmaster-2.1-2.20080218.2mdv2010.0.noarch.rpm
libdaq0-0.6.1-1-mdv2011.0.i586.rpm snortsam-2.70-1-mdv2011.0.i586.rpm from /var/cache/urpmi/rpms
warning: LOOP:
warning: removing snort-rules-2.4-6.noarch "Requires(hint): oinkmaster" from tsort relations.
warning: removing oinkmaster-2.1-2.20080218.2mdv2010.0.noarch "Requires: /etc/snort" from tsort relations.
warning: removing snort-2.9.1-1.i586 "Requires(post): snort-rules" from tsort relations.
Preparing...
########################################################################################################################
     1/8: snortsam
########################################################################################################################
     2/8: libdaq0
########################################################################################################################
     3/8: libdnet1
########################################################################################################################
     4/8: pcre
########################################################################################################################
     5/8: perl-Archive-Tar
########################################################################################################################
     6/8: snort
########################################################################################################################
     7/8: snort-rules
########################################################################################################################
     8/8: oinkmaster
########################################################################################################################
```

怎樣使用 Snort　[來源:本文出自:http://xfocus.org 作者: xundi　]

安裝方法：

如果你安裝好了 libpcap 後，對 snort 安裝將是很簡單，關於 libpcap 的安裝說明，
你可以看看 blackfire(http://go.163.com/~bobdai/的一些文章，關於 WINDOWS
下的 winpcap 你可以看我站上的 SNIFFER FOR NT 上的安裝說明。裝好 libpcap 後，
你可以使用通常的命令：

1.) ./configure

2.) make

3.) make install

裝好後你可以使用 make clean 清除一些安裝時候產生的檔。
（有些系統如 freebsd 已經支援了 libpcap，所以很輕鬆，不用再裝了）。

而 WINDOWS 更簡單，只要解包出來就可以了；

參數介紹：

命令列是 snort -[options] <filters>

選項：

-A <alert> 設置<alert>的模式是 full,fast,還是 none;full 模式是記錄標準的 alert 模式到 alert 檔中；Fast 模式只寫入時間戳記，messages, IPs,ports 到檔中，None 模式關閉報警。

-a 是顯示 ARP 包；

-b 是把 LOG 的資訊包記錄為 TCPDUMP 格式，所有資訊包都被記錄為兩進制形式，名字如 snort-0612@1385.log，這個選項對於 FAST 記錄模式比較好，因為它不需要花費包的資訊轉化為文本的時間。Snort 在 100Mbps 網路中使用"-b"比較好。

-c <cf> 使用設定檔<cf>,這個規則檔是告訴系統什麼樣的資訊要 LOG，或者要報警，或者通過。

-C 在資訊包資訊使用 ASCII 碼來顯示，而不是 hexdump，

-d 解碼應用層。

-D 把 snort 以守護進程的方法來運行，預設情況下 ALERT 記錄發送到/var/log/snort.alert 文件中去。

-e 顯示並記錄 2 個資訊包頭的資料。

-F <bpf>從<bpf>檔中讀 BPF 篩檢程式（filters），這裡的 filters 是標準的 BPF 格式篩檢程式，你可以在 TCPDump 裡看到，你可以查看 TCPDump 的 man 頁怎樣使用這個篩檢程式。

-h <hn>設置網路位址，如一個 C 類 IP 位址 192.168.0.1 或者其他的，使用這個選項，會使用箭頭的方式資料進出的方向。

-I <if> 使用網路介面參數<if>

-l <ld> LOG 資訊包記錄到<ld>目錄中去。

-M <wkstn> 發送 WinPopup 資訊到包含<wkstn>檔中存在的工作站列表中去，這選項需要 Samba 的支援，wkstn 檔很簡單，每一行只要添加包含在 SMB 中的主機名稱即可。（注意不需要\\兩個斜杠）。

-n <num> 是指定在處理<num>個資料包後退出。

-N 關閉 LOG 記錄，但 ALERT 功能仍舊正常。

-o 改變所採用的記錄檔，如正常情況下採用 Alert->Pass->Log order，而採用此選項是這樣的順序：Pass->Alert->Log order，其中 Pass 是那些允許通過的規則而不記錄和報警，ALERT 是不允許通過的規則，LOG 指 LOG 記錄，因為有些人就喜歡奇奇怪怪，象 CASPER，QUACK 就喜歡反過來操作。

-p 關閉雜亂模式嗅探方式，一般用來更安全的調試網路。

-r <tf> 讀取 tcpdump 方式產生的檔<tf>,這個方法用來處理如得到一個 Shadow(Shadow IDS 產生)檔，因為這些檔不能用一般的 EDIT 來編輯查看。

-s　　LOG 報警的記錄到 syslog 中去，在 LINUX 機器上，這些警告資訊會出現在/var/log/secure,在其他平臺上將出現在/var/log/message 中去。

-S <n=v>這個是設置變數值，這可以用來在命令列定義 Snort rules 檔中的變數，如你要在 Snort rules 檔中定義變數 HOME_NET,你可以在命令列中給它預定義值。

-v　　使用為 verbose 模式，把資訊包列印在 console 中，這個選項使用後會使速度很慢，這樣結果在記錄多的是時候會出現丟包現象。

-V　　顯示 SNORT 版本並退出；

下面是一些命令的組合介紹，當然更多的組合你可以自己去測試：

Snort 存在比較多的命令選項和參數，先來介紹一些基本的一些命令，如果你想要把資訊包的頭顯示在螢幕上，你可以使用：

snort -v

這個命令會運行 Snort 和顯示 IP 和 TCP/UDP/ICMP 頭資訊。
我使用了 ping 192.168.0.1 就顯示了如下資訊：

06/10-10:21:13.884925 192.168.0.2 -> 192.168.0.1
ICMP TTL:64 TOS:0x0 ID:4068
ID:20507　　Seq:0　　ECHO

06/10-10:21:13.885081 192.168.0.1 -> 192.168.0.2
ICMP TTL:128 TOS:0x0 ID:15941
ID:20507　　Seq:0　　ECHO REPLY

06/10-10:21:14.884874 192.168.0.2 -> 192.168.0.1
ICMP TTL:64 TOS:0x0 ID:4069
ID:20507　　Seq:256　　ECHO

06/10-10:21:14.885027 192.168.0.1 -> 192.168.0.2
ICMP TTL:128 TOS:0x0 ID:15942
ID:20507　　Seq:256　　ECHO REPLY

如果你想要解碼應用層，就使用:

snort -vd
再次使用 ping 192.168.0.1 就顯示了如下資訊：

06/10-10:26:39.894493 192.168.0.2 -> 192.168.0.1
ICMP TTL:64 TOS:0x0 ID:4076

