Learning Snort Rules by Capturing Intrusions in Live Network Traffic Replay

Jinsheng Xu[1] *Jinghua Zhang*[2] Triveni Gadipalli[1] *Xiaohong Yuan*[1] *Huiming Yu*[1]
[1] *North Carolina A&T State University* [2] *Winston-Salem State University*

Abstract – Rule development for Snort, which is one of the most popular network intrusion detection systems, is a critical skill to detect ever emerging new cyber attacks. This paper describes a Snort lab that helps students to learn Snort rules effectively. For beginners, it is difficult to determine if a rule is correctly written without being able to test them in a realistic setting. The uniqueness of this hands-on learning lab is that it allows students learn how to write Snort rules by testing and debugging their rules against the live network traffic replay. The lab requires students to learn and apply various features of Snort rules to successfully detect the intrusions. The intrusion traffic packets are real captures that were downloaded from various sources on the Internet. This lab could be extended by adding new intrusion traffic that requires students to use a new set of Snort rules. We piloted this Snort lab on a workshop that included nineteen instructors who teach information assurance courses in different universities and colleges. The lab was successfully completed within two hours and most of the participants gave very high reviews for the lab. Several of the instructors showed great interests in using the lab in their classes.

Index terms – Snort; Security; Education; Learning Technologies;

I. INTRODUCTION

Snort is a very popular open source network intrusion prevention and detection system (IDS/IPS) developed by Sourcefire. With millions of downloads and approximately 300,000 registered users, it has become the de facto standard for IDS/IPS [1, 2]. Snort is widely used in academia as a tool for teaching network security concepts [11, 12, 13]. The key component of Snort is the rule which captures intrusions of various types. Although there are official rule sets available from Snort website [1], new Snort rules are needed to counter new emerging cyber attacks. Therefore, it is critical to understand and learn how to write Snort rules. Besides the official Snort documentation, there are other good sources where Snort rules can be learned [4, 10]. The Snort rule syntax tree and rule process are analyzed in [5]. Snort rules are known to be challenging to learn and develop, therefore many Graphical User Interfaces have been developed to assist with Snort rule writing [6, 7, 8, 9].

Snort rules are usually learned by reading tutorials and it is difficult to verify the correctness of rules without actually seeing the rules in action capturing intrusions. According to lessons learned in [11] "rules need to be tested, modified, and further tested and modified again to reduce false positives and false negatives". A hands-on lab that requires students to test and debug their rules on real traffic is a critical step in understanding and developing Snort rules, which can help students gain confidence in developing new rules. We developed a Snort lab that allows students to learn rules by capturing intrusions in live network traffic. The goal of the project is to teach students various features of Snort rules and use those features to capture the live intrusion traffic. We downloaded multiple traffic captures from various sources and used them to design six exercises which require students to learn and use different Snort options and features. We combined the traffic captures and wrote a program with *pcap* library to replay the traffic to a Local Area Network where students do their labs. In the next sections, we describe our previous work, how we designed the lab, how it was used, and the evaluation results of the lab.

II. PREVIOUS WORK

Our Snort learning tool was developed based on our previous tool called "*LAN Attack on a Switched Network Simulator*". Source code developed for this tool was reused by the Snort learning lab. Because it is also an instructional tool for network security, we are going to describe it briefly here. The tool was designed to help students understand the methods that attackers use to become Man-In-The-Middle on a switched network. Although, several tools exist that can do Man-In-The-Middle, the technical details are hidden from the users [14, 15]. For example, "Cain & Abel" implements APR (ARP Poison Routing) which enables Man-In-The-Middle attacks to be carried out easily on switched network. However, students cannot learn the details of becoming Man-In-The-Middle which include poisoning the router's ARP table, poisoning the victim's ARP table and forwarding the packets between the router and the victim.

