Sharing Specifications

Christian Collberg

Todd Proebsting

Department of Computer Science
University of Arizona

http://repeatability.cs.arizona.edu
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 obtain an advantage from tampering with the hardware itself or the software it contains.

In our system, the trusted server collaborates with the untrusted client's analytical 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 attack target for the adversary, making tampering difficult without this 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 man-in-the-middle (R-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 centers 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 to trick a control server to send disconnection commands to other customers [59]. 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 unsecured environments such as the corner of a test bed, where they are vulnerable to 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 attack 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 severing connections, termination, etc.) can be launched.

Security mechanisms. In this paper we present a system that achieves protection against R-MATE attacks through the use of code diversity and continuous code replacement. In our system, the trusted server continuously and automatically generates diverse variants of client code, while these code updates are also received by the untrusted clients, and installs them in the client as it is running. The idea is to force the client to continuously analyze and re-analyze 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:

\[ t_1 \quad \text{Client tampers} \quad t_2 \quad \text{Client detects} \quad t_3 \quad \text{Server responds} \]

The \( t_1 \) are interaction events, points in time when clients communicate with servers either to exchange application data or to perform code updates. At time \( t_1 \), the client tampers with the code under his control. Until the next interaction event, during interval \( t_2 \), the client runs autonomously, and the server cannot detect the attack. At time \( t_2 \), after an interval \( t_2 \), consisting of a few milliseconds, etc., 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 \( t_3 \), 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? 😀
• $f: \mathbb{N} \rightarrow \mathbb{N}$?
• $\varphi$?
• 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

• Technical Report
• Conference Paper
• PhD Thesis

Haskell λ
A Purely Functional Language
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.
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.
Study
The 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.
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?

Can we find the code?

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

Can we build the code in 30 minutes?

Can we build the code in >30 minutes?

No

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.

**Versioning**
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.
[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.
[Therefore] we will not provide the source code outside the group.
... 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.

Industrial Lab Tradeoffs
... we have an agreement with the [business], and we cannot release the code because of the potential privacy risks...
Proposal
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.
<table>
<thead>
<tr>
<th>Location</th>
<th>• email address and/or web site</th>
</tr>
</thead>
</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

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

Sharing Data

Database

Share?

Sure!
Questions?