Intel SGX
The Next Generation of Security

Kristopher Willis
Computer Science and Engineering
University of South Florida
Tampa, FL, USA
kww@usf.edu

ABSTRACT

Computer security is an ongoing threat to users around the world. Intel has created a new, innovative way of securing personal and identifiable information using extra processor instructions called Software Guard Extension or (SGX). Intel SGX enables applications to execute code and protect sensitive data from within the protected enclave giving developers direct control over application security. This protection is accomplished by several different cryptographic schemes that are detailed throughout this paper. This paper also presents the use cases of Intel SGX. What attack surfaces may be achievable and how the enclave may have the potential to leak information. I also present SGX pitfalls and how Intel might be able to improve them. Finally, the paper gives insight into Intel CET technology and how it will revolutionize application security with the protection of return oriented programming.

INTRODUCTION

Intel has a new, innovative way to secure applications and data within the hardware layer. Intel refers to it as Software Guard Extension or (SGX). Intel SGX adds 17 new instructions that enable applications to run in a secure enclave. The secure enclave is a trusted space to execute and write data that should be protected from untrusted and unwanted rouge processes. Think of this technology as a more enhanced version of the TPM (Trusted Platform Module). The TPM allows users to store critical sensitive data such as credit card, medical records, biometric information, and passwords within encrypted hash values. Intel SGX takes this a step forward by enabling full applications to run inside a secure enclave environment.

Intel SGX also offers full random access memory encryption. Lately, there has been an increase in attack vectors going after data stored in RAM. Intel SGX would eliminate this attack vector and give users the peace of mind that their data would be safe when moving data between the processor, RAM, and storage. This is even more important in the cloud computing environment where you could have multiple users sharing RAM modules. This technology is a part of the Intel SGX suite and requires no extra steps beyond the Intel SGX compiler.

Intel SGX is not backwards compatible and currently only works with Intel 6th Generation Xeon, i7, and i5 processors. At first, Intel only developed the new instructions to work with Microsoft operating systems however, Intel later released a driver to support Linux based systems on GitHub. This means that, at the moment, the adoption rate of Intel SGX is very minimal. Intel also requires that application developers use a special compiler for their code to allow the new instructions. The Intel SGX compiler is only supported in Intel's development environment and Microsoft Visual Studio. Furthermore, Intel is only supporting C and C++ programming languages. Intel also requires that you must license for commercial use while also requesting access to their development services in order to create certificates which will be explained in more depth about later. Essentially, Intel has made it very difficult to use this technology out of the box which is why currently little applications take advantage of it.

AMD will have a competing technology that was introduced in 2016 which will be in their new Opteron and Zen processors coming out in early 2017. AMD has taken a different approach by separating their enclave technology from their RAM encryption technology. This will work well for AMD in the short term, as it allows application developers the opportunity to rollout security features. While AMD's technology is less documented, it would also require special compiled code which means we could see different executables for both AMD and Intel. This could also mean a divergence of security applications where we have competing technologies on competing x86-64 processors.

OVERVIEW

In this paper the intentions are to look at Intel SGX from the hacker perspective. Intel is mainly using SGX as a defensive technology however, could a hacker use this as an offensive weapon? Has Intel properly used encryption and certificate methods to keep data from leaking? Are the new instructions sound, and can disassemblers see the enclave? Are there ways to bypass the enclave and have applications run outside without

the user or developer knowing? These are just some of the questions that will be answered throughout this paper.

While Intel SGX will be the focus of this paper, Intel Control Flow Enforcement Technology or CET is actually the main reason of interest in Intel SGX. Intel CET gives protection to the application for attacks such as return oriented programming or (ROP). As a software reverse engineer we have several avenues to look at when attacking software. Mainly, we look at NX, ASLR, Canary, and ROP. For most applications today, the compiler will protect the application by turning on protections for NX and implement ASLR and Canary. ROP is the last resort in low hanging fruit for the attacker. If there were protections against ROP the software hacker will be at a much higher disadvantage when trying to reverse engineer the application. CET will drastically change the game when applications start using the technology. CET relies on Intel SGX in order to work properly.

ENCLAVE

Intel achieves trusted computing functionality by using secure enclaves which is the center around Intel SGX. Intel uses new instructions specially designed for SGX to enable the enclave. The enclave in its most basic form is a separated and encrypted region within the processor for sensitive code and data. The enclave is only decrypted within the Intel processor which allows data to be safe, even when it moves through the RAM. The enclave becomes part of the application and has full access to memory.