[1] *1601 E. Market St., Greensboro, NC, 27411, Tel: 336-334-7245, jxu@ncat.edu.*
[2] *601 Martin Luther King Jr. Drive, Winston-Salem, NC 27110, zhangji@wssu.edu.*

ISBN 1-933510-96-X/$15.00

Two programs with which students are asked to carry out a successful Man-In-The-Middle attack on a switched network have been developed. The first program, named "*sendarp*", is programmed to send an arbitrary ARP reply message. Students are asked to provide such information as MAC addresses of the source and the target in ARP reply message, IP addresses of the source and the target in ARP reply message, and destination and source MAC addresses in the Ethernet frame header. Only with clear understandings of Ethernet frames, ARP message, and goals of ARP poisoning, can students successfully carry out this attack. This program needs to be executed twice to poison both the router and the victim. To increase the chance of success, the program repeatedly sends the ARP reply message with fixed time interval.

The second program, named "*mim*", forwards the packets between the router and the victim. To become a successful Man-In-The-Middle without being detected by the victim, the attacker must forward the intercepted packet either to the victim or to the router and let the victim continue communicating without interruption. This tool dumps the intercepted traffic into a file in tcpdump format which can later be viewed using Wireshark or other packet analyzers. This program asks students under which condition a packet should be forwarded to the router or the victim. Students need to know the format of the IP packet intercepted by the attacker to correctly forward the packets. To assist the lab, we developed a shell script that runs on the victim and constantly contacts a web server that computes a simple mathematical function on the random number sent by the victim. The students need to intercept enough traffic to successfully guess the function computed by the web server.

III. LAB DESIGN

A. Collecting real world attack traffic

There are several websites on the Internet that provide free network traffic captures. Most of our data are collected from these websites: *www.openpacket.org*, *www.pcapr.net*, and *www.techtraces.com*. Two of the above sites: *www.openpacket.org* and *www.pcapr.net* are *web2.0* sites where users can upload and share their captures. *www.techtraces.com* is a site that provides free resources to VoIP developers. It has some sample captures that are related to VoIP traffic. Only a small portion of the captures on these sites are intrusion related packets. Much time is spent on identifying various intrusion traffic captures that meet our needs of learning various features of Snort rules. We have downloaded eleven traffic captures from multiple sources. These captures need to be replayed on the network for students to capture them by writing correct Snort rules. Instead of replaying them individually, we combined these traffic captures into a single file. In a lab environment where many students work on it at the same time, the captured packets need to be sent to all of the lab computers. The captured traffic packets are Ethernet frames whose destination MAC addresses are not equal to any of those computers in the lab. To be able to send them to all the computers, we wrote a program in C with *pcap* library that reads the packets from the traffic file and modify the destination MAC address to broadcast (*ff:ff:ff:ff:ff:ff*) address and send them out through the Ethernet interface. We looped the traffic data to allow students to have multiple chances of capturing intrusions during the lab time.

