FOR710: **Reverse-Engineering Malware: Advanced Code Analysis**

5 Day Program | 36 CPEs | Laptop Required

Course Topics

- Code deobfuscation
- Program execution
- Shellcode analysis
- Steganography
- Multi-stage malware
- WinDbg Preview
- Encryption algorithms
- Python scripting for malware analysis
- Dynamic Binary Instrumentation (DBI) Frameworks
- Payload and config extraction
- Scripting with Ghidra
- Malware correlation
- YARA rules
- Capa rules

What You Will Receive

- Windows 10 VM with pre-installed malware analysis and reversing tools
- Real-world malware samples to examine during and after class
- Coursebooks and workbook with detailed step-by-step exercise instruction

As defenders hone their analysis skills and automated malware detection capabilities improve, malware authors have worked harder to achieve execution within the enterprise. The result is modular malware with multiple layers of obfuscation that executes in-memory to hinder detection and analysis. Malware analysts must be prepared to tackle these advanced capabilities and use automation whenever possible to handle the volume, variety and complexity of the steady stream of malware targeting the enterprise.

FOR710: Advanced Code Analysis continues where FOR610: Reverse-Engineering Malware: Malware Analysis Tools and Techniques course leaves off, helping students who have already attained intermediate-level malware analysis capabilities take their reversing skills to the next level. Authored by SANS Certified Instructor Anuj Soni, this course prepares malware specialists to dissect sophisticated Windows executables, such as those that dominate the headlines and preoccupy incident response teams across the globe.

Developing deep reverse-engineering skills requires consistent practice. This course not only includes the necessary background and instructor-led walk throughs, but also provides students with numerous opportunities to tackle real-world reverse engineering scenarios during class.

FOR710 Advanced Code Analysis will prepare you to:

- Tackle code obfuscation techniques that hinder static code analysis, including the use of steganography
- Identify the key components of program execution to analyze multi-stage malware in memory
- Identify and extract shellcode during program execution
- Develop comfort with non-binary formats during malware analysis
- Probe the structures and fields associated with a PE header
- Use WinDBG Preview for debugging and assessing key process data structures in memory
- Identify encryption algorithms in ransomware used for file encryption and key protection
- Recognize Windows APIs that facilitate encryption and articulate their purpose
- Create Python scripts to automate data extraction
- Use Dynamic Binary Instrumentation (DBI) frameworks to automate common reverse engineering workflows
- Write scripts within Ghidra to expedite code analysis
- Correlate malware samples to identify similarities and differences between malicious binaries and track the evolution of variants
- Build rules to identify, group and classify malware

"As malware gets more complicated, malware analysis has as well. In recent years, malware authors have accelerated their production of dangerous, undetected code using creative evasion techniques, robust algorithms, and iterative development to improve upon weaknesses. Proficient reverse engineers must perform in-depth code analysis and employ automation to peel back the layers of code, characterize high-risk functionality and extract obfuscated indicators."

—Anuj Soni, **Course Author**

Section Descriptions

SECTION 1: Code Deobfuscation and Execution

Malware authors complicate execution and obfuscate code to hide data, obscure code, and hinder analysis. Using evasion techniques and in-memory execution, malicious developers continue to thwart detection and complicate reverse engineering efforts. To facilitate an in-depth discussion of code deobfuscation and execution, this section first discusses the creative use of steganography to hide malicious content. Then, we discuss the key steps in program execution, so we can identify how code is launched and label functions accordingly. This includes a review of the Windows loader and an inspection of the Portable Executable (PE) file format. Finally, we cover how to analyze shellcode with the support of WinDbg Preview, a powerful Windows debugger.

TOPICS: Analyzing Code Deobfuscation; Identifying Program Execution; Understanding Shellcode Execution

SECTION 2: Encryption in Malware

This section tackles a critical area of reverse-engineering malware: the use of encryption in malware. Cryptography is used by adversaries for a variety of reasons, including to encrypt files, protect keys, conceal configuration settings, and obfuscate command and control (C2) communications. To perform comprehensive investigations of high-impact malware, skillful reverse engineers must be prepared to investigate routines that implement encryption and articulate their purpose.

TOPICS: Encryption Essentials; File Encryption and Key Protection; Data Encryption in Malware

SECTION 3: Automating Malware Analysis

In this section, we discuss how to write scripts to automate our analysis. We introduce key aspects of Python scripting and write code to automate some of our work from prior sections. Next, we introduce Dynamic Binary Instrumentation (DBI) Frameworks and examine how DBI tools can complement and automate common reverse engineering workflows. We apply our knowledge of Python to automatically extract payloads and configs, accelerate debugging efforts, and support static code analysis with Ghidra.

TOPICS: Python for Malware Analysis; Malware Analysis with Dynamic Binary Instrumentation (DBI) Frameworks; Automating Analysis within Ghidra

SECTION 4: Correlating Malware and Building Rules

Correlational analysis helps identify similarities and differences between malware samples. This provides insight into code reuse and facilitates the creation of YARA and capa rules, allowing an organization to track malware families. Correlation analysis includes straightforward hash comparisons as well as more complex attempts to pinpoint function-level differences. We discuss several approaches to diffing binaries and assess their benefits and limitations.

TOPICS: Correlation Analysis; Building YARA Rules; Building capa Rules

SECTION 5: Advanced Malware Analysis Tournament (Exyended Access)

The final section of this course gives students an opportunity to flex their new knowledge and skills in a more independent, competitive environment. Participants will have extended access (beyond a 5-day live class) to a capture the flag (CTF) platform, where they will attempt a combination of multiple choice and short-answer challenges. Students must recall key concepts and perform workflows discussed in class to successfully navigate the tournament and accumulate points. Whether or not competition motivates you, this section presents an excellent opportunity to analyze real-world, complex malware samples and reinforce your new advanced code analysis skills.

"I was recently named our IR lead, and coming from purple teaming/pentesting I needed the content of this course to make meaningful improvements to the program. I feel well prepared to tackle the challenges ahead now."

—Ryan M.

Who Should Attend

- **Information security professionals** who want to improve upon their intermediate-level reverse-engineering skills.
- **Reverse engineers** who need to improve their abilities to analyze obfuscated code, assess encryption capabilities in malware, automate analysis tasks, and generate effective detection rules.