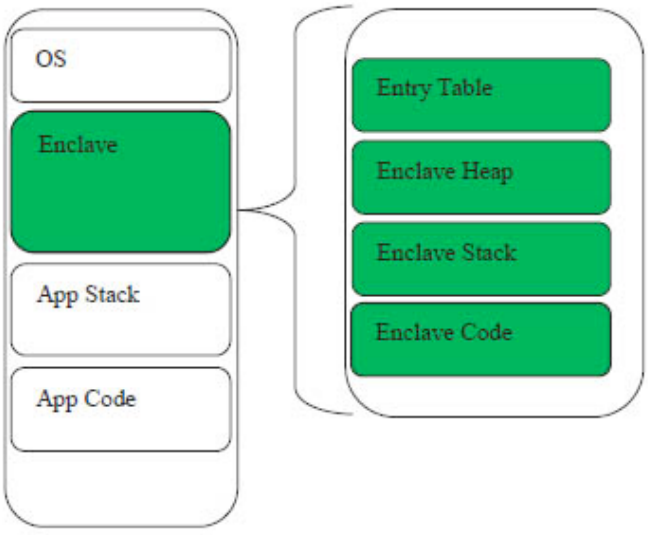

Figure 1: Enclave Stack [1]

When an application is compiled to use SGX, there are a couple different things that happen when a user opens up the application for the first time. First, the application is going to call home to Intel to make sure the application was developed by an identified trusted developer. Intel requires that all developers who want to use SGX be identified as a trusted developer by Intel. The reason for this has to do with Intel being concerned about malware and botnet creators from creating SGX enabled applications that would require extensive investigation in order to know if they were running inside the secure enclave. Once the application has been identified, the processor is then going to get a special certificate, which will be how the application will communicate through the enclave. At this point, the application can pass code marked as secure through the enclave where it will be safe and encrypted.

The enclave has several special properties that make Intel SGX unique. When an application runs through the enclave, data that goes to the processor is encrypted along with the RAM and storage. When the encrypted data gets to the processor the processor will then decrypt the data and allow the application to ran. This ensures that applications which run through SGX will be secure through the entire system. While there have been other solutions available that put data in secure spaces Intel SGX allows this at the application layer and drastically reduces the chances of leaked secure data to be accessed by rouge processes.

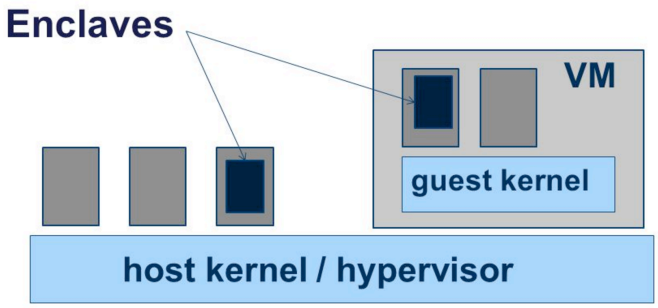

Figure 2: How the enclave interacts with the hypervisor [1]

Intel SGX is also supported in hypervisors. The enclave will be especially important in this space as we move more into mobile and cloud environments. In the not too distant future, many businesses will solely rely on cloud computing and hypervisors to run their enterprise. As we move closer to that reality, Intel SGX and the enclave can effectively secure data which is passed through. What is particularly unique about hypervisors and the Intel SGX enclave is that data can freely pass through the enclave as secure data. This means that everything that is run in the virtual machine will be encrypted and only executed only within the enclave. It will not matter at that point, if a developer has been trusted by Intel, if the operating system is supported, or if it's been compiled using SGX.

The enclave is most important part of the SGX platform. While there is a lot to SGX the main reason to use this environment is for the enclave. Security improvements in the last several years has been on how to compartmentalize data. VLANs have done it for networking to an extent. We have used hypervisor technology to separate workstations. We use

containers to separate applications from one another on a system. The processor enclave is the next logical step in securing application data and making it much harder for attackers to obtain data. Because of the enclave, Intel has made it much harder to attack systems through making the attack vector much smaller. We will talk about the attack vectors of SGX later on in this paper, but it's worth pointing out that Intel SGX has pioneered this new technology that is now being tested in many different spaces, including mobile and government entities that are trying to protect data on the go and our national security.

ARCHITECTURE

Intel SGX adds 17 new instructions to the x86 instruction set. These instructions are added by compiling C/C++ code in either the Intel SGX specific IDE or Microsoft Visual Studio that supports SGX. Let's dive into the instructions and see how they work.

SGX Supervisor Instructions

- ENCLS[EADD]: Add a page
- ENCLS[EBLOCK: Block an EPC page
- ENCLS[EDBGRD]: Create an enclave
- ENCLS[EDBGW]: Write data by debugger
- ENCLS[EEXTEND]: Extend EPC page measurement
- ENCLS[EINIT]: Initialize an enclave
- ENCLS[ELBD] Load an EPC page as blocked
- ENCLS[ELDU] Load an EPC page as unblocked
- ENCLS[EPA] Add version array
- ENCLS[EREMOVE] Remove a page from EPC
- ENCLS[ETRACK] Activate EBLOCK checks
- ENCLS[EWB] Write back/ invalidate an EPC page

The supervisor instructions are the instructions that are controlled by the compiler. The most important instructions here are the create and initialization of the enclave. These are the only two instructions that are able to be seen, before the rest is encrypted though the enclave.