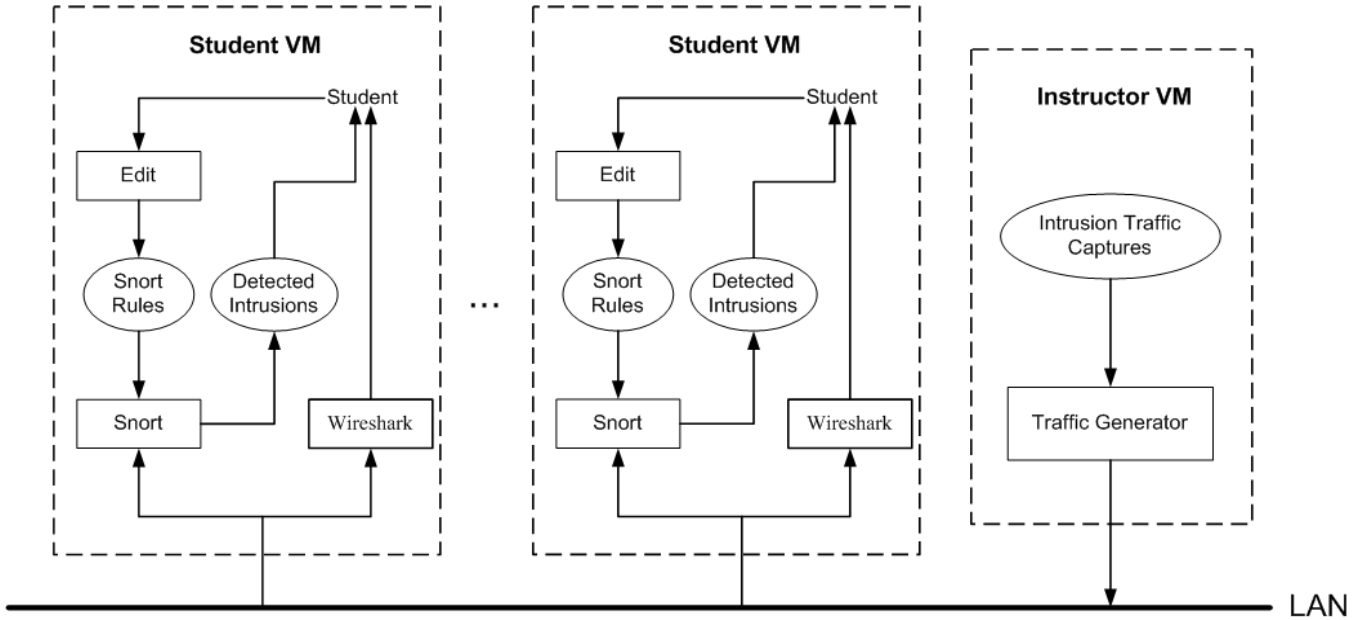

Figure 1. Lab Structure Diagram

B. Lab Environment

ISBN 1-933510-96-X/$15.00

We have created a Windows XP Professional virtual machine that includes everything needed for the lab. We installed Snort, Wireshark, and our own traffic player together with the combined traffic captures. The source code for the traffic generator is also included for students' reference as well as for sharing for possible future improvements. The lab handout, lectures slides and a Snort tutorial are also included in the virtual machine. The advantages of using virtual machine include not only saving time from installing Snort and Wireshark, but also avoiding changes to the configurations of the hosting computers.

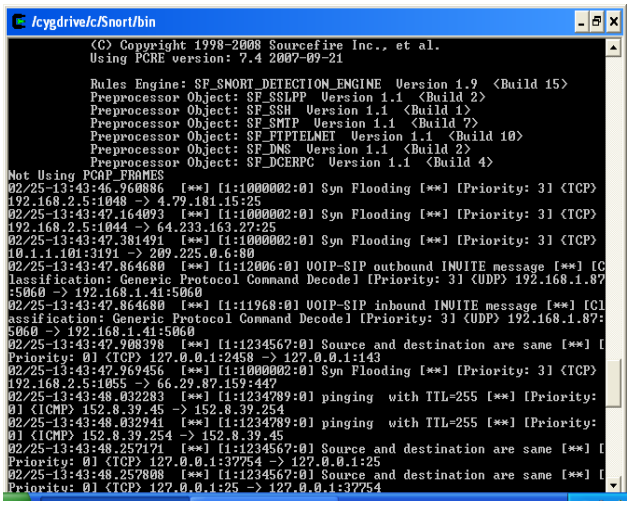

Figure 2. Screenshot of intrusions detected by Snort.

Lab Structure diagram is shown in Figure 1. The instructor's virtual machine will run the traffic generator which reads packets from the traffic capture file and broadcasts the captured intrusion traffic. Multiple copies of student virtual machines could be deployed on a local area network. On the student virtual machine, students examine the captured packets with Wireshark and develop the rules to capture the intrusions. Students can test and debug their rules based on the output of Snort. Figure 2 shows an example of intrusions detected by Snort on a student VM.

C. Lab Description

Students are asked to develop six Snort rules to capture six different attacks. A Snort rule has two logical parts as shown in Figure 3: rule header and rule options. Rule header format is shown in Figure 4. The rule header defines attributes of a packet, as well as what to do in the event that a packet with all the attributes indicated in the rule should show up. The action field of the rule header tells Snort what to do when it finds a packet that matches the rule criteria. Rule options follow the rule header and they are enclosed in closed braces. The rule option section contains alert messages and information on which parts of the packet should be inspected to determine if the rule action should be taken. Figure 5 shows an example of a simple Snort rule. The Snort rule in Figure 5 will generate an alert when an ftp (port number 21) packet from any IP address is sent to 152.54.23.89.

