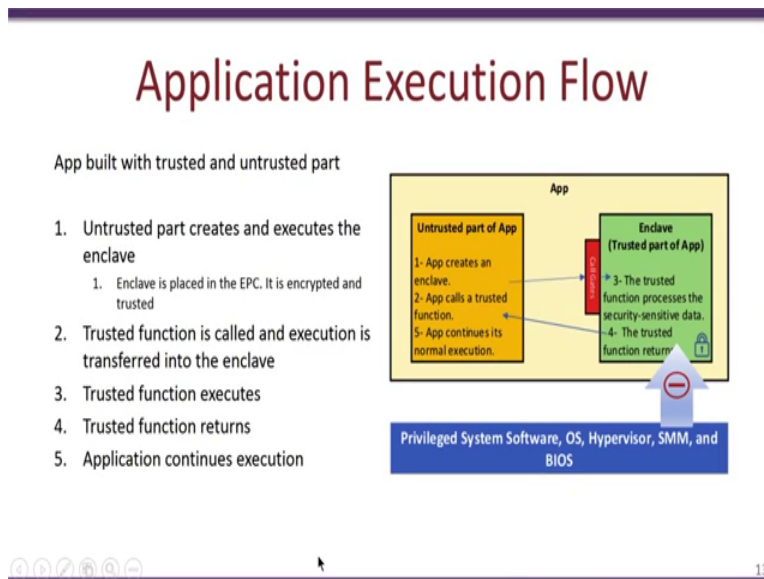


Information Security - 5 - Secure Systems Engineering
Professor Chester Rebeiro
Indian Institute of Technology, Madras
Intel's SGX part 2

Hello and welcome to the second part of the lecture on Intel SGX this in the course for secure systems engineering just part of the NPTEL series. In the previous video we had also looked at the Intel SGX we have seen what was capable with this particular feature in the processor. We looked at how the processor actually managed the memory, how it actually allocated memory regions for enclaves, how things were actually stored in the enclave and so on. In this part of the video lecture we will look at Intel SGX more from a software perspective. We will look at how applications would use the Intel SGX feature and we create enclaves.

What are the instructions are available for supporting these features. So a lot of what we are actually talking is present in this particular paper innovative instructions and software model for isolated in execution, this was published in hasp 2013. Also some of the figures are that are in this video have been actually borrowed from Intel.

(Refer Slide Time: 01:34)



So a typical application development on a system which has SGX supported would require that you actually split the application into two parts. One is the trusted part which you want to be part of the enclave and also the untrusted part where the remaining part of the application is present. So, the trusted part would be aspects such as cryptography, whether a region where passwords

are stored, a region where encryption and decryption is done, secret keys are stored and so on.

The remaining part could be the regular application like access like the graphical user interfaces other interfaces and so on. Now note that unlike the trustzone we have seen earlier both the untrusted application and the trusted application are present in the same address space, in fact the trusted part is just mapped to a certain region within that particular application. So when typically it would be untrusted part of the application which is executing the application would continue to execute until there is a call required to move to the trusted app.

So this essentially is a call to the enclave, so in order to actually incorporate SGX in an application the application would first need to create an enclave so the application would typically run in a untrusted mode and occasionally when there is enclave needs to be called rather than the application needs to call a trusted function. Then there is switch from the non-enclave mode into the enclave mode then the step 3 is where the enclave function executes and processes the security related sensitive data.

And finally the mode is switched back from the enclave mode back to the untrusted or the enclave mode and the application continues to execute. So these are the various steps involved when applications are developed with SGX enabled and the applications have enclaves. In order to actually support this form of application development Intel has a few instructions. So these instructions have been added on the instruction set of the Intel processors in order to create enclaves invoke trusted functions and so on. So we will first look at what this special functions are.

