

CSI 4139 / CEG 4399

Design of Secure Computer Systems

Laboratory #3

Due: November 8, 2018 (written report due November 12, 2018)

PART (a)

Goal: This lab is intended to give you experience with the Snort program. Snort is a simple and powerful network monitoring agent. In this lab you will install and configure Snort, as well as write Snort rules to identify various attacks and test the applied rules.

Requirements: 3 machines (VM or Real) (Target, attacker, and Snort IDS) + Network Equipment

Pre-lab Background: The Snort homepage is located at the following address: www.snort.org. On the website there are documents that may assist you in understanding Snort (in particular, you may wish to consult the Snort User's Manual, available at <https://www.snort.org/documents>). The section on writing Snort rules will be a helpful reference for writing the rules needed for this lab.

Deliverables: You are expected to write a document and demonstrate your results.

For this lab you need to run several different attacks (see below). Test the attacks with the Snort IDS and show how you detect and prevent these attacks using the rules that you have written.

The primary intent of this lab is for students to learn how to write effective rules. For example, it would be easy to write rules that match all IP datagrams regardless of content, but this would be ineffective for detecting anomalous malicious activity.

Include in your report the rules you have created as well as the /var/log/snort/alert output. (The alert file is appended each time Snort produces an output, so you should erase the alert file before each Snort run while experimenting with different rules.) Be sure to include a descriptive message ("msg") with each alert. Support your report by analyzing the results, including snapshots and graphics as appropriate.


Attacks: For this lab, students will conduct 3 different attacks. (To simplify the “guess” of information required for an attack, such as TCP sequence numbers and source port numbers, assume that attackers are on the same physical network as the victims. Therefore, sniffers can be used to get that information.)

The specific attacks to be implemented can be freely chosen, but must fall into the following 3 categories (see https://www.snort.org/rules_explanation):

- ICMP-based attack
- SNMP-based attack
- Attack from a blacklisted source

PART (b)

Goal: Implement a rudimentary antivirus program.

Details: Your program must successfully perform the following task 

- Read the latest virus definition file to find that a new virus has a particular signature (that is, the virus contains a specific sequence of bytes). Furthermore, assume that the virus may use a Caesar-cipher-type of encryption to make itself polymorphic.
- Check every file in a specific folder and in all of its subdirectories to find any that are infected.
- For any infected file found, render the virus inactive (set the first 8 bytes of the signature sequence to “xxxxxxx”) and quarantine the file (move it to a specified folder).
- Keep the user informed (through the UI) about what the program is doing, what infected files it has found, whether inoculation was done, and whether the dangerous file was quarantined.

During the lab demonstration of your program, the TA will supply a virus definition file and a folder containing multiple files of various types (some of which may be infected). The virus definition file will consist of one or more lines, with each line containing a virus signature (a specific sequence of bytes up to 96 characters in length). Your program must read each line and look for any viruses in the given files. The TA will observe the operation of your program.

Deliverables: you are expected to write a document and demonstrate your program.

Document: write a brief description (no more than three pages) of your program and the relevant choices you have made. In particular, you should describe your environment and programming language, and you should explain how your program uses the virus definition file and checks for the presence of the virus. Any underlying assumptions that must be true in order for your program to work should also be discussed. Provide a section describing what functionality would need to be added to your program to make it useful for a production environment.

Software: you must implement the antivirus software. This will consist of working code, as well as a UI sufficient to allow the TA to observe what your program is doing.

Laboratory #3, Parts (a) and (b), can be done in groups of up to 3 people.

Additional note for Lab #3, Part (b):



You may find it useful to look at the link http://www.eicar.org/anti_virus_test_file.htm

EICAR is essentially a “safe” file that was designed to test anti-virus and anti-malware programs. It does not do any damage, but looks like a real virus to such programs and thus allows people to confirm that their anti-virus programs are working properly. The file eicar.com has a 68-byte signature. This file, and its associated signature, may serve as a useful test for your laboratory code (you can see if your code will properly read the signature from a virus definition file, find the eicar.com file, inoculate it, and quarantine it).

Note, however, that the computers in the lab all have anti-virus software installed already. Consequently, they are likely to find eicar.com and quarantine it as soon as you try to download it (i.e., before you can get a chance to try your own program on it). To get around this, you may need to Caesar-cipher-encrypt the file eicar.com and have your program work with this instead of the original file.

Working with the EICAR file is not required for this lab; it is simply one interesting and real-world way to test the functionality of your implementation.