Another important instruction to note is the EDBGW instruction which writes data to the debugger. Intel has added a debugger to SGX for two reasons. First, the debugger helps in identifying code that could escape the enclave. Once the enclave has been made, no other parts of the code can be read which could cause issues. The debugger in this case will write data to a log that can only be read in the emulation environment with the proper certificate data from Intel. Intel protects the log data and removes the log capability after the code has been compiled. The other reason the debugger exists is that SGX will try to fix sigfaults from occurring. Intel has enabled a basic check as a last resort to protect data from leaving the enclave. If the debugger cannot properly fix the issue, the application will sigfault and send back the results to Intel.

The other instructions within the supervisor group help run the enclave successfully. The EPC in particular is how the enclave and the exposed hardware communicate instructions. Pages are created within the SGX module that becomes the buffer zone between the enclave and the hardware. The SGX supervisor instructions are the backbone of how SGX operates.

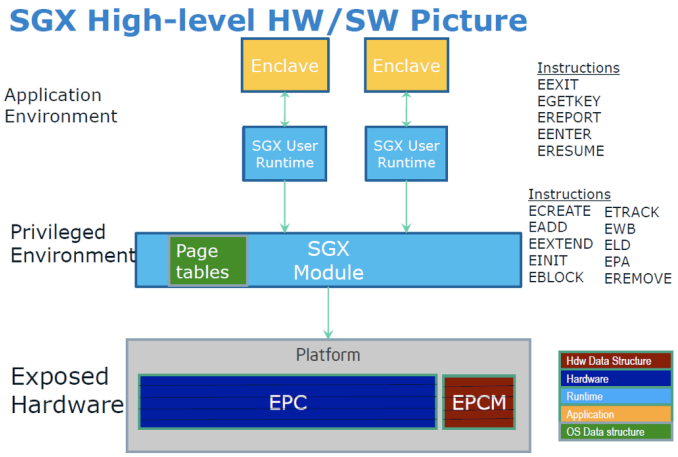

Figure 3: High level overview of the SGX instructions [1]

SGX User Instructions

- ENCLU[EENTER]: Enter an Enclave
- ENCLU[EEXIT]: Exit an Enclave
- ENCLU[EGETKEY]: Create a cryptographic key
- ENCLU[EREPORT]: Create a cryptographic report
- ENCLU[ERESUME]: Re-enter an Enclave

The SGX user instructions are called by user interactions within the enclave. These instructions allow for you to enter and exit the enclave. Most importantly it allows for applications to create the cryptographic keys in order to communicate between the application, RAM, and storage all with encryption. The only hardware decryption method at that point would be the processor, which holds onto one of the two keys.

What is really interesting in the user instructions is the ERESUME instruction. This instruction allows for the application to leave and reenter the secure enclave environment. No one has really came to the conclusion as to why Intel added this nor does Intel give a good reason as to why in the reference guide [1]. One possible reason would be to have an application like an internet browser that doesn't want to operate fully in the enclave to share browser history or search results. The browser, could initialize, check, and then enter the enclave. Once it has entered, it will then exit where it will wait for the browser to need the secure enclave to enter information like credit card data then exit the enclave once again.

CRYPTOGRAPHY

Intel SGX uses cryptography everywhere. From signing the enclave so that Intel can use trusted developers to sealing data so that secure information does not leak out of the enclave. SGX uses several forms of cryptography which I will now go over.

Random Generation

Cryptography relies on strong randomization in order to effectively secure data. Intel has put special care in making sure the cryptography used will have strong randomization. Intel has added several special libc operations with the SGX development kit to ensure proper randomization from the developer. In particular, Intel uses the RDRAND instruction to generate randomness. Intel has taken great care to make sure developers are properly building in randomization. In many ways developers will use RAND which is incredibly unsecure. When you call the RAND instruction the randomness is actually not random. In most cases developers believe that you can use RAND to create randomness in cryptography; however, RAND is only partially random and more computation is required to create true randomness. The RDRAND instruction is a little better with true randomness.

An interesting aspect to RDRAND is that it's one of the only, if not the only, non-SGX instruction to be available inside the SGX enclave. This means that RDRAND could actually force an exit within the enclave. Intel has put a lot of trust in the RDRAND instruction to be able to do this. Some of this may be due to the fact that Intel wants to make sure application developers do not use the RAND instruction. Although, a counter argument could be made that giving this access to the secure enclave could pose issues later on if a bug is found to be in the RDRAND instruction.

Standard Crypto

Intel SGX uses several different cryptographic algorithms within the environment.

- RSA3072 + SHA256: Enclave Signatures
- RSA2048 + SHA256: Attestation Process
- DSA-EC + SHA256: Launches the enclave
- ECDH: Remote key exchange
- AES-GCM: Sealing Data
- AES-CMAC: Key derivation
- AES-CTR: Memory Encryption

Every one of these cryptographic algorithms have strong security, with many being standardized, having lasted the test of time. If flaws were to be made, it would be in the implementation stage.

Custom Crypto

Since Intel has some very specific needs for SGX, Intel has made some custom cryptographic schemes. In particular, two stand out to be some of the most critical of how Intel SGX operates.

