# Collecting Metadata on an Instant Messaging Server

**Alexandre Pujol, Christina Thorpe and Liam Murphy  Performance Engineering Laboratory, School of Computer Science and Informatics,
University College Dublin, Ireland**
alexandre.pujol@ucdconnect.ie
christina.thorpe@ucd.ie
liam.murphy@ucd.ie

**Abstract:** In this work in progress paper, we worked on metadata leakage in the specific use case of an instant messaging server. In a first time, we define the different type of metadata that can be found, then in a second time, we provide a proof of concept using live forensic method to show how we can retrieve leaked metadata on a running server. Then we propose the use of an Oblivious RAM model as a solution of this leak.

**Keywords:** metadata, metadata definition, privacy, instant messaging, live forensic, oblivious RAM

## 1. Introduction

Instant messaging (IM) services are becoming a major communication tool for both private and enterprise use. Therefore, the security of the data is an important concern. Consequently, some instant messaging services allow end-to-end encryption.

However, even with end-to-end encryption, metadata leakage is an important issue. For instance, an attacker spying on an IM server can intercept the metadata and retrieve sensitive knowledge. This attack also works if all the data are encrypted; moreover, the metadata can potentially reveal more knowledge than the data themselves.

In this short paper, we define two types of metadata that can leak. Then, working on an IM use case, we created a novel approach in order to detect leaks of metadata. This technique uses a live forensic tool on the RAM process of an IM server to retrieve metadata even when the data is encrypted.

To the best of our knowledge, this is the first study that shows the leaks of metadata in an IM application. Furthermore, in order to prevent this leak, we propose to use an Oblivious Random Access Machine (ORAM) scheme - an academic cryptography technique that prevents the leak of all kinds of metadata. Doing that, we are covering a gap in literature between ORAM and real life leaks of metadata.

The remainder of this paper is organized as follows: Section 2 discusses the related work in the area. Section 3 defines the security model considered in this research. Section 4 presents a definition of two different types of metadata considered in the context of this work. Section 5 states the objectives, which form the motivation. Section 6 introduces the proposed solution. Section 7 describes a basic experiment conducted to demonstrate the target attack. Finally, Section 8 contains a summary of the work to date and proposes the planned future work.

## 2. Related work

There have been a few studies on the collection of metadata: Dubin et. Al. [4] shows how to extract information from encrypted YouTube video streams. Mayer et. Al. [5] studies the privacy issues that are associated with smartphones due to the fact metadata are usually not protected. However, most of these studies use local or network detection. To the best of our knowledge, there is no existing study where the attacker has a full access to the server in order to retrieve metadata. Our solution uses the state of the art in live forensic analysis tools. We use Rekall [2] as our main memory analysis framework. We also use Volatility [1], a similar framework.

## 3. Security model

This section details the security model that we consider in our work, including the use case and attacker model.

### 3.1 Use case

In this work in progress paper, we consider a simple Instant Messaging (IM) application use case. The IM server runs an IM instance. The messages sent between users can be either encrypted or unencrypted.

### 3.2 Attacker model

The adversarial model defined for this work is detailed in the following:

- The Server is vulnerable to an attacker and therefore cannot be trusted.
- The attacker has a root access to the server and is looking to retrieve as much metadata as possible from the IM server users.
- The attacker is regarded as a passive adversary seeking to stay on the server as long as possible and to gather information. However it will not pervert any data in the server itself.

The assumption that the attacker gains a root access on the server is realistic for this kind of attack. Moreover, because it does not tamper with the server itself, it can stay a long period of time on the server without being detected.

## 4. Metadata definition

In this paper, we define two different types of metadata: tangible metadata and intangible metadata. These definitions are one of the contributions of this work and are very important for the remainder of the paper.

### 4.1 Tangible metadata (TMD)

Tangible metadata are the metadata that can be written in the server. This type of metadata can be encrypted or not. For instance, they are: the file name, the file path, the timestamp, the owner id, etc. All the metadata are stored next to the file itself and we can encrypt them.

### 4.2 Intangible metadata (IMD)

Intangible metadata are the metadata created by an action and therefore we do not find them in a written form at any moment on the server drive. For example, when the server is uploading a file to a client, the packets sent by the server reveals an action is being executed. This action by itself leaks intangible knowledge on the file owner.

