Course Project: Milestone 5

Cesar Munguia

04/23/2024

—

IS-4543-001

—

Jose Mireles

Objectives

Each milestone submission will be a write-up that includes:

- What did you do?
- What did you learn?
- Documentation of your work
    - Screenshot
    - Picture
    - Video
    - Interpretive Dance

Project Proposal

The project involves in-depth analysis of a virtual machine infected with malware, focusing on understanding the malware's behavior, persistence mechanisms, and potential impact on the system. Through forensic techniques and dynamic analysis, I will dissect the malicious code, identify evasion tactics, and extract indicators of compromise. The project aims to enhance cybersecurity skills by providing hands-on experience in malware analysis and incident response, crucial for defending against and mitigating the effects of sophisticated cyber threats.

- Milestone 5: Document findings, and analysis techniques. Generate a brief report of summarizing the characteristics and behavior of the malware sample. Include possible recommendations for mitigation.

In Milestone 5 I will be focusing in documenting all my findings and final conclusions. I will also make a brief report summary about the specific actions this malware sample made while in execution. Finally, I will conclude this project talking about some recommendations of the sandbox environment itself, lessons learned, and ways to combat or mitigate similar malware.

Summary of Activities

My first step in this final milestone was to create tables that shows an overall description of the analyzed malware. This is usually done when a forensic/malware report is made. Authors start their report by creating some visual representations (like tables), to give the reader some type of context. Usually, reports tend to be very long, extensive, and in-depth, however, for the sake of this assignment I will try to be as brief as possible. I started by creating a table in my “Documentation of Work Completed” that shows high-level information about the malware sample I conducted an investigation on. Exhibit 1 states the file name, size, type, a variety of hashes including MD5, SHA1, and SHA256. It also includes the CPU type, compiler-stamp, and debugger-stamp. Then, Exhibit 2 shows low-level information about the malware, more specifically general PE information. It states, the section name, entropy, file-ratio, raw-address, virtual-address, and virtual-size. Then, in Exhibit 3 I made a table about the files I was able to spot that the sample had created. I was able to easily spot these created files because the program itself put them in the same directory where the executable resided. To be more exact, the binary created 1 directory and several .png files. Then, I created Exhibit 4 emphasized the overall behavior of the malware. The table is separated into two columns, one column states the objective and the second one states the actions taken by Core1Installer.exe. My final table, Exhibit 5, talks about some of the techniques used that stood out the most. Those were obfuscation, anti-debugging, C2C, input capture, timestomp to name them.

Finally, I wanted to visually represent the network diagram I was working with in my sandbox, as well as some other logical networks that would've worked a little better in this situation. I used the draw.io tool in the web to assist me in the creation of these logical networks.

In Logical Network 1, I drew the actual network used in my sandbox. This was composed of the Windows 10 VM (victim) and the REMnux VM only. There were no special networking devices like routers or switches. In Logical Network 2, I showed the network used whenever I tried to instrument the malware by changing its embedded IP to 192.168.204.203. Again, no specialized devices are present. In Logical Network 3, I created a combination of the previous two networks, which is composed of two separated networks with several routers. One network is 192.168.10.0 and the other one is 192.168.240.0 with the second one having the embedded IP that was modified by me. Finally, in Logical Network 4, I had the initial network where the Win 10 VM and the REMnux resided, but the only thing that changed here is that there's only one network. The actual embedded IP is not changed and is just standing outside of the network acting as an actual public IP address.

Description of Learning Completed

With all the analysis being done at this point there were a lot of things that I learned during the process. One of my initial discoveries was that the malware tried to imitate a regular file with a normal convention name like Core1Installer.exe. This sample could've been targeted to a specific group of people that might've needed a program update or a new tool in their computer. The program itself did not occupy a lot of storage, which in some cases may rise a flag indicating suspiciousness, but rather had a small weight (420 KB). During my static analysis, I learned that the malicious user used .NET assembly for an additional layer of security. If the malware was probably packed with other obfuscation tools like UPX, I might've been able to unpacked it, but I was unable to disassemble .NET.