1. Memory Encryption Engine: The (MME) protects data stored in the RAM. This new algorithm utilizes two special data sets in order to create the new scheme. MME combines the MAC address with a slightly tweaked version of AES.
2. Creation Scheme: In reality this is the most important algorithm in the entire implementation of SGX. Creation scheme is the certificate for the secure enclave. If this scheme is insecure than SGX is completely useless.

Looking at the creation scheme it has some very unique properties. It combines RSA with a symmetric key that then authenticates a unique signature which Intel provides. Figure 4 below gives a detailed outline graph on how the creation scheme works in more detail.

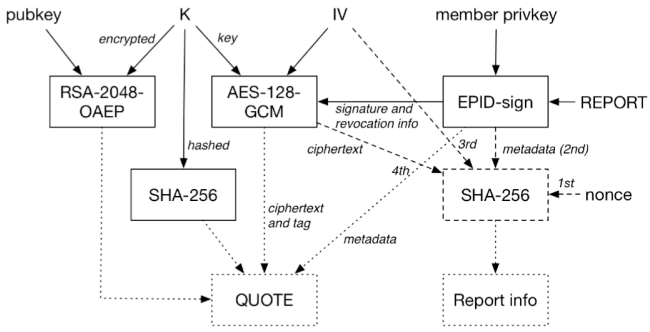

Figure 4: Creation Scheme Outline [5]

Custom Signatures

Intel also features a unique signature scheme that is called to enhance the privacy of the enclave called EPID. This signature continually works to ensure that the application which is running in the enclave is the application that is identifies as. This signature is more unknown than anything else in Intel SGX. This signature is only issued by Intel and no one else knows what the Intel Signing Key is. This key is tied to the identified developer and the specific application that is running. In some cases, security professionals have argued that this is a deliberate backdoor into the system. Intel believes this is one of the only ways to ensure that the application is safe to run inside the enclave. Regardless, this should raise some concern as this could cause applications to not work in the future if the developer no longer continues their relationship with Intel.

USE CASES

Intel SGX has several use cases that really standout. Cloud infrastructure is booming; United States National Security is at an all-time risk; businesses are being attacked every day. Intel SGX has so much potential to elevate some of the burden that security professionals have in protecting sensitive data.

Cloud

The cloud is becoming an emerging space for both business and personal data. Cloud providers such as Amazon, Microsoft, and Google support enterprises around the world. There is a misconception that cloud providers will provide security to the end user. This misconception can make users incredibly vulnerable to attacks. Intel SGX would provide a standard layer of software security beyond the typical firewall that the provider has in place. Intel SGX gives cloud providers the opportunity to protect virtual machine solutions. Every cloud service utilizes virtual operating systems for customers to use. In theory, Intel SGX could be utilized to house entire virtual machines inside the enclave. This technology would have to be adopted by hypervisor companies like VMware and Citrix but this technology would drastically improve hypervisor security.

Currently, cloud providers must purchase TPM devices, with costs in the thousands, in order to have the TPMs attached to servers. This cost gets passed on to the customer, who either chooses to use or not use the TPM. Intel SGX, other than purchasing an Intel processor is not an additional cost. This allows cloud providers to offer a security feature to the customer at no extra cost. Cloud providers like Microsoft and Amazon could even give customers the option to purchase SGX applications which could boost sales for application developers who take the time to create SGX enabled applications.

Government

Currently, the only reason why SGX is not fully adopted has been developer support and the price of entry for new Intel processors that support SGX. When attending this year's Department of Homeland Security SBIR conference there were several companies making secure enclaves for mobile devices. It would not be surprising if the US Government was already using applications utilizing the SGX enclave.

Most applications developed by the Government will never see the light of day to the general public. However, when developers start developing and designing code to be used with secure enclaves it will trickle down to business applications. Given the landscape of security concerns around business data we might see more government RFPs centered around contractors making applications around secure enclaves like Intel SGX. Intel SGX would be another layer to protect US assets and provide a much smaller attack surface which I go into later in this paper.

Business

Once Intel SGX penetrates the business application market we will see a drastic decrease in several known business security weak spots. In particular, businesses could see a drastic decrease in malware and even more so in ransomware. Ransomware has really taken businesses by storm. Florida Center for Cybersecurity has been seeing more sophisticated ransomware every week. Most of the ransomware released are actually not from the developer. Instead, developers choose to sell ransomware as a package on the darknet to go after low hanging fruit. This allows the developer to sell a product without committing a large scale crime. The attacker purchases the package for pennies on the dollar compared to what they can make attacking a company multiple times. Ransomware has become so prevalent that many Fortune 500 companies are starting to have bitcoin wallets just to pay off attackers who ask for amounts ranging from $2000 to $75,000 a hit. Intel SGX could prevent most of the ransomware attacks happening today. Ransomware requires root access to the system where it then tries to encrypt as much data as possible. Intel SGX would either trap the ransomware in the enclave and once the application closes would be wiped. The other alternative is that the ransomware would be shutout from the enclave environment which would only encrypt the data that was not a part of the enclave. Either way, the secure data would be safe.

Intel SGX can also be used to secure data such as biometrics and password data. The #1 cause of security breaches today is from employees who don't have the proper training to keep their information safe. Many companies have special security policies that get in the way rather than support the employee who is trying to do their job. Intel SGX would give businesses the ability to have more sophisticated password managers and the ability to use biometric capabilities that are easier for the employee to use.

