Oregon State University
College of Engineering

Malware Analysis

Sponsor: Bill Pfeil

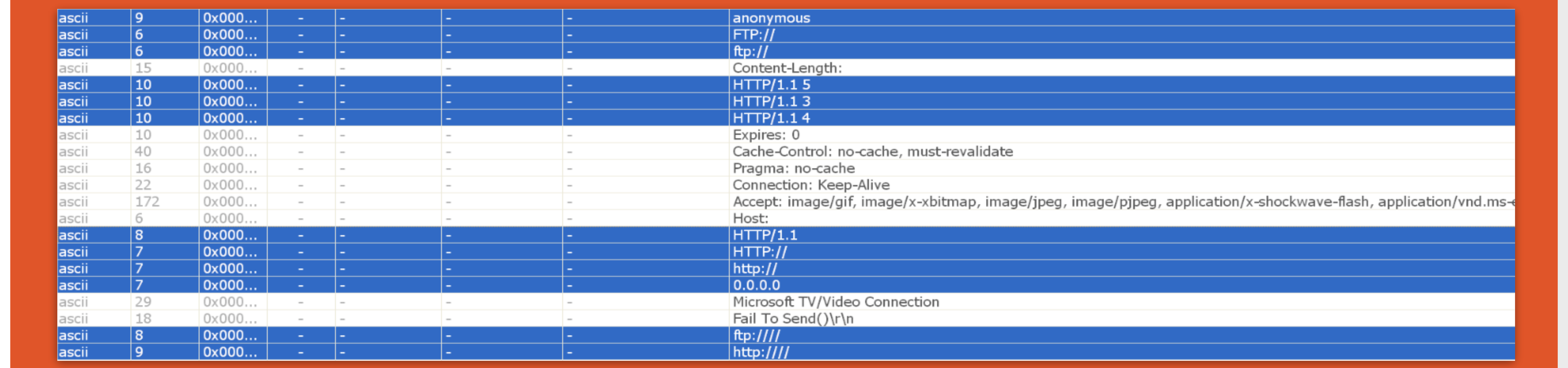
SPRING 2023

Michael Banks — crocshock911@comcast.net
Jennifer Bowers — bowerjen@lifetime.oregonstate.edu
Steven Hunt — huntste@oregonstate.edu
Mark Kaiser — kaisermar@oregonstate.edu

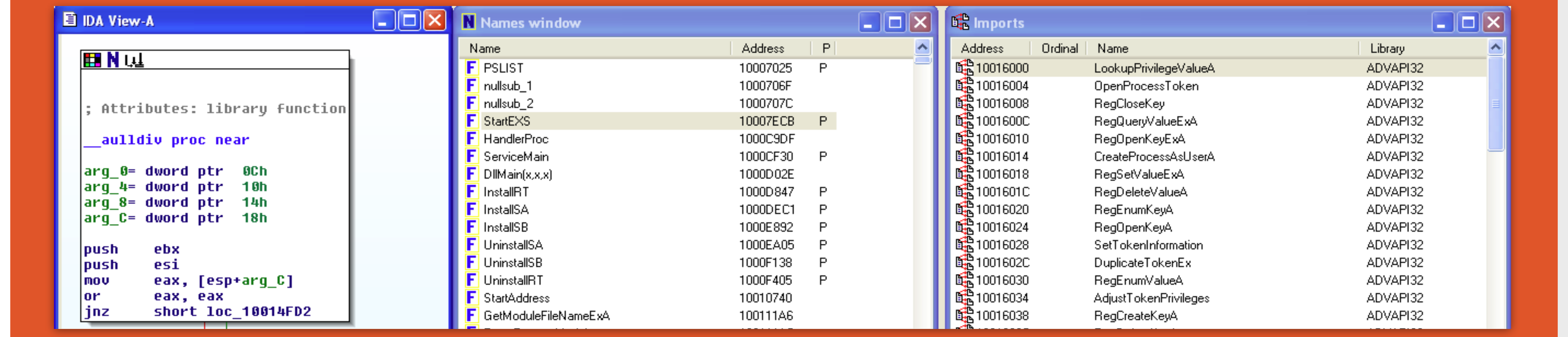
STATIC ANALYSIS

Static analysis examines the malware before execution and involves reading outputs of software tools. If the sample is a Windows portable executable, this could consist of using tools like *PEExplore*, *PEView* or *PeStudio*. Observations on which resources it imports or whether the executable is packed so as to obfuscate information are part of this consideration. Other static options include disassemblers such as *IDA*, which summarize how an executable or DLL runs in machine code, or multipurpose and decompilation tools like *Ghidra*.

