Intel® Software Guard Extensions Tutorial Series: Part 1, Intel® SGX Foundation

By John M., Benjamin O., published on July 7, 2016

The first part in the Intel® Software Guard Extensions (Intel® SGX) tutorial series is a brief overview of the technology. For more detailed information, see the documentation provided in the Intel Software Guard Extensions SDK. Find the list of all the tutorials in this series in the article Introducing the Intel® Software Guard Extensions Tutorial Series.

Understanding Intel® Software Guard Extensions Technology

Software applications frequently need to work with private information such as passwords, account numbers, financial information, encryption keys, and health records. This sensitive data is intended to be accessed only by the designated recipient. In Intel SGX terminology, this private information is referred to as an application's secrets.

The operating system's job is to enforce security policy on the computer system so that these secrets are not unintentionally exposed to other users and applications. The OS will prevent a user from accessing another user's files (unless permission to do so has been explicitly granted), one application from accessing another application's memory, and an unprivileged user from access OS-resources except through tightly controlled interfaces. Applications often employ additional safeguards, such as data encryption, to ensure that data sent to storage or over a network connection cannot be accessed by third parties even if the OS and hardware are compromised.

Despite these protections, there is still a significant vulnerability present in most computer systems: while there are numerous guards in place that protect one application from another, and the OS from an unprivileged user, an application has virtually no protection from processes running with higher privileges, including the OS itself. Malware that obtains administrative privileges has unrestricted access to all system resources and all applications running on the system. Sophisticated malware can target an application's protection schemes to extract encryption keys and even the secret data itself directly from memory.

To enable the high-level protection of secrets and help defend against these software attacks, Intel designed Intel SGX. Intel SGX is a set of CPU instructions that enable applications to create enclaves: protected areas in the application's address space that provide confidentiality and integrity even in the presence of privileged malware. Enclave code is enabled by using special instructions, and it is built and loaded as a Windows* Dynamic Link Library (DLL) file.

Intel SGX can reduce the attack surface of an application. Figure 1 demonstrates the dramatic difference between attack surfaces with and without the help of Intel SGX enclaves.

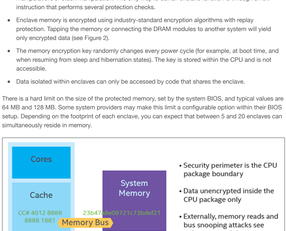

Figure 1: *Attack-surface areas with and without Intel® Software Guard Extensions enclaves.*

How Intel Software Guard Extensions Technology Helps Secure Data

Intel SGX offers the following protections from known hardware and software attacks:

- Enclave memory cannot be read or written from outside the enclave regardless of the current privilege level and CPU mode.
- Production enclaves cannot be debugged by software or hardware debuggers. (An enclave can be created with a debug attribute that allows a special debugger—the Intel SGX debugger—to view its content like a standard debugger. This is intended to aid the software development cycle.)
- The enclave environment cannot be entered through classic function calls, jumps, register manipulation, or stack manipulation. The only way to call an enclave function is through a new instruction that performs several protection checks.
- Enclave memory is encrypted using industry-standard encryption algorithms with replay protection. Tapping the memory or connecting the DRAM modules to another system will yield only encrypted data (see Figure 2).
- The memory encryption key randomly changes every power cycle (for example, at boot time, and when resuming from sleep and hibernation states). The key is stored within the CPU and is not accessible.
- Data isolated within enclaves can only be accessed by code that shares the enclave.

There is a hard limit on the size of the protected memory, set by the system BIOS, and typical values are 64 MB and 128 MB. Some system providers may make this limit a configurable option within their BIOS setup. Depending on the footprint of each enclave, you can expect that between 5 and 20 enclaves can simultaneously reside in memory.

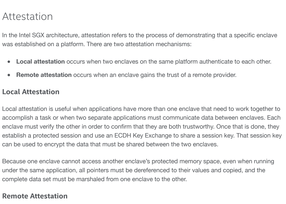

Figure 2: *How Intel® Software Guard Extensions helps secure enclave data in protected applications.*

Design Considerations

Application design with Intel SGX requires that the application be divided into two components (see Figure 3):

- **Trusted component.** This is the enclave. The code that resides in the trusted code is the code that accesses an application's secrets. An application can have more than one trusted component/enclave.
- **Untrusted component.** This is the rest of the application and any of its modules. It is important to note that, from the standpoint of an enclave, the OS and the VMM are considered untrusted components.

The trusted component should be as small as possible, limited to the data that needs the most protection and those operations that must act directly on it. A large enclave with a complex interface doesn't just consume more protected memory; it also creates a larger attack surface.