Business applications would see the most benefit to Intel SGX technologies. This is so brand new that there are more possibilities than even Intel realizes. While the cost of development and new equipment will be steep for the next couple of years, we should be seeing financial companies moving to secure enclaves like Intel SGX soon enough.

Personal

Intel SGX will take a much longer time to trickle down into the personal market space. Although the everyday person will see lots of benefits in SGX with virtual machines, cloud computing and business applications such as financial data. The biggest benefit to the consumer market will be less leaked personal data from negligent businesses. Millions of identities and leaked personal data roams the web and Intel SGX can be another win for security professionally trying to protect people's sensitive data.

ATTACK SURFACE

The attack surface is the landscape that can be used to try to attack the system or application environment. The current attack surface of modern computers is to exploit through a vulnerable application exploiting the operating system, which then gives the ability to the attacker to grab any and all information after the exploit occurs. In the case of Intel SGX, the attacker would have to attack the hardware layer, which is more difficult to accomplish. While Intel touts its security capabilities within SGX, it is not entirely secure out of the box. Even though Intel SGX can make executable applications nearly impossible to reverse engineer, the enclave code is not encrypted. The enclave becomes secure when the identified vendor application sends a certification to the processor which then checks to see if the application is allowed to create a secure enclave or not. When this happens, only then will the processor create the enclave and allow secure data to move and be executed. This allows for small opportunities for timing attacks which could leak secure information within the Intel SGX environment.

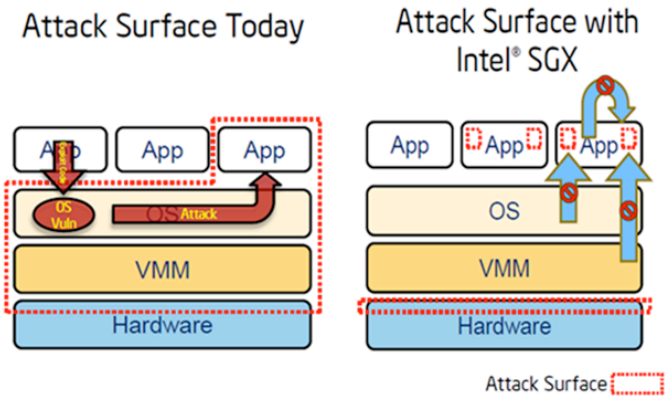

Figure 5: SGX Attack Surface [6]

When Intel SGX launched last year, the main concern of attack would be hackers using the enclave to their advantage. If a hacker was able to create a trusted certificate, then that would open the door for botnets and malware to run on systems without a trace. This is a valid concern; however, there are several arguments that should at least alleviate some of the concern. When applications run inside the enclave, they are severely limited and cannot perform syscalls nor perform I/O operations within the encrypted space. Currently, you have to become an identified vendor from Intel to distribute code that builds a secure enclave or the application will fail to run. The most important aspect thought, is that the code running inside the enclave does not run in kernel mode and would require privilege escalation to get identifiable information from the enclave.

Intel acknowledges that there are several security limitations that SGX cannot protect against. [5]

- **Cache-Timing**: Cannot prevent side-channel attacks that may happen when code labeled secret are executed at particular times when accessing cache.
- **Microcode**: Cannot protect against an attacker who changes functionality in the machine code.
- **Physical**: Someone who get ahold of your CPU before you and creates a hardware backdoor.

Additionally, there are the obvious attacks that could compromise SGX. For instance, Intel could have holes in the enclave software that could cause leaks, concurrency issues, and memory corruption. If the application developer was inexperienced in creating SGX applications, it could very well cause issues with leaked data from doing function calls outside the enclave. Of course, you could also have bugs within Intel trusted libraries such as libc. Also if some application developer didn't download the SGX SDK from Intel, or Intel's download manager was compromised, then an attacker could inject insecure code in the development environment.

The complexity of implementing Intel SGX correctly is one of the main hurdles that Intel and application developers will have to face down the road. It's not easy to detect when the application should be running in an encrypted environment. It will take further development time and carful debugging techniques to make sure no securely labeled data leaks from the enclave. Unfortunately, most of this will have to be done in a black box method of testing. This will ultimately delay applications or application developers will side track Intel SGX security to future updates instead of making sure Intel SGX works out of the box.

Attack Surface Reduction

Application developers have many opportunities to reduce the attack surface when utilizing Intel SGX. For instance, developers can utilize the debugger instruction within SGX that insures code securely by giving logs to confirm or deny security leaks. It's very important that the software developer creates secure code that will not have secure data from the enclave leak out.

Intel also needs to have more clear information about how one gets to be an identifiable vendor. While Intel does have some vague information on its website, it does not give clear instructions on how to become a vendor or what rules you must adhere to in order to be trusted by Intel. There is also speculation that Intel is not giving clear information on this, due to Intel not actually certifying developers; however, this claim has not been supported. Regardless, Intel would be better off giving this information, as no one really knows if they should be looking at I/O and file I/O in particular to potentially attacking SGX enclave applications.