```
ID:20763      Seq:0    ECHO
58 13 42 39 E0 BB 05 00 08 09 0A 0B 0C 0D 0E 0F    X.B9............
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F    ................
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F     !"#$%&'()*+,-./
30 31 32 33 34 35 36 37                                01234567

06/10-10:26:39.894637 192.168.0.1 -> 192.168.0.2
ICMP TTL:128 TOS:0x0 ID:15966
ID:20763      Seq:0    ECHO REPLY
58 13 42 39 E0 BB 05 00 08 09 0A 0B 0C 0D 0E 0F    X.B9............
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F    ................
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F     !"#$%&'()*+,-./
30 31 32 33 34 35 36 37                                01234567
```

如果要看到更詳細的關於有關 ethernet 頭的資訊，就要使用：

```
snort -vde
```

使用 ping 192.168.0.1 就顯示了如下資訊：

```
-*> Snort! <*-
Version 1.6-WIN32
By Martin Roesch (roesch@clark.net, www.clark.net/~roesch)
WIN32 Port By Michael Davis (Mike@eEye.com, www.datasurge.net/~mike)
06/10-10:32:01.345962 0:60:94:F9:5E:17 -> 0:50:BA:BB:4A:54 type:0x800 len:0x62
192.168.0.2 -> 192.168.0.1 ICMP TTL:64 TOS:0x0 ID:4079
ID:21787      Seq:0    ECHO
99 14 42 39 47 4C 0C 00 08 09 0A 0B 0C 0D 0E 0F    ..B9GL..........
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F    ................
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F     !"#$%&'()*+,-./
30 31 32 33 34 35 36 37                                01234567

06/10-10:32:01.346164 0:50:BA:BB:4A:54 -> 0:60:94:F9:5E:17 type:0x800 len:0x62
192.168.0.1 -> 192.168.0.2 ICMP TTL:128 TOS:0x0 ID:16090
ID:21787      Seq:0    ECHO REPLY
99 14 42 39 47 4C 0C 00 08 09 0A 0B 0C 0D 0E 0F    ..B9GL..........
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F    ................
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F     !"#$%&'()*+,-./
30 31 32 33 34 35 36 37                                01234567
```

當然上面的一些命令你只是在螢幕上看到，如果要記錄在 LOG 檔上，你可以先建立一個 log 目錄，在使用下面的命令：

```
snort -dev -l ./log -h 192.168.0.1/24
```

這個命令就使 Snort 把 ethernet 頭資訊和應用層資料記錄到./log 目錄總去了，並記錄的是關於 192.168.0.1 CLASS C 的資訊，

如果你想利用一些規則檔（一些記錄特定資料的規則檔，如 SYN ATTACK 等記錄）就使用：

snort -dev -l ./log -h 192.168.1.0/24 -c snort-lib

這裡的 Snort-lib 是你的規則檔的檔案名，這將採用 snort-lib 檔中設置的規則來決定是否記錄某個資訊包。而

snort -d -h 192.168.1.0/24 -l ./log -c snort-lib 可以不記錄一些 ethernet 頭資訊
如，我用./nmap -sS 192.168.0.1 -p 21 就在/log/alert.ids 中記錄如下資訊：

[**] IDS246 - MISC - Large ICMP Packet [**]
06/12-13:48:31.992395 192.168.0.1 -> 192.168.0.2
ICMP TTL:128 TOS:0x0 ID:36579
ID:46802 Seq:0 ECHO REPLY

我故意使用了 rules 出現的規則 php.cgi/?，如 192.168.0.1/cgi-bin/php.cgi/?，就顯示：

[**] IDS232 - WEB-CGI-PHP CGI access attempt [**]
06/12-13:53:35.106323 192.168.0.2:1789 -> 192.168.0.1:80
TCP TTL:64 TOS:0x0 ID:8945 DF
*****PA* Seq: 0xA070C880 Ack: 0xF113872 Win: 0x7D78

snort -d -h 192.168.1.0/24 -l ./log -c snort-lib -s 就會把日誌記錄在你規則檔中所定義的 LOG 檔中，而不是默認的 alert.ids 中。

snort -d -h 192.168.1.0/24 -l ./log -c snort-lib -o 此命令是讀規則檔的順序，有些人很奇怪，需要先讀允許的規則檔，再讀 alert 規則檔，然後來 LOG 記錄，那就按照上面的命令來操作。

如果你的網路請求相當多，你可以使用：

snort -b -A fast -c snort-lib
這樣，每一條規則內的警告消息就分開記錄，對於多點同步探測和攻擊的記錄可以不容易丟包。當然這樣記錄的 LOG 檔是兩進制的，類似與 tcpdump 的格式，你可以使用這樣的方法來查看這些 LOG：

snort -d -c snort-lib -l ./log -h 192.168.1.0/24 -r snort.log

---

structure of the alert:

<Rule Actions> <Protocol> <Source IP Address> <Source Port> <Direction Operator> <Destination IP Address> <Destination > (rule options)

規則檔中的選項的意義：