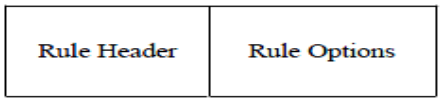

Figure 3. Snort Rule Structure

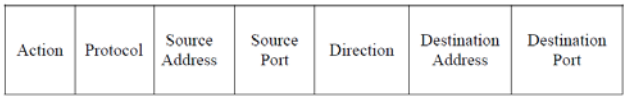

Figure 4. Rule Header Format

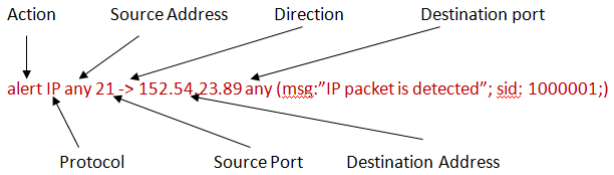

Figure 5. A Snort rule example.

Table 1 shows the learning objectives of six Snort exercises and their corresponding intrusions. To help student develop rules, we provided students with five examples of attacks and their corresponding rules as shown in Table 2 part a). Students could use them as a guide when they write rules for the lab exercises as shown in Table 2 part b).

Table 1. Learning objectives and their Intrusions

Exercise	Objectives	Intrusions
No.1	Learn rule header, basic protocol header fields.	Source and Destination IP are the same.
No.2	Learn how to check content in Payload	"show databases" command in MYSQL
No.3	Learn Regular Expression	Inbound VOIPSIP Invite message
No.4	Learn *depth*, *distance*, and *offset* options	Attempting DNS Zone Transfer
No.5	Learn *flowbits* option	Skype is started and then being used
No.6	Learn *threshold* option	TCP SYN Flooding Attack

For exercise No.1 students need to know the components of the Snort rule header. In addition, students need to learn general rule options and basic Snort rule options for

ISBN 1-933510-96-X/$15.00

protocol header fields (or non-payload detection rule options). For example, students are given documents about *ttl, flags, id, sameip* and other basic non-payload detection rule options. In exercise No. 2, students need to learn to use *content* option which belongs to Snort payload detection rule options. Students will learn how to match binary data when using *content* rule options. For example, students should be able to match binary string: *0f 00 00 00 0*3 that precedes "show databases" in a MySQL query. Regular expression is very important in matching a string pattern instead of an exact string. In exercise No.3, students need to learn *perl* compatible regular expressions to catch variations in inbound VOIP-SIP invite message. In exercise No.4, more payload detection rule options are introduced to students. These rules are to capture DNS Zone Transfer packets. The goal of exercise No.5 is to teach students how to use *flowbits* option which involves multiple TCP packets in a session. Students will learn how to set *flowbits* and how to detect whether *flowbits* is set in subsequent packets. In exercise No.5, students are asked to set *flowbits* when someone starts Skype and generate alert when Skype conversion has begun by detecting that *flowbits* has been set. The goal of exercise No.6 is to teach students how to use threshold options to reduce the number of logged alerts and false alarms. Students will use *threshold* option to capture TCP SYN Flooding attacks by detecting more than 10 Synchronization packets within 1 second.

Table 2 shows each example and exercise pair and the corresponding sample solutions. For each exercise except for No.6, students are given an example and the corresponding rule as the guide as shown in part a) of each exercise. Once students fully understood the Example of each exercise, they were asked to create the Snort rule for the part b) by themselves. Sample solution was presented and explained when most students finished the task. Some of the solutions are adapted from the source where the intrusion captures are downloaded. Except for exercise No.5, all other exercises just need one Snort rule. Exercise No.5 requires two rules because the first rule sets the *flowbits*, and the second rule checks if the *flowbits* has been set. The difficulties of these exercises are not necessarily increasing. However, the knowledge learned in previous exercises is needed to complete the current one. We suggest the exercises to be finished progressively. Students can test the rules for each exercise before moving to the next one. There are other features of Snort rules that are not included in this lab. We designed these exercises to help students learn the basic Snort rules within a two-hour intense learning period in a lab. With knowledge learned in this lab, we have laid foundation for students to continue exploring advanced rule options.