Enclaves should also have minimal trusted-untrusted component interaction. While enclaves can leave the protected memory region and call functions in the untrusted component (through the use of a special instruction), limiting these dependencies will strengthen the enclave against attack.

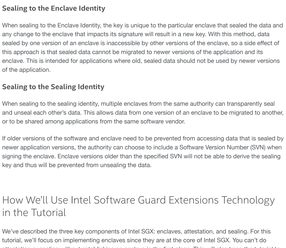

Figure 3: *Intel® Software Guard Extensions application execution flow.*

Attestation

In the Intel SGX architecture, attestation refers to the process of demonstrating that a specific enclave was established on a platform. There are two attestation mechanisms:

- **Local attestation** occurs when two enclaves on the same platform authenticate to each other.
- **Remote attestation** occurs when an enclave gains the trust of a remote provider.

Local Attestation

Local attestation is useful when applications have more than one enclave that need to work together to accomplish a task or when two separate applications must communicate data between enclaves. Each enclave must verify the other in order to confirm that they are both trustworthy. Once that is done, they establish a protected session and use an ECDH Key Exchange to share a session key. That session key can be used to encrypt the data that must be shared between the two enclaves.

Because one enclave cannot access another enclave's protected memory space, even when running under the same application, all pointers must be dereferenced to their values and copied, and the complete data set must be marshaled from one enclave to the other.

Remote Attestation

With remote attestation, a combination of Intel SGX software and platform hardware is used to generate a quote that is sent to a third-party server to establish trust. The software includes the application's enclave, and the Quoting Enclave (QE) and Provisioning Enclave (PvE), both of which are provided by Intel. The attestation hardware is the Intel SGX-enabled CPU. A digest of the software information is combined with a platform-unique asymmetric key from the hardware to generate the quote, which is sent to a remote server over an authenticated channel. If the remote server determines that the enclave was properly instantiated and is running on a genuine Intel SGX-capable processor, it can now trust the enclave and choose to provision secrets to it over the authenticated channel.

Sealing Data

Sealing data is the process of encrypting it so that it can be written to untrusted memory or storage without revealing its contents. The data can be read back in by the enclave at a later date and unsealed (decrypted). The encryption keys are derived internally on demand and are not exposed to the enclave.

There are two methods of sealing data:

- **Enclave Identity.** This method produces a key that is unique to this exact enclave.
- **Sealing Identity.** This method produces a key that is based on the identity of the enclave's Sealing Authority. Multiple enclaves from the same signing authority can derive the same key.

Sealing to the Enclave Identity

When sealing to the Enclave Identity, the key is unique to the particular enclave that sealed the data and any change to the enclave that impacts its signature will result in a new key. With this method, data sealed by one version of an enclave is inaccessible by other versions of the enclave, so a side effect of this approach is that sealed data cannot be migrated to newer versions of the application and its enclave. This is intended for applications where old, sealed data should not be used by newer versions of the application.

Sealing to the Sealing Identity

When sealing to the sealing identity, multiple enclaves from the same authority can transparently seal and unseal each other's data. This allows data from one version of an enclave to be migrated to another, or to be shared among applications from the same software vendor.

If older versions of the software and enclave need to be prevented from accessing data that is sealed by newer application versions, the authority can choose to include a Software Version Number (SVN) when signing the enclave. Enclave versions older than the specified SVN will not be able to derive the sealing key and thus will be prevented from unsealing the data.

How We'll Use Intel Software Guard Extensions Technology in the Tutorial

We've described the three key components of Intel SGX: enclaves, attestation, and sealing. For this tutorial, we'll focus on implementing enclaves since they are at the core of Intel SGX. You can't do attestation or sealing without establishing an enclave in the first place. This will also keep the tutorial to a manageable size.

Coming Up Next

Part 2 of the tutorial series, Intel® Software Guard Extensions Tutorial Series: Part 2, Application Design, will focus on the password manager application that we'll be building and enabling for Intel SGX. We'll cover the design requirements, constraints, and the user interface. Stay tuned!

Find the list of all the tutorials in this series in the article Introducing the Intel® Software Guard Extensions Tutorial Series.

Additional Resources

Intel® Software Guard Extensions

Intel® SGX Forum

For more complete information about compiler optimizations, see our Optimization Notice.

1 comment
Top
John M. said on Jul 27,2016
Fixed the broken link to Part 2 in the series. Sorry for the error and any inconvenience that caused!
Add a Comment
Sign in
Have a technical question? Visit our forums. Have site or software product issues? Contact support.