Network Intrusion Detection Lab

Contents

Lab Objectives and Setting

Overview

Network intrusion detection is a critical component of network defense. Through effective network intrusion detection, malicious activities and policy violations can be monitored and identified. In general, Network intrusion detection system (NIDS) can be classified into two types, signature-based detection, and anomaly-based detection. The former one detects intrusions by matching network patterns with predefined rules, and the latter one detects deviations from a model of “good” traffic. In this lab, students will learn how to do the signature-based detection.

Objectives

Upon completion of this lab, students will be able to:

- Describe the working mechanism of two popular open source network intrusion detection systems, Snort and Bro
- Apply Nmap to launch a simple network attack (e.g., HTTP brute-force attack)
- Apply Snort and Bro rules to detect network attacks

Prerequisites

- Practical experience with SSH and basic Linux commands;
- Basic knowledge of network protocols.

EZSetup

EZSetup is a Web application capable of creating various user-defined cybersecurity practice environments (e.g., labs and competition scenarios) in one or more computing clouds (e.g., OpenStack or Amazon AWS). EZSetup provides a Web user interface for practice designers to visually create a practice scenario and easily deploy it in a computing cloud, which allows for customization and reduces overhead in creating and using practice environments. End users are shielded from the complexity of creating and maintaining practice environments and therefore can concentrate on cybersecurity practices. More information about EZSetup can be found at https://promise.nexus-lab.org/platform/.

Environment Setting

In this lab, students can access two virtual machines (VM) in the cloud from EZSetup. One acts as an attacker that is equipped with an Nmap scanner, and the other works as a victim. The victim VM is installed with an Apache web server as well as two NIDS, Snort and Bro, which are used for detecting network attacks against the web server. The network topology for this lab is provided below in Figure 1.

The access information about these two virtual machines is provided below in Table 1. Please refer to the EZSetup dashboard for the actual public IP addresses and passwords.

Table 1 VM properties and access information

Name	Image	RAM	VCPU	Disk	Login account
Attacker	nids-attacker	1GB	1	30GB	See EZSetup
Victim	nids-victim	2GB	2	40GB	See EZSetup

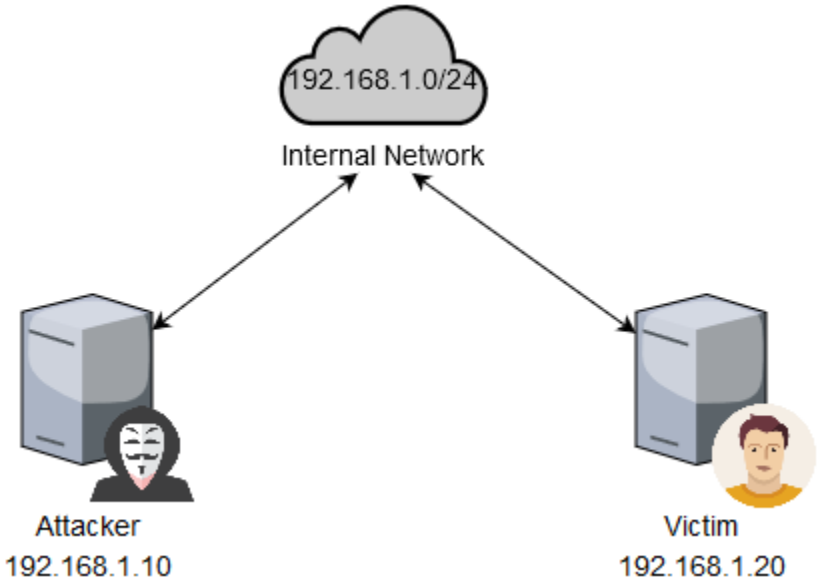

Figure 1 Lab network topology

License

1. Bro NIDS

Bro is an open-source network traffic analyzer, which is primarily a security monitor that inspects all traffic on a link in depth for signs of suspicious activities. Bro supports a wide range of traffic analysis tasks even outside of the security domain. The most immediate benefit that a site gains from deploying Bro is an extensive set of log files that record a network's activity in high-level terms. These logs include not only a comprehensive record of every connection seen on the wire but also application-layer transcripts. Bro provides users with a domain-specific, Turing-complete scripting language for expressing arbitrary analysis task.