(Refer Slide Time: 04:11)

Instruction set Extensions for SGX

- **Privileged Instructions**
 - Creation related: to create, add pages, extend, initialize, remove enclave
 - Paging related: evict page, load an evicted page
- **User level instructions**
 - Enter enclave, leave enclave
 - Interrupt related: asynchronous exit, resume

14

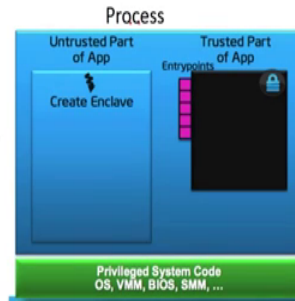
So first of all these special extensions for SGX are divided into 2 form one you have the Privileged set of instructions and the user level instructions. So the privileged instructions essentially should be used as a super user where you have instructions to create pages, add pages extend pages, remove enclave, and create an enclave and so on. Also the paging related aspects such as eviction of a page, loading of a page all done as part of the privileged instructions. The user level instructions on the other hand mostly just 2 or 3 of them one instruction will permit you to enter into the enclave and other one will permit you to leave the enclave and the third one would know as a resume would permit an application in user mode to resume after a interrupt or exception has occurred.

(Refer Slide Time: 05:12)

Enclave Life Cycle (creation)

ECREATE Instruction

- Creates a SECS (SGX enclave control structure)
 - Contains global information about the enclave
- System software can choose where (in the process virtual space) the enclave should be present
- Also specifies
 - Operating mode (32/64 bit)
 - Processor features that is supported
 - Debug allowed



15

So let us look at an overview of this various instructions, the first one is actually to create the enclave that is the ECREATE instruction and as we see over here create is a privileged instruction so whenever an application wants to create and enclave it would require to call make a system call within the system call there would be an ECREATE instruction which would initialize initiate the enclave creation. So when ECRETAE is invoke the processor would create an SECS structure in the PRM the processor related memory and this SECS structure as we know would contain the global information about the enclave.

Now the unique thing about the SGX is that the operating system can choose and manage where in the entire virtual space the enclave should be present. For example the operating system could choose a particular virtual address page and specify to the hardware that the enclave should be created in this particular page. Now the unique thing about this is that due to the hardware protection mechanisms this is just choosing the page or the address where the enclave is present is about all the operating system can do it will not be able to read or write to the contents within that particular page but rather just manage that page.

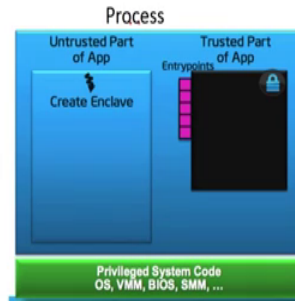
Also while creating the page the operating system would need to specify other aspects such as the operating mode whether it is 32 bit or 64 bit. It needs to specify the processor features that is to be supported and also where debug is allowed or not within that particular enclave.

(Refer Slide Time: 07:12)

Enclave Life Cycle (adding pages)

EADD Instruction

- System software should select free ECS page
- EADD will initialize EPCM with
 - Page type (TCS / REG)
 - Linear address that will access the page
 - RWX permissions
 - Associate the page in SECS structure
- EADD will then record EPCM information in a crypto log stored in the SECS
 - This is the measurement of the enclave
 - Used for gaining assurance
- Copy 4K bytes of data from unprotected memory into the enclave



16

So after creation is done the next step is an EADD instruction. So EADD is also a privileged instruction and therefore it can only be done by operating system. So as before when EADD is invoked the operating system would be capable of selecting a particular ECS page where the enclave code or data is used, so typically the operating system would be able to manage this particular page but would not be able to actually read or write the contents of that page.

So EADD will do 2 things first it will initialize the EPCM as we have seen in the previous lecture. Every ECS page has a corresponding EPCM entry that is the ECS page map which contains essentially the privileges and the access control for that particular ECS page, so for example as we have seen it has the read write execute permissions the address the linear address will access that particular page and also it has some information regarding the page type such as whether it is TCS or REG or so on.

Then done is that EADD completed implemented in the processor will then record EPCM information in a crypto log that is stored in the SECS. So note that the SECS was created during the ECREATE instruction which was done initially and during the EADD per ECS there will be some log which gets appended or added in the SECS. Then the SECS is filled and that is 4 kilobytes of data which is copied from the hard disk that is from the applications binary to the SECS page that is to the DRAM so note that all of this data is encrypted because this data present in this hard disk as well as the DRAM is present outside the processor and as we have seen before any

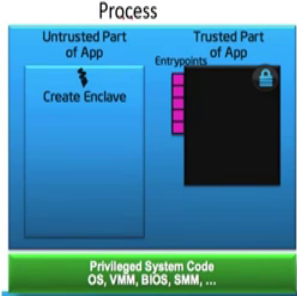
information outside the processor which is any enclave data or code present outside the processor is always in an encrypted state this ensures confidentiality and could also be used to build integrity.

