Experiment 7

Aim

To study and implement an IDS (Intrusion Detection System) using an open source tool (SNORT).

Theory

Intrusion Detection System

An Intrusion Detection System (IDS) is a device, typically another separate computer, that monitors activity to identify malicious or suspicious events.

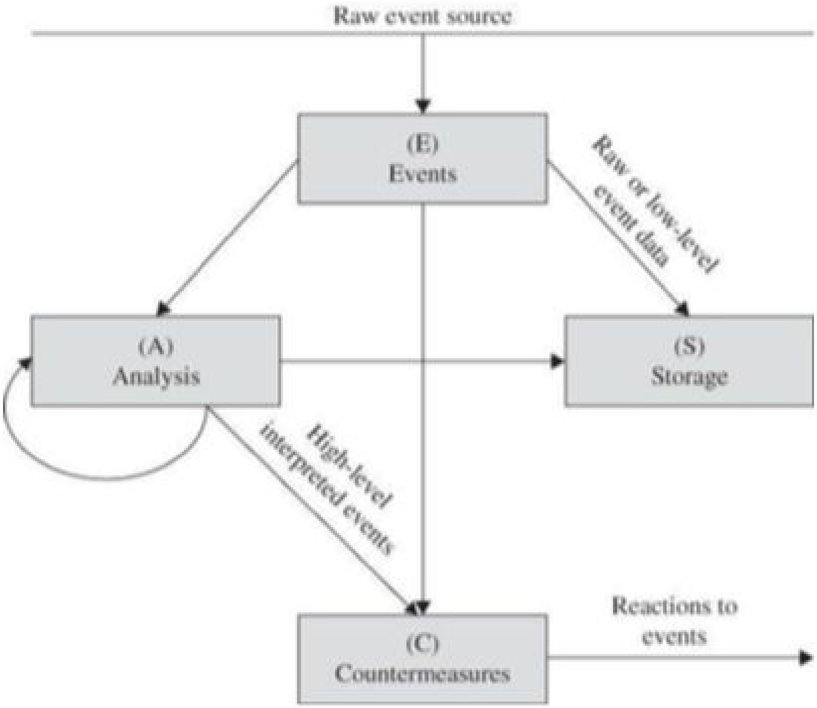


FIGURE 6-64 Model of an Intrusion Detection System

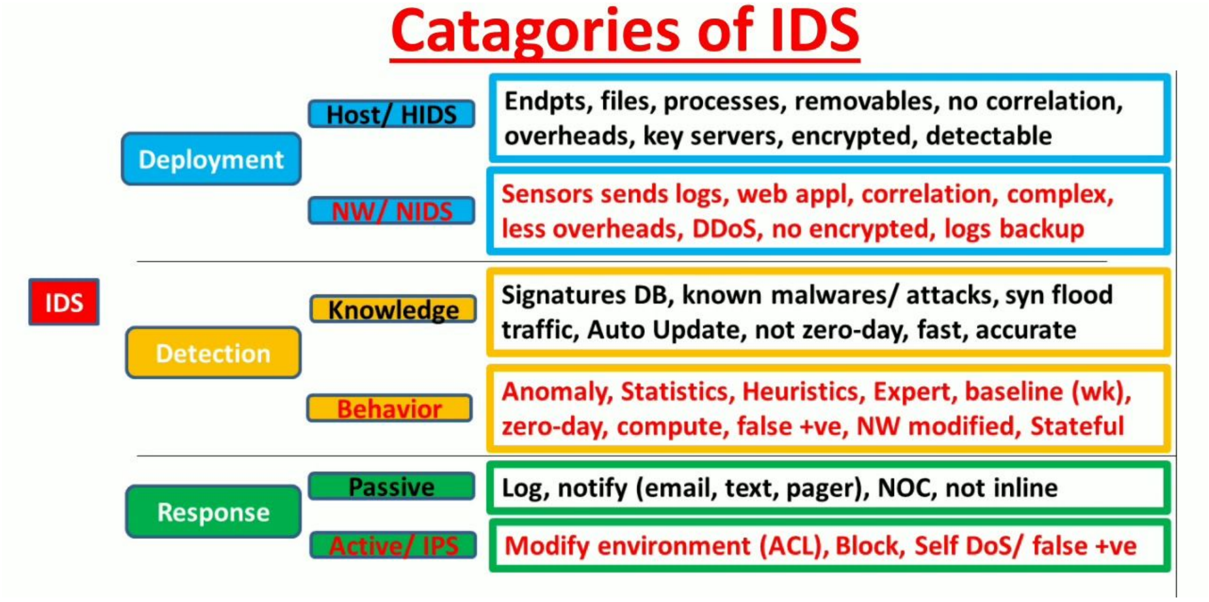
The components in the figure above are the four basic elements of an intrusion detection system, based on the Common Intrusion Detection Framework. An IDS receives raw inputs from sensors. It saves those inputs, analyzes them, and takes some controlling action.