Figure 2 Architecture of Bro and packet processing flow

Bro is made of two major components as shown in Figure 2. Its event engine reduces the incoming packet stream into a series of higher-level events. These events reflect the network activity in policy-neutral terms. Script interpreter is Bro's second main component, which executes a set of event handlers written in Bro's custom scripting language. These scripts can express a site's security policy, generate real-time alerts and execute arbitrary external programs on demand.

Tasks:

A. Basic operation with Bro: change to root privilege by entering sudo su, and then launch BroControl shell by typing broctl on the victim VM. Enter the following commands in the shell: deploy, status. Take a screenshot for each result returned by the above commands (Note: you may need to wait for a minute or two before getting the result of the deploy command).

B. Check the policies that have been installed in the Bro configuration. (Hint: check this log file /opt/bro/logs/current/loaded_scripts.log)

2. Snort NIDS

Snort is an open source NIDS that is able to perform real-time traffic analysis and packet logging. It performs protocol analysis, content searching and content matching. The program can also be used to detect probes or attacks. Snort has three main modes: sniffer, packet logger, and network intrusion detection. In sniffer mode, Snort captures the network packets and displays them on a console. In packet logger mode, Snort logs packets to the disk. In intrusion detection mode, Snort monitors network traffic and analyzes it against a rule set defined by the user and then performs a specific action based on what has been identified.

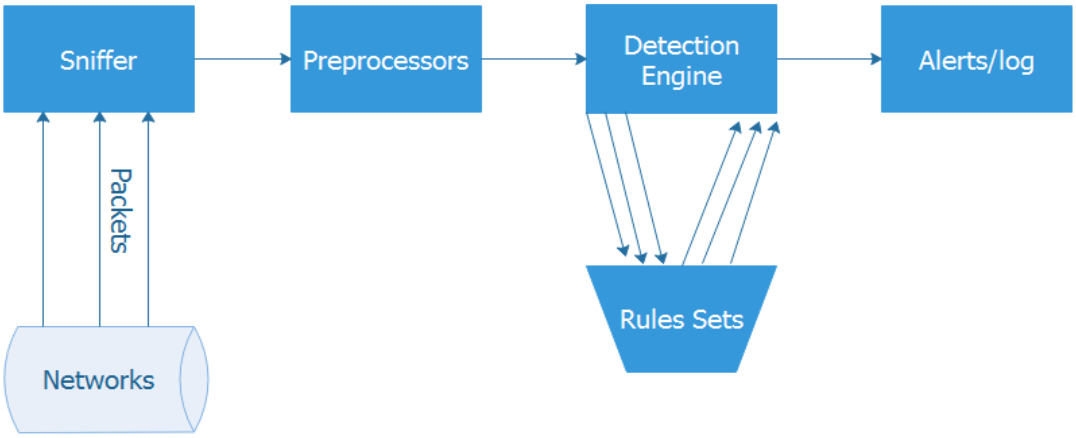

Figure 3 Architecture of Snort and packet processing flow

Snort is logically divided into multiple components as shown in Figure 3. These components work together to detect specific attacks and to generate output in a required format from the detection system. The four major components of Snort's architecture are listed as below:

a. *Sniffer*: The sniffer can collect packets from real-time network traffic covering the different type of network interface and prepare the raw packets to be sent to preprocessors.

b. *Preprocessors*: The preprocessor deals with the raw packet from the sniffer and checks them against certain plugins which determine what kind of packets or what kind of behavior is Snort dealing with. After that, the processors will send packets with a particular type of behavior defined in the plugins to the detection engine.

c. *Detection Engine*: The detection engine is the most important part of signature-based IDS in Snort. It will compare every packet with each rule from a predefined rule set and send the packets that match any rules to the output. If the packets did not match any rules, they will be dropped.

d. *Output*: It generates alerts and log messages depending upon the action rule that defined in the detection engine.

A Snort rule consists of two parts, as shown in Figure 4:

- *Rule header*: The rule header contains the rule's action (log or alert), protocol (TCP, UDP, ICMP, and IP), source and destination IP address and netmasks, and the source and destination ports information.
- *Rule options*: The rule options are optional.

