Runtime Microcode Updates with Intel® Software Guard Extensions

White Paper

September 2021

Revision 1.0

Notice: This document contains information on products in the design phase of development. The information here is subject to change without notice. Do not finalize a design with this information.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document, with the sole exception that you may publish an unmodified copy. You may create software implementations based on this document and in compliance with the foregoing that are intended to execute on the Intel product(s) referenced in this document. No rights are granted to create modifications or derivatives of this document.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software, or service activation. Learn more at intel.com, or from the OEM or retailer.

No computer system can be absolutely secure. Intel does not assume any liability for lost or stolen data or systems or any damages resulting from such losses.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Contents

Tables

Revision History

Revision Number	Description	Date
1.0	• Initial release of the document.	September/2021

1 Introduction

Intel® Software Guard Extensions (Intel® SGX) provide instructions to enable application software to instantiate a protected container, referred to as an enclave. Details of Intel SGX are described in CHAPTER 36 through CHAPTER 42 of Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3D.

Intel® SGX attestation allows remote parties to identify the security version number (SVN) of all the components of the SGX trusted computing base (TCB), including the SVN of the processor microcode. If processor microcode is updated with a newer version, then the Intel® SGX attestation will update to reflect the new SVN. Previously, once enclaves have been used during a boot cycle, updating the attestation required a system reboot resulting.

Increasingly, many computing systems are requiring higher Service Level Agreement (SLA) up time. Firmware updates such as microcode updates to address bug fixes and security updates leads to downtime making it difficult to meet these higher SLAs. Intel introduced runtime update of Intel® SGX TCB to reduce SLA impact.

This document explains additional Intel SGX capabilities introduced to perform runtime update of Intel SGX TCB attestation.

Note that some microcode updates are only fully effective when loaded via the BIOS. The new capabilities in this document will not prevent the need for a reset to effectively load these patches.

2 Intel® Software Guard Extensions and Microcode Updates

The processor microcode can be updated at any time. However, if it is updated while enclaves exist, it’s not accurate to assert that the security posture of those enclaves is based on the new update. Rather the enclave’s security posture is based on the oldest microcode that executed that enclave or created assets that the enclave relies on. For this reason, Intel SGX attestation reflects the oldest microcode that executed any enclave during this boot cycle.

A snapshot of the processor microcode SVN is taken during each boot cycle at the time when Intel SGX is first used. This results in microcode updates being loadable at any time, fixing microcode issues. However, previously the attestation would not reflect the update unless the platform is restarted and the microcode update is loaded before Intel SGX is first used.

A new ENCLS leaf (EUPDATESVN) has the capability to update this snapshot after it has been taken, if executed while the EPC is completely empty. This ensures that no enclaves that executed under a previous microcode update still exist. EUPDATESVN also regenerates new internal security assets (such as the paging key). If enclaves were operating, they must be shut down to yield an empty EPC.

After a microcode update is loaded at runtime, it is up to system software to empty the EPC and execute the new ENCLS[EUPDATESVN]. Enclave use can resume and any enclaves that were shut down can be relaunched if needed.

2.1 Runtime TCB Recovery Flow

This section contains a high-level example flow of how system software might load a Microcode Update and update the attestation to reflect the new SVN.

1. During boot, system software confirms support for updating the microcode update SVN by confirming CPUID.(EAX=12H, ECX=00H):EAX[10] is set.
2. Existing management infrastructure deploys the microcode update to platform, and existing software flows load microcode update on all cores.
3. Software uses management-infrastructure-specific mechanism to determine that the update corrects an SGX-related issue that requires existing enclaves need to be removed.
4. System software must prevent other use of SGX while it conducts steps 5-7. VMM can suspend or shut them down VMs using SGX, or configure the processor so attempts to manage SGX resources or enter an enclave will result in a VMEXIT.

5. System software prepares the EPC by marking all in use pages as unused with the ENCLS[EREMOVE] instruction.

6. System software executes ENCLS[EUPDATESVN], which verifies that EPC is ready.

7. If EUPDATESVN fails, the error code will indicate why.

   - If RAX = SGX_INSUFFICIENT_ENTROPY, software should retry step 6.
   - RAX = SGX_EPC_NOT_READY reflects that the EPC was not ready. Current HW will return this if the EPC is not empty. If software removes remaining EPC page, it should retry step 6.
   - RAX = SGX_NO_UPDATE indicates success but provides information that the SVN did not need updating. Software should confirm the correct microcode update was loaded in step 2.

8. System software allows use of Intel SGX and necessary enclaves can be rebuilt.

3 ISA Impact

Support for updating the attestation at runtime is provided by a new ENCLS instruction leaf, a new register for monitoring the state of the EPC, new Intel SGX error codes, and an enumeration bit in CPUID.

