CSCI 2951U: Topics in Software Security

Introduction

Vasileios (Vasilis) Kemerlis
January 29, 2018

Department of Computer Science
Brown University
What is this course about?

State-of-the-art in software exploitation and defense

- Memory unsafe code (written in C/C++, asm, ...)

Software Security

1. Prevalent software defects
   - Stack/Heap smashing
   - Format string bugs
   - Pointer errors
   - ...
2. Modern defenses
   - W^X, ASLR
   - Stack/Heap canaries
   - RELRO, BIND_NOW
   - BPF_SECCOMP, FORTIFY_SRC
   - CFI, CPI, ...

Software Exploitation

1. Code injection
2. Code reuse
   - Return-to-libc (ret2libc)
   - Return-oriented prog. (ROP)
   - Just-In-Time ROP (JIT-ROP)
   - Blind ROP (BROP)
   - Signal-oriented prog. (SROP)
   - ...
3. Data-only attacks

vpk@cs.brown.edu (Brown University)
Course Overview (1/2)

▶ What is this course about?

✔️ State-of-the-art in software exploitation and defense ➔ **CSCI 1650++**
What is this course about?

- **State-of-the-art** in software exploitation and defense ➔ **CSCI 1650++**
- Memory unsafe code (written in C/C++, asm, ...)

Software Security

1. Prevalent software defects
   - Stack/Heap smashing
   - Format string bugs
   - Pointer errors
   - ...

2. Modern defenses
   - W^X, ASLR
   - Stack/Heap canaries
   - RELRO, BIND_NOW
   - BPF_SECCOMP, FORTIFY_SRC
   - CFI, CPI, ...

Software Exploitation

1. Code injection
2. Code reuse
   - Return-to-libc (**ret2libc**)
   - Return-oriented prog. (ROP)
   - Just-In-Time ROP (JIT-ROP)
   - Blind ROP (BROP)
   - Signal-oriented prog. (SROP)
   - ...

3. Data-only attacks
Course Overview (2/2)

Why take this course?

- Understand the boundaries of protection mechanisms and argue about their effectiveness.
- Familiarize with experimental mitigation techniques.
- Learn how and why (certain) defenses can be bypassed.
- Exploit "weaponization".

Why are these useful?

To design effective (and efficient) software protection mechanisms you need to:

(a) understand what sorts of attacks are possible
(b) how exactly these attacks work
(c) why previous attempts failed
Why take this course?

**Offense**

- Learn *how* and *why* (certain) defenses can be **bypassed**
  - Exploit “weaponization”
### Why take this course?

<table>
<thead>
<tr>
<th>Defense</th>
<th>Offense</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓ Understand the <strong>boundaries</strong> of protection mechanisms and argue about their <strong>effectiveness</strong></td>
<td>✓ Learn <strong>how and why</strong> (certain) defenses can be <strong>bypassed</strong></td>
</tr>
<tr>
<td>✓ <strong>Familiarize</strong> with experimental mitigation techniques</td>
<td>• Exploit “weaponization”</td>
</tr>
</tbody>
</table>
Course Overview (2/2)

**Why take this course?**

<table>
<thead>
<tr>
<th>☺ Defense</th>
<th>☺ Offense</th>
</tr>
</thead>
<tbody>
<tr>
<td>✅ Understand the <strong>boundaries</strong> of protection mechanisms and argue about their <strong>effectiveness</strong></td>
<td>✅ Learn <strong>how</strong> and <strong>why</strong> (certain) defenses can be <strong>bypassed</strong></td>
</tr>
<tr>
<td>✅ Familiarize with experimental mitigation techniques</td>
<td>• Exploit “weaponization”</td>
</tr>
</tbody>
</table>

**Why are these useful?**

- To design effective (and efficient) software protection mechanisms you need to:
  - (a) understand **what** sorts of attacks are possible
  - (b) **how** exactly these attacks work
  - (c) **why** previous attempts failed
Prerequisites

- **CSCI 1650** *(Software Security and Exploitation)*
  - Control-flow Hijacking
  - Code Injection (Shellcode dev.)
  - Code Reuse (ROP)

- **CSCI 1670** *(Operating Systems)*
  - C/C++, x86 **asm**
  - Linking and Loading
  - Virtual Memory
Prerequisites

- **CSCI 1650 (Software Security and Exploitation)**
  - Control-flow Hijacking
  - Code Injection (Shellcode dev.)
  - Code Reuse (ROP)

- **CSCI 1670 (Operating Systems)**
  - C/C++, x86 **asm**
  - Linking and Loading
  - Virtual Memory

✔ Having taken the following courses is a plus, but not required:
  - **CSCI 1660** (Computer Systems Security)
  - **CSCI 2951E** (Topics in Computer System Security)
Prerequisites

- **CSCI 1650 (Software Security and Exploitation)**
  - Control-flow Hijacking
  - Code Injection (Shellcode dev.)
  - Code Reuse (ROP)