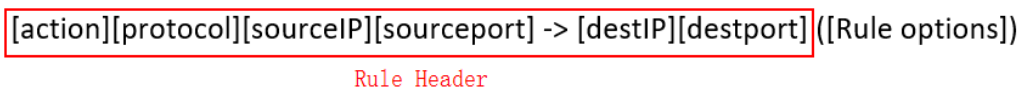

Figure 4 The Snort rule format

Tasks:

A. Verify that Snort has been properly installed: change to root privilege by entering sudo su, and then type command snort -V to get the version of the Snort installed on the victim machine. A screenshot is needed to show you have finished this task.

3. HTTP Brute-Force Attack

Concept: Before understanding the concept of HTTP brute-force attack, we should know the concept of brute force attack, which can manifest itself in many different ways, but primarily consists in an attacker configuring predetermined values, making requests to a server using those values, and then analyzing the response. For the sake of efficiency, an attacker may use a dictionary attack.

One of the most common methods of HTTP access authentication is basic access authentication, which requests clients identify themselves with a login name and password. For the HTTP brute-force attack, the attacker will try the massive combination of login names and passwords. If the credentials are valid, the server sends the requested content. Otherwise, the server responds with HTTP status code 401.

Tool – Nmap: Nmap (Network Mapper) is a security scanner, which is used to discover hosts and services on a computer network, thus building a "map" of the network. To accomplish its goal, Nmap sends specially crafted packets to the target host(s) and then analyzes the responses. The software provides a number of features for probing computer networks, including host discovery and service and operating-system detection. These features are extensible by scripts that provide more advanced service detection, vulnerability detection, and other features. In this experiment, we use the script of http-brute, which performs brute force password auditing against HTTP basic authentication.

The task A and B below assume you use VNC link to access the victim VM

Tasks:

A. On the victim VM, first, make sure the apache web server is running by entering service apache2 status in a terminal window, and then type firefox to launch the browser. Type http://localhost in the URL field to bring up the website hosted on the victim VM. In the pop-up authentication window, enter web as the username and password as the password, and then click “OK” button. Take a screenshot of the web page displayed.

B. On the victim VM, launch Wireshark by typing sudo wireshark-gtk in a terminal window, and then select **ens3** as the interface to capture traffic from and click start to begin. On the attacker VM, enter the command below:

   $ nmap --script http-brute -p 80 192.168.1.20

   Wait until nmap finishes, what is the output of nmap? What do you see from the HTTP traffic captured by the Wireshark (Hint: type http in the filter window)?

4. Detection Using Bro

The HTTP brute-force attack can be detected by analyzing the Bro log file http.log which logs request/response pairs and all relevant metadata. http.log is in the /opt/bro/logs/current directory. There are many lines in the log showing that attacker (192.168.1.10) failed to log in the webpage hosted on the victim (192.168.1.20). Figure 5 shows a portion of the log file. The **administrator** is an account that the nmap brute-force attack tries.

```
1522333700.112352       CJJ72u2lz6zoIcK0rh      192.168.1.10    59656   192.168.
1.20    80      1       GET     192.168.1.20    /       -       1.1     Mozilla/
5.0 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html) 0
459     401     Unauthorized    -       -       (empty) administrator   -
-       -       -       -       FX4FOv1XlW4EEKpNHe      -       text/html
```

Figure 5 A log entry of http.log file

To enable Bro's automatic detection of HTTP brute-force attack, we can add a detection script (here we name it as detect-bruteforce.bro) and load it into Bro. If Bro detects such an attack, it will create alerts in a log file named **notice.log**, which records specific activities that Bro recognizes as potentially interesting, odd, or bad.

To use detect-bruteforce.bro script to detect the HTTP brute-force attack (i.e., too many rejected usernames and passwords occurring from unauthorized requests), we define a new notice type HTTP_Auth_Bruteforce_Attack, a threshold for the number of attempts (**30**) and a monitoring interval (**1 minute**).

chen@ubuntu: /opt/bro/share/bro/policy/protocols/http

```
16 @load base/protocols/http
17 @load base/frameworks/sumstats
18 @load base/utils/time
19
20 module HTTP;
21
22 export {
23         redef enum Notice::Type += {
24                 HTTP_Auth_Bruteforce_Attack
25         };
26         const bruteforce_threshold: double = 30 &redef;
27         const bruteforce_measurement_interval = 1mins &redef;
28 }
```

