CSE 410/510 Special Topics: Software Security

Dates: 1/31/2022 - 5/11/2022
Time: Monday, Wednesday 5:00 PM - 6:20 PM

Instructor: Dr. Ziming Zhao
Email: zimingzh@buffalo.edu
Office Hours: Wednesday 3:30 PM - 4:30 PM or by appointment
You will need to log in to UB Zoom before you join the office hours at
`https://buffalo.zoom.us/j/95299258797?pwd=QlBhbjJIUlM5WmlETmFtOE5qT1Z5dz09`

TA: Md. Armanuzzaman Tomal
Email: mdarmanu@buffalo.edu
Office Hours: Friday 3:30 PM - 4:30 PM or by appointment
You will need to log in to UB Zoom before you join the office hours at
`https://buffalo.zoom.us/j/95299258797?pwd=QlBhbjJIUlM5WmlETmFtOE5qT1Z5dz09`

1   Course Description

This course is designed to provide students with good understanding of the theories, principles, techniques and tools used for software and system hacking and hardening. Students will study, in-depth, binary reverse engineering, vulnerability classes, vulnerability analysis, exploit and shellcode development, defensive solutions, etc. to understand how to crack and protect native software. In particular, this class covers offensive techniques including stack-based buffer overflow, heap security, format string vulnerability, return-oriented programming, etc. This class also covers defensive techniques including canary, shadow stack, address space layout randomization, control-flow integrity, etc. A key part of studying security is putting skills to the test in practice. Hacking challenges known as Capture The Flag (CTF) competitions are a great way to do this. In this class the progress of students are evaluated by lab assignment and in-class Capture-The-Flag (CTF) competitions. The course can be used to satisfy the MS project requirement.

2   Student Learning Outcomes

1. Identify the aforementioned types of software vulnerabilities at source code and binary code level

2. Exploit the discovered software vulnerabilities

3. Understand the existing defensive mechanisms to defeat attacks, their pros and cons

3   Topics

1. Background knowledge

   (a) Principles of compiler, linker and loader
   (b) x86 and x86-64 architectures
   (c) x86 and x86-64 instruction sets
   (d) Assembly and reversing engineering with AT&T and Intel syntax
   (e) Linux file permissions
   (f) Set-UID programs
   (g) ELF file format, PLT, GOT

(h) Memory map of a Linux process (x86 and x86-64)

(i) System calls

(j) Environment variables

(k) Tools: `gcc`, `gdb`, `pwndbg`, `objdump`, `ltrace`, `strace`, `env`, `readelf`, `/proc/PID/maps`, `pmap`, `tmux`, IDA Pro, Ghidra, binary ninja

2. Stack-based buffer overflow (2 sessions)

   (a) C local and global variables

   (b) C calling conventions (x86 and x86-64)

   (c) How stack works

   (d) Overflow local variables

   (e) Overflow RET; shellcode in exploits; shellcode in environment variable; shellcode in program arguments;

   (f) Frame pointer attack

   (g) Return-to-libc

   (h) Tools: `pwntools`

3. Defenses against stack-based buffer overflow (2 sessions)

   (a) Base and bound check

   (b) Data Execution Prevention

   (c) Canary: GCC implementation for x86 and x86-64

   (d) Shadow stack: 1) traditional; 2) parallel

   (e) SafeStack

   (f) Address space layout randomization (ASLR) and bypasses

   (g) Tools: `ldd`

4. Developing Shellcode (2 sessions)

   (a) Writing, compiling, and testing x86 and x86-64 assembly code

   (b) System call on Intel (x86 and x86-64)

   (c) Constraints on shellcoding: 1) zero-free shellcode; 2) ASCII-only shellcode; 3) English shellcode

5. Format string vulnerability

   (a) Format string

6. Heap security

   (a) Dynamic allocator, ptmalloc, tcache, `malloc()`, `free()`

   (b) Use after free (UAF)

   (c) Double free vulnerability

7. Integer overflow vulnerability

   (a) Integer overflow vulnerability

8. Return-oriented programming (2 sessions)

   (a) ROP

(b) Blind ROP

(c) Jump-oriented programming

(d) Control-flow integrity (CFI)