SGX SECURITY

Intel has tried its best to cover every possible issue in order to make Intel SGX secure. There are however, several different avenues an attacker could utilize to take advantage of SGX. Many of these advantages have been thought of but it's worth mentioning because Intel has done very little besides controlling signature data.

Intel's Goals for SGX

When Intel set out to create Intel SGX it had several goals in mind.

- Protect data from unauthorized access from rogue software running at higher privilege levels.
- Preserve sensitive code and data without disrupting the ability to schedule and manage through the OS.
- The freedom to install and uninstall applications as the user chooses.
- An application should have trusted code and produce a signed attestation, rooted in the processor.
- Use trusted development environments with familiar tools
- Secure applications should have the ability to scale.
- Software developers should be able to deliver trusted applications with updates on their schedule.
- The application can define secure regions of code and data to maintain confidentiality.

The Intel objectives have been met and, in many ways, have exceeded what they wanted to achieve. As Intel progresses the technology, we will see significantly more improvements to how the verified developer works and operates.

Creating an Application

Creating an application for Intel SGX is very simple. All you need to do is use Microsoft Visual Studio and call the SGX library. From that point you can mark which data you want ran inside the enclave, and then mark it as secure. Figure 6 below shows a test of the enclave on buffers, strings, and integer data after being marked as secure.

```
App: Buffertests:
App: Buffer before change: Hello World!
App: Buffer after change: Hello Enclave!

App: Stringtests:
App: Returned Secret: Default Enclave savedText
App: Saved Secret: My secret string
App: Load Secret: My secret string

App: Integertests:
App: secretIntValue first load: -1
App: saved a 1337 to the enclave.
App: secretIntValue second load after 1337 was saved: 1337
```

Figure 6: SGX Code Test

Malware

If a malware developer was able to be identified as a trusted developer it would severely hurt Intel SGX. The scenario that malware could live inside the secure enclave has very real implications. While in the personal space, this would not be a main concern, in the cloud space, it would be devastating. In a cloud environment it would be extremely hard to detect an application in the enclave. Currently, the only known ways to potentially detect rogue applications would be to either monitor I/O data or monitor file disk usage. Neither of these mitigation techniques would work effectively in a cloud infrastructure.

To create malware within the enclave it would take very little time. Most malware today is coded in C which Intel SGX supports. When coding for Intel SGX there is very little you have to do in order to get started. First, you need to call the SGX library which allows for the compiler to enable the instructions. At that point, once you develop the malware all you would need to do is compile. At this part, Intel would need you to be an identified developer. As of right now, attackers would have an extremely hard time in getting a trusted developer certificate. This is why there have been no known cases of malware developed for Intel SGX.

In protecting against malware developers, Intel does not run applications as root in the enclave. The user would still need to run as root and for Microsoft Windows users, it would prompt to accept the UAC when running the application as root. This countermeasure also protects the user from unintentionally downloading malware that could run in the enclave. Regardless, it's a valid concern that malware developers could be looking at SGX as a great attack vector which would linger in systems for quite a long time.

Botnets

Another concern is that botnet creators could use Intel SGX as a way to gain more access to systems. Botnets today are mostly utilized for email spam, denial of service attacks, crypto attacks against IoT devices, and bitcoin mining. Botnets have been notorious for finding ways to go undetected in systems. Currently, the main method to avoid detection is to stay dormant in systems until the host node decides to activate the botnet clusters to launch an attack or to mine resources. This method has been incredibly successful in getting through sandboxes and black box detection as the botnet uses very little to no resources most of the time.

If a developer was to develop a botnet for Intel SGX, it would prove to be the best way to gather a lot of nodes to launch attacks. In most cases, botnets do not even need root privileges to run correctly. The botnets could run inside the enclave undetectable even through I/O scans. This would allow the attacker to send requests to the nodes through the enclave and be untraceable. The only protections in stopping botnet developers from creating botnets for SGX would be lack of

Intel 6th generation processors on the market and the ability to gain access to the trusted developer platform.

PITFALLS

While Intel has tried its best to make a great product, there are still some pitfalls that really keep Intel SGX from being successful currently. There is also some stuff they have left. Here are some of my gripes with Intel SGX.

Encrypting the I/O

I'm not quite sure why Intel left out encrypting the data that goes through I/O. Sure, it's one of the only ways as a last resort to see attackers in the enclave, but the data still could leak from I/O. Nothing else makes sense as to why they left this out. Encrypting the I/O would have completely sealed the system.

Just encrypting the I/O because it would make the system much more secure. It would have also delivered a product that no one else has done before. While the Intel SGX enclave will be eventually a standard for security of applications; the enclave is not unique. Giving full I/O encryption would have closed another gap that exists for security professionals to close.

