SYLLABUS

ADVANCED REVERSE ENGINEERING OF SOFTWARE

VERSION 1.2

The most practical and comprehensive training course on reverse engineering

eLearnSecurity
Forging security professionals

eLearnSecurity has been chosen by students in over 140 countries in the world and by leading organizations such as:

hp Microsoft intel cisco AT&T verizon U.S. AIR FORCE pwc gemalto

COURSE DESCRIPTION

This fundamental self-study course teaches you the theoretical and practical knowledge required to perform advanced reverse engineering of third-party software and malware on the assembly language level.

Through a series of lessons, and several challenges, you will be taught all the necessary skills to succeed as a professional, and not just acquire a superficial understanding of how to use reversing tools.

This training is based on Windows NT architecture (XP, Vista, 7, 8), since malware & vulnerability researchers as well as software pirates still typically target these operating systems.

During your advanced reverse engineering training, you will learn several methods used to identify, isolate, and finally, analyze portions of code which are of high interest. You will also learn about the most common Windows APIs utilized for file, memory and registry manipulation by either software protections (such as packers) or malware.

Additionally, the training focuses on several packers in order to give the student all the essential knowledge and understanding of manual unpacking software. This is one of the most important parts of advanced reverse engineering.

On top of all these exciting topics, you will also get insights into the most common anti-reversing tricks, including different code obfuscation methods. Not only will you analyze their mechanisms, but also learn how these can be bypassed in order to successfully perform the reverse engineering process.

WHO SHOULD TAKE THIS COURSE?

This advanced reverse engineering training course is highly practical, meaning you will learn things by yourself and not just listen to some instructors and watch videos. If you like the "learning-by-doing" approach, then this is for you. This is NOT a "learn–repeat–forget"-type of training. The course's guidance ensures that you will get all the necessary knowledge along the way.

The Advanced Reverse Engineering of Software training course provides the foundation for current or future malware researchers. If you are involved software

development, you will benefit from learning how pirates attempt to bypass your protection. In turn, you will be able to create more sophisticated and smarter ways to keep pirates away, as efficiently as possible.

This course definitely benefits you if you are a:

- penetration tester
- security analyst
- antivirus researcher
- software developer
- software tester
- malware researcher
- government IT staff
- computer forensics expert
- IT security expert
- mobile application developer
- game developer
- incident response team member
- vulnerability researcher
- web application security expert

Since reverse engineering is based on the complete understanding of computing architecture, this course serves as a great foundation for everyone working in IT positions. With this foundation, you will understand even the most complex IT topics easier.

WHO SHOULD NOT TAKE THIS COURSE?

If you are looking to quickly memorize some theory which you can dump out on paper during an exam to get another certificate, this course is NOT for you. If you are simply looking for user-manuals of reverse engineering tools in course format, then you won't be happy with this highly interactive training course either.

HOW AM I GOING TO LEARN?

The fun way of course!!!

Don't worry, eLearnSecurity courses are very interactive and addictive, and presents content in such a way that it appeals to all learning styles. During this training, you will have to deal with several guided reversing challenges, that will provide you with relevant and hands-on practical application experience. Don't expect the outdated way of learning by reading pages and pages of theoretical methodologies.

CAN I TRACK MY LEARNING PROGRESS?

…or will I only find out during the exam if I learned something?

The answer to these questions is very simple. Your achievements will tell. Each practical chapter of the course has some cool reversing challenges associated with it. We will solve these together, while we explain to you all the necessary concepts. You are then free to practice the labs as long as you want. If you can solve a challenge, that demonstrates that you learned and properly understood the concepts.

IS THERE A FINAL EXAMINATION?

Yes. The final examination consists of two parts. The first part is a multiple-choice quiz test. Once you have passed this, you will proceed with the hands-on examination. During this second part of your exam, you will have to solve a complex Reverse Engineering Challenge.

WILL I GET A CERTIFICATE?

Once you passed the complete final examination, you are an "eLearnSecurity Certified Reverse Engineer" and will hold the eCRE certification.

