Code Reuse Attacks and Defenses against them

Peng XU, Paul Muntean
peng@sec.in.tum.de, paul@sec.in.tum.de

Lehrstuhl für Sicherheit in der Informatik,
I20, TU-München

February 2, 2017
Outline

1 Class style
   - Organization and Requirement
   - Grading
   - Time Table
   - Literature Research
   - Next Steps

2 Content
   - Problems
   - Attacks
   - Defense mechanisms
     - Randomization based defense
     - Compiler-based defense
     - Binary-rewriting based defense
This seminar will be organized as a scientific conference:

- Familiarization phase (approx. 2 Week)
- Manufacturing phase (approx. 6 Week)
- Review phase (approx. 2 Week)
- Improvement phase (approx. 2 Week)
- Talk preparation (approx. 1 Week)
- Talk and Discussion
Requirements

Report Elaboration

- Delivery of a scientific paper with about $\geq 10$ pages in length
- Usage of $\LaTeX$ is mandatory for all
- Formatting with the $\LaTeX$-Style of Springer (LNCS)

Reviews

- Each one of you creates two anonymous reviews about other two reports
- Size of the one review: approximately one page in $\LaTeX$
- Additionally each of you will get an review from us
Requirement

- Demonstrate with the example code for the attacks’ papers
- Demonstrate with the new version compilers for the compiler-based works
- Show which framework will be used and how to change the binary with example for the binary-rewriting works
- Create results using the idea and approach from the paper idea
- Put the results in your report and evaluate the method
# Time Table

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>27.01.</td>
<td>Kick-off</td>
</tr>
<tr>
<td>18.04. - 27.06.</td>
<td>Regular meetings (presence mandatory)</td>
</tr>
<tr>
<td>02.05.</td>
<td>Delivery of the literature research,</td>
</tr>
<tr>
<td></td>
<td>Outline of the report</td>
</tr>
<tr>
<td>09.05. - 27.06.</td>
<td>Presentations</td>
</tr>
<tr>
<td>Bis 30.06.</td>
<td>Delivery of the 1. version of the report</td>
</tr>
<tr>
<td></td>
<td>End of the presentation phase</td>
</tr>
<tr>
<td>04.07.</td>
<td>Distribution of the review topics</td>
</tr>
<tr>
<td>Bis 07.07.</td>
<td>Delivery of the reviews</td>
</tr>
<tr>
<td>08.07.</td>
<td>Return of the reviews</td>
</tr>
<tr>
<td>Bis 11.07.</td>
<td>Delivery of the final version of the report</td>
</tr>
</tbody>
</table>
The Grading is comprised of all personal contributions of this seminar and is composed of:

- Report (50%)[10% for the demonstration project ]
- Presentation (25%)
- Delivered review (15%)
- Participation and discussion (10%)
Topic assignment

- Who wants which topic?
Goal:

- To find relevant literature
- Main arguments, Techniques or Approaches...
  1. find,
  2. understand,
  3. explain,
  4. prove them
- Structure Topics
  - Report structure
Literature Research & Sources

Good

- Books, Library
- http://portal.acm.org/
- http://www.springerlink.com/
- http://www.computer.org/
- http://citeseer.ist.psu.edu/
- http://scholar.google.com/

Wrong

- Heise-Newsticker, Wikipedia, e.g.
Access to Literature

Through the Authors Website

- Authors publish the papers mostly on their websites
- Other resources can be found through Google Scholar

Through Springer, ACM, IEEE

- Download of papers costs
- TUM has full rights to download papers
- Usage on an Proxy-Server required: www.lrz.de
- Access through the proxy in the TUM web is restricted
Next Steps

**\LaTeX**-Introduction

- Is there the need?
- Schedule a date?
Next Steps

**\LaTeX-Introduction**

- Is there the need?
- Schedule a date?

**ToDos in the Familiarization phase**

1. Literature research
2. Create report structure
Questions

Q&A?
Problems

- Sophisticated, complex, Different developers
- Unsafe language (C/C++) development
Attacks

- **Runtime attacks**
  - Memory corruption: Stack overflow, dangling pointer
  - Code injection attacks
  - Code reuse attacks
Code Reuse Attacks

- Types of CRAs
  - Instruction-level reuse: ROP, JIT-ROP, ...
  - Function-level reuse: vTable Hijacking, COOP, ...
Attacks

- Ret2libc
- Return-oriented programming (ROP)[1]
- Jump-oriented programming (JOP)[2]
- Hacking Blind(Blind-ROP)[3]
- JIT-ROP[4]
- Counterfeit object-oriented programming(COOP)[5]
Attacks

1. Return-oriented programming without returns

2. Jump-oriented programming: a new class of code-reuse attack

3. Hacking blind
Attacks

4 Just-in-time code reuse: On the effectiveness of fine-grained address space layout randomization


5 Counterfeit object-oriented programming: On the difficulty of preventing code reuse attacks in C++ applications

Defense mechanisms

Defenses

- Randomization
- Control Pointer Integrity
- Control Flow Integrity
Defense mechanisms

Randomization based defense

- Against ROP attack on Android[6]
- Against JIT-ROP attacks[7]
- Against function reusing attacks[8]
6 Blender: Self-randomizing address space layout for android apps


7 Isomeron: Code Randomization Resilient to (Just-In-Time) Return-Oriented Programming


8 It’s a TRaP: Table randomization and protection against function-reuse attacks

Crane, Stephen J., et al. "It’s a TRaP: Table randomization and
Defense mechanisms

Control Flow Integrity

- Compiler-based defenses
- Binary-rewriting-based defenses
Compiler-based defense

- SafeDispatch[9]
- Interleaving[10]

9 SAFE DISPATCH: Securing C++ Virtual Calls from Memory Corruption Attacks


10 Protecting C++ dynamic dispatch through vtable interleaving

Binary-rewriting based defense

- VTint[11]
- V-TIP[12]

11 VTint: Protecting Virtual Function Tables Integrity


12 Towards automated integrity protection of C++ virtual function tables in binary programs

Questions

Q&A?