Oregon State University
Computer Science

Malware Analysis

Project Archive - Final

Michael Banks
Jennifer Bowers
Steven Hunt
Mark Kaiser

Project Sponsor: Bill Pfeil

CS467 Online Capstone Project
Spring Term, 2023

POST-MORTEM REPORT

INTRODUCTION

Our team's journey through the Malware Analysis project has been engaging, fulfilling, and dynamic. From the start, we've valued the opportunity to broaden our skills through collaborative malware analysis. We immersed ourselves in the complex world of exploitation, using disassemblers like *IDA* and decompilers such as *Ghidra*, deepening our understanding of malware propagation, persistence, and evasion techniques. This project has provided us with unique exposure to security concepts, extending beyond those typical of a standard computer science course.

Throughout the project, each team member seized the opportunity for independent work, personal exploration through trial and error, and the sharing of individual knowledge and insights. While some pieces of malware required more rigorous analysis than initially anticipated, our collaborative spirit, willingness to adapt, and the mutual sharing of knowledge secured our way to the finish line. In the end, we leveraged malware analysis tools and our collective insights to conduct successful static and dynamic analyses of three malware samples, all while prioritizing the detailed documentation of our findings.

Our team's accompanying demonstration video, illustrative poster, and this comprehensive final project archive serve as confirmation of the Malware Analysis Project's success. As a Malware Analysis Project team, we are thrilled to present, at the Career Showcase event, a unified platform that exhibits our successful journey. We believe our final product is a testament to our team's resilience, collaborative spirit, and dedication to deepening our understanding of the intriguing world of malware.

TEAM DEVELOPMENT

As the term draws to a close, the Malware Analysis team has successfully analyzed three samples of malware. The first sample was an executable PE file, while the second was a DLL file that was not entirely executable. The third sample, also an executable PE file, was a much more sophisticated specimen than the first. The team has employed two types of analysis: static and dynamic. Static analysis examines the malware before execution and requires an understanding of programming languages, compiler optimizations, and sometimes even assembly code. Dynamic analysis involves execution of the malware, which requires handling the exploit with care, as well as a thorough understanding of computer architecture, runtime processes, and network communications.

Our team strategically split into pairs for each phase of malware analysis, gleaning samples sourced from the Practical Malware Analysis [1] textbook. Each member contributed a reasonably even amount of work per analysis, while also bringing diverse experiences to the table. Our teammate Mark, specifically, assumed a leadership role with his extensive security-focused history. Mark's expertise proved invaluable in offering insights into the effective use of various malware analysis tools, how to set them up properly, and when to use them. Our bi-weekly virtual team gatherings served not only as a platform for discussions around our individual discoveries and documentation, but also as a venue for Mark to guide us through thoughtful, insightful walkthroughs.

PERFORMANCE REVIEW

Upon reflection, our team's drive and determination emerged as significant strengths. Our Malware Analysis team maintained a focused and collaborative approach throughout the term. Bi-weekly virtual meetings served as platforms to share insights, deepen understanding, and lend support to any team member facing challenges. Despite occasional difficulties in the analysis process or with the deployment of tools, our team's resilience and cooperation turned challenges into productive learning opportunities. Our team's agreed upon project plan facilitated an equitable distribution of rotating roles and responsibilities, ensuring a balanced workload for analysis, and that nobody was placed in an uncomfortable or demanding position. Thus, even challenges with specific malware samples served as invaluable learning opportunities.

The project, however, was not without challenges. One issue we encountered was the selection of a malware sample better suited for static analysis, rendering our dynamic analysis attempts ineffective. This hitch, while frustrating, emphasized the importance of careful malware selection for our work. On the one hand, for future projects, a predetermined list of suitable malware samples would streamline the analysis process and prevent such hurdles. On the other hand, our team acknowledges that the malware sample challenged us to try every analytic tool we could get our hands on! The experience was enriching and honed our skill sets further. Given that the primary objective of our project was to advance skill development and gain hands-on experience with malware analysis, it is evident our holistic approach to the project was a success.