The PE information indicated me that most of the code was going to reside in the .text and the .rsrc sections for the simple reason that they had the largest file-ratio. I was able to prove this with the use of disassembler tools. During my disassembly process, I was not able to fully reverse-engineer the whole portion of the code because of my lack of knowledge of assembly language. However, I was able to grasp high-level information about the structure of the code that helped me in my analysis further way. The program was structured with conditional rules like if-and-else statements, as well as it included error-check code with try, do, and except statements. I'm assuming that the conditional rules were more focused in the logic of the program, and the error-check code was more focused to components of the operating system itself that enforces my belief that the executable had VM-aware capabilities.

In my dynamic analysis, I was able to confirm that my hypothesis made in the prior analysis was true. This hypothesis I made stated that the program was more than likely going to establish a HTTP connection with an external IP, as well as to create/modify/delete files and/or directories. I noticed that the after execution, the binary created a "Images" directory where it stored all of the screenshots taken. This is shown in one of the tables I created. Also, I learned in tcpview64.exe that the executable was in fact trying to communicate to a public IP address. However, it failed because there was no such IP in the sandbox environment.

At first, I thought the REMnux VM was going to come very handy in this situation but it ended up not being useful at all. Even though the machine was acting as the default router and also as the internet, the program failed to establish a connection. This was due to the nature of situation. Let me explain, the malware was trying to communicate to an inexistent IP, and whenever REMnux received the packets from the Win 10 VM it tried to redirect them to that IP but did not where to send it exactly. Also, since the connection was stated to be through HTTP using port 9000, then REMnux was also unable to process this data because HTTP packets regularly go through port 80, not 9000. For that reason, I couldn't get anything relevant from the logs and/or Wireshark. Even when I modified the embedded IP to a different one, it still failed to communicate properly.

So, as shown in my second logical network, one of my recommendations for the future could be to add another VM, and leave the REMnux for a different use. This other VM could have the modified IP address and be listening to port 9000 using netcat, and with that I could test to see if I'm successful in the communication. This would later give me more relevant information for my analysis, like possible payloads or additional malware-downloads being installed in the victim's machine.

Another approach would be to create two different networks composed of two routers. One network would contain the REMnux VM and the Windows 10. The second router would contain the modified IP found in my analysis. These two networks would be connected by the routers, and I could have either two or three routers, where the third one would connect the other two. Essentially, it would be the same thing so in the future I could stick to only two routers. This is shown in my Logical Network 3. Finally, the situation or setup that I believe would work the best is going to require me to have one network only. Again, this network would be only composed of the router, the switch, and my two VMs. Then, I would have the actual suspicious IP sitting outside of the network by itself. I would then add a route to the router's routing table so that it knows where to redirect the traffic if it's trying to send it to a specific node. In this case, whenever the malware in the Win 10 VM wants to establish the connection with 188.120.240.203, it would start by sending the traffic to the switch, then to the router, then the router having the needed route it would send the packets to the node. This other VM sitting outside the network could be in listening for communication in port 9000 using netcat in server-mode.

There could be several ways when it comes to mitigating malware infection, or intrusions inside a enterprise/corporate network. One of them is to incorporate a NIDS (network-based intrusion detection system) and a HIDS (host-based intrusion detection system) for every device in the network. This would not only detect intrusions, but it would also prevent them if strong policies are in place. Another recommendation that never does bad to a firm is to train the employees and make them aware about the techniques used by malicious users when trying to compromise networks. This training would not only make them aware of the techniques, but it would also show them how to properly check phishing emails and what to click and what not to click. My final recommendation would be is to have well-written incident response plan for several reasons. Firstly, it serves as a proactive shield against potential damage caused by various incidents, be it cybersecurity breaches or operational errors, by swiftly containing the situation and mitigating its impact. Secondly, it aids in minimizing downtime, crucial for preserving revenue streams and productivity levels. Thirdly, a well-executed plan safeguards the firm's reputation and brand image, instilling trust among stakeholders by demonstrating the company's commitment to security and resilience.

