Sharing Specifications

or

the State of Repeatability in Computer Systems Research

Christian Collberg

Todd Proebsting

Department of Computer Science
University of Arizona

http://repeatability.cs.arizona.edu

To appear in The Communications of the ACM

Supported by the private foundation that must not be named
Opening Gambit
Study
Proposal
Future Work
Abstract

We present a new general technique for protecting clients in distributed systems against Remote Man-in-the-Middle (R-MATE) attacks. Such attacks occur in settings where an adversary has physical access to an untrusted client device and can gain an advantage from tampering with the hardware or software it composes.

In our system, the trusted server coordinates the untrusted client’s analytics abilities by continuously and automatically generating and pushing to him diverse client code variants. The diversity subsystem employs a set of primitive code transformations that provide an ever-changing target for the adversary, making tampering difficult without being detected by the server.

1. Introduction

Man-in-the-Middle (MATE) attacks occur in settings where an adversary has physical access to a device and compromises it by tampering with its hardware or software. Remote MATE attacks occur in distributed systems where untrusted clients are in frequent communication with trusted servers over a network, and malicious users target an advantage by compromising an untrusted device.

To illustrate the ubiquity of R-MATE vulnerabilities, consider the following four scenarios. First, in the Advanced Metering Infrastructure (AMI) for controlling the electrical power grid, networked devices (“smart meters”) are installed at individual households to allow two-way communication with control servers of the utility company. In an R-MATE attack against the AMI, a malicious consumer tampers with the meter to emulate an imminent blackout, or tricks a control server to send disconnection commands to other customers [23].

Second, massive multiplayer online games are susceptible to R-MATE attacks since a malicious player who tampers with the game client can get advantage over other players [10]. Third, wireless sensors are often deployed in unattended environments such as the homes of elderly people to detect tampering attempts. A compromised sensor could be coached into supplying the wrong observations to a base station, causing real-world damage. Finally, while electronic health records (EHR) are typically protected by encryption while stored in databases and transmitted to doctors’ offices, they are vulnerable to R-MATE attacks if an individual doctor’s client machine is compromised.

1.1 Overview

In each of the scenarios above the adversary’s goal is to tamper with the client code and data under his control. The trusted server’s goal is to detect any such integrity violations, after which countermeasures (such as severe connections, termination, etc.) can be launched.

Security mechanisms. In this paper we present a system that achieves protection against R-MATE attacks through the extensive use of code diversity and continuous code replacement. In our system, the trusted server continuously and automatically generates diverse variants of client code, probes these code updates to the untrusted clients, and installs them as the client is running. The intention is to force the client to constantly analyze and reanalyze incoming code variants, thereby overwhelming his analytical abilities, and making it difficult for him to tamper with the continuously changing code without being detected by the trusted server.

Limitations. Our system specifically targets distributed applications which have frequent client-server communication, since client tampering can only be detected at client-server interaction events. Furthermore, while our use of code diversity can delay an attack, it cannot completely prevent it. Our goal is therefore the rapid detection of attacks; applications which need to completely prevent any tampering of client code for even the shortest length of time, are not suitable targets for our system. To see this consider the following timeline in the history of a distributed application running under our system:

The $e, s$, and $t_i$ are invocation events, points in time when clients communicate with servers either to exchange application data or to perform code updates. At time $e_1$, the client tampers with the code under his control. Until the next invocation event, during interval $t_1$, the client runs autonomously, and the server cannot detect the attack. At time $s_2$, after an interval $s_2$, consisting of a few milliseconds events, the client’s tampering has caused it to display anomalous behavior, perhaps through the use of an out-of-date communication protocol, and the server detects this. At time $t_2$, finally, the server issues a response, perhaps by shutting.
To: authors@cs.ux.edu

Cool paper! Can you send me the system so I can break it? 😆
Reimplement!
Reimplement!

type operator =
| A
| B of operand * value * binop
| C of operand * value * operand * binop
| D of operand * value * operand * binop
| E of operand * operand