IDSs perform a variety of functions:

- Monitoring users and system activity
- Auditing system configuration for vulnerabilities and misconfigurations

- Assessing the integrity of critical system and data files
- Recognizing known attack patterns in system activity
- Identifying abnormal activity through statistical analysis
- Managing audit trails and highlighting user violation of policy or normal activity
- Correcting system configuration errors
- Installing and operating traps to record information about intruders

Categories of IDS



Catagories of IDS

IDS

Deployment

Host/ HIDS

Endpts, files, processes, removables, no correlation, overheads, key servers, encrypted, detectable

NW/ NIDS

Sensors sends logs, web appl, correlation, complex, less overheads, DDoS, no encrypted, logs backup

Detection

Knowledge

Signatures DB, known malwares/ attacks, syn flood traffic, Auto Update, not zero-day, fast, accurate

Behavior

Anomaly, Statistics, Heuristics, Expert, baseline (wk), zero-day, compute, false +ve, NW modified, Stateful

Response

Passive

Log, notify (email, text, pager), NOC, not inline

Active/ IPS

Modify environment (ACL), Block, Self DoS/ false +ve

SNORT

Snort is an open source network intrusion prevention system, capable of performing real-time traffic analysis and packet logging on IP networks. It can perform protocol analysis, content searching/matching, and can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts, and much more.

Snort has three primary uses:

1. A straight packet sniffer like tcpdump
2. A packet logger (useful for network traffic debugging, etc)
3. A full blown network intrusion prevention system.

Snort Rules

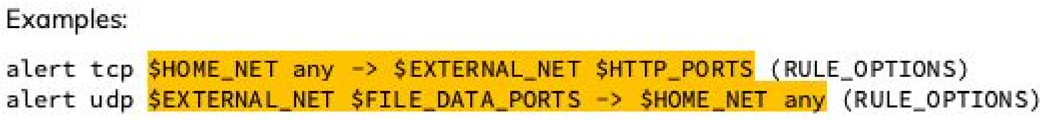
Rules are a different methodology for performing detection, which bring the advantage of 0-day detection to the table. Unlike signatures, rules are based on detecting the actual vulnerability, not an exploit or a unique piece of data. Developing a rule requires an acute understanding of how the vulnerability actually works.

Community rules refer to all rules that have been submitted by members of the open source community or Snort Integrators. These rules are freely available to all Snort users and are governed by the GPLv2.

Rule Headers

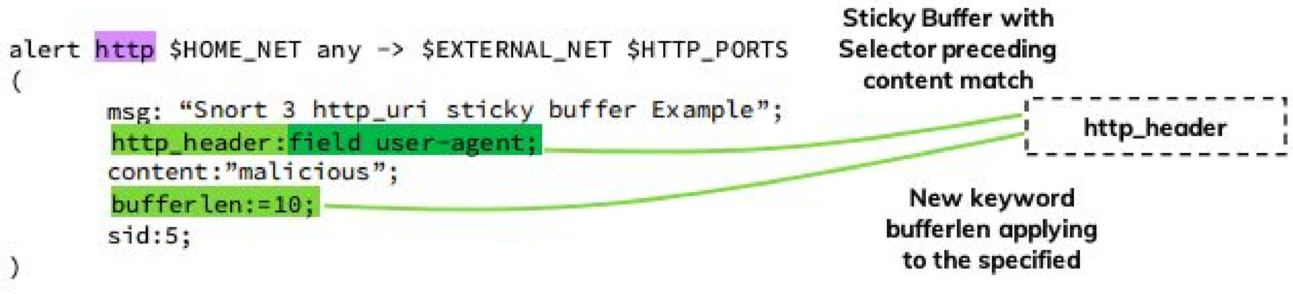
The rule header follows a specific format:

Action Protocol Networks Ports Direction Operator Networks Ports



Examples:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (RULE_OPTIONS)
alert udp $EXTERNAL_NET $FILE_DATA_PORTS -> $HOME_NET any (RULE_OPTIONS)
```



```
alert http $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS
(
        msg: "Snort 3 http_uri sticky buffer Example";
        http_header:field user-agent;
        content:"malicious";
        bufferlen:=10;
        sid:5;
)
```

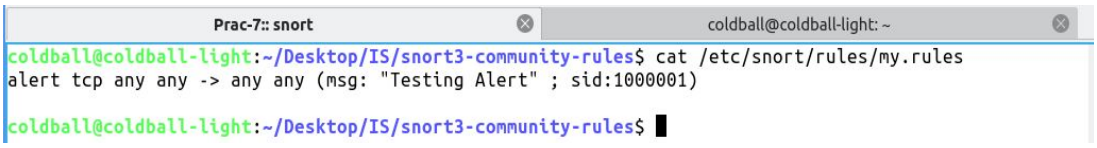
Sticky Buffer with Selector preceding content match

http_header

New keyword bufferlen applying to the specified

Output

my.rules



```
coldball@coldball-light:~/Desktop/IS/snort3-community-rules$ cat /etc/snort/rules/my.rules
alert tcp any any -> any any (msg: "Testing Alert" ; sid:1000001)

coldball@coldball-light:~/Desktop/IS/snort3-community-rules$
```

snort.conf

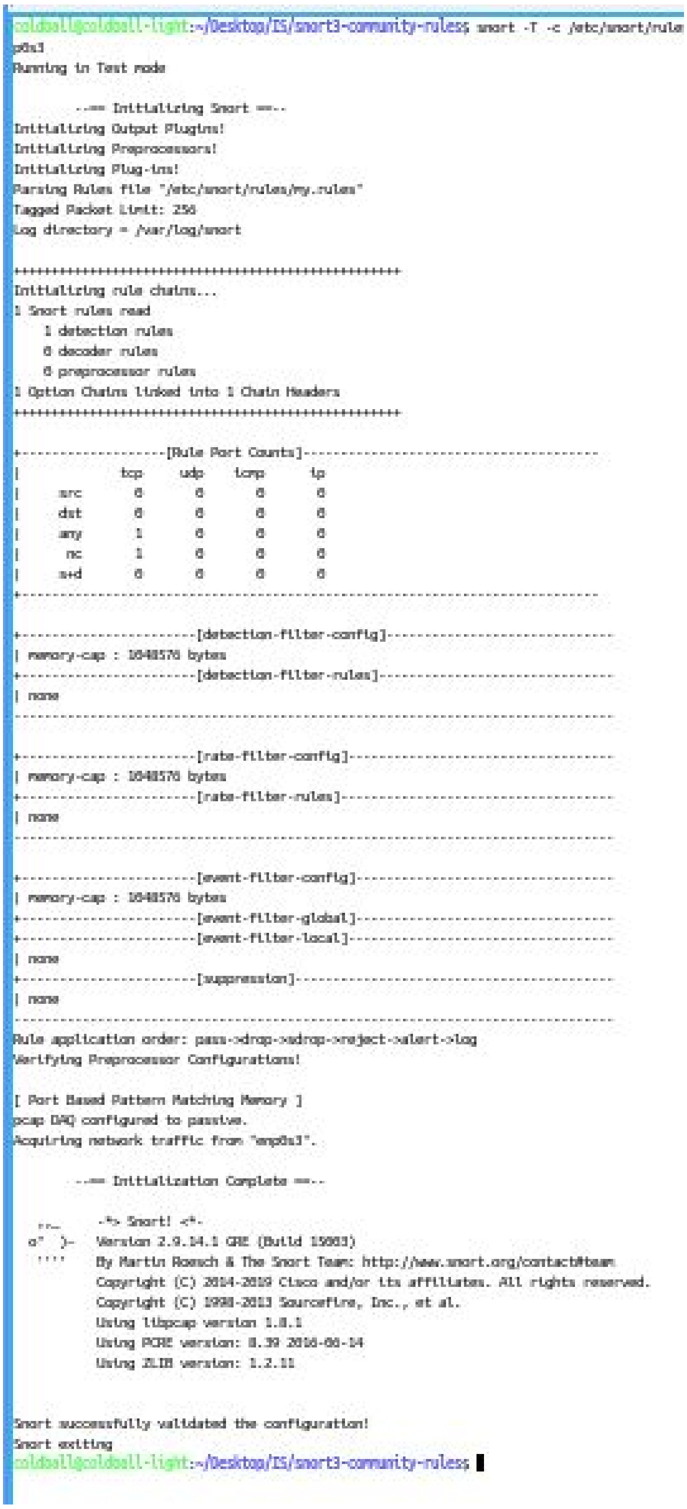
```
# Setup the network addresses you are protecting
ipvar HOME_NET 10.0.2.0/24