However, it is important to note there is not a strong limit between tangible and intangible metadata. Indeed, in order to execute an action (generating intangible metadata), the server usually needs to access tangible metadata to check the validity of the action using; for instance, the owner id, the file name, etc.

## 5. Motivation

Our work is motivated by the desire to meet the following objectives:

- Show and quantify the leak of metadata from a server.
- Provide a proof of concept in order to detect the leak of metadata.
- Show the importance of IMD in the leak.
- Provide a test-bed in order to test the security of future construction that will prevent the leak of metadata (see section 7)

## 6. Proposed solution

The method we propose is based on our new metadata definition in order to reveal both TMD and IMD from a running IM server. To collect TMD we can simply read the information on the server hard drive. However, TMD encryption is possible and will easily prevent this attack. Moreover, as we explained in section 4, TMD are usually needed in order to generate IMD. Therefore, we can focus our system on the detection of IMD, which is possible using two methods:

- Collect the trace leaked by an action on the server hard drive. E.g., collect the log file. However this method can easily be prevented using either encrypted logs or disabling (or limiting) the use of logs on the server.
- Collect the trace leaked by an action on the server RAM. This technique also has the advantage of retrieving the TMD used in order to validate the IMD.

Therefore, the solution proposed in this paper is to analyse the state of the RAM of a running server in order to collect metadata. Our method is based on live forensic technique. As the early goal of our study is to provide a proof of concept, and in order to make the system as simple as possible in the preliminary stages, we are not using a real IM server but a dummy IM server. In this way, we can concentrate our study on the live forensic memory acquisition and analysis. However, the use of a real IM server is the real practical use case for this technique.

## 7. Experiment

Our technique to collect metadata on an IM server uses live forensic software. Our system is constructed as the following:

- The server runs a simplified IM server.
- Two dummy users simulate a conversation responding to each other with a new message every 10 seconds. The content of the messages does not matter and is random.
- The attacker runs a program on the server that uses the Volatility and Rekall framework in order to retrieve and analyse the RAM system in real time.

### 7.1 Results

Although our detection system is in an early development stage, our program has been able to retrieve the record of every action it has been showed. The retrieved record consist of:

- IMD: An action has been executed on the server from a given user to another. Moreover, because the collection of information has been conducted over time, access patterns are also revealed.
- TMD: IDs of the users are revealed.

## 8. Discussion and future work

### 8.1 Improve our attack.

Although our collection system is not finalised yet and needs improvement, we created a proof of concept that shows how to collect metadata on a IM server. The improvements that need to be done are multiple. Then can be summarised as:

- Add the support for real IM server (with both encrypted and unencrypted messages).
- Develop a new metric and method to quantify the leak of metadata.

### 8.2 Oblivious RAM

Oblivious Random Access Machine (ORAM) [3] is a cryptographic construction that allows clients to access encrypted data residing on an untrusted storage server, while completely hiding the access patterns to storage. Particularly, the sequence of physical addresses accessed is independent of the actual data that the user is accessing.

One item of future work will focus on Oblivious RAM in order to show that Oblivious RAM techniques can be a good solution for preventing the specific intangible metadata attack we presented in this paper. Moreover, the attacker model presented here is compatible with the typical model detailed for traditional ORAM schemes.

## Acknowledgements

This work was supported, in part, by Irish Research Council grant GOIPG2016479 (research.ie), in part, by Science Foundation Ireland grant 10CEI1855 to Lero — the Irish Software Research Centre (www.lero.ie) and in part by

Science Foundation Ireland grant 13RC2094 and co-funded under the European Regional Development Fund through the Southern & Eastern Regional Operational Programme to Lero — the Irish Software Research Centre (www.lero.ie).

## References

Cohen, M. Rekall memory forensics framework. DFIR Prague, 2014.

Devadas, S., van Dijk, M., Fletcher, C. W., Ren, L., Shi, E. and Wichs, D. Onion oram: A constant bandwidth blowup oblivious ram. In Theory of Cryptography Conference, pages 145–174. Springer, 2016.

Dubin, R., Pele, O., Dvir, A. and Hadar, O. I know what you saw last minute-the chrome browser case.

Mayer, J., Mutchler, P. and Mitchell, J. C. Evaluating the privacy properties of telephone metadata. Proceedings of the National Academy of Sciences, page 201508081, 2016.

The volatility foundation - open source memory forensics. 2014.