- **CSCI 1670 (Operating Systems)**
  - C/C++, x86 *asm*
  - Linking and Loading
  - Virtual Memory

- **Having taken the following courses is a plus, but not required:**
  - **CSCI 1660 (Computer Systems Security)**
  - **CSCI 2951E (Topics in Computer System Security)**

- **We will review (most of) the important concepts**
Logistics

Meetings
• Mondays, 3PM – 5:20PM (M hour)
• CIT 368

Grading
- Paper reviews: 10%
- Paper presentations: 20%
- Discussion part: 20%
- Project report: 40%
- Project presentation: 10%

Communication
- http://cs.brown.edu/courses/csci2951-u/
- course.csci.2951u.2018-spring.s01@lists.brown.edu
- Check the website!
- Announcements
- Lecture slides
- Readings

Study material
- No required textbook
- Assigned readings

vpk@cs.brown.edu (Brown University)  CSCI 2951U  Spring ’18
Logistics

Meeting

• Mondays, 3PM – 5:20PM (M hour)
• CIT 368
Logistics

 Meetings

- Mondays, 3PM – 5:20PM (M hour)
- CIT 368

 Communication

- http://cs.brown.edu/courses/csci2951-u/
- course.csci.2951u.2018-spring.s01@lists.brown.edu

vpk@cs.brown.edu (Brown University)  CSCI 2951U  Spring ’18
### Meetings
- Mondays, 3PM – 5:20PM (M hour)
- CIT 368

### Communication
- [http://cs.brown.edu/courses/csci2951-u/](http://cs.brown.edu/courses/csci2951-u/)
- course.csci.2951u.2018-spring.s01@lists.brown.edu

**Check the website!**
- Announcements
- Lecture slides
- Readings

vpk@cs.brown.edu (Brown University)
# Logistics

<table>
<thead>
<tr>
<th>Meetings</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Mondays, 3PM – 5:20PM (M hour)</td>
</tr>
<tr>
<td>• CIT 368</td>
</tr>
</tbody>
</table>

## Grading

- ✔ Paper reviews → 10%
- ✔ Paper presentations → 20%
- ✔ Discussion part. → 20%
- ✔ Project report → 40%
- ✔ Project presentation → 10%

### @ Communication

- http://cs.brown.edu/courses/csci2951-u/
- course.csci.2951u.2018-spring.s01@lists.brown.edu

- ⭐ Check the website!
  - Announcements
  - Lecture slides
  - Readings
### Logistics

<table>
<thead>
<tr>
<th>Meetings</th>
</tr>
</thead>
</table>
| • Mondays, 3PM – 5:20PM *(M hour)*  
| • CIT 368 |

<table>
<thead>
<tr>
<th>Grading</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓ Paper reviews ➔ 10%</td>
</tr>
<tr>
<td>✓ Paper presentations ➔ 20%</td>
</tr>
<tr>
<td>✓ Discussion part. ➔ 20%</td>
</tr>
<tr>
<td>✓ Project report ➔ 40%</td>
</tr>
<tr>
<td>✓ Project presentation ➔ 10%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Communication</th>
</tr>
</thead>
</table>
| • [http://cs.brown.edu/courses/csci2951-u/](http://cs.brown.edu/courses/csci2951-u/)  
| • course.csci.2951u.2018-spring.s01@lists.brown.edu |

<table>
<thead>
<tr>
<th>Study material</th>
</tr>
</thead>
<tbody>
<tr>
<td>■ No required textbook ➔ Assigned readings</td>
</tr>
</tbody>
</table>

*Check the website!*
- Announcements
- Lecture slides
- Readings

vpk@cs.brown.edu (Brown University)

CSCI 2951U
Instructor

Vasileios (Vasilis) Kemerlis

- vpk@cs.brown.edu
- https://www.cs.brown.edu/~vpk

Office hours: Mon. 6PM – 8PM (CIT 505)
Memory Safety Circus

Figure 1. Attack model demonstrating four exploit types and policies mitigating the attacks in different stages

- Make a pointer go out of bounds
- Make a pointer become dangling
- Use pointer to write (or free)
- Use pointer to read

VI. Memory Safety

- Modify a data pointer
- Output data variable

- Modify code ...
- Interpret the output data

- Modify a code pointer ...
- Data-only attack

- Modify a data variable ...
- Information leak

VII.A. Data Integrity

- Code Integrity
- Code Pointer Integrity
- Address Space Randomization
- Control-flow Integrity
- Data-flow Integrity

VII.B. Data Space Randomization

- Non-executable Data / Instruction Set Randomization
- V.A. Data Integrity

- Use pointer by indirect call/jump
- Execute available gadgets / functions
- Use pointer by return instruction
- Execute injected shellcode

VIII.A. Address Space Randomization

- ... to the address of shellcode / gadget
- Use corrupted data variable

VIII.B. Data-flow Integrity

- ... to the attacker specified value


vpk@cs.brown.edu (Brown University)