```
#     msg => message to output in the alert/log files
#     flags => TCP flags, use 0 for no flags at all
#     ttl => the TTL value you want to key on (nice for catching traceroutes)
#     content => the packet application layer, look for buffer overflows here
#     itype => the NUMBER of the ICMP type
#     icode => the NUMBER of the ICMP code
#     minfrag => minimum fragment payload size
#     seq => tcp sequence number
#     ack => tcp ack number
#     id => IP header fragment ID number
#     logto => file to log specific alerts to
#     dsize => match on the packet payload size
#     offset => start a content search <offset> bytes into the payload
#     depth => only search <depth> bytes into the payload for a pattern match
#     session => record the session traffic from clear text protocols like
#                        ftp or telnet
#     ipopts => check for a specific IP option
```

使用 Oinkmaster 自動升級 Snort 規則
作者：JP Vossen TechTarge (2005-04-01 11:25:23)

當你使用 Snort 的時候，你可能會發現一些預設的規則對你毫無用處，而一些你需要的規則卻不能在預設情況下使用，甚至有時候你必須修改一些規則。由於 Snort 的規則像你的防毒工具一樣需要定期更新，而每次下載新的規則後手動恢復以前的設置又很不切實際。

值得慶幸的是，有一個免費工具 Oinkmaster 可以幫你做到這一切，它可以運行在 UNIX 和 Windows 系統上。（由於 Oinkmaster 是用 Perl 語言編寫的，因此在 Windows 系統上運行時需要 ActivePerl 檔的支援）。

在經過幾年的測試運行後，2004 年 5 月發佈了第一個正式版本的 Oinkmaster。事實上在 Snort 團體內部，Oinkmaster 一直 被當作一個規則更新器來使用。Oinkmaster 按照設定檔中你所定義的操作運行。首先，你要通過超文字傳輸協定、安全超文字傳輸協定、檔案傳輸協議、檔和 SCP 方法獲得最新的規則。然後，你定義哪些檔需要更新而哪些檔需要跳過；哪些

簽名 ID（SIDs）需要修改；哪些簽名 ID 應該使能而哪些 簽名 ID 應該禁用。考慮到可以用範本方式進行配置，你也可以根據需要任意引用檔。Oinkmaster 自帶的設定檔具有完備的歸檔資訊，而且包含有用 的預設配置，但是你仍需要仔細流覽以確認你下載的規則快照與你運行的 Snort 版本相符合。

你可以使用 Oinkmaster 直接升級你的嗅探器，而另一個更好的辦法是：首先用一個設定檔去更新一個分段目錄，並且報告更新的詳細資訊。只要你確認並通過了這些修改，就可以使用另一個設定檔去實際升級嗅探器。（詳情請見 http://oinkmaster.sourceforge.net/faq.shtml 上 Oinkmaster 的常見問答集 Q3）。

你只需要將簽名 ID 添加到“使能”或者“禁用”列表中，就能輕易地使能或者禁用它。修改一個簽名 ID 也很簡單，只要用一個規則運算式描述你的改 變就可以。利用範本還可以簡化大量的重複修改操作。由於 Snort 規則也是使用規則運算式描述的，因此，如果你對規則運算式不是很熟悉，還是有必要去瞭解 一些。因為在其它程式設計領域規則運算式的應用也非常廣泛。

如何在 Snort 之中使用 Bleedingsnort Rules – 快速解說版

(http://www.bleedingsnort.com/)

Sample *oinkmaster-bleedingsnort.conf* for use with the Bleeding Snort Ruleset:

```
---------- Sample Start ----------
url = http://www.bleedingsnort.com/downloads/bleeding.rules.tar.gz
path = /bin:/usr/bin:/usr/local/bin
tmpdir = /tmp
update_files = \.rules$|\.config$|\.conf$|\.txt$|\.map$
skipfile local.rules
---------- Sample End ----------
```

Then run oinkmaster like so:

```
oinkmaster.pl -q -C Oinkmaster-bleedingsnort.conf -o ./rules
```

Adjust ./rules to be your rules dir of course.

Add these lines to your snort.conf:

```
include $RULE_PATH/bleeding-virus.rules
include $RULE_PATH/bleeding-attack-response.rules
include $RULE_PATH/bleeding-policy.rules
include $RULE_PATH/bleeding-custom.rules
```

```
include $RULE_PATH/bleeding-dos.rules
include $RULE_PATH/bleeding-exploit.rules

include $RULE_PATH/bleeding-inappropriate.rules
include $RULE_PATH/bleeding-malware.rules
include $RULE_PATH/bleeding-p2p.rules
include $RULE_PATH/bleeding-scan.rules
include $RULE_PATH/bleeding-web.rules
```

or just:

```
include $RULE_PATH/bleeding-all.rules
```

And finally, only if you are using Barnyard or some other tool that relies on a sid-msg.map you need to add the bleedingsnort map to the stock file like so:

```
cp sid-msg.map sid-msg.map.orig
cat bleeding-sid-msg.map sid-msg.map.orig | sort -u > sid-msg.map
```

And you're ready to go!!!

Snort 與其他工具的比較。

Snort 的主要用途就是網路監視、資料包的記錄和檢測入侵行為，下面是與分別具有上述兩種功能的典型工具的比較。

>--snort 與 tcpdump 的比較
Tcpdump 是最為經典的嗅探工具，主要是用於記錄網路資料，網路故障的探測診斷工具。Snort 與它的最大的共同之處在於都是基於 libpcap 的並且支持 BPF 過濾機制，所以本質上都是調用的捕獲資料包的庫函數，但是 snort 的目的不僅僅是在於記錄這個資料包而是從安全的角度考慮出發區解析它，並且 tcpdump 主要是分析第二層或者第三層的報文來進行網路故障診斷，而 snort 則主要針對于應用層的資料進行分析從而實現檢測入侵行為。除此之外，由於 tcpdump 旨在快速完整地記錄流量，所以它制定了特殊的輸出格式，速度快但是不易看懂，而 snort 就提供了更為友好的輸出格式，有利於系統管理員的直接分析。

--Figure 1 - Typical Snort telnet packet display:

```
---------------------------------------------------------------------------
20:59:49.153313 0:10:4B:A9:66 -> 0:60:97:7:C2:8E type:0x800 len:0x7D
192.168.1.3:23 -> 192.168.1.4:1031 TCP TTL:64 TOS:0x10 DF
***PA* Seq: 0xDF4A6536 Ack: 0xB3A6FD01 Win: 0x446A
FF FA 22 03 03 E2 03 04 82 0F 07 E2 1C 08 82 04 .."...........
09 C2 1A 0A 82 7F 0B 82 15 0F 82 11 10 82 13 FF ..............
```

F0 0D 0A 46 72 65 65 42 53 44 20 28 65 6C 72 69 ...FreeBSD (elri
63 2E 68 6F 6D 65 2E 6E 65 74 29 20 28 74 74 79 c.home.net) (tty
70 30 29 0D 0A 0D 0A p0)....

--Figure 2 - The same telnet packet as displayed by tcpdump:

20:59:49.153313 0:10:4b:d:a9:66 0:60:97:7:c2:8e 0800 125: 192.168.1.3.23 >
192.168.1.4.1031: P 76:147(71) ack 194 win 17514 (DF) [tos 0x10] (ttl 64,
id 660)
4510 006f 0294 4000 4006 b48d c0a8 0103
c0a8 0104 0017 0407 df4a 6536 b3a6 fd01
5018 446a d2ad 0000 fffa 2203 03e2 0304
820f 07e2 1c08 8204 09c2 1a0a 827f 0b82
150f 8211 1082 begin_of_the_skype_highlighting 免費 8211 1082
end_of_the_skype_highlighting 13ff f00d 0a46 7265 6542 begin_of_the_skype_highlighting 免
費 7265 6542 end_of_the_skype_highlighting
5344 2028 begin_of_the_skype_highlighting 免費 5344 2028 end_of_the_skype_highlighting
656c 7269 632e 686f 6d65 2e6e
6574 2920 2874 7479 7030 290d 0a0d 0a

2.--原理

snort 作為一個 NIDS，其工作原理為在基於共用網路上檢測原始的網路傳輸資料，通過分析捕獲的資料包，主要工作為匹配入侵行為的特徵或者從網路活動的角度檢測異常行為，進而採取入侵的預警或記錄。從檢測模式而言，snort 屬於是誤用檢測（misuse detection）。[注：該方法對已知攻擊的特徵模式進行匹配，包括利用工作在網卡混雜模式下的嗅探器被動地進行協議分析，以及對一系列資料包解釋分析特徵。]

從本質上上來說，snort 是基於規則檢測的入侵偵測工具，即針對每一種入侵行為，都提煉出它的特徵值並按照規範寫成檢驗規則，從而形成一個規則資料庫。其次將捕獲得資料包按照規則庫逐一匹配，若匹配成功，則認為該入侵行為成立。目前，snort 的檢測規則庫主要包括了以下幾類的入侵行為：

snort 的結構主要分為三個部分如圖 n-1：

l--資料包捕獲和解析子系統（Capture Packet Mechanism from link layer and the packet decoder ）：

該子系統的功能為捕獲網路得傳輸資料並按照 TCP/IP 協定的不同層次將資料包進行解析。Snort 利用 libpcap 庫函數進行採集資料， 該庫函數可以為應用程式提供直接從鏈路層捕獲資料包的介面函數並可以設置資料包的篩檢程式以來捕獲指定的資料。(的詳細介紹請參閱附錄 N)。網路資料獲取和解析機制是整個 NIDS 實現的基礎，其中最關鍵的是要保證高速和低的丟包率，這不僅僅取決於軟體的效率還同硬體的處理能力相關。對於解析

機制來說，能夠處理資料包的類型的多樣性也同樣非常重要，目前，snort 可以處理乙太網，權杖環以及 SLIP 等多種鏈路類型的包。

I--檢測引擎（the detect engine）
檢測引擎是一個 NIDS 實現的核心，準確性和快速性是衡量其性能的重要指標，前者主要取決於對入侵行為特徵碼的提煉的精確性和規則撰寫的簡潔實用性，由於網路入侵偵測系統自身角色的被動性——只能被動的檢測流經本網路的資料，而不能主動發送資料包去探測，所以只有將入侵行為的特徵碼歸結為協定的不同欄位的特徵值，通過檢測該特征值來決定入侵行為是否發生。後者主要取決於引擎的組織結構，是否能夠快速地進行規則匹配。

Snort 採用了靈活的外掛程式形式來組織規則庫，即按照入侵行為的種類劃分為相應的外掛程式，
使用者可以根據需要選取對應的外掛程式進行檢測。目前包括的外掛程式分別如下：
每一類外掛程式中包括了數十條的檢測規則，分別代表同一類型的不同入侵行為。對於規則的定義，snort 使用了一種簡單的，羽量級的規則描述語言，上述已經提及到檢測的最終行為就是檢測資料包中協定的不同欄位，例如埠號就是重要的入侵線索。為了更為清楚地闡述這個問題，我們舉一例說明：

攻擊名稱--NT IIS Showcode ASP
攻擊種類--獲取非法存取權限。
攻擊描述--通過構造特定的 URL 請求可以非法閱讀伺服器上的其他檔：
