J. Parallel Distrib. Comput. 122 (2018) 70–80

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Distributed shielded execution for transmissible cyber threats analysis

Yuxia Cheng [a], Qing Wu [a,*], Wenzhi Chen [b], Bei Wang [b]

[a] Hangzhou Dianzi University, 1158 No.2 Avenu, Baiyang Street, Hangzhou, China
[b] Zhejiang University, Zheda Road 38, Xihu District, Hangzhou, China

HIGHLIGHTS

- A distributed shielded execution framework for network cyber threats analysis is presented.
- A brief secure partition mechanism is proposed to decouple secure and normal spaces.
- A context-switch secure interface is proposed to reduce Iago attacks.
- New key–value encryption operations are integrated to prevent rollback and replay attacks.

ARTICLE INFO

Article history:
Received 3 January 2018
Received in revised form 20 April 2018
Accepted 14 July 2018
Available online 25 July 2018

Keywords:
Data security
Distributed shielded execution
Secure enclaves

ABSTRACT

Transmissible cyber threats have become one of the most serious security issues in cyberspace. Many techniques have been proposed to model, simulate and identify threats' sources and their propagation in large-scale distributed networks. Most techniques are based on the analysis of real networks dataset that contains sensitive information. Traditional in-memory analysis of these dataset often causes data leakage due to system vulnerabilities. If the dataset itself is compromised by adversaries, this threat cost would be even higher than the threat being analysed. In this paper, we propose a new distributed shielded execution framework (Disef) for cyber threats analysis. The Disef framework enables efficient distributed analysis of network dataset while achieves security guarantees of data confidentiality and integrity. In-memory dataset is protected by using a new encrypted key–value format and could be efficiently transferred into Intel SGX enabled enclaves for shielded execution. Our experimental results showed that the proposed framework supports secure in-memory analysis of large network dataset and has comparable performance with systems that have no confidentiality and integrity guarantees.

1. Introduction

Transmissible cyber threats have become one of the most serious security issues in cyberspace, such as computer viruses spreading on the Internet, rumours propagating in social networks, and rolling blackouts in smart grids. Many techniques have been proposed to model, simulate and identify threats' sources [21] and their propagation in large-scale distributed networks. Most analysis of transmissible cyber threats [20,44] are based on real network dataset that contains sensitive information. However, in-memory analysis of these dataset in traditional systems including both cloud and local servers often causes data leakage due to system vulnerabilities. If the dataset itself is compromised by adversaries during the cyber threat analysis process, this threat cost would be even higher than the threat being analysed [1].

Today's cloud and local computer systems are built based on a traditional hierarchical security model that mainly targets at protecting the privileged code of system software from untrusted code of applications, and does not aim at protecting user data from being accessed by privileged code [6]. Therefore, the cyber threats analysis code must trust: (i) the provider's hardware used to run their applications; (ii) the privileged system software (host OS, hypervisor, firmware) and the full stack of system management software; (iii) the system administrators and other staffs that have entitlements to access hardware facilities. From this point of view, the trusted computing base (TCB) to the cyber threats analysis users is very large and uncertain. And there were many data leakage incidents [11] happened due to software and administrative vulnerabilities. Although privacy preserving [12] could be used to hide sensitive information, there are many scenarios that require encrypt/decrypt processing to protect data security [49].

With the increasing security concerns in the cyberspace, new hardware technologies for trusted computing have evolved

* Corresponding author.
E-mail address: wuqing@hdu.edu.cn (Q. Wu).

https://doi.org/10.1016/j.jpdc.2018.07.014
0743-7315/

rapidly. Trusted execution environments (TEEs) can provide applications with a secure execution context. Even if the rest of software (hypervisor, OS, etc.) are compromised, the application inside TEE can still remain trusted. The recent Intel Software Guard Extensions (SGX) [17] and the ARM TrustZone [2] are among the promising techniques that protect user's sensitive code and data against malicious software that attempts to compromise its integrity and confidentiality.

Our objective is to provide cyber threat analysis users a distributed shielded execution system to protect their sensitive in-memory data in cyberspace. Particularly, we focus on the distributed key–value store system to process network dataset for cyber threats analysis. The in-memory key–value store systems are widely used in the distributed systems. User's data resides in the memory are most vulnerable to leakage, while data stored in the storage devices can be securely encrypted.

In this paper, we aim to protect in-memory key–value store system based on the Intel SGX trusted techniques. Cloud users need only trust the hardware and their own applications, thus reducing TCB to the minimum. By design and implementation of secure key–value store system, we try to achieve the following goals: (i) Users' code and data are guaranteed to reside in the memory of the trusted physical machine. (ii) Confidentiality and integrity of user's private data are protected even if the system software stack is compromised and controlled by adversaries. (iii) The performance gap between the secure key–value store system and the original system without security enhancement should be kept small.