IV. RESULT AND CONCLUSION

The lab was part of a project that was supported by the National Science Foundation (NSF) of the United States. Part of goals of this project titled "Faculty Development Workshop on Cyber Games and Interactive Simulations" includes (a) Develop interactive educational tools and materials, (b) Conduct the faculty development workshop, and (c) Distribute education tools and materials to attendees of the faculty development workshop. The purpose of Snort lab is to teach students not the instructors. However, these instructors will transfer their own learning experience in the workshop to their students. This Snort lab described in this paper was first used in the summer of 2009 at Faculty Workshop held at University of North Carolina at Charlotte. There were nineteen attendees who are faculty members of universities and colleges across the United States. Although all of them teach information assurance related courses in their organizations, they do have various backgrounds in Snort. Many of them have little or no knowledge about Snort. The first thirty minutes were spent on introducing Snort rules and lab environment. Two hours of lab time were given to the attendees to complete the six lab exercises. Two attendees formed a group to work on the lab collaboratively. All of the groups were able to finish the lab on time, but about half the groups needed some help and only two groups needed significant assistance. In post workshop evaluation, we asked attendees if they have learned things are useful to teach information assurance course. Fifteen of them strongly agreed and three agreed. Many of attendees showed great interest in using this Snort lab in their own information assurance classes to teach their students.

There are several lessons learned through this workshop experience. First, the instructor should ask students to read basic Snort documentations before the lab. Especially, students who do not have any background in Snort should spend more time reading the documents. Some of the attendees who did not have Snort background spent significant amount of time just trying to get themselves started. Second, many students had trouble writing rules in basic text editing tools. We should have provided students with visual rule editing tools which can assist students develop rules faster with less syntax errors.

We will make all the materials for this Snort lab available online. The materials include lectures slides, lab descriptions, intrusion traffic captures and the source code for replaying the traffic captures. The lab could be extended by adding new intrusion traffic that teaches students a new set of rules. This lab could also be used in a network security competition where students compete with one another to finish the intrusion detections first.

ISBN 1-933510-96-X/$15.00

Table 2. Examples (a), Exercises (b) and Sample Solutions

Exercise Number	Exercises	Sample Solutions
No. 1	a) How to send a send an alert when someone is pinging with ttl 255?	alert icmp any any -> any any (msg:"pinging with TTL=255";ttl: 255; sid:1234789;)
	b) How can we write a rule to send an alert when source and destination are same?	alert ip any any -> any any (msg: "destination and source are same" ; sameip; sid: 123456;)
No. 2	a) How to send an alert when someone tries to use "show databases" command in MYSQL	alert tcp any any -> any 3306 (msg:"MYSQL show databases attempt"; flow:to_server,established; content:"|0F 00 00 00 03|show databases"; classtype:protocol-command-decode; sid:1776; rev:3;)
	b) How can we write a rule to detect an SNMP connction over UDP using the default "public" is made?	alert udp any any -> any 161 (msg:"SNMP public access udp"; flow:to_server; content:"public";classtype:attempted-recon; sid:1411;)
No. 3	a) How to send an alert when someone invites for VOIP-SIP connection?	alert udp any 5060 -> any any (msg:"VOIP-SIP outbound INVITE message"; content:"INVITE"; depth:6; nocase; content:"SIP/2.0"; distance:0; nocase; pcre:"/^INVITE\s+(sips?|tel|https?)\x3A[\w-'"]+\x40[\w-'"\x2E]+\s+/smi"; classtype:protocol-command-decode; sid:12006;)
	b) How to generate an alert for an inbound VOIP-SIP INVITE message?	alert udp any any -> any 5060 (msg:"VOIP-SIP inbound INVITE message"; content:"INVITE"; depth:6; nocase; content:"SIP/2.0"; distance:0; nocase; pcre:"/^INVITE\s+(sips?|tel|https?)\x3A[\w-'"]+\x40[\w-'"\x2E]+\s+/smi"; classtype:protocol-command-decode; sid:11968;)
No. 4	a) How to send an alert when an attempt is made for DNS Zone transfer ?	alert tcp any any -> any 53 (msg:"DNS zone transfer TCP"; flow:to_server,established; content:"|00 00 FC|"; offset:15; classtype:attempted-recon; sid:255;)
	b) How to generate an alert when activity relating to network chat clients is detected?	alert tcp any any <> any 1863 (msg:"CHAT MSN message"; flow:established; content:"MSG "; depth:4; content:"Content-Type|3A|"; nocase; content:"text/plain"; distance:1; classtype:policy-violation; sid:540;)
No.5	a) How to generate an alert when someone wants to start skype?	alert tcp any any -> any any (msg:"P2P Skype client login startup"; flow:to_server,established; dsize:5; content:"|16 03 01 00|"; depth:4; flowbits:set,skype.login; classtype:policy-violation; sid:5998;)
	b) How to generate an alert when network traffic that indicates Skype is being used?	Rule 1: alert tcp any any -> any any (msg:"P2P Skype client login startup"; flow:to_server,established; dsize:5; content:"|16 03 01 00|"; depth:4; flowbits:set,skype.login; classtype:policyviolation; sid:567890;) Rule 2: alert tcp any any -> any any (msg:"P2P Skype client login"; flow:to_client,established; flowbits:isset,skype.login; dsize:5; content:"|17 03 01 00|"; depth:4; classtype:policy-violation; sid:567891;)
No. 6	How to generate an alert when "syn-flooding" happens?	alert tcp any any -> any any (msg:"Syn Flooding"; flags:S; flow:to_server; threshold: type threshold, track by_src, count 10, seconds 1; priority:3; sid:678901;)

