HARDWARE-ASSISTED SECURE EXECUTION ON REMOTE, UNTRUSTED SYSTEMS

SANDEEP KUMAR

AMAR NATH AND SHASHI KHOSLA SCHOOL OF INFORMATION TECHNOLOGY

INDIAN INSTITUTE OF TECHNOLOGY DELHI

JULY 2024

HARDWARE-ASSISTED SECURE EXECUTION OF PROGRAMS ON UNTRUSTED PLATFORMS

by

SANDEEP KUMAR

AMAR NATH AND SHASHI KHOSLA SCHOOL OF INFORMATION TECHNOLOGY

Submitted

in fulfillment of the requirements of the degree of **Doctor of Philosophy**

to the

Indian Institute of Technology Delhi

JULY 2024

For my grandfather,

Late Sh. Mahendra Prasad

Certificate

This is to certify that the thesis titled **HARDWARE-ASSISTED SECURE EXECUTION OF PROGRAMS ON UNTRUSTED PLATFORMS** being submitted by **Mr. Sandeep Kumar** for the award of **Doctor of Philosophy** in **Information Technology** is a record of bona fide work carried out by him under my guidance and supervision at the **Amar Nath and Shashi Khosla School of Information Technology, Indian Institute of Technology Delhi**. The work presented in this thesis has not been submitted elsewhere, either in part or full, for the award of any other degree or diploma.

Smruti Ranjan Sarangi

Professor

Department of Computer Science and Engineering

Indian Institute of Technology Delhi

New Delhi- 110016

Acknowledgements

I had a great experience during my Ph.D. and will cherish it for the rest of my life. I learned great skills that made me a better person and researcher, and I made a few lifelong friends.

I want to start by thanking my Ph.D. supervisor, **Dr. Smruti R. Sarangi**. I am grateful for his constant guidance, which has made me a better researcher. I also deeply appreciate his push to understand the fundamentals of any problem before starting a project. This one piece of advice has helped me with all of my projects. I also loved our conversation about life and different non-Ph.D.-related topics. He taught me the importance of hard work. I will not forget working with him at two in the morning to meet a paper submission deadline. I thank him again. I also thank my student research committee members, **Prof. Anshul Kumar**, **Prof. Kolin Paul**, and **Prof. Shouribrata Chatterjee**, for providing valuable feedback on my research.

I also thank my master's advisor, **Prof. K. Gopinath**. He was gracious enough to accept me as his student and guided me during my master's and my stint as a research assistant. I will be forever grateful to him for his guidance and support.

I am grateful to the administrative staff in the department: Mr. Rajesh Kumar, Mr. Suresh, Ms. Vandana, Ms. Manju, and Ms. Aditi. I am especially grateful to Ms. Aditi. She went beyond her duties to help me fulfill the formalities for various things.

During the course of my Ph.D., I found many great friends: Dr. Arindam Bhatacharya, Dilpreet Kaur, Dr. Ovia Shesadri, Dr. Dishant Goel, Dr. Vinayak Gupta, Dr. Diksha Moolchandani, Dr. Priyanka Singla, Dr. Anupam Sobti, Omais Shafi, Saurabh Tewari, Dr. Shubhankar Suman, Dr. Hameedah Sultan, Dr. Janibul Bashit, Abhisek Panda, and Rahul Kanyal.

I am deeply grateful to Dr. Diksha Moolchandani for collaborating on multiple projects and fruitful lab discussions. I learned many things from her regarding resilience, technical background, and general research direction. I would also like to thank Abhisek Panda. We collaborated on many projects, and I will be grateful for his help in project discussions, experiment setup, and finalizing the projects. He is a great person, friend, and researcher. I wish him all the

best for his Ph.D. journey. I thank Arindam, Dilpreet, Ovia, Dishant, Vinayak, Sikha, and Anupam for those fun lab discussions on research and life, impromptu lunches, and walks around the campus.

I would also like to thank Kaushal Bisht and Oindrila Mitra for being there with me through the Ph.D. journey and for all those house parties that acted as a stress buster and motivated me. I am deeply thankful to them. Apart from them, I would like to thank Akshar Kaul, Remish Minz, Chandrahas Dewangan, Rakesh Malviya, Rahul Sharma, Ankit Dixit, Saurabh Mondal, Kiran Prasad, Anirban Laha, and Suvam Mukherjee for being a part of my life at different phases and making it fun to live. I thank Aditi Puri for the constant push to submit the thesis.