Reaching the above three goals faces some challenges.

(1) Due to the SGX hardware restrictions, the trusted space (named enclave) could only use a limited number of EPC (Enclave Page Cache) pages (a total of 128 MB for current processor). However, a typical key–value store system requires large memory capacity that far exceeds the available SGX memory resources. Although the SGX provides paging mechanism, paging brings very high overhead that is intolerable for latency sensitive applications.

(2) The data transmission between enclave and outside world will bring potential security vulnerabilities. The system calls invoked in the enclave must switch to the untrusted OS, which is vulnerable to Iago attacks [9]. The untrusted OS may return malicious data into the enclave, and the enclave may leak secret data to the outside world. The key–value store system has frequent system calls of memory and network I/O operations. Simply porting the key–value store system into the enclave will not only expose many interfaces that increase the potential attack surfaces, but also incurs high performance penalty.

(3) While the attackers cannot directly access code and data inside enclaves, they can potentially raise known attacks from outside the enclave, such as replay attacks [40], and rollback attacks [22]. How to prevent or mitigate these attacks without significantly compromising performance is also a big challenge in the design of a secure key–value store system.

In this paper, we propose a distributed shielded execution framework for cyber threats analysis (short for Disef) that leverages key–value store system and the new Intel SGX processors to protect in-memory private dataset. The main contributions of this paper is described as follows:

(1) We analyse the workflow of key–value store system and propose a brief method to classify security sensitive operations (related to updating key–value pairs) and other non-sensitive operations. Only the security sensitive operations are executed inside the enclave, which significantly reduces memory footprint required in the enclave. Instead of putting all code and data into the enclave, we properly encrypt key–value pairs and put them in the non-enclave memory region. In this manner, Disef overcomes the hardware limitation of small enclave memory and the large overhead caused by SGX paging. Disef can use normal memory pages to store large amount of key–value pairs with near native performance and still preserve the confidentiality and integrity of user data.

(2) We design one secure interface between the trusted enclave and the untrusted non-enclave space. The minimum number of interfaces could reduce many potential vulnerabilities. As memory allocation and network I/O operations are irrelevant with modifying the encrypted key–value pairs, we put these codes and system calls outside the enclave to reduce possible Iago attacks. This design not only reduces security risks caused by invoking system calls inside the enclave, but also reduces the performance penalty caused by data transmission across the enclave boundary. The parameters passed through the secure interface are carefully examined before running the enclave code.

(3) We propose a client-controlled data protection mechanism to mitigate some potential attacks, such as replay attacks and rollback attacks. Preventing replay and rollback attacks are expensive in an untrusted cloud environment. We propose two efficient protection methods integrated with the Disef system: (i) we design a new KEY–VALUE format and integrate with a version number to prevent rollback attacks. (ii) we design a new update VALUE format and add a monotonic update number to prevent replay attacks. These protection mechanisms are processed along with normal key–value operations without extra communication, which have low performance overhead. In the client, Disef establishes a fast version tracking table (VTT) to record the version number of certain key–value pairs that are stored in the server. The version number of a key–value pair retrieved from the server is checked to prevent rollback attacks. When the key–value pairs are updated in the Disef enclave, the enclave checks both the version number and the update counter to prevent possible rollback and replay attacks.

We implement the Disef system based on the memcached project [28]. The Disef system consists of three parts, the Disef enclave, the Disef memcached, and the Disef client. We leverage the Intel SGX remote attestation mechanism to establish secure channel and share secrets between the enclave and client. Our experimental results on the real SGX machine showed that the Disef system has comparable performance with the baseline memcached, while Disef achieves stronger security guarantees of user dataset.

The remainder of this paper is organised as follows. Section 2 describes the related work. Section 3 provides background of Intel SGX. Sections 4 and 5 describe the Disef system design and implementation. Section 6 presents performance evaluation. And finally, Section 7 concludes our work.

2. Related work

There have been extensive studies on transmissible cyber threats analysis techniques, including malware propagation analysis and control [45,46], rumour propagation modelling and restraining [7,44], and threat sources identification [20,21]. These techniques mainly focus on theoretical methods based on pre-acquired network dataset. However, their practical deployment in real systems still faces many limitations, including how to protect the network dataset while in-memory analysis. New cyber threats analysis approaches can be deployed in practical systems under the distributed shielded execution with Disef. Therefore, Disef is orthogonal to the design of cyber threats analysis techniques.

The Intel SGX technique was first introduced in [13,14,26]. Many SGX-related studies [3,6,15,30,38] have been proposed in the cloud environment. Researchers in [6] proposed the architecture named Haven that enables unmodified windows applications to run on the SGX mode in an untrusted cloud. Haven loads both a trusted application and a trusted library OS into an enclave. S. Arnautov et al. [3] proposed a secure container mechanism named