(Refer Slide Time: 09:42)

Enclave Life Cycle (measuring pages)

EEXTEND

- Measure a 256 byte region in an EPC page
 - This region is specified by the developer
 - The measurement comprising of a 64 bit address and a 256 byte information in the SECS
 - 16 invocations EEXTEND needed to measure the whole page
- Correct construction of the enclave would result in a matching with the enclave owner
 - The enclave owner's signature is stored in a SIGSTRUCT structure
 - This can also be remotely verified



The diagram illustrates the Enclave Life Cycle. It shows a 'Process' box divided into two sections: 'Untrusted Part of App' on the left and 'Trusted Part of App' on the right. The 'Untrusted Part of App' contains a 'Create Enclave' button. The 'Trusted Part of App' contains a 'lock' icon and is labeled 'Entrypoints'. Below the process box is a green bar labeled 'Privileged System Code OS, VMM, BIOS, SMM, ...'.

17

So after all pages present in the enclave had been added that is using the EADD instruction so for example if there are like 20 different pages then EADD would be invoked 20 different times one system would then need to call EEXTEND so the EEXTEND would essentially measure a 256 byte region in an EPC page. So this region is specified by the developer the measurement actually comprises of a 64 bit address and 256 byte information present the in SECS.

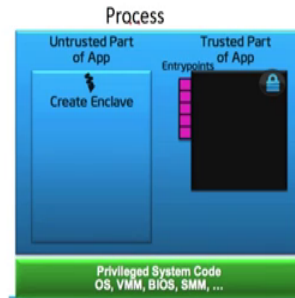
Since we are actually measuring 256 bytes at a time so therefore we have 16 invocations of EEXTEND needed to measure the whole page. So construction of the enclave would actually result in a matching with the enclave owner so this is where we actually would obtain some level of integrity because when an enclave gets create by an application developer the application developer could actually specify which regions in the various EPC pages should be measured. Also, he could also specify a particular signature for those particular EPC pages. Thus what is a certain by the EEXTEND instruction is that the creation of these various EPC pages is exactly similar or has exactly the same signature of what as what the application developer intended.

(Refer Slide Time: 11:27)

Enclave Life Cycle (initializing pages)

EINIT

- Should be invoked after all pages have been added
- Verify that the signature matches that of the owner's signature
- If EINIT is successful, it allows the enclave to be entered



18

So after EEXTEND the next instruction get invoked is EINIT, so EINIT is again a privilege instruction and therefor is should be done by the operating system. So it is invoked after all the pages have been added and essentially it could verify that the signature matches that of the owner signature. So if EINIT is successful it allows the enclave to be entered. So after EINIT that is the first step which is over here so the first step as we seen over here evolves the E create E add E extend and EINIT, so these 4 steps are all done the operating system by application invoking system calls and then requesting application system to create this enclave. So after these 4 steps are done the enclave is ready to use and then the application could actually use various instruction EENTER and EEXIT to enter and leave the enclave.

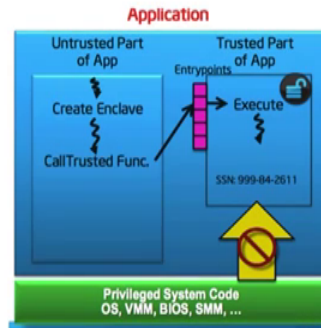
(Refer Slide Time: 12:34)

Enclave Life Cycle (enter/exit)

Process invokes the enclave through pre-defined entry points using EENTER instruction

EENTER

- Changes made to enclave mode
- Need to know the location to transfer control and location where to save state in case of an interrupt
- Defines an Asynch. Exit pointer, which where IRET returns to after servicing an interrupt
 - It is outside the enclave
 - And typically will have an instruction ERESUME



19

So we look more in detail about EENTER where untrusted function could invoke a trusted function which present in the enclave. When the untrusted part of the application invokes EENTER there certain things which are to be specified, so the EENTER instruction needs to know the location within the enclave where the transfer should be done. It should also define something known as asynchronous exit pointer which we will actually see a bit later. So we look at more detail about the asynchronous exit pointer and essentially this is related to interrupt management in an enclave mode.