KEY TECHNOLOGIES

The malware analysis team utilized a variety of technologies to safely analyze each sample. A foundational technology that our analysis environment relied upon heavily is *VMware* and custom configured virtual machines. This required advanced network configuration to isolate the malware and hypervisor to prevent malware virtual machine escape. In addition to this configuration several open source and well known tools were installed. Once everything was set up a baseline snapshot was taken. This allowed us to return to a clean state after different phases of analysis. Throughout the setup, configuration and usage there were pros, cons and the learning curve.

While setting up our analysis environment the team found there were subtle differences between the *VMware* versions for different Operating Systems such as Mac, Windows and Linux. This caused some confusion and required additional effort to research where the settings were hidden in each *VMware* version's user interface. Another challenge was figuring out how to move files between our different VMs that had a virtualized host-only network between each other. The solution we came up with was to sideload a custom python script that uses the Python simple web server module and serves up an upload page form for use in a web browser. This allowed for download and upload to our *REMnux* virtual machine from our *Windows XP* virtual machine.

Our secure analysis environment ended up working really well. No malware escaped and we were able to analyze each sample despite this locked down environment. The team could backup artifacts and findings to the *REMnux* virtual machine before reverting the *Windows XP* virtual machine to the clean snapshot. Screenshots and other evidence were able to be taken from our bare metal host.

During our analysis many of the tools were easy to use, however there were a few advanced tools that required significant research to understand usage. Tools such as *Strings*, *Floss Flare*, *PEID*, *PeStudio*, *Process Monitor*, *Process Explorer* and *Wireshark* were all simple

to learn and use. Tooling such as *IDA*, *Ghidra*, *x64dbg*, *InetSim*, and *FakeDNS* took additional time and effort to become familiar with. These tools provided unique capabilities to pull out properties from the sample or even interact with the sample after execution.

LESSONS LEARNED

Michael Banks

My personal accomplishments with the malware samples are as follows. For sample #1 I performed a static analysis using all of the static analysis tools we have readily available, finding *PEid* to be the most useful as it reveals imports, exports, and strings. In one of our meetings our leader showed us another method of analyzing strings using a program called *Floss*, which helped us find the most important information about what services the malware deployed. For sample #2 I performed dynamic analysis using all of the dynamic analysis tools we have readily available, however those tools are only useful if you can run the malware, which I was unable to figure out how to do with the particular DLL malware sample. As a team in our meetings, we looked at it again and continued to try running the DLL, but nobody was able to do it. This not only gave me confidence that I was not doing anything particularly wrong, but also gave us a nice learning experience about how different tools can be used to approach the same problem from multiple angles. For sample #3 I performed dynamic analysis just as with the previous sample, however this sample worked out great. The static analysis team found a piece of the malware's code that could be modified to get it to run for dynamic analysis, and once I made that modification everything worked smoothly. This was a nice shift from the previous sample's failure.

For the project plan document, I created most of the visuals and gave a description for each of some of the analysis tools we are using, and I have written a great portion of this project archive document. For the career showcase poster, I pieced together the elements of the poster that were decided on by the group and concentrated on making it visually appealing by using images with transparent backgrounds, adding in screenshots of six of the programs we used in action, and writing in descriptions of each that the rest of the team could then further polish from what I wrote. I was also responsible for the overall formatting in the final product after following guidelines we determined as a team. As for the reading, I read most of the chapters in the textbook and have been sure to stay at the rate of the malware samples we are working on.

Jennifer Bowers

Being an active participant in the process of creating this malware analysis project has enabled me to gain considerable knowledge, skills, and insights. I have become increasingly adept at examining malware, since our team obtained practical experience with both static and dynamic analysis on three distinct malware samples. I remained on track with our team's agreed-upon project plan, as did other members of my team, making the process of working together feel seamless. The designated textbook readings were fascinating and an essential part of my learning process as well. Some valuable insights I garnered from the experience include setting up a secure virtual environment, deploying and using new malware analysis tools, simulating networks, running DLL files, learning to be comfortable with assembly language, conducting memory forensics, and becoming familiar with disassemblers and debuggers.

