CS6264 Project 5 Tutorial

DIY Network-based IDS: A Tutorial

Introduction

Now that you have hooked syscalls for an operating system, you have essentially created a sandbox to run malicious code in! These are crucial for setting up a network-based intrusion detection system as you are now able to inspect every executable that flows through your network. All you need to do is hook it up and automate the process! Simple, right?

Before you begin

- Install Vagrant
- Download the Vagrantfile for this project here (it can also be found under the Files tab in Canvas): https://gatech.instructure.com/files/20628443/download?download_frd=1
- Make sure you have VirtualBox or one of the other providers installed
- Initialize your environment by first cd-ing into the directory with the Vagrantfile (hopefully its own directory for your lab), and typing vagrant up into your command line

Before you begin

- This will set up 3 different machines which are all headless. You can enter them by typing (sudo) vagrant ssh [hostname]:
  - mal: the ‘malicious’ machine, that sends malicious binaries over the network. The client vicky will attempt to receive from mal.
    - Username/password: vagrant/vagrant
    - Home: /home/mal/
    - Utilities already installed: vsftpd
    - IP: 192.168.50.2

Before you begin

- vicky: the ‘victim’ machine that will receive the files from mal. For every executable file downloaded from mal, vicky will need to forward the Snort log/PCAP file to sandy.
  - Username/password: vagrant/vagrant
  - Utilities already installed: vsftpd, snort (and dependencies), git, python, watchdog, pwn
  - IP: 192.168.50.3
- Sandy: the ‘sandbox’ machine, which is basically the starting OVA from Lab3. It drops all packets to mal to simulate the box being isolated from the outside. It will receive the executables from vicky and run them to check if they are malicious or benign. Then, it will need to record the alert in /var/log/ids_alert.log
  - Username/password: lab3/captainhook
  - Utilities newly installed: vsftpd
  - IP: 192.168.50.4

Part 0 - 10pts

The first part is to confirm that vicky is able to receive a file from mal. For that, you will have to SSH into mal and try to connect to vicky via the FTP protocol.

You can do this by typing `$ ftp [IP]` and following the prompts to send a file (you can log in using the username and passwords below). The manpage for vsftpd is found here: https://help.ubuntu.com/community/vsftpd.

You may have to confirm that FTP is actually running on all three boxes first.

Part 1 - 20pts

Next, you will have to write a Snort rule in vicky that will detect on an executable file that is sent over the network. The full documentation for this is found here, but you may just want to Google around. Make sure that this is logged, which appear as files in /var/snort/log.

Note: among the provided samples, there is only one sample that does not contain executable file. Your Snort rule should not trap the file.

Part 2 - 20pts

Then, you will want to write an automatic process that will send the log file create over to sandy (best if over FTP) any time there is a new log file created. One way to do this can be with the use of a daemon, and you can find a tutorial here: http://shahmirj.com/blog/beginners-guide-to-creating-a-daemon-in-linux. This can also be done as a shell script.

Make sure you do not send the same log twice!

Part 3 - 30pts

Part 3a - Now, your work will be in sandy. You will have to tweak your current script somewhat so it is no longer an anomaly detection engine: it will solely be used to detect malicious syscalls (the same syscalls identified in the last project) and print them to the kernel log. This should be a simple copy-paste. If you do not wish to use your script from lab3, we are also providing a sample hooks.c script with all the hooks you might need for this assignment, which can be found here (it can also be found under the Files tab in Canvas).

Part 3b - The next step is to write a daemon for sandy such that whenever it receives a new log (hopefully always to the same directory), it will extract the binary from the PCAP and run the binary (with the LKM already inserted).

Part 4 - 10pts

You may have noticed that we mentioned “kernel log” in step 3a. This is because whenever you call printk() in you LKM, all of these messages are being logged at /var/log/kern.log.

Today, while syslogd, the syslog daemon, manages some of the logging in Ubuntu, most of it is done through a service called rsyslogd, which extends the functionality of syslog (incidentally, extending it to log kernel messages). This service is also configurable, so you are able to direct messages to different log files of your choosing (even new ones!).

You will need to configure rsyslogd so that it will output any messages from your LKM to be put into a log file called ids_alerts.log. The full documentation for this can be found here: https://www.rsyslog.com/doc/v8-stable/configuration/index.html

Bonus – 5 pts + 5 pts

Part a - you will have to think of what potential behaviors malicious binaries might have. You will alert these behaviors in the rsyslog. Then, after analyzing the resyslog, Sandy will transfer back the benign files to client Vicky and drop the malicious files.

Part b – Zeek (or originally called Bro) is an alternative network IDS tool for snort. For this part, you will use Zeek for Part 1 as an alternative of snort on the detection of the executable files sent over the network. Similar to snort, you will have to log the binary drop and extract the executable file.

Deliverables

- Zip the following files in a .tar.gz file called [GTusername]_cs6264_lab5.tar.gz
  - Snort configuration
  - Various daemons that you wrote for both the sandbox and the client
  - Your LKM C file and Makefile
  - The new rsyslog configuration
- A report of your NIDS that includes:
  - Screenshots of each step of the process from when you send the file to the client to when an alert is logged. These should probably include: sending the executable from mal; receiving the executable in vicky; sending the PCAP to sandy; the sandbox alerting/not alerting on the executable
  - Any implementation details that you want to share
  - **DETAILS PERTAINING TO HOW TO RUN YOUR CODE (THIS IS VERY IMPORTANT!!!! IF WE HAVE TO GUESS HOW TO RUN YOUR CODE WE MAY NOT BE VERY CHARITABLE IF YOU LOSE POINTS)**

Rubrics

- Proof of successful file transfer over FTP: 10 pts
  Snort rule correctly written: 20 pts
  Daemon to send file from Vicky to Sandy correctly written: 20 pts
  LKM updated for new malicious program: 10 pts
  Daemon written to analyze program on sandy: 20 pts
  Rsyslogd config correctly written: 10 pts
- Report: 10 pts
- Bonus: 10 pts