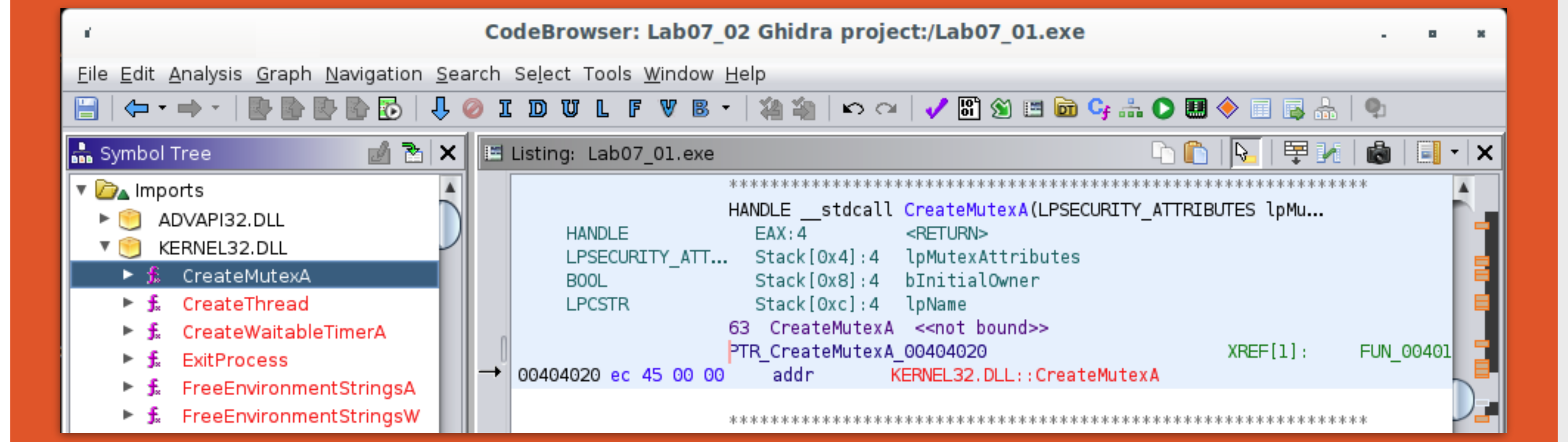
PeStudio is a multipurpose tool that detects functionality signatures for different types of malware and can detect abnormalities in imported files. Sometimes import purposes must be researched.
In this image, *PeStudio* is being used on our second malware sample to observe that that it is capable of making HTTP and FTP socket connections, likely to contact a hacker's server and send them private information about the host computer that will enable them easy access in the future: a backdoor.



IDA is popular and powerful disassembler/debugger software that provides users with the capability to analyze and understand the low-level assembly code of executable files, allowing for reverse-engineering and inspection of the inner workings of malware.
In this image, *IDA* is being used on our second malware sample to observe its assembly code, names of internal functions, and import functions. With this information alone we were able to see that the malware installs and uninstalls services and makes changes to the registry.



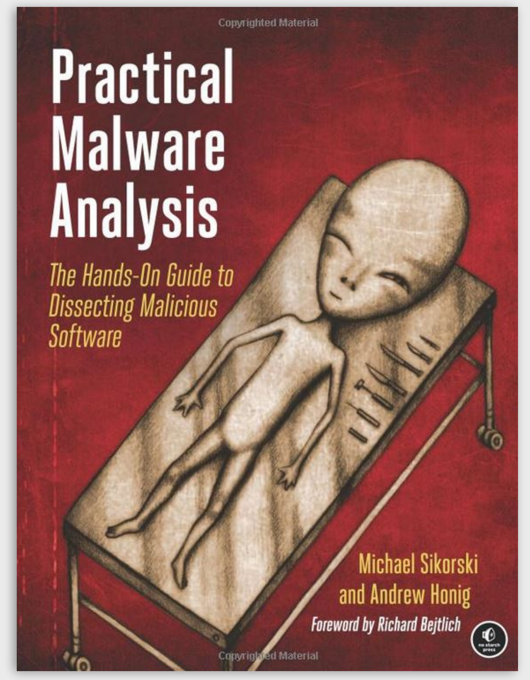
Ghidra is a multipurpose tool which can decompile a binary program and provide insight into the code pages, source code, and code flow for a malware sample.
In this image, *Ghidra* is being used on our third malware sample to observe creation of a mutex in the assembly code. The mutex allows the malware to know if it already exists on the target system.