Documentation of Work Completed

Exhibit 1

TARGET	
File name	Core1Installer.exe
File Size	430080 bytes (~420KB)
File Type	Executable (.exe)
MD5	7DC4BD6B762DC2E36CB201282D419148
SHA1	704E8CB2DC1282DFC17BC0ED0326CFFF943B693D
SHA256	031ED94B13F6292CA38061AC20D5C784C6470D3F52B207A959BEDF0ED12C0665
Code-page	Unicode UTF-16, little endian
CPU	32-bit
Compiler-stamp	Wed Oct 13 10:18:32 2038 | UTC
Debugger-stamp	Wed Feb 23 16:32:35 2101 | UTC

Exhibit 2

PE INFORMATION			
Section name	.text	.rsrc	.reloc
Entropy	6.315	6.236	0.102
File-ratio	52.14%	47.62%	0.12%
Raw-address	0x00000200	0x00036E00	0x00068E00
Raw-size	224256 bytes	204800 bytes	512 bytes
Virtual-address	0x00002000	0x0003A000	0x0006C000
Virtual-size	223928 bytes	204588 bytes	12 bytes

Exhibit 3

FILES CREATED	
Type	Name
Directory	C:\Users\admin\Desktop\malware-sample\Images
PNG	2024-04- 10 12-17-15.png

PNG	2024-04- 10 12-17-20.png
PNG	2024-04- 10 12-17-25.png
PNG	2024-04- 10 12-17-30.png
PNG	2024-04- 10 12-17-35.png
PNG	2024-04- 10 12-17-40.png
PNG	2024-04- 10 12-17-45.png
PNG	2024-04- 10 12-17-50.png
PNG	2024-04- 10 12-18-00.png
… + 100	…

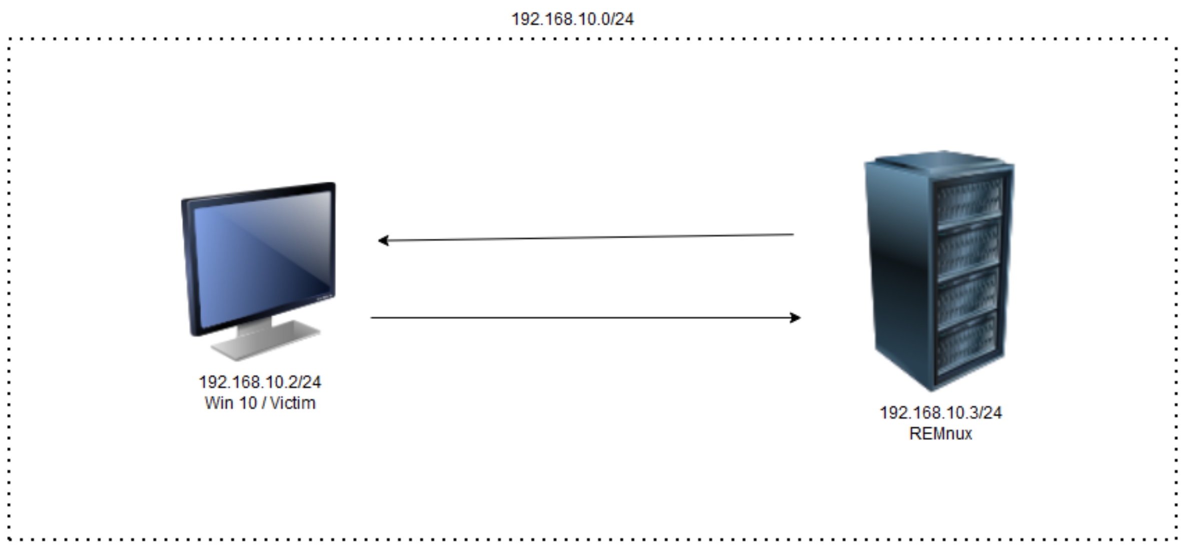
Exhibit 4

OVERALL BEHAVIOR	
Objective	Actions
Communication	Create/Send HTTP request Receive HTTP response
File system	Create directory Delete/Create/Read file Set file attributes
Cryptography	Generate Pseudo-random sequence: Use API
Collection	Creates a DirectInput object (capturing input)

