CSE 410/510 Special Topics: Software Security

Dates: 08/30/2021 - 12/10/2021
Time: Monday, 5:00 PM - 7:50 PM

Instructor: Dr. Ziming Zhao
Email: zimingzh@buffalo.edu
TA: TBD
Office: 338B Davis Hall
Office Hours: Monday 3:50 PM - 5:00 PM or by appointment

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 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-based attacks
   (a) Data-flow attacks

10. Race conditions
   (a) Race in file system
   (b) Process and thread

    (c) Race in memory

11. Automatic vulnerability discovery
    (a) Static analysis
    (b) Fuzzing
    (c) Symbolic execution, concolic execution
    (d) Tools: `llvm`, `afl`, `KLEE`

3 Grading Policy

Area	No. Items	Points per Item	Points for Area
Exams			160
Written Midterm	1	80	
Written Final	1	80	
Homework	14	45	630
Final CTF	1	200	200
Attendance	14	1	14
Course Evaluation Bonus	2	8	16
Total			1020

Table 1: 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
Academic Dishonesty	>F<

Table 2: Final Letter Grades

4 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.

5 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, 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.

6 Syllabus Update

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