Intel's Management of Signature Keys

While most people would trust Intel with the signature key management, as a security professional, it really doesn't sit right that Intel is the only one who knows the Intel Signature Key. In fact, this issue has come up in many different discussions about the product. Intel should design a new way to handle the signature key. If not, they should at least consider the ability for a third party to control access to the keys. They also do very little in reassuring customers, who are actively looking at a security tool the assurance that their key is safe. It requires reading the Intel SGX manual to even discover that Intel generates a unique secure key that the end user must have in order to run the application in the enclave. Much of the bad press on this technology stems from the signature key management. While it's understandable that you want to make sure the developers who use this product, will abide and even champion the product it's no excuse to use this as a way to have control of the signature keys.

Human Implementation

The biggest downfall of Intel SGX is the implementation by humans. Humans make mistakes and humans make a lot of mistakes when it comes to security. In the case of Intel SGX, Intel is giving trust to the developers to implement Intel SGX correctly. To many developers SGX probably looks like an overwhelming undertaking and a huge responsibility. There are just too many ways to potentially leak data from the enclave by mistake. While coding errors often happen the secure enclave has the ability to have a lot of secure information pass through it.

One of the implementations I would like to see added is the ability to have a debugger tool check the code and present based on making the code more secure. Intel has disallowed certain calls such as the RAND instruction call, which is a step in the right direction. What would be much more beneficial though would be if Intel went a step further and actually scanned the code for known bad code implementation.

Intel should also develop a way to integrate Intel Security their security tool company into Intel SGX. This would further help develop more tools that could be used with Intel SGX. It would also give more of an incentive for companies to upgrade to new Intel 6th generation processors and promote more Intel Security tools.

Ongoing Development

The biggest pitfall of Intel SGX is that there is no clear direction for where this is headed in the future. With most Intel products they provide roadmaps to let customers know when new products will be released and what customers can expect for new features. With Intel SGX no news has come from Intel in months. The new Kaby Lake processors came out with zero mention that Intel SGX is supported. While Intel might be developing new products that utilize Intel SGX it hurts perceptions that SGX will die off before it ever reaches the market.

Another issue I see with development is that new features that should be added to the Intel SGX library are being added as separate components like AMD's competing product. This will create market confusion and separate products. Intel should combine these products and allow developers more access to new security features instead of releasing planed progression on new products without release dates.

Backwards Compatibility

The TPM module took very little time before it hit market and became a viable security solution. In the case of Intel SGX however, it will take years for it to see any progress in the security tool industry. This is mainly due to the fact that Intel did not find a way to make Intel SGX fully, or even partially backwards compatible to other i-series generations. This lack of backwards compatibility has made the growth of Intel SGX very slow and may never see traction in the market.

Intel SGX has a lot going for it. While the pitfalls I have mentioned have given SGX negative reviews within the security industry, it has done very little to negatively persuade developers against using it. This is the first iteration of Intel SGX and Intel will take these concerns and make it a better product.

CONTROL-FLOW ENFORCEMENT TECHNOLOGY (CET)

Control-flow Enforcement Technology or CET was developed by Intel to provide capabilities to defend against return oriented programming. It works by creating a shadow stack and performing indirect branch tracking. CET is reliant on Intel SGX in order to work properly.

Return Oriented Programming

Return oriented programming is a security exploit technique that allows the attacker the ability to execute code even when there are security defenses in place. ROP is executed by taking control of gadgets. Each gadget usually ends with a return instruction allowing you to get further into the library code. By chaining gadgets together, you can achieve arbitrary operations and smash the stack. Currently, there is no way to protect against ROP. One of the only minor defenses is to utilize address space layout randomization (ASLR) to better randomize gadgets and make it harder to chain gadgets together. By not having a proper defense mechanism this allows attackers a serious advantage.

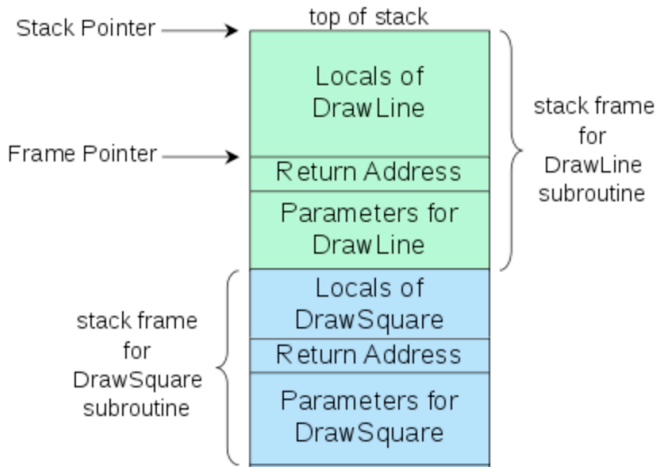

Figure 7: ROP Stack [7]

Shadow Stacks

A shadow stack is a secondary stack for the program to use exclusively for control transfer operations. This stack is separated from the data stack and can be enabled by either the user or by the compiler in supervisor mode. The shadow stack is enabled from a call instruction that pushes the return address from the data stack to the shadow stack.

The shadow stack is protected through page table protections that store instructions within the shadow stack. The page table protections prevent malicious attempts to cause overflow or underflow of the buffer in the stack. These protections are why the shadow stack is so important to prevent ROP and other stack smashing techniques.