I was fortunate to complete a 7-month internship at Intel Labs in Bangalore, India. I want to thank Aravinda Prasad and Sreenivas Subramoney for giving me the opportunity and their constant guidance during the internship, which helped me grow as a researcher.

Finally, I would like to thank my parents, Mr. Ramjee Thakur and Mrs. Rema Thakur, for their unconditional love and support throughout my life. Nothing would have been possible without them. I also thank my brother, Dr. Nilesh Kumar, for his constant life, fitness lessons, and all those fun moments we shared. I am deeply grateful for the love and support I received from my grandparents, uncles, aunts, and cousins. I especially thank my grandfather, Mr. Mahendra Prasad, for his immense love, support, and belief that I can achieve anything. I wish he were here to see this.

Sandeep Kumar

Abstract

Securing code and data in a cloud computing environment is a quintessential problem. Cloud-based remote systems are untrusted, and thus, there is a need to ensure their trustworthiness. Moreover, many such applications may be dealing with sensitive data, which increases the incentive for a malicious adversary to try to subvert the execution flow.

The traditional, software-only solution to protect access to code and data is to encrypt them when they are resident in secondary storage media such as the hard disk or to use special access-controlling functions such as license managers that check the existence of a key with the client machine. In this thesis, we show that such software-only solutions, using a provable, strong, unbreakable cryptographic primitive, are vulnerable to a malicious entity with admin-level privileges. The ingenuity of such an attack is that it does not interfere with the functioning of the license check manager method but rather with its outcome, rendering the entire defense method ineffective. We designed a novel *control flow bending* attack on the execution of popular open-source software using their control flow graph (or CFG) representation and broke the license check manager of many commercial applications, including MySQL.

The community in this space has also proposed hardware-based defense solutions where the execution platform guarantees the security of an application. Although effective, pure hardware-based solutions have a long development cycle and are not easily patchable in case of a new attack. Hence, *hybrid methods*, where hardware and software work in tandem, are gaining popularity. In this setting, the hardware does the heavy lifting and exposes a common interface to gain insight into the execution of an application. The information obtained from these modules is then processed in the software to provide security to an application.

In this thesis, our main focus is to ensure secure execution on a remote, untrusted server. Hence, we focus on trusted execution environments (or TEEs) designed to ensure an application's security in the presence of a privileged attacker. In a TEE, the hardware ensures the code and data's confidentiality, integrity, and freshness. We selected Intel Software Guard eXtension (or Intel SGX), a TEE solution from Intel, for our experiments due to its wide availability and

popularity. In SGX, the code and data remain encrypted on the disk and the main memory and are only decrypted when brought into the chip. This approach ensures protection from privileged attackers, such as a malicious operating system (OS) or hypervisor. However, there are certain challenges here. As the OS is not a part of the trusted computing base (TCB), the "secure" part of applications cannot make system calls, issue certain instructions, or take advantage of shared libraries.

We start by addressing a basic requirement for working with Intel SGX: developing *a standard benchmark suite* for evaluating the performance of applications built on SGX. Prior work in this area used workloads that did not provide a holistic view of the execution characteristics of the SGX. We created a suite of applications from different domains showing different characteristics in terms of CPU usage and memory usage to gain an in-depth insight into the workings of SGX. In order to measure the efficiency of our proposed benchmark suite, we also define a scoring method to gauge the quality of a benchmark suite. The definition of these scores is based on a list of desirable features that have been qualitatively enunciated in prior work (notably by the authors of the Parsec benchmarks). We use the scores to show that our benchmark suite for SGX (SGXGauge) qualitatively outperforms other suites in this domain. Furthermore, using our novel set of metrics, we were able to critically analyze the pros and cons of the following benchmark suites and explain qualitative observations made by other authors more quantitatively: SPEC, Parsec, and Ligra.

