CS 499/ISA 564: Lab 5 – Network Hunting and C2

Lab Submission Guidelines

Submissions must be in Microsoft Word or PDF format. Be sure to clearly label what question you're answering. Where possible, screenshots should be embedded directly in the document. Screenshots should be cropped to only include what is necessary to answer the question. If you need to save them as a separate file, save them in compressed format (gif, jpg, png) and name them after the question they pertain to. If your submission contains multiple files, archive them (zip, 7z, tar.gz) and submit the file via Blackboard.

Lab Requirements

- 32-bit Windows 7 VM
- Kali Linux VM
- Files in the Lab 5 Resources folder

Kali VM Setup

- Configure networking as NAT to allow downloading updates
- Update package repositories – *apt-get update*
- Upgrade installed packages – *apt-get dist-upgrade -y*
- Install Snort and pcregrep – *apt-get install snort pcregrep*
- Snort install will ask for a local network definition. Just press enter
- Change networking back to host-only
- Open /etc/inetsim/inetsim.conf and change the service_bind_address and dns_default_ip to the IP address of your Kali VM and uncomment the lines
- Start inetsim (it will say simulation running when it's done launching) – *inetsim*
- Extract the PCAPs from AnalysisPCAPs.zip into an easily accessible folder

Windows VM Setup

- Configure networking as host-only
- Go to Start → Run → Type in ncpa.cpl and hit enter
- Right-click your Local Area Connection interface and go to properties
- Click on Internet Protocol Version 4 (TCP/IPv4) and then properties
- Manually configure the same IP address/subnet mask the VM had before but change the default gateway and preferred DNS server to that of your Kali VM
- Install the Visual C++ 2010 Redistributable from vcredist_x86.exe (YARA prerequisite)
- Copy yara.exe and extract the malware files from AnalysisFiles.zip into an easily accessible folder

Lab 5: Task 1 – C2 Protocol Analysis

For this exercise we will be using Lab5.1.exe. The purpose of this exercise is to learn how to fool malware into performing its C2 operations within a monitored environment as well as to capture its C2 data and reverse engineer it by analyzing its network traffic.

Launch Wireshark and start capturing on the Local Area Connection interface. Optionally disable promiscuous mode to reduce packet spam. Launch Lab5.1.exe. It will briefly flash a command prompt and finish executing. If it crashes then it was unable to resolve DNS. If it ran successfully then stop and save your capture.

Question 1.1 – Provide a PCAP file or screenshot within Wireshark showing successful beaconing of the malware to your Kali VM running INetSim. Screenshots must show the entire C2 session and must highlight the actual C2 protocol traffic (the packets from Windows → Kali after the connection is established).

Question 1.2 – Provide a complete annotation of the malware's C2 protocol. Your annotation must account for all protocol bytes. Some bytes are just random binary data. The easiest way to extract the protocol bytes is to Follow →TCP Stream then under Show and save data as, select Hex Dump then Save as.

Lab 5: Task 2 – C2 Code Analysis

For this exercise we will continue to use Lab5.1.exe. The purpose of this exercise is to learn how to reverse engineer a C2 protocol by analyzing the code used to parse it from within a compiled binary.

There is a function in Lab5.1.exe (sub_401529) that performs C2 protocol validation on an input string to ensure that the string is correctly formatted. You must first develop and understanding of how it validates input and then provide an input string that will pass validation. Here are some helpful tips for these questions:

- There is a local variable that keeps track of the number of bytes read, look for add instructions with that variable as the destination operand
- One of the data parses happens in a loop, find the loop termination condition
- The final parse can be difficult to, eh, parse. Keep track of eax. The same size for malloc() is used for memcpy(). It may behoove you to debug the binary to better understand the function arguments

Question 2.1 – Identify the address of every instruction used to update the number of bytes read (use text view), as well as the technique used by the parser to identify where one data chunk ends and another begins.

