Instructions for PA #3

Jinwoo Kim, *jinwoo.kim@kaist.ac.kr*
Jaehyun Nam, *namjh@kaist.ac.kr*

May 16, 2017

1 Overview

You will implement **a Network Intrusion Detection System (NIDS)** in this assignment. Here are some instructions that you **MUST** follow in your assignment. If you have any question or any unclear thing about the assignment, please send an email to TAs.

We will evaluate your program as below.
1) Run a submitted NIDS program with prepared NIDS rules (i.e. Snort rules)
2) Transmit a set of arbitrary packets to the NIDS program
3) Check whether the NIDS captures and analyzes received packets or not.
4) Check whether the NIDS detects packets that are matched to the rules or not.

2 Input Interface

Your NIDS must have an input interface to get a rule file from a user.

$ ./my_nids rule_file.txt

3 Rule Syntax

3.1 Requirements

We will follow the **Snort rule syntax**[1] to write NIDS rules when evaluating your program. You don't need to parse all of the elements described in Snort documentation. Instead, make sure your NIDS parses below fields.

- IP Header
  - ToS (Differentiated Services Code Point)
  - Length
  - Fragment Offset
  - Protocol
  - Source/Destination IP Address
- TCP Header
  - Source/Destination Port Number
  - Sequence Number
  - Acknowledgement Number

  - TCP Flags
- UDP Header
  - Source/Destination Port Number
- Payload
  - HTTP header
    - GET, PUT, POST, HEAD, and URI
    - e.g., "GET /index.html"
  - Non-HTTP header
    - Normal string patterns
    - e.g., "I am an attacker"

3.2 Rule Format

Basically, we will construct rules as following.

<action> <protocol> <source ip> <source port> -> <destination ip> <destination port> (<option 1>; <option 2>; ...)

3.2.1 Action

There are several defined actions in the snort rules, but we will only use the **alert** action that generate an alert for declared rules.

3.2.2 Protocol

Consider three protocols, which are **tcp, udp, and http** in our assignment. In the case of http, we will consider the packets including HTTP header as http packets.

3.2.3 IP Address

The IP address fields (i.e. <source ip> and <destination ip>) are composed of numeric **IP address and a CIDR block**. The CIDR block represents the netmask that indicates routing prefix. If you specify "/24", it indicates Class C network. You need to consider subnetwork that matches the specified CIDR block. When you specify exact IP address you need to skip the CIDR notation.

Example)

alert tcp 192.168.1.0/24 any -> 192.168.1.0/24 22 (content:"/bin/sh"; msg:"Remote shell execution message!")
alert tcp any any -> 143.248.5.153 80 (msg:"A packet destined to www.kaist.ac.kr")

3.2.4 Port Number

We use two ways for representing port numbers. When we use a colon ":", it indicates **the range operator** to match multiple ports. We can also use **static port numbers** as a comma separated list.

Example)

alert udp any any -> 192.168.1.0/24 1:1024 (msg:"udp traffic from any port and destination ports ranging from 1 to 1024")
alert tcp any any -> 192.168.1.0/24 :6000 (msg:"tcp traffic from any port going to ports less than or equal to 6000")
alert tcp any :1024 -> 192.168.1.0/24 500: (msg:"tcp traffic from privileged ports less than or equal to 1024 going to ports greater than or equal to 500:")
alert tcp any any -> any 80,22,21 (msg: "tcp traffic from any port going to Web, FTP, and SSH server ports")

3.2.5 Rule Options

Now we consider rule options that match our required fields.