Figure 6 Define a new notice type in Bro HTTP policy

Using the http_reply event, we check whether the HTTP status code is 401. If so, we use the Summary Statistics (SumStats) Framework to keep track of the number of failed attempts.

```
event http_reply(c: connection, version: string, code: count, reason: string)
        {
        if (c$http?$status_code && c$http$status_code == 401 && c$http?$username)  # && c$http?$password
                {
                        SumStats::observe("http-auth.failed_auth", [$host=c$id$orig_h], [$str=cat(c$id$resp_h)]);
                }

        }
```

Figure 7 Observe failed HTTP authentication response in SumStats

We use the SumStats framework to produce a notice of attack when the number of attempts without valid credential exceeds the specified threshold during a measuring interval. The function threshold_crossed in Figure 8 below is a callback that is called when a threshold has been crossed. If the number of attempts is over 30 within 1 minute, a notice will be generated in notice.log.

```
event bro_init()
	{
	local r1: SumStats::Reducer = [$stream="http-auth.failed_auth", $apply=set(SumStats::UNIQUE), $unique_max=double_to_count
(bruteforce_threshold+2)];
	SumStats::create([$name="http-auth-detect-bruteforcing",
			$epoch=bruteforce_measurement_interval,
			$reducers=set(r1),
			$threshold_val(key: SumStats::Key, result: SumStats::Result) =
				{
				return result["http-auth.failed_auth"]$num+0.0;
				},
			$threshold=bruteforce_threshold,
			$threshold_crossed(key: SumStats::Key, result: SumStats::Result) =
				{
				local r = result["http-auth.failed_auth"];
				local dur = duration_to_mins_secs(r$end-r$begin);
				local plural = r$unique>1 ? "s" : "";
				local message = fmt("%s had %d failed logins on %d HTTP basic auth server%s in %s", key$host, r$n
um, r$unique, plural, dur);
				NOTICE([$note=HTTP::HTTP_Auth_Bruteforce_Attack,
					$src=key$host,
					$msg=message,
					$identifier=cat(key$host)]);
				}]);
	}
```

Figure 8 Use SumStats framework to produce attack notice

Next, move detect-bruteforce.bro file into /opt/bro/share/bro/policy/protocols/http directory, and also add /opt/bro/share/bro/policy/protocols/http/detect-bruteforce.bro in the config file /opt/bro/share/bro/site/local.bro (it may be already there, but you need to check)**.** After that, go to BroControl by entering the command broctl with root privilege in a terminal window. In the BroControl shell, enter deploy to install the new rule.

Tasks:

A. On the attacker VM, launch the attack by entering the following command

$ nmap --script http-brute -p 80 192.168.1.20

On the victim VM, wait for a few minutes, and check if there is a notice.log file in the /opt/bro/logs/current directory. If yes, check the content of the file and take a screenshot.

5. Detection Using Snort

We created a rule file named local.rules in /etc/snort/rules folder. The rule inside the local.rules is shown below, which is used to detect HTTP brute-force attacks:

```
alert tcp $HOME_NET 80 -> any any (msg:"Unauthorized login attempt on webserver";content:"401"
;threshold: type threshold, track by_dst, count 30, seconds 60; sid:1000006;rev:1;)
```

Figure 9 A Snort rule for detecting HTTP brute-force attacks

- The *msg* keyword includes the message that will be displayed once HTTP brute-force attack is detected.
- The *content* keyword can search for content with “401” that is an error status response code in the packet payload and trigger response based on that data.
- The *threshold* keyword means that this rule logs 30th event on this *sid* during a 60-second interval. After an event is logged, a new time period starts for *type* threshold.

- The *track by_dst* keyword means track by destination IP.
- The *count* keyword means count number of events.
- The *seconds* keyword means time period over which the count is accrued.
- The *sid* keyword is used to uniquely identify Snort rules.
- The *rev* keyword is used to uniquely identify revision of Snort rules

The local.rules file has to be included in /etc/snort/snort.conf

include $RULE_PATH/local.rules

Note: Before starting Snort, checksum offloading has to be disabled on the NIC by typing

$ sudo ethtool --offload ens3 rx off tx off

Then start Snort program with the following command:

$ sudo snort -q -c /etc/snort/snort.conf -i ens3

Tasks:

A. On the attacker VM, launch the attack by entering the following command

   $ nmap --script http-brute -p 80 192.168.1.20

   On the victim VM, check the content of /var/log/snort/alert file, take a screenshot of the alerts for HTTP brute-force attacks.

Assignment

Complete all the tasks and save your answer (with screenshots) to each of the tasks into a PDF file. Submit the PDF file.