http://attackhost/msadc/Samples/SELECTOR/showcode.asp?source=/msadc/Samples/../../../../../boot.ini
入侵特徵碼 --IP 位址：保護網段以外的 IP 地址。
--協議類型：TCP
--埠：80
--TCP 標誌位元：PUS, ACK
--資料段內容：/selector/showcode.asp
CVE ID--CAN-1999-0736
Bugtraq ID --167

分析以上的這個入侵實例，我們可以看出其實檢測該入侵行為的關鍵是判斷埠號和數據段內容，IP 位址、協定類型和 TCP 標誌位元只是輔助的特徵碼。但是當開始分析原始資料包時，是否應該就直接匹配埠和資料段的內容？無疑針對該入侵行為上述做法的匹配效率是最高的。但是實際上這樣做會降低整體檢測的效率，因為入侵系統要對龐大的網絡資料逐一進行檢測，應該遵循先檢測所有入侵行為的共同特徵其次才是個體特徵的原則，例如如果首先檢測 IP 地址，一旦發現並不屬於檢測範圍之內，就立即檢測下一個數據包而並非繼續檢測該包的其他欄位。這樣既保證了檢測的快速性，又提高了報警的實時性。

Snort 正是按照上述原則定義規則的，將檢測規則劃分成兩個部分：規則頭和規則選項。前者是所有規則共有的包括 IP 位址、協議類型、埠號，後者根據不同規則包括相應的

欄位關鍵字，例如 TCP 的標誌位元或者視窗大小等。檢測規則除了包括上述的關於“要檢測什麼”還應該定義“檢測到了該做什麼”，snort 定義了三種處理方式——alert(發送報警資訊),log（記錄該資料包）和 pass（忽略該資料包）並定義為規則的第一個匹配關鍵字，這樣設計的目的非常簡單，旨在在程式中組織整個的規則庫，即將所有的規則按照處理方式組織成三個鏈表以用於更快速準確地進行匹配，體現了設計者的巧妙之處。
下面我們來舉一實例來具體說明規則的定義：
alert tcp !$HOME_NET any -> $HOME_NET 80 (msg:"CAN-1999-0736 - IIS-showcode"
;flagsA; content:"/selector/showcode.asp"; nocase

該實例正式針對表（N）中所示的入侵行為所定義的檢測規則，通過該例可以看出 snort 的規則語言簡明實用，基本格式為：
規則動作 協定類型 IP 位址 埠號 -> 協議類型 IP 位址 埠號 （規則選項）
源發送方 目的接受方

關於規則的具體書寫規範不再贅述，下面就其關鍵和特別之處加以說明：

1.-- 變數和操作符
snort 作為一個 NIDS，主要的目的就是能夠保護本網段即及時發現外部網對內部網的攻擊，所以規則中的 IP 位址的定義主要是針對外部網和內部網位址兩種。由此 snort 引入了變量的機制，即可以在規則中用變數表示 IP 位址欄位，使用者在運行前可根據實際的子網地址來定義該變數，這樣在解析檢測規則時 snort 會自動替換變數值，增加了規則的靈活性，不過只適應於像 IP 位址這種基本所有規則都具有同一值。
為了更為準確地表達規則和精確地表示檢測範圍，snort 還定義了三類操作符：
l--否定操作符——“ ！”
用於表示 snort 還增加了否定符“！”來區分內部網和外部網。例如例 n 的!$HOME_NET。

l--方向操作符——“->”和”<>”
用於表示傳輸的方向，分別表示單向和雙向傳輸。
l--埠描述符——“ ：”
用於表示埠的範圍。例如： “600:“表示大於 600 的埠號。

2. 規則選項
規則選項作為檢測時的重要標準組成了 snort 入侵偵測引擎的核心，既易用又非常靈活強大。首先其靈活性是指可以根據不同的不同行為制定相應的檢測選項內容，其次其強大性是指不僅檢測具有一定的廣度和深度並且定義了檢測到時該做什麼 。snort 中有 15 個規則選項關鍵字，其中有三個關鍵字是做為檢測到後的回應：
msg - 在報警和封包日誌中列印一個消息
logto - 把包記錄到使用者指定的檔中而不是記錄到標準輸出
resp - 主動反應（切斷連接等）
Resp 關鍵字可以對匹配一條 Snort 規則的流量進行靈活的反應（flexible reponse -FlexResp）。FlexResp 代碼允許 Snort 主動地關閉惡意的連接。該模組合法的參數如下：
rst_snd - 向發送方發送 TCP-RST 資料包

rst_rcv - 向接受方發送 TCP-RST 資料包
rst_all - 向收發雙方發送 TCP_RST 資料包
icmp_net - 向發送方發送 ICMP_NET_UNREACH
icmp_host - 向發送方發送 ICMP_HOST_UNREACH
icmp_port - 向發送方發送 ICMP_PORT_UNREACH
icmp_all - 向發送方發送上述所有的 ICMP 資料包。

作為入侵偵測系統，理論上只需要檢測入侵，並不需要去回應入侵行為的。所以該功能應該是作為 SNORT 的附加功能，但是值得一提的是，發送 RST 和 ICMP UNREACH 資料包向攻擊方可以暫緩其對目標主機的攻擊，我們所研究的一個工具叫做 dsniff 中的 tcpkill 就是利用這個原理進行切斷非法連接，但是對於一般的拒絕服務攻擊，該方法的作用就不甚明顯了。對於 SNORT 來說，實現該功能必然會降低檢測的的效率尤其是在網路流量特別大的時候。

另外 12 中關鍵字都是針對協議中的不同欄位設置的：
關鍵字--檢測內容--主要針對的攻擊行為
ttl--檢測 ip 頭的 ttl 的值 --用於對 traceroute 探測的檢測
id--檢測 ip 頭的分片 id 值--駭客的固定攻擊，例如設置為 31337
dsize--檢測包的淨荷尺寸的值--緩衝區溢位攻擊。