Our analysis of the three malware samples yielded interesting results. The first malware sample created a fake process that launches a keylogger to capture user keystrokes and saves

them in a file on the computer. The second malware sample was capable of several advanced techniques, including DLL process injection, token manipulation/impersonation, time stomping of files, registry manipulation, and networking capabilities using both FTP and HTTP protocols. It also had the ability to install itself as a persistent service, manipulate the graphical user interface (GUI), log information in a file named "*xinstall.log*", and potentially cover its tracks with a file. The capabilities of the second sample were fascinating!

The third malware sample covertly hijacked Internet Explorer to aid in its creation of a service named *Malservice*. The program also created a mutex to ensure a copy of itself remained running – but only one copy. In this way, the exploit achieved persistence and planned to wait undetected until January 1, 2100 to deploy its nefarious attack. On this fateful night, our team was able to verify that the malware planned to spring into action by deploying a loop repeating 20 times, creating *CreateThread* 20 times, making 20 HTTP calls to *www.malwareanalysisbook.com*, and continuing in this fashion indefinitely. The third exploit assumed a Denial-of-Service attack! Considering the potential reach of the malware, and its capacity to infect an infinite number of computers in the wild, I am struck by the profound scale of devastation a single attack could inflict on our interconnected world.

This project has also provided invaluable training in teamwork and collaboration. Working cooperatively with my team, alternating between different partners for static or dynamic analysis, has demonstrated the importance of adaptability and clear communication. The bi-weekly online Microsoft Teams meetings were an ideal platform for us to share ideas, help one another, and offer new perspectives on the malware sample analyses. Additionally, one of our team members, Mark, thoughtfully and thoroughly provided our team with exploit "walk-throughs" and insights based on his extensive experience with offensive security. Mark's contributions have significantly helped to solidify my understanding of new malware-related concepts. Thank you, Mark, for your generosity.

Lastly, working with assembly language offered a profound lesson. Prior to this project, I found assembly language intimidating and challenging. In the process of analysis, however, I gradually learned to better read and understand it, clarifying what once seemed like a jumble of instructions. In addition, using *Wireshark* to "see" network activity enriched my understanding of network communication and familiarity with what certain protocols look like in action. The Malware Analysis Project challenged me to think strategically, work collaboratively, and face difficult tasks with patience and determination. The lessons I have learned continue to resonate with me and will shape my approach to cybersecurity and teamwork moving forward.

Steven Hunt

Prior to this project, the extent of my malware analysis knowledge came from an early version of the "Defense Against the Dark Arts" class in this program plus the few utilities that are tested on in the CompTIA Security+ exam. As far as utilities, I got some more practice on *Process Monitor*, *Explorer*, *regedit* and *Wireshark* and was introduced to several, including *PeStudio*, *IDA*, *INetSim*, *Windbg* and *Reshacker*, among others. I also expanded my general knowledge of the typical strategies that are used in malware thanks to my experienced teammate, Mark, who was able to boil down the robust but verbose textbook to a few useful lessons during our regular meetings.

Notably, while most of my education on the topic was categorical and broad before, I became accustomed this term to recognizing certain patterns in Windows malware, such as recognizing which DLL imports are indicative of what capabilities. This, combined with

observing the operation of malware within a disassembler environment will be valuable to me whether I continue on with my stated goal of system administration or train into security.

As far as soft skills and miscellany, I was surprised a few times this term to discover the proverbially blistering pace that can be achieved by working in parallel or simultaneously on certain projects if labor can be divided that way. Of course, I've worked shoulder-to-shoulder with others on computing projects before, but their goals were a little more nebulous. In those cases, we all benefited from each other's knowledge but rarely worked together in a way that felt synergistic, and in this way I find that this group was very successful.