...
Technical Report

Conference Paper

PhD Thesis

• $f: \mathbb{N} \rightarrow \mathbb{N}$?
• $\varphi$?
• typecheck?

type operator =
| A |
| B of operand * value * binop |
| C of operand * value * operand * binop |
| D of operand * value * operand * binop |
| E of operand * operand |

...
To: PI,DC@cs.ux.edu

I ... request under the OPEN RECORDS ACT ... ALL SOURCE CODE ...
From: legal@cs.ux.edu

... to the extent such records may exist, they will not be produced pursuant to ORA.
PhD Thesis

10 TOP

NSF NSF

DARPA

git? backups?
Pursuant to ORA, I request copies of all electronic mail...
... we estimate a total cost of $2,263.66 to search for, retrieve, redact and produce such records.
We will also make our data and software available to the research community when appropriate.
Consequences

By not sharing their code, and by (perhaps unintentionally) leaving holes in their publications, the authors have effectively guaranteed that their claims can never be refuted.
Study
[T]he ability to re-run the exact same experiment with the same method on the same or similar system and obtain the same or very similar result.

Vitek, Kalibera: R3 – Repeatability, Reproducibility and Rigor
Weak Repeatability

Do authors make the source code used to create the results in their article available, and will it build?
Results are backed by code?

ASPLOS'12, CCS'12, OOPSLA'12, OSDI'12, PLDI'12, SIGMOD'12, SOSP'11, VLDB'12, TACO'9, TISSEC'15, TOCS'30, TODS'37, TOPLAS'34
Results are backed by code?

Can we find the code?

1. Article?
2. Web?
3. Email?
Results are backed by code?

Can we find the code?

1. Article?
2. Web?
3. Email?

Can we build the code in 30 minutes?

No

Can we build the code in >30 minutes?

No

Does the author believe the code builds?

No

Weakly Repeatable
Reasons for not Sharing?

The email responses we received were pleasant, accommodating, and apologetic if code could not be provided.
The good news ... I was able to find some code. I am just hoping that it matches the implementation we used for the paper.
Unfortunately the current system is not mature ... We are actively working on a number of extensions ...
The code was never intended to be released so is not in any shape for general use.

No Intention to Share
[Our] prototype ... included many moving pieces that only student knew how to operate ... he left.
[Our] prototype … included many moving pieces that only student knew how to operate … he left.
... the server in which my implementation was stored had a **disk crash** ... three disks crashed ... Sorry for that.
... the server in which my implementation was stored had a disk crash ... three disks crashed ...
Sorry for that.
[Our system] continues to become more complex as more PhD students add more pieces to it.
... when we attempted to share it, we [spent] more time getting outsiders up to speed than on our own research.
[Therefore] we will not provide the source code outside the group. Academic Tradeoffs
... we can't share what did for this paper. ... this is not in the academic tradition, but this is a hazard in an industrial lab.
... we have no plans to make the scheduler's source code publicly available. ... because [ancient OS] as such does not exist anymore
... few people would manage to get it to work on new hardware.
We would like to be notified [if] the implementation [is used] to perform (and ... publish) comparisons with other developed techniques.
... based on earlier (bad) experience, we [want] to make sure that our implementation is not used in situations that it was not meant for.
… we have an agreement with the [business], and we cannot release the code because of the potential privacy risks…
The code ... is ... hardly usable by anyone other than the authors ... due to our decision to use [obscure variant of obscure language]
Proposal
Three Modest Proposals

1. Funding agencies should encourage researchers to request additional funds for **repeatability engineering**
Three Modest Proposals