Subsequently, we propose a novel approach to prevent control-flow bending (or CFB) attacks by leveraging SGX. Our approach is to "*handicap*" a binary by placing a small portion of secure functions – the license check and its forward pass – inside Intel SGX. Access to the secure function is only allowed after successfully validating the license file within SGX. Attempting to bypass the license check manager (as in a CFB attack) will result in a crash. Although secure, this approach has a high overhead of checking license checks due to a remote attestation step for every access to a secure function. To address this performance issue, we designed a novel solution to reduce the cost of multiple license checks by enabling a partial number of authentication requests to be processed locally. Next, we enabled access to an SGX-optimized file system for an application running within SGX. Our design uses a combination of encryption and hashing to ensure the confidentiality, integrity, and freshness of the data blocks written to the file system. Our design maintains a small cache of data and metadata blocks to ensure fast read and write operations.

As our final contribution, we combine all the previous methods and propose a novel method to enable live migration of large SGX enclaves from one machine to another. The first version of SGX supported a secure, usable memory of 92 MB. However, the second generation of SGX

called *scalable SGX*, can support secure memory up to 512 GB. Prior work solved the migration challenge by using the traditional stop-and-copy approach. However, this approach fails to scale to the gigabytes of secure memory in the scalable SGX. We identified key challenges in adopting a live migration scheme for secure enclaves. We propose a novel solution that addresses these challenges and enables secure live migration of a large SGX enclave from one machine to another.

संक्षेप

क्लाउड कंप्यूटिंग वातावरण में कोड और डेटा सुरक्षित करना एक सर्वोत्कृष्ट समस्या है। बादल-आधारित रिमोट सिस्टम अविश्वसनीय हैं, और इस प्रकार उनकी विश्वसनीयता सुनिश्चित करने की आवश्यकता है। इसके अलावा, ऐसे कई एप्लिकेशन संवेदनशील डेटा से निपट सकते हैं - इससे निष्पादन प्रवाह को नष्ट करने की कोशिश में दुर्भावनापूर्ण प्रतिद्वंद्वी का प्रोत्साहन बढ़ जाता है।

कोड और डेटा तक पहुंच की सुरक्षा के लिए पारंपरिक, सॉफ्टवेयर-केवल समाधान उन्हें एन्क्रिप्ट करना है जब वे हार्ड डिस्क जैसे द्वितीयक भंडारण मीडिया में रहते हैं, या लाइसेंस प्रबंधकों जैसे विशेष एक्सेस-नियंत्रण कार्यों का उपयोग करते हैं जो डेटा के अस्तित्व की जांच करते हैं। क्लाइंट मशीन के साथ कुंजी। इस थीसिस में, हम दिखाते हैं कि ऐसे सॉफ़्टवेयर-केवल समाधान, एक सिद्ध मजबूत, अटूट क्रिप्टोग्राफ़िक प्रिमिटिव का उपयोग करते हुए, व्यवस्थापक-स्तर के विशेषाधिकारों के साथ एक दुर्भावनापूर्ण इकाई के प्रति संवेदनशील होते हैं। इस तरह के हमले की सरलता यह है कि यह लाइसेंस जांच प्रबंधक पद्धति के कामकाज में हस्तक्षेप नहीं करता है, बल्कि इसके परिणाम के साथ, संपूर्ण रक्षा पद्धति को अप्रभावी बना देता है। हमने उनके नियंत्रण प्रवाह ग्राफ (या सीएफजी) प्रतिनिधित्व का उपयोग करके लोकप्रिय ओपन सोर्स सॉफ़्टवेयर के निष्पादन पर एक नया नियंत्रण प्रवाह झुकने वाला हमला डिज़ाइन किया और MySQL सहित कई वाणिज्यिक अनुप्रयोगों के लाइसेंस जांच प्रबंधक को तोड़ दिया।