Mark Kaiser

On the malware analysis project I have had the opportunity to work with other security minded students. Prior to this project my experience drew from experience in other related cybersecurity fields. I have a background and currently work in the offensive security (penetration testing and red teaming) field. There is some overlap in knowledge and skills with that of a malware analyst. I also received some training in a malware analysis course in 2019 through the SANS organization. This project allowed me to refresh my knowledge on analysis tools I have used in the past while also learning about new analysis tools covered by the book and through our collective team research. I have enjoyed this project very much, especially leading brown bag sessions for some of the more advanced topics with the team. For example I have shared experience and taught how to set up our secure malware analysis environment, how to use certain advanced tools like *IDA* and *Ghidra* and most recently how to dump memory with *Dumpit* and analyze this memory with *Redline* and *Volatility*. I have contributed to all phases of the project, analysis, research, and reporting. I have contributed in both the static and dynamic phases of analysis. In the static phase I leveraged tools such as: *PeStudio*, *Strings*, *IDA*, *Ghidra*, and *Floss Flare*. In the dynamic phases of analysis I used tools such as *Process Explorer*, *Process Monitor*, *Wireshark*, and *Netstat*. In our first malware sample, I found that it performs process hollowing of a legitimate "*svchost.exe*" and has a range of local capabilities such as keylogging. During the analysis of the second sample I found a large range of capabilities. This sample included FTP and HTTP network capabilities, DLL process injection, persistence by installing itself as a service, file timestomping, and even a cleanup script to cover tracks. While analyzing the third malware sample I found that it installs itself as a service for persistence, performs process injection to hide, reaches out to malwareanalysisbook.com for instructions, checks the current system for *HGL345* mutex to not install repeatedly on the same system and executes actions via system APIs instead of shell commands for stealth purposes. Next I contributed to the team poster where we outlined the phases of analysis, highlighted our favorite tools with screenshots and short descriptions.

If there were any lessons to take away from this project we could have been more deliberate in our selection of malware samples to analyze from the book. Despite this, as a team we were very successful because we communicated well, planned our objectives and stuck to them, and divided up work in a way that allowed for parallelism where possible. Each person on the team brought different talents and abilities which complemented and enhanced each assignment we worked on together. . While I did my best to lead with experience and share what I have learned from my training and my industry experience, others on the team brought great creativity and attention to detail. It has been a pleasure working on this project throughout the term.

PROJECT CONTINUANCE

ACCOMPLISHMENTS

In short, three malware samples were heavily analyzed statically and dynamically in a distributed, virtualized environment. The first sample was a primitive keylogger, the second was a DLL that intentionally evaded analysis, and the third was a time-gated internet denial-of-service platform. Each of these samples was first studied in various ways that precluded execution, constituting static analysis, and then allowed to run in the extent that they were capable of for observation, constituting dynamic analysis. The team of four split into pairs for either task and kept central documents detailing their findings for future reference, such as during the sections that follow.

THE KEYLOGGER

In the static analysis of the keylogger, one of the first tasks for Michael and Jennifer was to take a hash of the malware and check to see if anyone had encountered it before using the popular website *VirusTotal*. This availed a threat label of *explorerhijack/ao9ui3p*, and indeed when the pair opened the malware executable in *PEID*, *PE Explorer* and *Floss* to observe the strings and imports listed with the executable's resources, among the Windows functions listed were those related to manipulation of files via "*kernel32.dll*" as well as a reference to svchost.exe. This inclusion would become understood during dynamic analysis.

Mark and Steven carried out the dynamic portion and used primarily *Process Explorer*, *Process Monitor* and *Wireshark* to observe what execution entailed. The fruit of this was the observation that the executable carried out a chain of registry and file edits that allowed it to leave an impostor "*svchost.exe*" behind in *Explorer*, whose function was to oversee the recording of keystrokes from the top window into a plain text file in the directory where the original executable was contained. Normally, some form of data extraction method is expected, but use of Wireshark and netstat confirmed that there was no attempt to send the logged data through the internet.