You can print your shiny new certificate directly or have it shipped to you internationally.

ORGANIZATION OF CONTENTS

The student is provided with a suggested learning path to ensure the maximum success rate and the minimum effort.

THEORY PART

- Module 1: The Necessary Theory Part 1/3
- Module 2: The Necessary Theory Part 2/3
- Module 3: The Necessary Theory Part 3/3
- Module 4: VA/RVA/OFFSET & PE File Format

TECHNICAL PART

All the following chapters include practical challenges, which we discuss in the written part and/or during the video demos:

- Module 5: String References & Basic Patching
- Module 6: Exploring the Stack
- Module 7: Algorithm Reversing
- Module 8: Windows Registry Manipulation
- Module 9: File Manipulation
- Module 10: Anti-Reversing Part I
- Module 11: Anti-Reversing Part II
- Module 12: Anti-Reversing Part III
- Module 13: Code Obfuscation
- Module 14: Analyzing Packers & Manual Unpacking
- Module 15: Debugging Multi-Thread Applications

MODULE 1: THE NECESSARY THEORY PART 1 of 3

The first three modules aim to cover all the necessary theory as well as the concepts in which the practical part of this course is based.

We will start with a short description of what Reverse Engineering is and the reasons why someone might need it, and then we'll proceed with more technical concepts.

During the first three chapters, we will discuss the basics behind the Intel IA-32 CPU architecture (x86), the stack, the heaps, as well as exceptions, Windows APIs with some Windows Internals, and the most common types of reversing tools used these days.

MODULE 2: THE NECESSARY THEORY PART 2 of 3

The second module is also dedicated to the theoretical knowledge necessary for this course.

What you always need to keep in mind during this course, is that 'theoretical' doesn't actually mean that you might need it…or not.

In fact, the theory discussed during these first three chapters covers all the fundamental knowledge and the concepts that you will need, not just for this course and its technical assignments, but for the rest of your time as a reverser.

MODULE 3: THE NECESSARY THEORY PART 3/3

The third theoretical module of this will briefly touch on the concept of heaps, as well as discuss handles, exceptions, some basic Windows Ring3 Internal structures, and review Windows APIs.

Finally, we'll go through the most common types of reversing tools used today for software reverse engineering.

MODULE 4: VA/RVA/OFFSET & PE FILE FORMAT

In this chapter, we will discuss virtual addresses, relative virtual addresses, offsets, as well as some basic information regarding the Portable Executable File Format which describes the basic structure of all Windows executable files.

MODULE 5: STRING REFERENCES & BASIC PATCHING

This chapter is dedicated to 'String References' as well as Basic Memory and File Patching.

We demonstrate the use of data strings in order to locate the algorithm we are interested in, and then we reverse its logic.

Finally, we explain how we can manually calculate the offset of a byte inside the physical file by knowing its virtual address in memory.

Challenge and video included in this module

5. String References & Basic Patching
- **5.1. Introduction**
- **5.2. String References**
- **5.3. A few words before starting**
- **5.4. String References & Basic Patching**
  - 5.4.1. Run the target executable and observe its functionalities.
  - 5.4.2. Load the executable to Olly Debugger
  - 5.4.3. Search for string references
  - 5.4.4. Reversing the logic
  - 5.4.5. Basic Memory Patching
  - 5.4.6. Executable Patching through Olly
  - 5.4.7. VA -> OFFSET manual calculation
  - 5.4.8. Manual Byte Patching
  - **Conclusion**

MODULE 6: EXPLORING THE STACK

This chapter focuses on exploring the data that we can retrieve from the stack in order to trace back an algorithm. This is a very important technique when dealing with on-the-fly encryption and decryption of data.

