CURRENT TOPICS IN
INFORMATION SECURITY
&
CRYPTOGRAPHY
ORGANISATION

- Paper Assignment: Today
- Paper Presentation: 2 weeks Dec
- Research Proposal: 3 weeks Mid Dec
- Research Paper: End of Semester Feb ‘14
ORGANISATION

1) Paper Selection
ORGANISATION

2) Group Assignment
3) Related Work
4) Everybody presents one paper
In two weeks from now

- 15 minutes each
- spread over 2 days
RESEARCH PROPOSAL

- Read the assigned paper
- Read the related work mentioned in the paper
- Figure out shortcomings
- What can be improved?
- What can be built upon that?
- What is your own idea?
RESEARCH PROPOSAL
(It's not as difficult as it seems)

While reading the paper

Interesting

Build upon

Good

Lack

You would have done it differently

Improve
RESEARCH PROPOSAL
(It's not as difficult as it seems)

Approve

Build upon

Improve

Research Proposal
RESEARCH PAPER

• written like a ‘real’ scientific paper
• adhere to the standard IEEE template
• 6 pages long (with references)
PAPERS TO CHOOSE FROM

Cloud Security

A1 AmazonIA - When Elasticity Snaps Back

Stefan

Mobile Security

A2 Android Permissions: User Attention, Comprehension, and Behavior
PAPERS TO CHOOSE FROM

Anonymous Communication

B1 Provably Secure and Practical Onion Routing

B2 Users Get Routed: Traffic Correlation on Tor by Realistic Adversaries
PAPERS TO CHOOSE FROM

**Case Law**

C1: Formalizing and Enforcing Purpose Restrictions in Privacy Policies

C2: Honeywords: Making Password-Cracking Detectable

**Passwords**

C1: Formalizing and Enforcing Purpose Restrictions in Privacy Policies

C2: Honeywords: Making Password-Cracking Detectable
Anonymous Communication

D1  Nymble: Blocking Misbehaving Users in Anonymizing Networks

D2  Practical Repudiation (or Traceability) for Anonymous Communication Networks
PAPERS TO CHOOSE FROM

Censorship Resistance

E1 Routing Around Decoys
E2 The Parrot is Dead: Observing Unobservable Network Communications
PAPERS TO CHOOSE FROM

e-Voting

F1 Using Mobile Device Communication to Strengthen e-Voting Protocols  

Martin
WHAT WE EXPECT FROM YOU

• Presentation (15 min + 5 discussion)
• Research Paper Proposal (1 page)
• Research Paper (6 pages IEEE)
WHAT WE CAN OFFER

• Insight into scientific community and processes

• Maybe a student assistant (HiWi) position

• Idea for a bachelor’s / master’s thesis

• Good preparation if you want to pursue a PhD

… and of course

7 CP
PAPER SELECTION
DATES FOR THE PRESENTATION

<table>
<thead>
<tr>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

December
Gadge Me If You Can

Secure and Efficient Ad-hoc Instruction-Level Randomization for x86 and ARM

Lucas Davi, Alexandra Dmitrienko, Stefan Nürnberger, Ahmad-Reza Sadeghi

ABSTRACT

Code reuse attacks such as return-oriented programming are one of the most powerful threats to contemporary software. ASLR was introduced to impede these attacks by dispersing shared libraries and the executable in memory. However, in practice its entropy is rather low and, more importantly, the leakage of a single address reveals the position of a whole library in memory. The recent mitigation literature followed the route of randomization, applied it at different stages such as source code or the executable binary. However, the code segments still stay in one block. In contrast to previous work, our randomization solution, called XIFER, (1) disperses all code (executable and libraries) across the whole address space, (2) re-randomizes the address space for each run, (3) is compatible to code signing, and (4) does neither require offline static analysis nor source code. Our prototype implementation supports the Linux ELF file format and covers both mainstream processor architectures x86 and ARM. Our evaluation demonstrates that XIFER performs efficiently at load-and during run-time (1.2% overhead).

Categories and Subject Descriptors

D.4.8 (Operating Systems): Security and Protection

General Terms

Security

Keywords

software diversity; randomization; ASLR; return-oriented programming; return-into-lib

1. INTRODUCTION

Security-critical operations such as online banking are increasingly performed by widespread everyday-software. This makes them an appealing target for various attacks, in particular runtime attacks which subject a process to an adversary’s control. Albeit context-free attacks on software are known for about two decades, they are still one of the major threats to software on desktop PCs and mobile devices. The NIST vulnerability database reported 663 buffer overflows in 2011, and 274 for 2012 [36]. The broad introduction of non-executable memory, write-not-execute (W / X) for short, successfully mitigates code injection attacks but gives rise to a form of attacks that reuse existing code by intelligently stitching small code fractions, so-called gadgets, together in order to execute arbitrary code. These gadgets are well selected so that they end in an instruction that transfers control to the next gadget, e.g. a `ret` (return) instruction which pops its target off the stack. Hence, the same return-oriented programming (ROP) [31].

Since these attacks rely on exact addresses of the instructions they want to abuse, Address Space Layout Randomization (ASLR, e.g. [29]) debuted as the next move in that cut-and-choose game. By randomizing the base address of loaded code and data in memory, ASLR in theory makes it impossible for an attacker to predict the location of gadgets in memory. However, low entropy [32] and the fact that a single leaked pointer makes it possible to calculate relative addresses called for yet another step of defense. Such a means of defense has come to light in the form of finer and finer code randomization that, in contrast to ASLR, also shuffles the code itself, not just its base address. Despite the fact that each randomization is a simple idea, its implementation is highly involved and several approaches exist in the literature, ranging from compiler-based solutions [6, 11, 18] to run-time solutions [36] that randomize the program either once or even constantly during its lifetime [13].

As we elaborate in Section 3, most of the existing works have at least one of the following drawbacks: they (i) need access to source code, (ii) do not cover the whole address space, e.g. no loaded libraries and the code segment stays in one block, (iii) do not re-randomize at each process start, or (iv) touch the executable file rendering them incompatible to code signing which is prevalent for commercial software and mandatory in modern app stores.

In order to compare and measure existing software diversity methods, we establish a set of properties that make a randomization solution ideal, i.e., featuring the best trade-off among these properties. These properties are: (1) mitigation of code reuse attacks (ROP and return-into-lib),
HOW IS A RESEARCH PAPER WRITTEN?

- Introduction
- Context
- Problem Description
- Contribution
- Related Work
- Actual Meat