(e) Tools: `ROPgadget`, `pwntools`

9. Data-oriented attacks

(a) Direct data manipulation

(b) Data-oriented programming

10. Race conditions

(a) Race in file system

(b) Process and thread

(c) Race in memory

4 Course Schedule

Date	Topics	Dues and Remarks
Week-1 Class-1 1/31	Overview and background knowledge	
Week-1 Class-2 2/2	Background knowledge	
Week-2 Class-1 2/7	ELF	
Week-2 Class-2 2/9	Stack-based buffer overflow 1	Hw-1
Week-3 Class-1 2/14	Stack-based buffer overflow 2	
Week-3 Class-2 2/16	Stack-based buffer overflow 3	Hw-2
Week-4 Class-1 2/21	Stack-based buffer overflow 4	
Week-4 Class-2 2/23	Stack-based buffer overflow 5	Hw-3
Week-5 Class-1 2/28	Stack-based buffer overflow defense 1	
Week-5 Class-2 3/2	Stack-based buffer overflow defense 2	Hw-4
Week-6 Class-1 3/7	Shellcode 1	
Week-6 Class-2 3/9	Shellcode 2	Hw-5
Week-7 Class-1 3/14	*** Midterm Exam and CTF First Half***	
Week-7 Class-2 3/16	*** Midterm Exam and CTF Second Half***	Hw-6
Week-8 Class-1 3/21	=== Spring Recess No class ===	
Week-8 Class-2 3/23	=== Spring Recess No class ===	
Week-9 Class-1 3/28	Format string vulnerability 1	
Week-9 Class-2 3/30	Format string vulnerability 2	Hw-7, Hw-8
Week-10 Class-1 4/4	Format string vulnerability 3	
Week-10 Class-2 4/6	Format string vulnerability 4	Hw-9
Week-11 Class-1 4/11	ROP 1	
Week-11 Class-2 4/13	ROP 2	Hw-10
Week-12 Class-1 4/18	ROP 3	
Week-12 Class-2 4/20	Heap-based buffer overflow 1	Hw-11
Week-13 Class-1 4/25	Heap-based buffer overflow 2	
Week-13 Class-2 4/27	Heap-based buffer overflow 3	Hw-12
Week-14 Class-1 5/2	Race conditions 1	
Week-14 Class-2 5/4	Race conditions 2	Hw-13
Week-15 Class-1 5/9	Cache side-channel, meltdown, spectre 1	
Week-15 Class-2 5/11	Cache side-channel, meltdown, spectre 2	Hw-14

| Exam Week 5/16 | Final CTF 7:15PM - 10:15PM | |
| Exam Week 5/18 | | Take-home Final Exam |

Table 1: Tentative Class Schedule: 1 session per week

5 Grading Policy

Students will be evaluated on their performance on the homework, exams and CTFs. Attendance check will be performed in each class. Table 5 shows the grade breakdown.

Area	No. Items	Points per Item	Points for Area
Homework	14	45	630
Exams	2		160
Written Midterm	1	80	
Written Final	1	80	
CTFs	2		200
Midterm CTF	1	80	
Final CTF	1	120	
Attendance	14	1	14
Anonymous Course Evaluation Bonus	2	10	20
Total			1024

Table 2: Grades Breakdown

Points	Grade
930 -	A
900 - 930	A-
870 - 900	B+
830 - 870	B
800 - 830	B-
770 - 800	C+
700 - 770	C
670 - 700	D+
600 - 670	D
0 - 600	F

Table 3: Final Letter Grades

6 Prerequisite

Students are expected to have a background in the C programming language (CSE 220 Systems Programming or equivalent). Solid background from the following classes helps significantly: i) CSE 341 Computer Organization; ii) CSE 365 Introduction to Computer Security; iii) CSE 421/521 Operating Systems. The instructor strives to keep this class self-contained.

7 Plagiarism and Cheating

Students are encouraged to discuss the assignments online and offline. However, students are not allowed to share code, exploits, write-ups with other students.

Plagiarism or any form of cheating in homework, assignments, labs or exams is subject to serious academic penalty. As an institution of higher learning, UB expects students to behave honestly and ethically at all times, especially when submitting work for evaluation in conjunction with any course or degree requirement. All students are encouraged to become familiar with UB's Academic Integrity Policy (`https://www.buffalo.edu/academic-integrity.html`), Honor Code, and Student Conduct Policy. There is a zero tolerance policy in this class. Any violation of the academic integrity policy will result in a `0` on the homework, lab or assignment, and even an `F` or `>F<` on the final grade. The violation will be reported to the department chair and the Dean's office. The instructor takes plagiarism very seriously, e.g., in Spring 2018, two students received an `F` for plagiarism behaviors.

8 Syllabus Update

Information in the syllabus may be subject to change with reasonable advance notice.