3.1 ENCLS[EUPDATESVN]

Table 3-1: EUPDATESVN - Update CPUSVN

Opcode/Instruction	Op/En	64/32-bit Mode Support	CPUID Feature Flag	Description
EAX = 18H ENCLS [EUPDATESVN]	None	V/V	Bit 10	Update CPUSVN if microcode has been updated and EPC is ready.

Description

If EPC is ready, this instruction updates CPUSVN to the currently loaded microcode update SVN and generates new cryptographic assets.

The EPC is ready when no page in the EPC is valid. EREMOVE should be used to mark all pages as unused.

It is the responsibility of system software to ensure that no other thread is executing or attempts to execute any ENCLS leaf while executing EUPDATESVN. Concurrency violations between EUPDATESVN and some ENCLS leaves may cause the ENCLS leaf to #GP(0) in ways unexpected to legacy software. System software should also prevent unnecessary software from having access to EUPDATESVN. For example, ENCLS exit controls should be used to prevent VMs not a part of the management system software from using EUPDATESVN.

The EUPDATESVN instruction fails if an ENCLS instruction is in progress on any thread, the EPC is not ready for an update, or insufficient entropy in the random number generator. The ZF flag will be set to indicate an error, and a code returned in RAX. If EUPDATESVN was successful but CR_CPUSVN was already up to date, the CF flag will be set and RAX will indicate that no update occurred.

If insufficient entropy causes a failure, software should repeat the instruction.

Table 3-2: EUPDATESVN Return Value in RAX

Return Code (See Table 3-4)	Type	Description
No Code	Success	EUPDATESVN was successful
SGX_NO_UPDATE	Success with Info	EUPDATESVN was successful, but CPUSVN was not updated because current SVN was not newer than CPUSVN.
SGX_LOCKFAIL	Error	An instruction concurrency rule was violated.

Return Code (See Table 3-4)	Type	Description
SGX_INSUFFICIENT_ENTROPY	Error	Insufficient entropy in RNG.
SGX_EPC_NOT_READY	Error	EPC is not ready for SVN update

Table 3-3: Base Concurrency Restrictions of EUPDATESVN

Leaf	Base Concurrency Restrictions	
	Access	On Conflict
EUPDATESVN	Exclusive	SGX_LOCKFAIL

Table 3-4: Additional Concurrency Restrictions of EUPDATESVN

Leaf	Base Concurrency Restrictions	
	vs EADD, EAUG, ECREATE, ELDB, ELDU, ELDBC, ELDUC, EPA, EREMOVE, EWB	
	Access	On Conflict
EUPDATESVN	Exclusive	SGX_LOCKFAIL

Operation

Table 3-5: EUPDATESVN Temp Variables

Variable Name	Type	Size	Description
TMP_CPUSVN	CPUSVN	128 bit	Temporary copy of CPUSVN before update
TMP_KEY	key	128 bit	Temporary copy of new paging key

```
(* Initialize flags *)
RFLAGS.ZF,CF,PF,AF,OF,SF := 0;
RAX := 0;

IF (Other instruction is accessing EPC) THEN
   RFLAGS.ZF := 1
   RAX := SGX_LOCKFAIL;
   GOTO ERROR_EXIT;
FI

(* Verify EPC is ready *)
IF (the EPC contains any valid pages) THEN
    RFLAGS.ZF := 1;
    RAX := SGX_EPC_NOT_READY;
    GOTO ERROR_EXIT;
FI

(* Refresh paging key *)
IF (NOT RDSEED(&TMP_KEY, 16)) THEN
    RFLAGS.ZF := 1;
    RAX := SGX_INSUFFICIENT_ENTROPY;
    GOTO ERROR_EXIT;
FI

(* Commit *)
CR_BASE_KEY := TMP_KEY;

TMP_CPUSVN := CR_CPUSVN;
(* Update CPUSVN to current minimum patch even if locked *)

(* Determine if info status is needed *)
IF (TMP_CPUSVN = CR_CPUSVN) THEN
    RFLAGS.CF := 1;
    RAX := SGX_NO_UPDATE;
FI

ERROR_EXIT:
```

Flags Affected

ZF	Cleared if instruction completed successfully. Set if error occurred. RAX is set to the error code
CF	Set if instruction completed successfully, but no SVN update was needed. RAX is set to info code.
PF,AF,OF,SF	Cleared

3.2 New and Updated Error Codes

Table 3-6 Instruction Error/Information Codes

Value	Error Name	Returned by
7	SGX_LOCKFAIL	EBLOCK, EMODT, EMODPR, EUPDATESVN
29	SGX_INSUFFICIENT_ENTROPY	EUPDATESVN
30	SGX_EPC_NOT_READY	EUPDATESVN
31	SGX_NO_UPDATE	EUPDATESVN

3.3 Enumeration

Availability of ENCLS[EUPDATESVN] can be enumerated via CPUID.(EAX=12H, ECX=00H):EAX[10] being set.