ISBN 1-933510-96-X/$15.00

V. REFERENCES

[1] Snort . Available at http:// www.snort.org/

[2] Martin Roesch, "Snort - Lightweight Intrusion Detection for Networks", Proceedings of the 13th USENIX conference on System administration, November 07-12, 1999, Seattle, Washington.

[3] Hideki Koike, Kazuhiro Ohno, "SnortView: visualization system of snort logs", Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security, October 29-29, 2004, Washington DC, USA.

[4] Li Xiaofang, Yao Yuan, "Master and Use Snort Tools for Intrusion Detection [J]; Computer Applications and Software; 2006-03.

[5] Zhang Yue-lian, Guo Wen-dong, "Analysis of the Rule of Snort and the Module of Snort Rule Process" [J]; Journal of Hebei University of Science and Technology; 2003-04.

[6] Cobalt. http://www.zeffie.com/cobalt_snort.html

[7] Analysis Console for Intrusion Databases. http://www.andrew.cmu.edu/user/rdanyliw/snort/snortacid.html

[8] SnortCenter. http://sourceforge.net/projects/snortcenter2/

[9] Christopher R. Evans, "A Graphical User Interface for Writing Snort 2.0 Rules". Unpublished. http://technology.asu.edu/files/documents/tradeshow/May04/ChristopherEvans.pdf

[10]R. Rehman, "Intrusion Detection with Snort", Upper Saddle River: Prentice Hall, 2003

[11]Jim Chen, Victor Tsao, Barry Williams, Tokunbo Olojo, John Smet, Nicole Regobert, Lamin Kamara, Michael Hughes, "Lessons Learned from Teaching Intrusion Detection and Intrusion Prevention with Snort", Secure IT conference, Mar. 2006. http://www.slideshare.net/amiable_indian/lessons-learned-from-teaching-intrusion-detection-and-intrusion-prevention-with-snort

[12]Rose Shumba, Towards a more effective way of teaching a cybersecurity basics course, Working group reports from ITiCSE on Innovation and technology in computer science education, June 28-30, 2004, Leeds, United Kingdom .

[13] S. K. Sharma and J. Sefchek. Teaching information systems security courses: A hands-on approach. Computers & Security, 26(4):290--299, June 2007.

[14] Cain & Abel, http://www.oxid.it/cain.html

[15] Ettercap, http://ettercap.sourceforge.net/

ISBN 1-933510-96-X/$15.00