content--在包的淨荷中搜索指定的樣式--最為重要的一個選項，用於在資料包的資料段中搜索指定的內容並根據資料觸發回應，可以搜索包含混合的文本和二進位資料。並設置了三個輔助關鍵字：offset,dsize,nocase
Flags--檢測 tcp flags 的值--非法埠掃描或者其他非法探測主機作業系統類型等。
Seq--檢測 tcp 順序號的值--檢測主機發送的序號集是否是固定的集合。入侵者可以利用該值冒充合法用戶向被入侵者發送資料，偽裝正常的通信以竊取資訊或者其他非法活動。
Ack--檢測 tcp 應答（acknowledgement）的值--Nmap 的 TCP PING 會設置該項的值為 0,從而判斷可能正在用 Nmap 進行非法掃描。
Itype--檢測 icmp type 的值--拒絕服務攻擊。注：只作為其中的一種特徵。
Icode--檢測 icmp code 的值--可疑的流量。
Session--記錄指定會話的應用層資訊的內容--記錄在 TCP 會話中的會話資料。
Icmp_id--檢測 ICMP ECHO ID 的值--
Icmp_seq--檢測 ICMP ECHO 順序號的值--
Ipoption--監視 IP option 的特定代碼--
Rpc--監視特定應用/進程調用的 RPC 服務--檢測非法的 RPC 請求，查看 RPC 請求，並自動將應用（Application），過程（procedure）和程式版本（program version）解碼，如果所有三個值都匹配的話，該規則就顯示成功。

3.預處理程式
預處理程式從 Snort 版本 1.5 開始引入，其代碼在檢測引擎被調用之前先被運行，為檢測做鋪墊，從而提高檢測的準確性和速度。而且預處理機制採用外掛程式形式，用戶和程式師能夠將模組化的外掛程式方便地融入 Snort 之中。目前 snort 現有的預處理程式模組有以下三種：

l--Minfrag

Minfrag 預處理程式檢查給定尺寸限制的分片資料包。資料包被分片通常是由源和目的主機之間的路由器引起的。一般說來，商業網絡設備不會產生小於 512 位元組的分片包。可以利用這個事實，來監控含有小分片的流量。

l--HTTP Decode

HTTP Decode 用於處理 HTTP URI 字串，將串中的資料轉化為可讀的 ASCII 字串，用於檢測 HTTP 的資料資訊對付隱蔽的 WebURL 掃描器和惡意的入侵者。

l--Portscan Detector

Snort Portscan 預處理程式的用處：

向標準記錄設備中記錄從一個源 IP 位址來的埠掃描的開始和結束。

如果指定了一個記錄檔，在記錄掃描類型的同時也記錄目的 IP 位址和埠。埠掃描定義為在時間 T（秒）之內向超過 P 個埠進行 TCP 連接嘗試，或者在時間 T（秒）之內向超過 P 個埠發送 UDP 資料包。埠掃描可以是對任一 IP 位址的多個埠，也可以是對多個 IP 位址的同一埠進行。現在這個版本可以處理一對一和一對多方式的埠掃描，下一個完全版本將可以處理分散式的埠掃描（多對一或多對多）。埠掃描也包括單一的秘密掃描（stealthscan）資料包，比如 NULL，FIN，SYNFIN，XMAS 等。如果包括秘密掃描的話，埠掃描模組會對每一個掃描資料包告警。

network to monitor - 監視埠掃描的目標網路以 network/CIDR 表示

number of ports - 在探測期間訪問的埠數目

detection period - 以秒計數的埠存取時間限制

logdir/filename - 告警資訊存放的目錄/檔案名，告警也可以寫入標準的告警檔中。

l--日誌及報警子系統（logging/alerting subsystem）

入侵偵測系統的輸出結果系統的必要特徵是即時性和多樣性，前者指能夠在檢測到入侵行為的同時及時記錄和報警，後者是指能夠根據需求選擇多種方式進行記錄和報警。一個好的 NIDS，更應該提供友好的輸出介面或發聲報警等等。

Snort 是一個羽量級的 NIDS，它的另外一個重要功能就是資料包記錄器，所以該子系統主要提供了方式：

1.--fast model :採取 TCPDUMP 的格式記錄資訊

2. readable model :按照協定格式記錄，易於使用者查看。

3.alert to syslog: 向 syslog 發送報警資訊。

4.alert to text file :以明文形式記錄報警資訊。

值得提出的是，snort 考慮到使用者需要高性能的時候，即網路資料流程量非常大，可以將數據包資訊進行壓縮從而實習快速的報警。

3.-- 程式結構

1)--snort 的整體結構

Snort 共有 64 個 c 檔和 h 檔，首先介紹

程式的整體結構，其流程圖如下：

其中最為關鍵的函數就是 ProcessPacket(),--其流程圖如下：

2)--資料結構--

snort 的主要資料結構就是幾個鏈表，上述已經提及，snort 組織規則庫的巧妙之處就是按照規則的處理動作來劃分成三個鏈表，其中每個鏈表又按照協議類型：TCP,IP 和 ICMP

分成三個鏈表，所以所有的規則都會被分配到這個三個鏈表中。鏈表中的成員就是描述每條規則的結構——RuleTreeNode，該結構中的一個重要成員就是記錄該規則的處理函數鏈表——RuleFpList，一條規則有時候需要調用多個處理函數來進行分析。該結構中的另外一個重要成員就是規則選項的結構，該結構同樣包括該規則的選項資訊以及其處理函數鏈表。
值得提出的是，並不是每一條規則都分配一個 RuleTreeNode 結構，因為很多規則的選項前的頭部分是相同的，只需要根據不同的規則選項鍊取不同的選項函數處理鏈表。基本整體的結構如圖 n 所示，所有鏈表的初始化都是在捕獲資料包前進行的。
除以上鏈表外，snort 還定義了預處理、輸出的關鍵字和處理函數鏈表，設計鏈表的主要意圖是為了實現外掛程式的思想，即使用者何以根據需求添加刪除預處理的功能模組。其只要
的資料結構如下：

