Intel® SGX for Dummies (Intel® SGX Design Objectives)

Submitted by Matthew Hoekstra (Intel) (https://software.intel.com/en-us/user/335935) on September 26, 2013

Last updated on November 30, 2015

Today the Intel® Software Guard Extensions (Intel® SGX) programming reference manual was published (/en-us/intel-sgx-programming-reference) (more information is available here (/en-us/isa-extensions/intel-sgx)). Given the significant time and effort that my colleagues and I have spent defining Intel® SGX, I can't find a strong enough word in my thesaurus to describe how thrilled/elated/ecstatic I am to finally be able to write about it publicly.

At its root, Intel® SGX is a set of new CPU instructions that can be used by applications to set aside private regions of code and data. But looking at the technology upward from the instructions is analogous to trying to describe an animal by examining its DNA chain. In this short post I will try to uplevel things a bit by outlining the objectives that guided the design of Intel® SGX and provide some more detail on two of the objectives. In future posts, I will dive deeper into the remaining objectives and review some of our experiences using Intel® SGX to protect various software applications.

Much of the motivation for Intel® SGX can be summarized in the following eight objectives:

1. **Allow application developers to protect sensitive data from unauthorized access or modification by rogue software running at higher privilege levels.**
2. **Enable applications to preserve the confidentiality and integrity of sensitive code and data without disrupting the ability of legitimate system software to schedule and manage the use of platform resources.**
3. **Enable consumers of computing devices to retain control of their platforms and the freedom to install and uninstall applications and services as they choose.**
4. **Enable the platform to measure an application's trusted code and produce a signed attestation, rooted in the processor, that includes this measurement and other certification that the code has been correctly initialized in a trustable environment.**
5. **Enable the development of trusted applications using familiar tools and processes.**
6. **Allow the performance of trusted applications to scale with the capabilities of the underlying application processor.**
7. **Enable software vendors to deliver trusted applications and updates at their cadence, using the distribution channels of their choice.**
8. **Enable applications to define secure regions of code and data that maintain confidentiality even when an attacker has physical control of the platform and can conduct direct attacks on memory.**

Here is a little more detail behind the first two objectives

Objective 1 – Allow application developers to protect sensitive data from unauthorized access or modification by rogue software running at higher privilege levels.

Several aspects of Objective 1 are worth amplifying. First, protecting sensitive data demands both *confidentiality* (preventing data disclosure) and *integrity* (preventing data tampering). Second, it implies a need to protect sensitive *code* as well as *data* (consider, for example, that an attacker can easily obtain unauthorized access to data by modifying or skipping authorization checks). Third, data must be protected not only when it is stored in encrypted form, but also at run-time when the data is unencrypted and being actively used for computation. Finally, it is critical to maintain run-time protection despite attacks from rogue software that has subverted legitimate system software to gain amplified privilege levels.

Objective 2 – Enable applications to preserve the confidentiality and integrity of sensitive code and data without disrupting the ability of legitimate system software to schedule and manage the use of platform resources.

While sensitive data must be protected from attack by *rogue* software running at high privilege levels, *legitimate* system software must be allowed to do its job. It is unacceptable to require protected applications to take over or significantly disrupt the basic operating system features (job scheduling, device management, etc.). Operating systems have evolved over many generations to perform these roles well, and requiring a duplicate, parallel environment would be impractical.

I will follow up shortly with more details on the remaining objectives.

Part 2 (/en-us/blogs/2014/06/02/intel-sgx-for-dummies-part-2) and Part 3 (/en-us/blogs/2014/09/01/intel-sgx-for-dummies-part-3) are now available.

Categories: Security (/en-us/search/site/field_topic/security-20870/language/en) , Software Guard Extensions (/en-us/search/site/field_technology/software_guard_extensions-43865/language/en) , Developers (/en-us/search/site/field_audience/developers-17152/language/en)

Tags: SGX (/en-us/search/site/field_tags/sgx-43863/language/en)

Comments (4)

Top

Matthew Hoekstr... said on Tue, 06/03/2014 - 10:23

Thank you for the encouraging feedback and the interest in the technology. Intel is not ready to announce plans for availability of SGX emulation platform yet, but this forum will be updated when we are ready.

Chris H. said on Tue, 04/22/2014 - 11:17

How can developers learn of SGX emulation availability once it's released?

Thanks.

Raoul said on Wed, 11/06/2013 - 06:03

Interesting question! As a PhD student working in this field, I'm also very interested in optaining SGX-enabled hardware. When can we expect this hitting the retail market? I'm not asking for a specific date, but a ball park figure. Should it be measured in months or years?

paradium said on Mon, 11/04/2013 - 09:26

Hi great blog post and exciting new tech - I've been interested in trusted computing since it all started back in 2002/03 with Palladium, LaGrande, NGSCB etc. and have experimented with writing my own MLE for the TXT technology that eventually came out. I think SGX is a great step in the right direction to make many of the use cases that were originally envisioned practical. Of particular importance is the fact that once the enclave has been built (through participation by the OS) the application can then control enclave entry/exits. Hopefully this will make it easier to integrate the technology into operating systems and applications and make it relevant for a wider audience. With TXT no OS supports it natively as far as I know, so you have to write a device driver and 'steal' control from the OS when entering an MLE, which is impractical. Also the fact that all the tech, including the platform keys, are now hosted in the CPU rather than in an extra add-on component (like the TPM) is a huge step forward, as is the fact that it doesn't require a specific chipset (at least as far as I've read), which was the case for TXT.

I already have some interesting use cases in mind. Which CPU generation will feature this new tech? Is there a development program to sign up for and/or a way to get access to a test platform or emulator? With TXT I believe there was a program where one could get early access to a hardware platform.

Thanks!

Add a Comment

Top

(For technical discussions visit our developer forums. For site or software product issues contact support)

Please sign in to add a comment. Not a member? Join today ›