THE EVASIVE DLL

In the course of malware design, it's a common practice to include strategies to avoid detection in order to ensure the persistence or success of the malware. This particular malware package consisted of a dubious DLL packed with a Python script which was able to alter its contents being displayed in the popular disassembler, *IDA*. This core purpose of the sample led to the difficulties mentioned elsewhere in the final report, because it meant that almost all of the features were suited to a phenomenon that could technically be noticed during dynamic analysis, but only so far as influencing the results of what might be observed in a static environment. That is, the sample had no practical effect other than deception.

In terms of static analysis carried out by Mark and Steven, disassembly in *IDA* bore several typical red flags as far as functions imported from other DLLs. If the DLL were paired with a permitting executable or script, it was capable of process injection (an obfuscation method in itself), service creation and internet communication. The problem with confirming any of this was that the script included with the DLL used none of these functions, totally confounding analysis. When Michael and Jennifer went to search for any signs of these methods being used during dynamic analysis, they were hard pressed to come up with anything. They would have

had to have crafted a novel executable to take advantage of what the DLL was capable of, which goes beyond basic analysis.

THE DENIER OF SERVICE

The last sample analyzed was somewhat more substantive. Static analysis was done by Mark and Jennifer and consisted mainly of checking strings and imports in *Visual Studio* and *PeStudio* and the structure of the machine code in *IDA*. The former suggested the ability to covertly hijack Internet Explorer and open up a URL (imports included *CreateServiceA*, *InternetOpenUrlA*, and *Sleep*) and the latter revealed functions to create a service known as *MalService* which was responsible for persistence.

Dynamic analysis by Steven and Michael involved the usual *Process Monitor* and *Explorer*, but also gave itself at last to network monitoring with *INetSim*, *Wireshark*, and *TCPView*. These utilities revealed that the executable would make a DNS request to "case" the website for our textbook, would leave itself a backdoor to boot itself perpetually as a startup program in Windows, and then wait until a certain year to begin rapidly sending bursts of HTTP requests to the website through repeated attempts to load it in Internet Explorer. This was confirmed when Michael used the debugger *OllyDbg* to locate the memory value the executable used to define in what year the denial of service would occur, 2100, and subsequently changed it to 2023.

REMAINS TO BE ACCOMPLISHED

At this point, our malware analysis team has completed the tasks outlined in our agreed-upon Project Plan to the best of our abilities. However, there are a few important items remaining on our agenda. These items include the following: the analysis of a fourth sample of malware; the completion of the Project Archive – Final report; the recording of a Malware Analysis Project Demonstration video; and the presenting of a unified platform in the Career Showcase event. Our team is pleased to have completed our Project Poster and is actively finalizing the Project Archive - Final paper. This Project Archive-Final paper, along with the Project Poster and Demonstration Video, will serve as a comprehensive documentation of our project.

Despite our best efforts, a fourth malware sample remains to be analyzed. Several factors contributed to our difficulties completing the fourth analysis as planned. Initially, estimating the time required for each malware analysis proved challenging. Our team estimated we could tackle four malware samples, since we had four people in our group. In retrospect, it seems we bit off a bit more than we could chew. Next, the evaluation of Lab05_01.dll [1] exceeded our anticipated timeframe. In hindsight, this sample was specifically designed for educational static analysis, rendering our dynamic analysis attempts ineffective. Wrestling with the intricacies of this particular malware further impacted our timeline. Lastly, with the end of the term approaching, we collectively decided to prioritize our completed analyses, dedicating our efforts to creating a high-quality Career Showcase deliverable rather than attempting to squeeze in one more malware sample. Nevertheless, the team is confident our successfully analyzed samples have provided us with sufficient, valuable insights and a solid foundation for our project.

In addition to the ongoing completion of the Project Archive paper and the creation of the demonstration video, our team remains committed to maintaining regular communication. We will continue to meet bi-weekly to address any challenges, to provide support to team members,

and to create a quality finished product. Our team understands the importance of asynchronous communication and continues to actively engage with each other as needed. By prioritizing the completion of the final archive paper, diligently working on the demonstration video, and maintaining regular communication, we are confident that we will successfully wrap up our project. These remaining tasks allow us to showcase the knowledge and skills we have acquired, even though the completion of the fourth malware sample is not feasible within our given timeframe.