Challenge and video included in this module

6. Exploring the Stack
- **6.1. Introduction**
- **6.2. A few words before starting**

6.3. Exploring the Stack
- 6.3.1. Run and Observe
- 6.3.2. Load to Olly and search for strings
- 6.3.3. How is this possible?!?!
- 6.3.4. Exploring the stack
- 6.3.5. Evaluating the MessageBox API parameters
- 6.3.6. Reversing the logic
- 6.3.7. Patching the code

Conclusion

MODULE 7: ALGORITHM REVERSING

During this module, we dig deeper into Reverse Engineering by analyzing, in detail, all the important algorithms of the executable which include the data encryption/decryption algorithm, as well as the input data validation algorithm.

Challenge and video included in this module

7. Algorithm Reversing
- **7.1. Introduction**
- **7.2. A few words before starting**
- **7.3. Algorithm Reversing**
  - 7.3.1. Important algorithms
    - String Decryption/Encryption
      - Call Stack Window
      - Calls to decrypt/encrypt string function
      - Setting SW BPs
      - Pushing parameters to decrypt/encrypt string function
      - Parameter value
    - Code Validation
    - Custom Exception Handler
  - **Conclusion**

MODULE 8: WINDOWS REGISTRY MANIPULATION

This module is dedicated to Windows Registry. We start with an overview of this important Windows component and then we proceed with the detailed analysis of an executable that attempts to read data from the registry and validate it according to a custom algorithm which we finally Reverse Engineer.

Furthermore, during this chapter, we also make use of Hardware Breakpoints, and we demonstrate their importance.

Challenge and video included in this module

8. Windows registry Manipulation
- **8.1. Introduction**
- **8.2. Windows Registry**
- **8.3. A few words before starting**
- **8.4. Windows Registry Manipulation**
  - 8.4.1. Retrieving data from Windows Registry
  - 8.4.2. Using Hardware Breakpoints
  - 8.4.3. Algorithm analysis
  - 8.4.4. Reversing the logic
  - **Conclusion**
  - **References**

MODULE 9: FILE MANIPULATION

During this module, we reverse engineer an executable that attempts to locate a specific file in the system and read data from it.

Once again, we analyze, in detail, the custom algorithm used to validate that data in order to extend our skills in reverse engineering custom algorithms.

Challenge and video included in this module

9. File Manipulation
- **9.1. Introduction**
- **9.2. A few words before starting**
- **9.3. File Manipulation**
  - 9.3.1. Getting a Handle
  - 9.3.2. What do we know by now?
  - 9.3.3. Reading the file contents
  - 9.3.4. Algorithm Analysis
    - Read Buffer Contents
  - **Conclusion**

MODULE 10: ANTI-REVERSING TRICKS PART I

This is the first module dedicated to anti-reversing tricks which includes some basic direct and indirect ways to detect a Ring3 debugger.

Challenge and video included in this module

10. Anti-Reversing Tricks Part I
- **10.1. Introduction**
- **10.2. Categories of Anti-Reversing tricks**
- **10.3. A few words before starting**
- **10.4. Anti-Reversing Tricks Part I**
- **10.5. PART I**
  - 10.5.1. Direct Debugger Detection
    - PEB.BeginDebugged
    - PEB.NtGlobalFlag
    - CheckRemoteDebuggerPresent
  - 10.5.2. Indirect Debugger Detection
    - OutputDebugString
    - OpenProcess
  - 10.5.3. Window Debugger Detection
  - **Not-a-conclusion…**

MODULE 11: ANTI-REVERSING TRICKS PART II

In this module, we continue talking about anti-reversing tricks regarding debuggers and reversing tools detection methods.

Challenge and video included in this module

11. Anti-Reversing Tricks Part II
- **11.1. Introduction**
- **11.2. Anti-Reversing Tricks Part II**
- **11.3. PART II**
  - 11.3.1. Process Debugger Detection
    - CreateToolhelp32Snapshot
    - Process32First
    - Process32Next
  - 11.3.2. Parent Process Detection

11.3.3. Module Debugger Detection
CreateToolhelp32Snapshot
Process32First
Process32Next
11.3.4. Code Execution Time Detection
Read Time-Stamp Counter
GetTickCount
Not-a-conclusion…

MODULE 12: ANTI-REVERSING TRICKS PART III

This module also focuses on anti-reversing tricks. In this case, we discuss differences between SW and HW breakpoints and how these can be detected. We also talk about more advanced tricks that involve the use of exceptions, and finally, we talk about some well-known methods for detecting a few popular VM environments.