इस क्षेत्र में समुदाय ने हार्डवेयर-आधारित रक्षा समाधान भी प्रस्तावित किया है जहां निष्पादन प्लेटफ़ॉर्म किसी एप्लिकेशन की सुरक्षा की गारंटी देता है। हालांकि प्रभावी, शुद्ध हार्डवेयर-आधारित समाधानों का विकास चक्र लंबा होता है और नए हमले की स्थिति में इन्हें आसानी से पैच नहीं किया जा सकता है। इसलिए, हाइब्रिड तरीके, जहां हार्डवेयर और सॉफ्टवेयर मिलकर काम करते हैं, लोकप्रियता हासिल कर रहे हैं। इस सेटिंग में, हार्डवेयर भारी भारोत्तोलन करता है और किसी एप्लिकेशन के निष्पादन में अंतर्दृष्टि प्राप्त करने के लिए एक सामान्य इंटरफ़ेस को उजागर करता है। इन मॉड्यूल से प्राप्त जानकारी को किसी एप्लिकेशन को सुरक्षा प्रदान करने के लिए सॉफ़्टवेयर में संसाधित किया जाता है।

इस थीसिस में, हमारा मुख्य फोकस दूरस्थ, अविश्वसनीय सर्वर पर सुरक्षित निष्पादन सुनिश्चित करना है। इसलिए, हम विश्वसनीय निष्पादन वातावरण (या टीईई) पर ध्यान केंद्रित करते हैं जो एक विशेषाधिकार प्राप्त हमलावर की उपस्थिति में एप्लिकेशन की सुरक्षा सुनिश्चित करने के लिए डिज़ाइन किया गया है। टीईई में, हार्डवेयर कोड और डेटा की गोपनीयता, अखंडता और ताजगी सुनिश्चित करता है। हमने इसकी व्यापक उपलब्धता और लोकप्रियता के कारण अपने प्रयोगों के लिए इंटेल से एक टीईई समाधान इंटेल सॉफ्टवेयर गार्ड एक्सटेंशन (या इंटेल एसजीएक्स) का चयन किया। एसजीएक्स में, कोड और डेटा डिस्क और मुख्य मेमोरी पर एन्क्रिप्टेड रहते हैं और चिप में लाए जाने पर ही डिक्रिप्ट होते हैं। यह दृष्टिकोण दुर्भावनापूर्ण ऑपरेटिंग सिस्टम (ओएस) या हाइपरवाइजर जैसे

विशेषाधिकार प्राप्त हमलावरों से सुरक्षा सुनिश्चित करता है। हालाँकि, यहाँ कुछ चुनौतियाँ हैं। चूंकि ओएस विश्वसनीय कंप्यूटिंग बेस (टीसीबी) का हिस्सा नहीं है, इसलिए एप्लिकेशन का "सुरक्षित" हिस्सा सिस्टम कॉल नहीं कर सकता, कुछ निर्देश जारी नहीं कर सकता और साझा लाइब्रेरी का लाभ नहीं उठा सकता।

हम इंटेल एसजीएक्स के साथ काम करने के लिए एक बुनियादी आवश्यकता को संबोधित करते हुए शुरुआत करते हैं: एसजीएक्स पर निर्मित अनुप्रयोगों के प्रदर्शन के मूल्यांकन के लिए एक मानक बेंचमार्क सूट विकसित करना। इस क्षेत्र में पहले के काम में ऐसे कार्यभार का उपयोग किया गया था जो एसजीएक्स की निष्पादन विशेषताओं का समग्र दृष्टिकोण प्रदान नहीं करता था। हमने एसजीएक्स के कामकाज में गहराई से जानकारी हासिल करने के लिए सीपीयू उपयोग और मेमोरी उपयोग के संदर्भ में विभिन्न विशेषताओं को दिखाने वाले विभिन्न डोमेन से अनुप्रयोगों का एक सूट बनाया। हमारे प्रस्तावित बेंचमार्क सूट की दक्षता को मापने के लिए , हम बेंचमार्क सूट की गुणवत्ता को मापने के लिए एक स्कोरिंग विधि भी परिभाषित करते हैं। इन अंकों की परिभाषा वांछनीय विशेषताओं की एक सूची पर आधारित है जिन्हें पिछले काम में गुणात्मक रूप से प्रतिपादित किया गया है (विशेषकर पारसेक बेंचमार्क के लेखकों द्वारा)। हम यह दिखाने के लिए स्कोर का उपयोग करते हैं कि SGX (SGXGauge) के लिए हमारा बेंचमार्क सूट, गुणात्मक रूप से इस डोमेन में अन्य सुइट्स से बेहतर प्रदर्शन करता है। इसके अलावा, मेट्रिक्स के हमारे नए सेट का उपयोग करके, हम निम्नलिखित बेंचमार्क सुइट्स के पेशेवरों और विपक्षों का गंभीर रूप से विश्लेषण करने और अन्य लेखकों द्वारा की गई गुणात्मक टिप्पणियों की व्याख्या करने में सक्षम थे।
अधिक मात्रात्मक रूप से: स्पेक, पारसेक और लिग्रा ।