```
typedef struct _PreprocessKeywordNode
{
char *keyword;
void (*func)(char *);
} PreprocessKeywordNode;
//  預處理關鍵字資訊結構。
typedef struct _PreprocessKeywordList
{
PreprocessKeywordNode entry;
struct _PreprocessKeywordList *next;
} PreprocessKeywordList;
//預處理關鍵字鏈表。
typedef struct _PreprocessFuncNode
{
void (*func)(Packet *);
struct _PreprocessFuncNode *next;
} PreprocessFuncNode;
//預處理函數鏈表。
```

所有鏈表的初始化都是在捕獲資料包前進行初始化的，一旦鏈表都已建立完畢，開始捕獲資料包，每收到一個資料包都會現首先調用預處理程式鏈表中的函數進行處理後，其次按照預設地順序遍歷 AlertList,PassList 和 LogList 三個鏈表。遍歷時首先根據資料包的協定類型定位規則鏈表，其次調用遞迴函數進行規則的逐一匹配，即首先匹配規則頭，若匹配則繼續遞迴匹配規則選項，若不匹配，直接匹配下一條規則。為了加快遍歷的速度，snort 在規則選項中的”content”內容匹配時調用了 Boyer-Moore 演算法。

如何微調 IDS 的設定 (Network IDS Tuning Methodology (White Paper from Cisco))

Tuning sensors is critical to a successful network IDS implementation.Without tuning, IDS sensors generate alerts in response to all traffic matching an established criteria. The number of false

alarms can easily overwhelm security personnel and reduce the value of the information the IDS provides. IDS sensors that are not tuned may also raise an alarm on an attack that does not impact the network resource being protected. Given enough false alarms, the security personnel will tend to ignore the IDS sensor—increasing the odds that a real attack will be successful—or will disable it, which leaves the network segment unprotected. The following are guidelines for tuning network IDS sensors.

Step 1. Identify Potential Locations for Sensors

To properly tune IDS sensors, the first step is to identify network locations where the sensors can be placed for maximum efficiency. In the public Internet segment, placing an IDS sensor in a location without traffic filtering can overwhelm the sensor's capabilities. Place IDS sensors behind a traffic filtering device such as a firewall, or in the case of a DMZ, a router with ACLs. If traffic is filtered upstream of the IDS, only traffic destined for the network resources being protected will reach the IDS, which reduces the sensor's workload.

Step 2. Apply an Initial Configuration

The objective of step 2 is to take a first pass at configuring the network IDS sensors. First, sensors are classified and grouped according to active signatures and are then configured by group with a common signature profile. The sensors in a group are managed collectively, which simplifies the management of large groups of sensors. A decision must be made to either deploy signature profiles using the default values or to tune specific signatures. Refer to Appendix A, "Intrusion Detection Primer," for a discussion of signature types and how they are implemented on different sensors throughout the network. A general guideline, if this is the initial deployment of network IDS, is to use the default settings at this step and then to tune them in one of the later steps. The destination for sensor alarms should also be determined during this step

Step 3. Monitor the Sensor While Tuning

The objective of steps 3 and 4 is to monitor IDS sensor alarms and tune out any alarms caused by normal background traffic rather than malicious activity. As steps 3 and 4 are executed, there should be a decrease in the number of false alarms. The monitoring period can last from several days to a week or more. If this is an initial network IDS implementation with a large number of sensors, the number of alarms may be quite large.

Step 4. Analyze Alarms, Tune Out False Positives, and Implement Signature Tuning (If Needed)

During the initial tuning period, you will need to determine the cause of every alarm in order to identify false positives. This task could be tedious, but it is necessary for your network IDS deployment to be of any use in detecting malicious activity.

How do you determine if an alarm is a false positive? A few suggestions follow:

1) Consult the IDS Network Security Database (NSDB) to determine whether the alarm in question is typically due to malicious activity or normal network background activity.

2) If the alarm appears to be the result of normal activity, determine the source of the alarm.

3) If the source of the alarm is a server, work with the appropriate applications and operating system support groups to determine if they think that this is normal behavior, malicious activity, or perhaps an unknown process or application running on the server. If possible, duplicate conditions in a known secure lab with a known, uncompromised server. The server may have to be built from scratch to guarantee this. It is critical that a list of all systems, their operating systems, and the applications running within the monitored environment be available in order to refine the IDS tuning as well as to determine the applicability of any patches released by vendors.

4) If the source is a network device (router, switch, etc.), follow step 3 procedures, but consult the appropriate network operations staff.

5) Deploy a threat analysis system in order to help validate IDS alarms as well as evaluate their significance, impact, and possible responses.

Once a false positive is identified, determine first if the activity that caused the alert can be modified so that an alarm is not generated. For instance, if NetBIOS is running on a server and is unnecessary on that server, disable NetBIOS before tuning the network IDS sensors to ignore alerts caused by the traffic. If the service or application generating the alarm is required, however, or cannot be disabled, an alternative is to configure the network IDS sensors to ignore the alarm. The following configurations are possible:

- Do not generate an alarm if this particular signature is seen for all sources
- Do not generate an alarm if this particular signature is seen from this particular address

It is always better to be specific when implementing security. Therefore, it is recommended that the network IDS sensor be tuned such that it does not generate an alarm for the specific signature from the specific address. However, if additional servers that also require the application or service are later implemented on the same network segment, it may be necessary to go back and tune these sources out as well. The administrative overhead is reduced if the server IP addresses can be grouped together into a single profile.

As mentioned in step 2, IDS sensors can be grouped and collectively managed according to their alarm profiles. This tuning feature reduces the administrative impact when sensors are added to the network.

Similarly, IDS signatures can be grouped into four broad categories:

- Exploit signatures—These signatures indicate attempts to compromise network systems through buffer overflows, Structured Query Language (SQL) injection, brute force password attacks, and other well-known exploits.
- Connection signatures—These signatures indicate reconnaissance activity: an attacker is enumerating systems and services on the network.
- String-match signatures—These signatures indicate corporate policy violations detected by sensors searching for custom text strings in the network traffic. For example, an IDS sensor could signal an alarm on any connection that transmits the phrase “Company Confidential” by e-mail or FTP.
- Denial-of-service (DoS) signatures—These signatures indicate attempts by attack tools such as Trinoo, tribal flood network (TFN), Stacheldracht, and TCP SYN floods to consume bandwidth or computing resources to disrupt normal operations.

These signatures are discussed in detail in Appendix A. If the default signature tuning templates were deployed in step 2, then specific tuning should now be attempted. Implement the specific tuning in stages, rather than all at once. If the sensors are grouped into four profiles, for example, tune each profile separately. Tune each category of signature separately as well. If a group of new false positives appears after pushing a new configuration to the sensors, this tuning method helps to identify the specific tuning responsible. The potential confusion that tuning all of the sensors at once would create must be weighed against the time that it takes to tune the sensors in stages.

Step 5. Selectively Implement Response Actions

Once the false positives are tuned out and logging due to IDS tuning changes is sufficiently reduced, response actions such as TCP resets, shunning, and IP logging can be implemented. Please see the section titled “Response Actions” for usage guidelines. Again, deploy the response actions in stages, rather than all at once.

Step 6. Update Sensors with New Signatures

Automatic signature updates should be implemented for deployments with large numbers of sensors. See the subsection titled “Configuration Management” under “Scaling Network IDS” for a discussion of the benefits of implementing these updates. After a signature upgrade, repeat steps 3 through 5 to tune each new signature appropriately. The time required to monitor the deployment for false positives should be reduced to just a day or two, because the number of new signatures introduced with each signature pack is small. This is another reason to keep the signature levels on the sensors up to date: the greater the difference in signature pack levels, the larger the number of new signatures introduced and the greater the length of time required to monitor and tune out false positives.

FAQ 問：有一大堆某種型態的警報(alert)產生出來了，我該怎辦？可以從哪裡發現更多相關資訊？

答：
有一些規則會比其他規則容易產生正常而誤認的警報(false positive)。這通常會隨著網路不同而變動。你首先要確定這是正常而誤認的警報。有些規則會參用 ID 號碼。下列是一些常見的識別系統，去那兒找關於某個警報的更多訊息。

系統 範例 URL 位元址

IDS IDS182 http://www.whitehats.com/IDS/182
CVE CVE-2000-0138 http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0138
Bugtraq BugtraqID 1 http://www.securityfocus.com/vdb/bottom.html?vid=1
McAfee Mcafee 10225 http://vil.nai.com/vil/dispVirus.asp?virus_k=10225

Log snort message to MySQL

Go down to the output section and uncomment the following line. Change it to be like the following except the password. Remember what you make it because you will need it later when you set up the snort user in mysql.