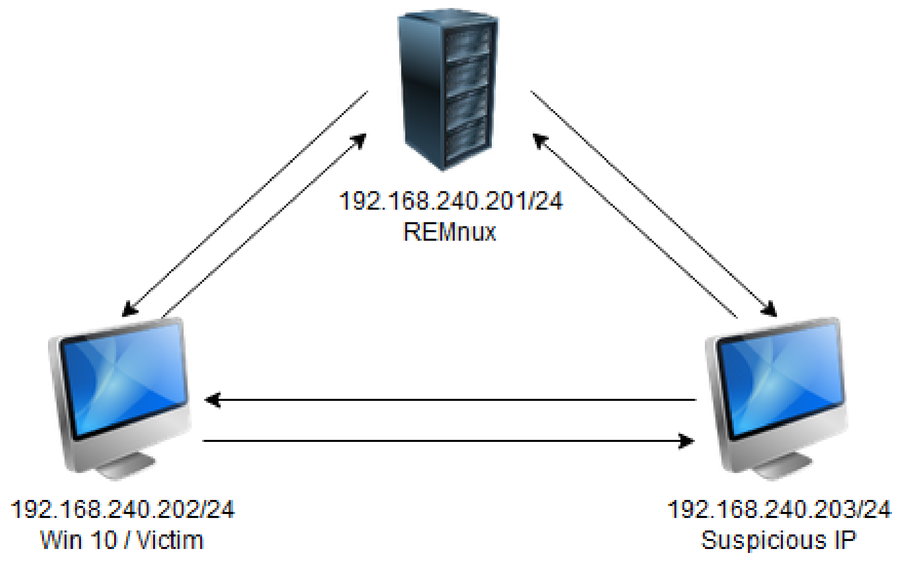
Exhibit 5

Techniques	Description
Obfuscation	.NET assembly layer
Anti-debugging	Extra layer of protection due to obfuscation
C2C	Tries to establish a connection with suspicious IP
Input Capture	Takes screenshots
Timestomp	Compiler time was modified with anti-analysis purposes