Question 2.2 – Execute the binary and pass an argument containing a string of hex bytes (no spaces or prefixes like \x or 0x) that will be successfully parsed. The binary should say C2 Test Succeeded. Provide a screenshot of your input and the binary's output.

Lab 5: Task 3 – Writing YARA Malware Rules

For this exercise we will be using the malware files in AnalysisFiles.zip. The purpose of this exercise is to learn how to write quality YARA signatures against a set of malware binary files.

The malware files contained in this zip file are all variants of njRAT. The binaries are .NET executables not Windows PE files. This means that you cannot analyze the code using tools like IDA Pro as .NET executables are not compiled to machine code, but rather to an intermediate byte code (like Java).

Your objective is to write a YARA rule that will trigger on all 5 malware binaries. Without code to examine, the YARA rules for these binaries must be based on their strings. Since the binaries are not packed, each of them has thousands of strings to choose from. Deciding which strings are good candidates for rules requires experience to develop a good heuristic. Here are some tips:

- If you find a string that you think is unique enough to be interesting, Google it. If you get search results entirely unrelated to malware then you've likely chosen a red herring. If you get very few results then your string is unique enough. If your string shows up in security reports about malware, you've chosen well.
- Combining individually weak strings into a strong condition is a perfectly valid way to build a rule. Not every string needs to stand on its own since the rule is evaluated as a whole.
- Try to find strings that represent something the malware author must have written themselves rather than something that exists elsewhere (like a library function name).

Run YARA with the following command line:
yara <rules file> <directory with your malware files>

Question 3.1 – Provide your YARA rule and a screenshot showing your rule hitting on all 5 malware binaries.

Extra Credit (up to 10 points) – Write a YARA rule that not only hits on all 5 malware samples but also meets the following conditions: at least 4 strings and regular expressions with at least 1 of each, and a condition that is not just "all of them" or similar. Points will be awarded based on the creativity and efficacy of the rule. This can be the same rule used to satisfy Question 3.1. The purpose of this extra credit is to encourage you to become more familiar with YARA's functionality and rule capabilities. Treat this as a learning experience.

Lab 5: Task 4 – Writing Snort Malware Rules

For this exercise we will be using the packet capture files in AnalysisPCAPs.zip. The purpose of this exercise is to learn how to write quality Snort signatures against a set of malware network traffic data.

The PCAPs contained in this zip file are from variants of njRAT. They contain C2 traffic sent from victims to the malware's controller. All PCAPs contain only TCP sessions relevant to the C2, there is no extraneous data. TCP.pcapng is a large PCAP containing innocuous TCP data and serves as a control for this exercise.

Your objective is to write a Snort rule that detects on all 5 njRAT PCAP files but not on TCP.pcapng. Your rule should be placed in /etc/snort/rules/local.rules. You should also modify /etc/snort/snort.conf to comment out all rules files besides local.rules. Your rule should not detect based on information obviously unique to this exact PCAP traffic such as hostnames, IP addresses, OS identifiers, and filenames. Instead, detect based on the protocol structure of the RAT.

Run snort using the following command line:
snort –c /etc/snort/snort.conf –q –k none –A console --pcap-show --pcap-dir <directory with your PCAP files>

Question 4.1 – Provide your Snort rule and a screenshot showing a Snort alert for that rule on the 5 njRAT PCAP files but not on TCP.pcapng. Your screenshot must show Snort running against all 6 .pcapng files.

Extra Credit (up to 10 points) – Research njRAT and provide a write-up of how it works, what malware campaigns it has been used in, how its C2 works, what functionality it has, etc. The more thorough your write-up, the more points you will receive.

Grading

- **Task 1 – 30 points (15 points per question)**
- **Task 2 – 30 points (15 points per question)**
- **Task 3 – 20 points (20 points per question)**
- **Task 4 – 20 points (20 points per question)**
- **Extra Credit – Up to 20 points**

Lab Survey - https://cryptomancer.typeform.com/to/GmDK2A