- msg: *"<message text>"*;
  - Represent the string when a packet is captured. It is a necessary option field.
- tos: *<number>*;
  - Check the IP header TOS field.
- len: *<number>*;
  - Check the IP header length. You just need to implement it as an exact match.
- offset: *<number>*;
  - Check the IP fragment offset field.
- seq: *<number>*;
  - Check the TCP sequence number field.
- ack: *<number>*;
  - Check the TCP acknowledgement number field.
- flags: *<flag values>*;
  - Tests the TCP flags for a match. You need to implement **5 flags** in this assignment
  - F - FIN
  - S - SYN
  - R - RST
  - P - PSH
  - A - ACK
  - E.g., alert any any -> 192.168.1.0/24 any (flags:SF; msg:"Possible SYN FIN scan";)
- http_request: *"<request line>"*;
  - Check the HTTP request line as an exact match. (i.e., GET, PUT, POST, HEAD, and URI)
  - Example)

  - alert http any any -> any any (http_request:"GET"; content:"daum"; msg:"DAUM!";)
  - alert http any any -> any 80 (http_request:"PUT"; content:"naver"; msg:"NAVER!";
)
  - alert http any 80 -> any any (http_request:"POST"; content:"google"; msg:"GOOGLE!";)

- content: *"<content string>"*;
  - Search specific string patterns in the payload.
  - Example) alert tcp any any -> 143.248.5.153 80 (http_request:"GET"; content:"attack"; msg:"an attack packet against kaist web server";)

• **NOTE 1:** "any" represents "Don't care".
• **NOTE 2:** We will only consider one white space as a delimiter (i.e. you don't need to consider multiple white spaces or tab).

4 Output Format

Your NIDS must display all fields of captured packets including payload regardless of matching. If there is a matched packet, you have to highlight matched fields so that a user can notice which field is matched. We provide an example output as below.

Example)

Rule: alert http any any -> any 50450 (http_request:"GET"; content:"naver"; msg:"NAVER detected!")
==========
[IP header]
Version: 4
Header Length: 20 bytes
ToS: 0x12
Fragment Offset: 0
Source: 143.248.57.246
Destination: 143.248.5.153

[TCP header]
Source Port: 5022
Destination Port: 50450
Sequence Number: 84207331
Acknowledgement Number: 3214731849
Flags: PSH, ACK

[TCP payload]
HTTP Request: GET
Payload: GET / HTTP/1.1 Host: www.**naver**.com Connection: keep-alive Upgrade-Insecure-Requests: 1 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/57.0.2987.133 Safari/537.36 Accept: text/html,application/xhtml+xml,applicati
Accept-Encoding: gzip, deflate, sdch Accept-Language: ko-KR,ko;q=0.8,en-US;q=0.6,en;q=0.4
==========
Message: NAVER detected!

• **IMPORTANT:** If a packet has a matched pattern, **highlight(apply red color) the fields and print the matched rule and the string included in the 'msg' option**.

5 Score

TBU

6 Submission

- You MUST submit your prog. assignments to the KLMS website.
- The website automatically marks the time of the last submission/modification. (The website often becomes unavailable, so please make sure that you submit your assignment early)
- 50% off for any late submissions.
- *Please DO NOT e-mail Prof. Shin or TAs to submit your assignments.*
- You will be submitting two files to the system. (ONE tarball + ONE report)
- E.g.) For prog. Assignment #3 (P3), the name will be "P3_yourIDnumber_yourNAME.tar.gz"
- (1) Create a tarball (tar.gz) with all the source codes, README, and executables.
- Tarball structure: (for prog. assignment 3)
  - Create a folder called "p3"
  - Put your sourcecode in the folder named "p3/src"
  - Put your executable(s) in the folder named "p3/bin"
  - README file goes to the root path - "p3/"
  - install.sh goes to the root path as well - "p3" (optional)
  - Then, compress the entire "p3" folder into a single tarball.
- (2) P1_yourIDnumber_yourNAME.pdf (max. five pages)

7 README

- No README, NO points
- Let us know how to run your program exactly. We won't be spending our time on figuring out how to run your program.

8 Test Environments

- We will evaluate your submitted programs on 14.04.5 LTS (Trusty Tahr) 64-bit platform.
- We will use a machine Intel Xeon Processor E5-2620 with 16GB RAM.

References

[1] `http://paginas.fe.up.pt/~mgi98020/pgr/writing_snort_rules.htm`