IT University of Copenhagen

(Thesis)

Information-Flow Secure Programming on Matrix: A Case Study

Authors:
Ans Uddin anud@itu.dk

Supervisor:
Willard Rafnsson

Co-supervisor:
Carsten Schürmann

February, 2018

Contents

1 Introduction

1.1 Data privacy and protection

With GDPR becoming effective in 2018 the focus on data privacy is at its peak. Privacy violation is when sensitive data is exposed to unauthorized actors. OWASP top ten ranks *Sensitive Data Exposure* as 3rd biggest security risk. Recent cases of data leakage has put more attention on data privacy and protection. Some cases are due to poor security measures and could arguably have been prevented. Examples of cases are:

- The Facebook - Cambridge Analytica scandal. Third parties were able to collect data through Facebook Login API.
- Google Plus leak. Third parties could access private data through APIs.
- Medicaid leak. A medical assistant had accessed patients health records and exchanged mails with another employee containing the patients' private data.

The cases above failed to achieve end-to-end security and could have been prevented with proper security policies and enforcement technique that enforces these policies.

There is more awareness on how applications deal with data. This add extra concern to the programmer and the application being developed on how sensitive data is handled and protected.

The well-known security enforcement techniques like access controls, firewalls and encryption are inadequate alone and does not ensure end-to-end security hence improperly handle sensitive data.

1.2 Information Flow Control

Information-flow security aims at protecting confidentiality and integrity of information.

Information flow systems controls the flow of information by enforcing policies.

Common security enforcement mechanisms such as firewalls, encryption, and antivirus software are useful for protecting confidential information. However, these mechanisms do not provide end-to-end security.

Access control permissions prevents processes not authorized by the file owner from reading the file. However, access control does not control how the data is used after it is read from the file. To soundly enforce confidentiality

using this access-control policy, it is necessary to grant the file access privilege only to processes that will not improperly transmit or leak the confidential data. Access-control mechanisms cannot identify these processes; therefore, access control, while useful, cannot substitute for information-flow control.

A firewall protects confidential information by preventing communication with the outside. A firewall permit some communication in both directions whether this communication violates confidentiality lies outside the scope of the firewall mechanism.

Encryption can be used to secure an information channel so that only the communicating endpoints have access to the information. However, this encryption provides no assurance that once the data is decrypted, the computation at the receiver respects the confidentiality of the transmitted data.

What these mechanism lack is a way language-based way to control how information propagates. A mechanism for information flow control is one that enforces information flow policies. There have been proposed many methods to enforce information flow policies. Run-time mechanisms that tag data with information flow labels have been employed at the operating system level and at the programming language level. Static program analyses have also been developed that ensure information flows within programs are in accordance with policies.

1.3 Matrix

Matrix is an open standard protocol for messaging over HTTP and synchronizing data. Matrix provides secure real-time communication over a decentralized federated network. Matrix secures data by providing end-to-end encryption.

Matrix defines their goal as to act as a generic HTTP messaging and data synchronisation system for the whole web - allowing people, services and devices to easily communicate with each other, empowering users to own and control their data and select the services and vendors they want to use.

Why Matrix?

Even though Matrix is intended for chat and instant messaging they define their longterm goal as "to act as a genering HTTP messaging and data synchronization for the whole web". The large number of software systems in the public sector has created a need for a common way of exchanging data. The Dabish public sector is facing a similar issue as Matrix.org has tried to solve...

1.4 The case study

Medical privacy is a relevant topic

Health care records.

Issues with Sundhedsplatformen more specifically the patient journal. 90.000 medical employees have access to ones journal. In practice a Health care assistant can access the journal. There are clear policies on who should have access and under what conditions to view the journal. But the mechanism to ensure these policies are lacking.

The following policies

1. The medical employee must have a employee certificate
2. The employee has to login to the journal system
3. Logging in or accessing a journal triggers a log entry
4. The employee must have the patient in care
5. The lookup most be relevant for the employee

There currently is no limitation to what the employees can see. Safety measures have been applied through logging and monitoring, and by but is inadequate.

The medical study example shows that it is possible to give another party private information and receive the results of its computation while remaining confident that the data given to it is not leaked.

As mentioned earlier Matrix provide end-to-end encryption. However in the presence of end-to-end encryption, apps can still leak through their application logic; a content-filtering chat bot running at the receiving end of an end-to-end encrypted connection can leak anything it receives from this connection.

Leaks through the application logic can be prevented with Information Flow Control (IFC). IFC prevents leaks by enforcing policies for secure information flow in a program. There exist tools (e.g JIF, Paragon and Fabric) that aid in building software with secure information flow.

Scope

The objective of the project is to do a case study on the security of Matrix, apply IFC tools to improve the Matrix security model and demonstrate the improvements.

A successful project is one that fulfills these criteria:

- Evaluation of Matrix security model
- Survey IFC tools to improve the Matrix security model
- Implement a prototype distributed system running on Matrix, using the chosen tools
- Demonstrate improvements to the Matrix security model

1.5 Threat model

*** Comment ***
*** Is it called adversary model / attacker model / threat model ***
*** Not sure if this section is correct ***

The major threat is to confidentiality and integrity Preservation of confidentiality and integrity of information is obtained with information-flow security.

The adversary has control of public input and output and is able to observe data over the network. It is assumed that the adversary has limited access to the system with restricted privileges and must not gain sensitive data.

1.6 Contribution

1.7 Structure of thesis

Chapter 1 sets the foundation for the thesis and introduces relevant information and background. In chapter 3 the method used in the thesis is elaborated on. Chapters 4 and 5 goes in depth with the analysis and design of the solution. The results are then presented and discussed in 6. The thesis is wrapped up in the conclusion section.

2 Background

2.1 Information Flow Control

Confidentiality

Integrity

The Lattice Model

Noninterference

Static policies

Dynamic policies

Declassification

Taking some specific information and changing it to a lower security classification.

Identify: What to classify, who declassifies, where the declassification happens and when the declassification happens

2.2 Matrix

Architecture

Client/Server API

Security model

3 Method

4 Analysis

4.1 Survey of IFC Tools

JIF

Fabric

Paragon

JSFlow

Not possible to use Matrix library with JSFlow because of missing support for libaries such as require (in node). Also overhead with configuring JSFlow to be the interpretor.

Swift, SIF, FlowR, JFlow

Selection of IFC tool

The selection of IFC tool put emphasis on the practical usage in combination with Matrix.

4.2 Evaluation of Matrix security model

5 Design

5.1 Object Oriented Analysis and Design

Requirement

Analysis

Design

5.2 Matrix Design

Design choices

6 Results

6.1 Implementation

6.2 Discussion

7 Conclusion

Appendices

Bibliography

[1] Website example. `https://google.com`. Accessed: xxxx-xx-xx.

[2] John Doe. *Lorem ipsum*. Unknown, 2018. ISBN 0521865719, 9780521865715.