इसके बाद, हम एसजीएक्स का लाभ उठाकर नियंत्रण-प्रवाह झुकने (या सीएफबी) हमलों को रोकने के लिए एक नया दृष्टिकोण प्रस्तावित करते हैं। हमारा दृष्टिकोण इंटेल एसजीएक्स के अंदर सुरक्षित कार्यों - लाइसेंस जांच और इसके फॉरवर्ड पास - का एक छोटा सा हिस्सा रखकर बाइनरी को "विकलांग" करना है। एसजीएक्स के भीतर लाइसेंस फ़ाइल को सफलतापूर्वक मान्य करने के बाद ही सुरक्षित फ़ंक्शन तक पहुंच की अनुमति दी जाती है। लाइसेंस जांच प्रबंधक को बायपास करने का प्रयास (सीएफबी हमले के रूप में) के परिणामस्वरूप दुर्घटना होगी। हालांकि सुरक्षित, इस दृष्टिकोण में एक सुरक्षित फ़ंक्शन तक प्रत्येक पहुंच के लिए दूरस्थ सत्यापन चरण के कारण लाइसेंस जांच की जांच करने का एक उच्च ओवरहेड है। इस प्रदर्शन समस्या का समाधान करने के लिए, हमने आंशिक संख्या में प्रमाणीकरण अनुरोधों को स्थानीय स्तर पर संसाधित करने को सक्षम करके एकाधिक लाइसेंस जांच की लागत को कम करने के लिए एक नया समाधान तैयार किया है। इसके बाद, हमने SGX के भीतर चल रहे एप्लिकेशन के लिए SGX-अनुकूलित फ़ाइल सिस्टम तक पहुंच सक्षम की। हमारा डिज़ाइन फ़ाइल सिस्टम में लिखे गए डेटा ब्लॉक की गोपनीयता, अखंडता और ताजगी सुनिश्चित करने के लिए एन्क्रिप्शन और हैशिंग के संयोजन का उपयोग करता है। हमारा डिज़ाइन तेजी से पढ़ने और लिखने के संचालन को सुनिश्चित करने के लिए डेटा और मेटाडेटा ब्लॉक का एक छोटा कैश बनाए रखता है।

अपने अंतिम योगदान के रूप में, हम सभी पिछले तरीकों को जोड़ते हैं और बड़े एसजीएक्स एन्क्लेव के एक मशीन से दूसरे मशीन में लाइव माइग्रेशन को सक्षम करने के लिए एक उपन्यास विधि का प्रस्ताव करते हैं। एसजीएक्स का पहला संस्करण 92 एमबी की सुरक्षित, उपयोग योग्य मेमोरी का समर्थन करता था। हालाँकि, SGX की दूसरी पीढ़ी, जिसे स्केलेबल SGX कहा जाता है, 512 जीबी तक सुरक्षित मेमोरी का समर्थन कर सकती है। पिछले कार्य ने पारंपरिक स्टॉप-एंड-कॉपी दृष्टिकोण का उपयोग करके प्रवासन चुनौती को हल किया। हालाँकि, यह दृष्टिकोण स्केलेबल SGX में मौजूद गीगाबाइट सुरक्षित मेमोरी को स्केल करने में विफल रहता है। हमने सुरक्षित एन्क्लेव के लिए लाइव माइग्रेशन योजना को अपनाने में आने वाली प्रमुख चुनौतियों की पहचान की और एक उपन्यास समाधान का प्रस्ताव दिया जो इन चुनौतियों का समाधान करता है और एक बड़े एसजीएक्स एन्क्लेव के एक मशीन से दूसरे मशीन में सुरक्षित लाइव माइग्रेशन को सक्षम बनाता है।

Contents

List of Figures

List of Tables