CSCI 2951U: Topics in Software Security

Introduction

Vasileios (Vasilis) Kemerlis

January 27, 2020

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
What is this course about?

- State-of-the-art in software exploitation and defense → CSCI 1650++
# Course Overview (1/2)

► **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

vpk@cs.brown.edu (Brown University)
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 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?

<table>
<thead>
<tr>
<th>Defense</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓ Understand the <strong>boundaries</strong> of protection mechanisms and argue about their <strong>effectiveness</strong></td>
</tr>
<tr>
<td>✓ Familiarize with experimental mitigation techniques</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offense</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓ Learn <strong>how</strong> and <strong>why</strong> (certain) defenses can be <strong>bypassed</strong></td>
</tr>
</tbody>
</table>
  * Exploit “weaponization” |

---

Why are these useful?

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

1. Understand what sorts of attacks are possible
2. How exactly these attacks work
3. Why previous attempts failed
Why take this course?

**Defense**
- Understand the **boundaries** of protection mechanisms and argue about their **effectiveness**
- Familiarize with experimental mitigation techniques

**Offense**
- 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:
  - understand **what** sorts of attacks are possible
  - **how** exactly these attacks work
  - **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

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)
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 (1/2)

Meetings
• Mondays, 3PM – 5:20PM (M hour)
  • CIT 506

Grading
• 4 Paper reviews Ü 10%
• 4 Paper presentations Ü 20%
• 4 Discussion part. Ü 20%
• 4 Project report Ü 40%
• 4 Project presentation Ü 10%

Communication
• https://cs.brown.edu/courses/csci2951-u/
• course.csci.2951u.2020-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)
## Meetings

- Mondays, 3PM – 5:20PM (M hour)
- CIT 506
### Meetings
- Mondays, 3PM – 5:20PM (**M hour**)
- CIT 506

### Communication
- https://cs.brown.edu/courses/csci2951-u/
- course.csci.2951u.2020-spring.s01@lists.brown.edu
Logistics (1/2)

<table>
<thead>
<tr>
<th>Meetings</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Mondays, 3PM – 5:20PM (M hour)</td>
</tr>
<tr>
<td>• CIT 506</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <a href="https://cs.brown.edu/courses/csci2951-u/">https://cs.brown.edu/courses/csci2951-u/</a></td>
</tr>
<tr>
<td>• <a href="mailto:course.csci.2951u.2020-spring.s01@lists.brown.edu">course.csci.2951u.2020-spring.s01@lists.brown.edu</a></td>
</tr>
</tbody>
</table>

★ Check the website!

• Announcements
• Lecture slides
• Readings
Logistics (1/2)

Meetings
- Mondays, 3PM – 5:20PM (M hour)
- CIT 506

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

Communication
- https://cs.brown.edu/courses/csci2951-u/
- course.csci.2951u.2020-spring.s01@lists.brown.edu

Check the website!
- Announcements
- Lecture slides
- Readings
Logistics (1/2)

 Meetings

- Mondays, 3PM – 5:20PM (M hour)
- CIT 506

 Grading

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

 Study material

- No required textbook ➔ Assigned readings

 Communication

- https://cs.brown.edu/courses/csci2951-u/
- course.csci.2951u.2020-spring.s01@lists.brown.edu

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

vpk@cs.brown.edu (Brown University)
Interest Form

- https://forms.gle/5EecXuREsWwog5746
Instructor

Vasileios (Vasilis) Kemerlis

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

Office hours: Mon. 6PM – 8PM (CIT 505)
Figure 1. Attack model demonstrating four exploit types and policies mitigating the attacks in different stages.

Human-readable text:

- **Code Integrity**
  - Code corruption attack
  - Control-flow hijack attack

- **Data Integrity**
  - Data-only attack

- **Instruction Set Randomization**
  - Non-executable Data

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

- **Data Space Randomization**
  - Modify a data pointer
  - Modify a code pointer
  - Modify a data variable
  - Output data variable

**Code corruption attack**

- Execute available gadgets / functions
- Execute injected shellcode

**Control-flow hijack attack**

- Interpret the output data

**Data-only attack**

- Use pointer by return instruction

**Data-space leak**

- Use corrupted data variable


vpk@cs.brown.edu (Brown University)