VIRGINIA TECH · DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

Modern Binary Exploitation

ECE 5984 · Special Study

3H, 3C | Fall | Blacksburg & Virtual

Instructor: Prof. Binoy Ravindran · binoy@vt.edu

COURSE DESCRIPTION

Exploitation of computer software vulnerabilities to create malicious system behaviors or leak sensitive data is pervasive across a wide range of application domains and computer form factors. Central to crafting effective security defenses is understanding what the common classes of vulnerabilities are, how they are defended in modern systems, how those defenses can be bypassed by writing exploits, and gaining practical experience in programming such exploits.

The course introduces basic reverse engineering for the x86 computer architecture, vulnerability analysis, and classical forms of binary exploitation, with an emphasis on developing practical offensive security skills. Vulnerabilities are introduced in increasing complexity — ranging from memory corruption to shellcoding to return-oriented programming to static memory layout- and dynamic memory-based corruption to race conditions — and their defenses are introduced in increasing difficulty for exploit developers to bypass by programming exploits.

CATALOG DESCRIPTION

Overview of reverse engineering, vulnerability analysis, and classical forms of binary exploitation. Essential concepts of x86-64 assembly and assembly-level debugging. Binary exploitation through buffer overflows and stack-based memory corruption. Shellcoding, including common types of shellcode payloads and tailoring shellcode to fit input constraints. Stack cookies; detection of stack-based buffer overflows and analysis of scenarios for bypassing stack-cookie-based mitigations. Data Execution Prevention and prevention of code injection attacks including shellcoding. Return-Oriented Programming and bypassing Data Execution Prevention. Overview of Address Space Layout Randomization and bypassing techniques. Dynamic memory and exploitation of Use-After-Free vulnerabilities. Race conditions and exploitation of racy behaviors in shared-memory concurrent code. **Pre:** Graduate standing. (3H, 3C)

COURSE LEARNING OBJECTIVES

Having successfully completed this course, the student will be able to:

- **Analyze** and debug x86-64 assembly code.
- **Implement** stack-based memory corruption exploits.
- **Implement** shellcode payloads and tailor shellcode to fit input constraints.

- **Analyze** stack cookies and scenarios for bypassing stack cookies.
- **Implement** Return-Oriented Programming gadgets to bypass Data Execution Prevention.
- **Implement** techniques for bypassing Address Space Layout Randomization.
- **Implement** Use-After-Free heap exploits.
- **Implement** race-condition exploits for multithreaded code.

PREREQUISITES

Graduate standing. Strong knowledge of C is assumed.

REQUIRED TEXT & COURSE MATERIALS

Required text

None.

Interactive lectures and hands-on laboratory exercises

The course uses Ret2 Systems's "WarGames" platform (wargames.ret2.systems), which is intended to teach security concepts in a safe and educational environment. Students enrolled in the course have access to the WarGames platform at no cost.

Recommended texts

- Erickson, J. (2008). *Hacking: The Art of Exploitation.* No Starch Press, 488 pp. ISBN-13: 978-1593271442.
- Anley, C., Heasman, J., Lindner, F., & Richarte, G. (2007). *The Shellcoder's Handbook: Discovering and Exploiting Security Holes.* Wiley, 752 pp. ISBN-13: 978-0470080238.
- Yurichev, D. (2013). *Reverse Engineering for Beginners.* beginners.re

GRADING (TENTATIVE)

COMPONENT	WEIGHT	BASIS
Labs	100%	9 lab exercises equally weighted at 11.11% of the final grade. Each lab consists of three progression challenges: Level 1 (easy), Level 2 (medium), and Level 3 (hard). Lab exercises are individual.

COURSE TOPIC OUTLINE (TENTATIVE)

#	DESCRIPTION	% OF COURSE
1	Essential concepts: Linux, C, x86 assembly, ethical/legal aspects of reverse engineering and hacking.	10%
2	Introduction to reverse engineering: memory layouts, control-flow graphs (CFGs), tools.	10%
3	Introduction to memory corruption: Executable and Linking Format (ELF), the stack, calling conventions, buffer overflows.	10%
4	Shellcoding: writing shellcode, code injection, scenario-relevant payloads.	10%
5	Stack cookies: mitigation against overflows, techniques for bypassing stack cookies.	10%
6	Data Execution Prevention (DEP) and Return-Oriented Programming (ROP): overview of DEP, writing ROP gadgets, return-to-libc (ret2libc).	10%
7	Address Space Layout Randomization (ASLR) and memory leaks: overview, information leaks, partial overwrites, ASLR closure.	10%
8	Heap exploitation: heap structure and concepts, corruption, use-after-free.	10%
9	Race conditions: introduction to process timing, pipelining.	10%
10	Advanced topics (e.g., automated exploit generation).	10%