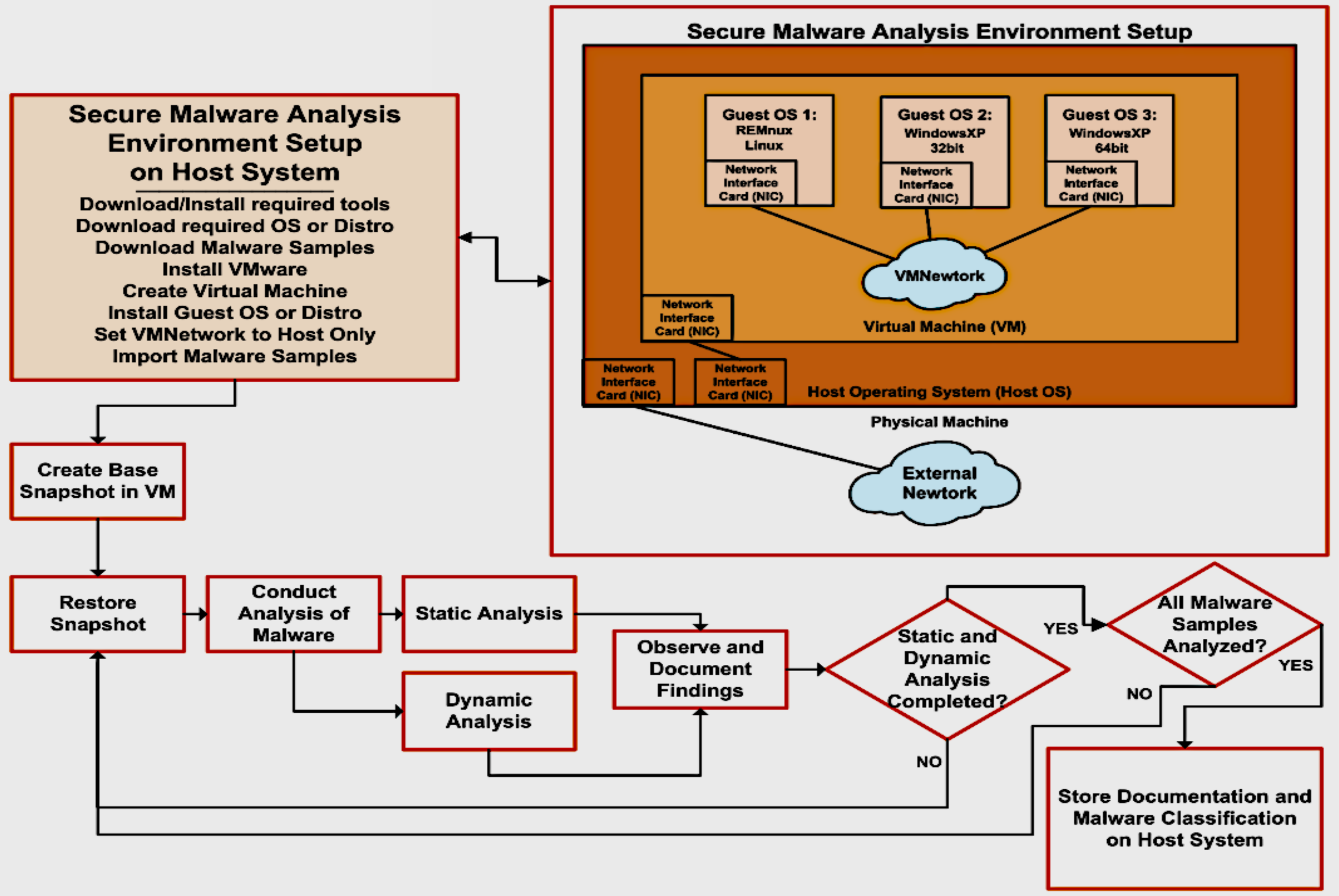
ABOUT THE PROJECT

Malware analysis consists of two types of analysis–static and dynamic–each presenting its own challenges. The team constructed a specialized malware analysis lab environment using the VMware hypervisor for virtualization and integrated the Windows XP and REMnux operating systems to establish a secure and versatile lab setting. For an added layer of security, we engineered the environments around a host-only network configuration, simulating network connections via INetSim and FakeDNS. To build a comprehensive toolkit, we installed and deployed sophisticated malware analysis software tools including Windows Sysinternals Suite, IDA, PeStudio, Ghidra and OllyDbg.

We tackled the analyses by dividing the team into two pairs, analyzing a set of malware samples sourced from the "Practical Malware Analysis" textbook. Each pair performed comprehensive investigations of three unique samples, systematically alternating between static and dynamic analyses and collecting our findings into documents to facilitate a broad understanding. The textbook, combined with our individual experiences, provided crucial insight into the correct usage, optimal setup, and appropriate application of malware analysis tools. During our bi-weekly virtual meetings, we engaged in thoughtful, detailed walkthroughs of malware analysis tools and shared our documented findings to further deepen our collective understanding.