1. Funding agencies should encourage researchers to request additional funds for **repeatability engineering**

2. Agencies should conduct **random audits** to ensure that research artifacts are shared in accordance with what was promised in the grant application
3. Publishers should require articles to contain a sharing contract specifying the level of repeatability to which its authors will commit.
3. Publishers should require articles to contain a **sharing contract** specifying the level of repeatability to which its authors will commit.
3. Publishers should require articles to contain a sharing contract specifying the level of repeatability to which its authors will commit.
3. Publishers should require articles to contain a **sharing contract** specifying the level of repeatability to which its authors will commit.
3. Publishers should require articles to contain a **sharing contract** specifying the level of repeatability to which its authors will commit.
3. Publishers should require articles to contain a sharing contract specifying the level of repeatability to which its authors will commit.
<p>| Location      | • email address and/or web site |</p>
<table>
<thead>
<tr>
<th>Location</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>• email address and/or web site</td>
<td>• <strong>types</strong>: code, data, media, documentation</td>
</tr>
<tr>
<td></td>
<td>• <strong>availability</strong>: no access, access, NDA access</td>
</tr>
<tr>
<td></td>
<td>• <strong>expense</strong>: free, non-free, free for academics</td>
</tr>
<tr>
<td></td>
<td>• <strong>distribution form</strong>: source, binary, service</td>
</tr>
<tr>
<td></td>
<td>• <strong>expiration date</strong></td>
</tr>
<tr>
<td></td>
<td>• <strong>license</strong></td>
</tr>
<tr>
<td></td>
<td>• <strong>comment</strong></td>
</tr>
<tr>
<td>Location</td>
<td>• email address and/or web site</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------</td>
</tr>
</tbody>
</table>
| Resource          | • **types**: code, data, media, documentation  
|                   | • **availability**: no access, access, NDA access  
|                   | • **expense**: free, non-free, free for academics  
|                   | • **distribution form**: source, binary, service  
|                   | • **expiration date**  
|                   | • **license**  
|                   | • **comment**  
| Support           | • **kinds**: resolve installation issues, fix bugs, upgrade to new language and operating system versions, port to new environments, improve performance, add features  
|                   | • **expense**: free, non-free, free for academics  
|                   | • **expiration date**  

Sharing Contract

sharing
repeatability.cs.arizona.edu;
collberg@gmail.com;
code: access,free,source;
data: access,free,source,"sanitized";
support: installation,bug fixes,free, 2015-12-31;
Discussion and Future Work
| TODS'37 | Davide Martinenghi, Marco Tagliasacchi | Proximity measures for rank join | Practical | Link from google | Not sent | - | Builds | Database Entry | Build notes |
| TODS'37 | Daniel Lemire, Owen Kaser, Eduardo Gutierrez | Reordering rows for better compression: Beyond the lexicographic order | Practical | Link from paper | Not sent | - | Builds | Database Entry | Build notes |
| TODS'37 | Benny Kimelfeld, Jan Vondrak, Ryan Williams | Maximizing Conjunctive Views in Deletion Propagation | Theoretical | - | - | - | - | Database Entry | - |
| TODS'37 | Yinan Li, Jignesh M Patel, Allison Terrell | WHAM: A High-Throughput Sequence Alignment Method | Practical | Link from google | Not sent | - | Build fails | Database Entry | Build notes |
| TODS'37 | Yufei Tao, Cheng Sheng, Jianzhong Li | Exact and approximate algorithms for the most connected vertex problem | Practical | - | Email sent | Replied yes | Builds | Database Entry | Build notes |
| TODS'37 | Junhu Wang, Jeffrey Xu Yu | Revisiting answering tree pattern queries using views | Practical | - | Email sent | Replied no | - | Database Entry | - |

To appear in The Communication of the ACM
1. Demanding everyone to share code always is unrealistic.

2. Sharing specifications are a low-cost alternative that can be implemented now.

3. We believe sharing specifications will be an incentive to authors to produce solid computational artifacts.
To: author@cs.ux.edu

Congrats on your new paper!

• Will you share?
• Under what license?
• URL to code/data?
Sharing Data Database

Share?

Sure!
1. Data for reproducibility research
1. Data for reproducibility research
2. Trending data for funding agencies
1. Data for reproducibility research
2. Trending data for funding agencies
3. Directory of research artifacts
1. Data for reproducibility research
2. Trending data for funding agencies
3. Directory of research artifacts
4. Motivating researchers to share
Questions?