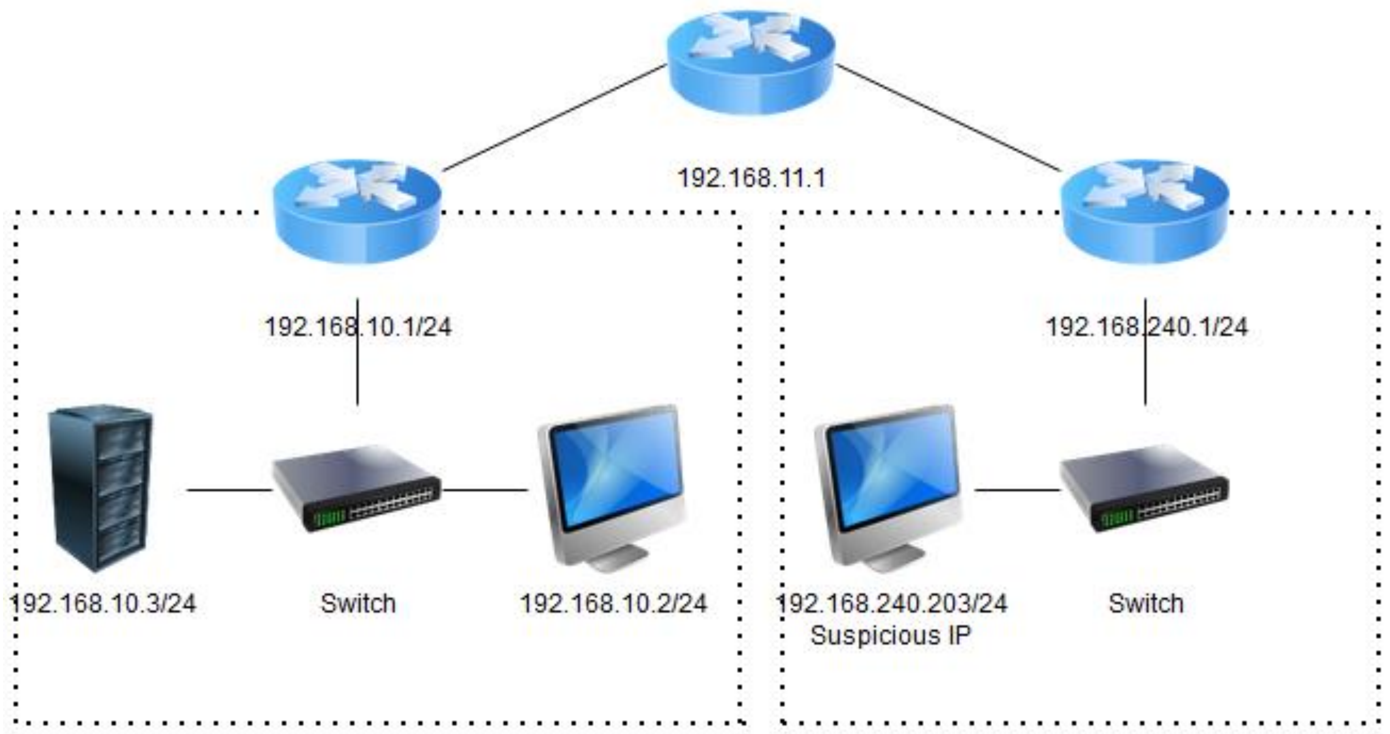
Logical Network 1



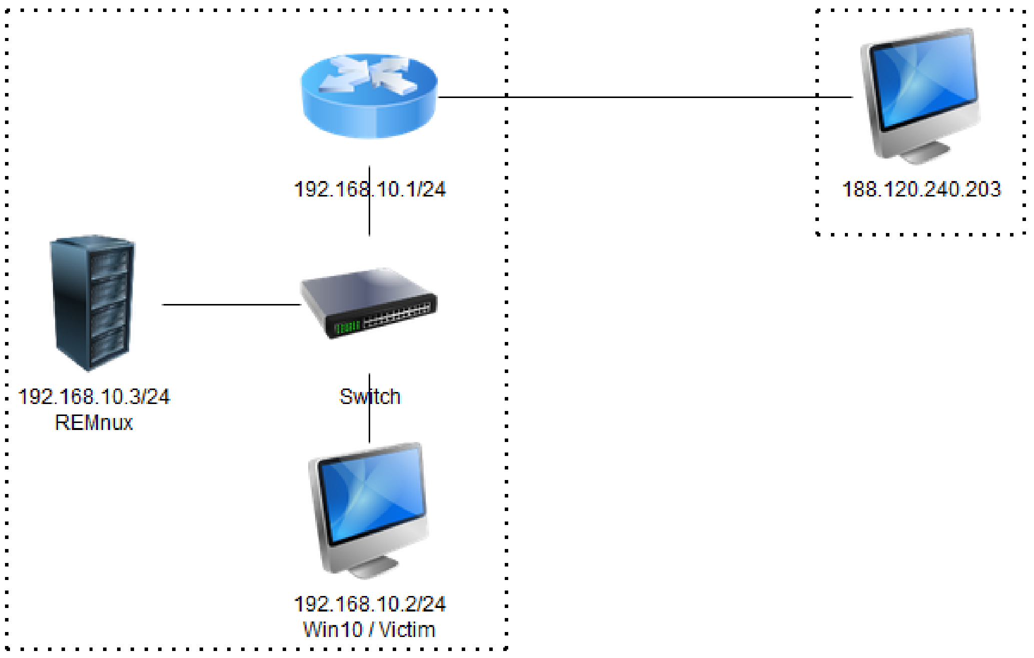
Logical Network 2



Logical Network 3



Logical Network 4


192.168.10.1/24
188.120.240.203
192.168.10.3/24
REMnux
Switch
192.168.10.2/24
Win10 / Victim