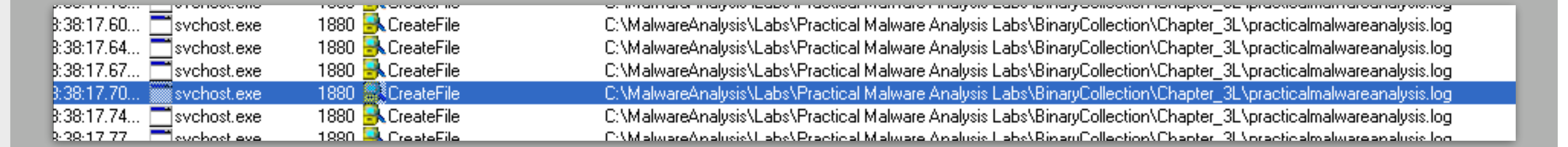
Malware Analysis Structural Diagram



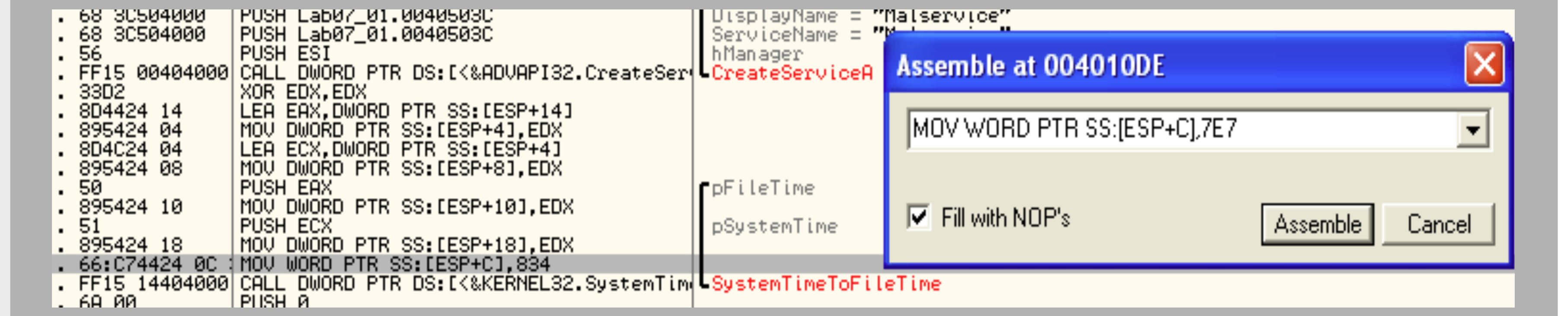
DYNAMIC ANALYSIS

Dynamic analysis involves executing malware, which requires containerization and knowledge of assembly code. Software from Windows Sysinternals such as *Process Monitor* are used to observe actions taken such as registry entries, file creations and network connections by the malware. Network analysis tools such as *Wireshark* or *Netstat* monitor network activity in great detail. Debuggers such as *OllyDbg* and *WinDbg* allow for modifying assembly code. With these tools put together malware activity can be closely monitored or forced into execution.

Process Monitor observes the files and registry entries as they are created, changed, or removed, and observes persistence mechanisms of malware such as service creation, scheduled tasks and auto runs.
In this image, *Process Monitor* is being used on our first malware sample to observeope's create and modify a log file called *practicalmalwareanalysis.log*, which was evidence for the malware being a keylogger. We were able to seek out and find the log file using its path and observed that while the malware runs each key press by the current user was logged. In practice this file could contain passwords and be automatically sent over a network connection to an adversary.



OllyDbg is a disassembler/debugger designed for analysis of x86 Windows executables (.exe), dynamic-link libraries (.dll), and other binary files that supports both static and dynamic analysis, allowing for analysis of a program's behavior in real-time. In this image, *OllyDbg* is being used on our third malware sample to find assembly code that specifies a time that the malware will run. The year it was set to run was 834 in hexadecimal, which translates to the year 2100. By changing this value to 7E7 were were able to run the malware in the year 2023. Note also that the XOR operation is used to zero out the EDX register before being moved into other time variables on the stack like day, month, and minutes. The malware was set to detonate on January 1st at midnight.



Wireshark aids in visibility of network activity to include the protocol, remote host, and possibly clear text communications between host and server.
In this image, *Wireshark* is being used on our third malware sample to identify an HTTP GET request to *www.practicalmalwareanalysis.com* performed by the malware. These GET requests were being sent really fast on an infinite loop: a Denial of Service (DoS) attack.