Challenge and video included in this module

12. Anti-Reversing Tricks Part III
12.1. Introduction
12.2. Anti-Reversing Tricks Part III
12.3. PART III
12.3.1. Software vs. Hardware Breakpoints
12.3.2. Software Breakpoint Detection
12.3.3. Hardware Breakpoint Detection
Most common way
A more obscure way
12.3.4. Ring0 Debuggers & System Monitoring Tools Detection
12.3.5. Structured Exception Handling (SEH)
12.3.6. Unhandled Exception Filter
12.3.7. VM Detection
VMware Detection
VirtualPC Detection
VirtualBox Detection
Conclusion

MODULE 13: CODE OBFUSCATION

In this module, we discuss different types of native code obfuscation methods. We explain how these are implemented, what obstacles are created, and how we can analyze and cleanup obfuscated code.

Challenge and video included in this module

13. Code Obfuscation
- **13.1. Introduction**
- **13.2. Logic flow obfuscation**
- **13.3. 'NOP' Obfuscation**
- **13.4. Anti-Disassembler Code Obfuscation**
- **13.5. Trampolines**
- **13.6. Instruction permutations**
  - 13.6.1. xor reg, reg
  - 13.6.2. add reg, reg1
  - 13.6.3. mov reg, reg1
  - 13.6.4. jump address
  - **Conclusion**

MODULE 14: ANALYZING PACKERS & MANUAL UNPACKING

This chapter focuses on executables packers, and more specifically, on different generic methods that we can use in order to successfully find the Original Entry Point of applications packed with common packers.

We give practical examples, and we unpack them together for fun and knowledge.

Challenge and video included in this module

14. Analyzing Packers & Manual Unpacking
- **14.1. Introduction**
- **14.2. A few words before starting**
- **14.3. Analyzing Packers & Manual Unpacking**
  - 14.3.1. Well-known EntryPoints
    - Microsoft Visual C++ 6.0
    - Microsoft Visual C++ 7.x

Microsoft Visual C++ 8.0-9.0
Borland C++ Builder
Dev C++ 4.9.9.2
Borland Delphi 6.0-7.0
Microsoft Visual Basic 5.0-6.0
MASM32/TASM32

14.3.2. Methods to reach the OEP
14.3.3. Packers & tools used
Conclusion

MODULE 15: DEBUGGING MULTI-THREAD APPLICATIONS

In this chapter, we discuss the debugging and the analysis of multi-thread applications, or in other words, of applications that can execute various blocks of code via different threads.

Reverse engineering multi-thread applications can sometimes be quite frustrating, especially for beginners.

Challenge and video included in this module

15. Debugging Multi-Thread Applications
15.1. Introduction
15.2. Multi-Threading in practice
15.3. Creating a new Thread
15.4. Threads Synchronization
15.5. Threads Manipulation
15.6. Debugging Multi-Thread Applications
Conclusion

ABOUT US

We are eLearnSecurity.

Based in Santa Clara, California, with offices in Pisa, Italy, and Dubai, UAE, Caendra Inc. is a trusted source of IT security skills for IT professionals and corporations of all sizes. Caendra Inc. is the Silicon Valley-based company behind the eLearnSecurity brand.

eLearnSecurity has proven to be a leading innovator in the field of practical security training, with best of breed virtualization technology, in-house projects such as Coliseum Web Application Security Framework and Hera Network Security Lab, which has changed the way students learn and practice new skills.

Contact details:

www.elearnsecurity.com
contactus@elearnsecurity.com

2040 Martin Ave.
Santa Clara, CA, USA

Via Matteucci 36/38
Pisa, Italy

Apricot Tower, Dubai Silicon Oasis
Dubai, UAE