The shadow stack is one of the most important security technologies to come out since NX and Canary. By eliminating return oriented programing from the hacker toolkit it would drastically decrease the available attack vectors.

```
gdb-peda$ checksec
CANARY    : disabled
FORTIFY   : disabled
NX        : ENABLED
PIE       : disabled
RELRO     : Partial
```

Figure 8: GDB Checksec

New Instructions and Indirect Branch Tracking

Intel has added new instructions calls and flags to support CET. These are add-on instructions that allow the shadow stack to be created and to enable the indirect branch tracking. What differs from Intel SGX instructions however, is that CET will work with both 32-bit and 64-bit modes.

When CET is enabled, the processor will support the SGX registers and call the shadow stack pointer. The processor will have to support the shadow stack feature in order to work correctly. Once the shadow stack is in place and SGX is enabled there will be little in the way of exploiting the application.

Indirect branch tracking is called by using the instruction ENDBRANCH. This new instruction is used to mark valid indirect call and jmp targets in the program. Indirect branch tracking is a new technology within software security and is being used mostly in machine language instruction sets. This will be the backbone in making sure that ROP cannot be used as the attack surface anymore.

Future with CET

The main reason for my interest in SGX was for the CET technology. CET will completely revolutionize the security industry once it becomes available. Intel is currently in its second revision of the reference guide and is close to release. By eliminating return oriented programming, it will eliminate the majority of binary stack exploits. This could have a long lasting impact on how security testing will be done in the future. While their always could be new and exciting exploit opportunities that come up, ROP was by far one of the most challenging exploits to land. In just over 20 years of security software testing there has been several stack exploitation techniques that have come and gone.

CONCLUSION

In this paper, I have explained how Intel SGX works and operates and how to enable the secure enclave, as well as the special and unique features the new secure enclave provides. I listed the new 17 unique x86 instructions to Intel SGX and what each one accomplishes. I gave a brief look at what use cases are available for SGX developers and what environments SGX would work well in. I talked about the attack surface and how Intel SGX protects up to the hardware layer. I presented some pitfalls of Intel SGX and gave suggestions on what they might be able to do to correct them. Finally, I gave a short introduction to Intel's new security technology CET that works with Intel SGX. Intel CET has the potential to revolutionize application security.

Intel SGX has accomplished quite a bit since being introduced less than 2 years ago. These 17 new instructions the implement the secure enclave have so much potential to revolutionize the security industry. The secure enclave has so many different use cases for securing applications of the future. Intel will hopefully keep developing the platform and make it better.

FUTURE WORK

There is still a lot to look at with Intel SGX. One of the first things I would like to get is access to run applications through the secure enclave. While I have the ability to create programs within the developer environment there is not a way currently to run the applications beyond the emulated environment without being a trusted developer. It would also be nice to have access to an Intel 6th Generation processor so I could run the applications and use a debugger and decompiler.

I would also like to look further into the enclave environment and whether or not you can get information to leak from it. My initial findings are that Intel may have made it easy for the developer to make mistakes and allow applications to leak information. If this is true, then than it will be very easy for attackers to grab the secure enclave information.

I would also like to further look into Intel CET. As a hacker, ROP has been the main exploit in many of the capture the flag competitions for the last several years. Once Intel finally releases the CET technology I plan to look into it more and see what potential issues could be exploited.

Lastly, I would like to compare Intel SGX technology with AMD's once it is officially out in Spring 2017. While we know they will be splitting up their technologies into multiple services it will be interesting to see which one has implemented a sounder environment.

Once I have received some additional data from Intel, I will be publishing more on Intel SGX and the security behind the technology. While it may take some time to get, it will be fantastic to disclose more in depth knowledge on Intel SGX and how to improve the technology going forward.

ACKNOWLEDGEMENTS

I would like to acknowledge Florida Center for Cybersecurity in allowing me to further my research interests and allow me to grow as a researcher.

Without participating in Whitehatters Computer Security Club [WCSC] at University of South Florida, I would not have had the knowledge to do this paper. To everyone I look up to including Vito, Gynophage, Hoju, Bspar, Duck, Fuzyll, Jetboy, GH0ST, Skolor, and the many others who inspire me every day. This is binary now tell me why you love it!

REFRENCES

[1] Intel Corporation. *Software Guard Extensions Programming Reference*, 2014. Reference no. 329298-002US.

[2] Intel Corporation. *Control-flow Enforcement Technology Preview*, 2016. Reference no. 334525-001US.

[3] Ittai Anati, Shay Gueron, Simon P Johnson, and Vincent R Scarlata. *Innovative technology for CPU based attestation and sealing*. Intel Corporation. 2013.

[4] Victor Costan, Srinivas Devadas. *Intel SGX Explained*. Computer Science and Artificial Intelligence Laboratory. MIT.

[5] JP Aumasson, Luis Merino. *SGX Secure Enclaves in Practice: Security and Crypto Review*. Kudelski Security. 2016

[6] Shaun Davenport, Richard Ford. *SGX: the good, the bad, and the downright ugly*. Computer Science Department. Florida Institute of Technology