```
output database: log, mysql, user=snort password=snort dbname=snort host=localhost
```

Setting up the database in MySQL:

I will put a line with a > in front of it so you will see what the output should be. (Note: In MySQL, a semi-colon ” ; “character is mandatory at the end of each input line)
(‘password’ is whatever password you want to give it, just remember what you assign. For the snort user use what you put in the output section of the snort.conf in the section above)

```
# mysql

mysql>  SET PASSWORD FOR root@localhost=PASSWORD('password');
>Query OK, 0 rows affected (0.25 sec)
mysql>  create database snort;
>Query OK, 1 row affected (0.01 sec)
mysql>  grant INSERT,SELECT on root.* to snort@localhost;
>Query OK, 0 rows affected (0.02 sec)
mysql>  SET PASSWORD FOR snort@localhost=PASSWORD('password_from_snort.conf');
>Query OK, 0 rows affected (0.25 sec)
mysql>  grant CREATE, INSERT, SELECT, DELETE, UPDATE on snort.* to snort@localhost;
>Query OK, 0 rows affected (0.02 sec)
mysql>  grant CREATE, INSERT, SELECT, DELETE, UPDATE on snort.* to snort;
>Query OK, 0 rows affected (0.02 sec)
mysql>  exit
>Bye
```

Execute the following commands to create the tables

```
mysql -u root -p  <  /usr/share/doc/snort-2.2.0/contrib/create_mysql snort
Enter password: the mysql root password
```

Then install the extra DB tables using the following command

```
zcat /usr/share/doc/snort-2.2.0/contrib/snortdb-extra.gz |mysql -p snort
Enter password: the mysql root password
```

Now you need to check and make sure that the Snort DB was created correctly

```
# mysql -p

>Enter password:
mysql>  SHOW DATABASES;
+------------+
| Database
+------------+
| mysql
| Snort
| test
+------------+
3 rows in set (0.00 sec)
mysql>  use Snort
>Database changed
mysql> SHOW TABLES;
+------------------+
| Tables_in_Snort
+------------------+
| data
| detail
| encoding
| event
```

```
| flags
| icmphdr
| iphdr
| opt
| protocols
| reference
| reference_system
| schema
| sensor
| services
| sig_class
| sig_reference
| signature
| tcphdr
| udphdr
+------------------+
19 rows in set (0.00 sec)
```