# Set up the external network addresses. Leave as "any" in most situations
ipvar EXTERNAL_NET any

# Path to your rules files (this can be a relative path)
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\rules
var RULE_PATH /etc/snort/rules
var SO_RULE_PATH /etc/snort/so_rules
var PREPROC_RULE_PATH /etc/snort/preproc_rules

# If you are using reputation preprocessor set these
# Currently there is a bug with relative paths, they are relative to where snort is
# not relative to snort.conf like the above variables
# This is completely inconsistent with how other vars work, BUG 89986
# Set the absolute path appropriately
var WHITE_LIST_PATH /etc/snort/rules/iplists
var BLACK_LIST_PATH /etc/snort/rules/iplists
```

Testing SNORT rules file my.rules



```
coldball@coldball-light:~/Desktop/IS/snort3-community-rules$ snort -T -c /etc/snort/rule
p0s3
Running in Test mode

        --== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file "/etc/snort/rules/my.rules"
Tagged Packet Limit: 256
Log directory = /var/log/snort

+++++++++++++++++++++++++++++++++++++++++++++++++++
Initializing rule chains...
1 Snort rules read
    1 detection rules
    0 decoder rules
    0 preprocessor rules
1 Option Chains linked into 1 Chain Headers
+++++++++++++++++++++++++++++++++++++++++++++++++++

+-------------------[Rule Port Counts]---------------------------------------
|             tcp     udp    icmp      ip
|     src       0       0       0       0
|     dst       0       0       0       0
|     any       1       0       0       0
|      nc       1       0       0       0
|     s+d       0       0       0       0
+----------------------------------------------------------------------------

+-----------------------[detection-filter-config]------------------------------
| memory-cap : 1048576 bytes
+-----------------------[detection-filter-rules]-------------------------------
| none
-------------------------------------------------------------------------------

+-----------------------[rate-filter-config]-----------------------------------
| memory-cap : 1048576 bytes
+-----------------------[rate-filter-rules]------------------------------------
| none
-------------------------------------------------------------------------------

+-----------------------[event-filter-config]----------------------------------
| memory-cap : 1048576 bytes
+-----------------------[event-filter-global]----------------------------------
+-----------------------[event-filter-local]-----------------------------------
| none
+-----------------------[suppression]------------------------------------------
| none
-------------------------------------------------------------------------------
Rule application order: pass->drop->sdrop->reject->alert->log
Verifying Preprocessor Configurations!

[ Port Based Pattern Matching Memory ]
pcap DAQ configured to passive.
Acquiring network traffic from "enp0s3".

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.14.1 GRE (Build 15003)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2019 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.8.1
           Using PCRE version: 8.39 2016-06-14
           Using ZLIB version: 1.2.11


Snort successfully validated the configuration!
Snort exiting
coldball@coldball-light:~/Desktop/IS/snort3-community-rules$ █
```

Conclusion

Thus we deployed SNORT on the VM’s network to listen for a simple set of SNORT rules and alert any matches on the console.