PROBLEMS AND WORKAROUNDS

Most of the malware that we studied was intended for either static or dynamic analysis, but not both. This is because the textbook separates the malware by the topic of the chapter, which was never both types of analysis together. There were many areas in the reading where it was described or implied that dynamic analysis can build off of static analysis, but the labs at the ends of the chapters that encompassed the malware that we looked at kept the static and dynamic separated and only focused on one for the chapter. This made our analysis a bit more difficult because the textbook clearly states how to apply one type of analysis to a given sample of malware, but we were on our own for the other type of analysis and found that it was not nearly as simple. The perfect example of this was in the second malware sample we looked at, which was a DLL file. The static analysis clearly exposed that the file has socket connection capabilities and can manipulate a network for communications. A dynamic analysis requires the ability to run the malware, and the nature of a DLL file is that it is simply a library that functions as an add-on for the source code in an executable that calls it. There are methods for running DLL files if they have enough standalone runnable code that does not require too much setup from the calling program, but we had no way of knowing whether that was the reason we could not run the malware, and the textbook did not cover dynamic analysis at all for that specific sample. As a team, we theorized many different ways to try to extract the information that would have been extracted by running the malware, including using memory dumps, but in the end we still needed to be able to run the malware for any of it to work. Since we were unable to figure out how to dynamically analyze the second sample, we used the opportunity to reason out what it could possibly mean, and we played around with more tools to learn as much as we could by the process. This was helpful in the long run for finding a path forward and theorizing some of the difficulties with how malware works.

The third sample that we looked at was a breath of fresh air. We were afraid that it was going to be similar to the second sample because it was yet again from a chapter of the textbook that focused on static analysis and needed to be analyzed dynamically. It started off a bit overwhelming because in our team meeting we discussed manipulating the assembly code to see if we can make the malware act differently, and there was *a lot* of assembly code and far too many places to manipulate things that finding the right piece of code to change seemed ridiculous. Initially, the dynamic analysis part of the team was a little stuck, however the static analysis team found a critical piece of code that decided whether the malware could run or not, completely based on time and date. Just by editing that one piece of assembly code, the malware was able to run and the dynamic analysis was made easy.

The takeaway from these two experiences is that when problems arise with an analysis it can result in a deeper learning experience, which can be used with future samples when similar problems arise. A sample of malware may or may not be able to run to be analyzed dynamically, and sometimes a static analysis may not be possible either because of something like encryption.

In situations like these skills can be further honed and developed just by the process of continuing to try, and if it still does not work out then the scientific method can be applied by theorizing and coming up with new conjectures based on those experiences, which may be useful for other applications.

NEXT PHASE PRIORITIES

If there were another phase for this project, our next phase priorities would center around analyzing advanced malware samples from the book. This would entail learning advanced analysis techniques and really digging into *IDA*, *Ghidra*, and *x64dbg* capabilities. We did not finish reading all of the chapters in the supplied book but the remaining chapters definitely cover more advanced topics. Additionally, there are entire books about using *IDA* and *Ghidra* and a substantial amount of time would be spent reading these books to make this level of analysis possible. The two books that I found would be most helpful are the *Ghidra* textbook “The Definitive Guide by Chris Eagle and Kara Nance” and the *IDA Pro* textbook “The Unofficial Guide to the World's Most Popular Disassembler” by Chris Eagle.

If time were not a limitation the team would want to consider creating our own malware analysis tools to speed up the analysis process and better understand how some of the existing analysis tools make analysis so much easier. Going a step further would be the study of writing our malware samples to understand how the code would look before compilation and then learning about obfuscation techniques employed by malware developers.

REFERENCES

[1] M. Sikorski and A. Honig, *Practical malware analysis : the hands-on guide to dissecting malicious software*. San Francisco No Starch Press, 2012.

[2] L. Zeltser, “Install from Scratch - REMnux Documentation,” *Remnux.org*, 2023. https://docs.remnux.org/install-distro/install-from-scratch (accessed May 10, 2023).