(Refer Slide Time: 13:14)

Entry into the Enclave

- Set TCS to busy
- Change mode to enclave mode
- Save state of SP, BP, etc. for return in case of async. Exit
- Save AEP
- Transfer control from outside the enclave to inside

20

So when the EENTER instruction is executed by the processor, the processor would set TCS bit the TCS in enclave too busy stating that this particular enclave is not used. It would change the mode from the non-enclave mode to the enclave mode then it would save the state of the stack pointer, base pointer and so on and it will also save something known as the AEP.

So these state saving like context saving in the stack and base pointer and so on are saved so that after returning from the enclave the normal mode of operation can continue. And finally there is a transfer from outside the enclave to inside the enclave and if all permission have been check are correct the hardware will permit the enclave function to execute. So all of these steps are done during the EENTER instruction and as you as we have seen EENTER is a user mode instruction and therefore an application for example wants to do an encryption would invoke EENTER to trigger the encryption in the enclave mode.

(Refer Slide Time: 14:33)

Exit from Enclave

- **EEXIT**
 - Clear enclave mode and flush TLB entries
 - Mark TCS as free.
 - Transfer control outside the enclave

21

Now after the encryption is complete in the enclave mode the application could return from the enclave mode to the non-enclave or the normal mode of operation using the EEXIT system call essentially in this instruction the processor will ensure that the enclave mode flag is cleared which will actually indicate that it is going back to the normal mode and also it will flush the various TLB entries it will mark the TCS as free and it will transfer the control from inside the enclave to outside the enclave. Now typically what is expected is that whenever there is EENTER instruction the only way to exit from the enclave is by this Enclave instruction.

However, there are exceptions to this particular case. The exception occurs when there is an interrupt that occurs when in enclave mode. So for example let us say that the enclave mode trusted function let say an encryption is executing and during this particular period an interrupt occurs during this time the processors would require to service the interrupt.

(Refer Slide Time: 15:44)

Asynchronous Exit (AEX)

- Occurs when an interrupt / exit occurs
- Processor state is securely saved inside the enclave and replaced with synthetic states
- AEP pushed onto the stack
(AEP is a location outside the enclave where execution goes to after IRET)
- After AEX completes, the logical processor is no longer in enclave mode

- Resuming after an interrupt
 - EERESUME instruction is invoked, which restores all registers
 - Typically EERESUME is present at the AEP location
- Resuming after a fault that occurred in the enclave?
 - Eg. A divide by zero

22

Now this makes things a bit complicated because interrupts are asynchronous events and therefore the processor would need to do service this interrupt without compromising on the security of the enclave. So in order to do this there is a special technique known as the AEX or the asynchronous exit which is supported by SGX and this feature would actually permit interrupts to be handled when in enclave mode as well.

So critical in this asynchronous exit is something known as the AEP which stands for the asynchronous exit pointer, so note that the asynchronous exit pointer is actually set up during the EENTER instruction. So what is done here is that some memory location outside the enclave is actually specified and the fact is whenever so it would typically point to a function which gets invoked whenever there is a return from the interrupt. So this deviates from how interrupts are typically managed so as we know from earlier classes that whenever an interrupt completes its execution this is done by the IRET instruction and the IRET instructions would essentially enforce the previous context and the program which was executing prior to the interrupt occurring but continue to execute. Here with SGX things are slightly bit different whenever there

is the IRET instruction that gets executed instead of going back directly to the enclave the function pointed to by this AEP is executed.

So this function would then invoke something known as the ERESUME instruction and ERESUME instruction would then force the processor to move from the non-enclave space to the enclave space. So the ERESUME instruction would essentially restore all the registers and so on.

(Refer Slide Time: 18:02)

Attestation

- system proves to somebody else that it has a particular SGX enclave
- Two attestation techniques
 - Intra machine (prove to another enclave in the same machine)
 - Inter machine (prove to a third party)
- Makes use of a register called MRENCLAVE
 - Contains the SHA-256 hash of an internal log that measures the activity done by the enclave
 - The log contains the pages (code, data, stack, heap) in the enclave
 - Relative position of the pages in the enclave
 - Security flags associated with the pages

Another very important aspect which is supported by Intel SGX is something known as Attestation where this system can actually prove to somebody else may be over the network or another server that it actually has a particular SGX. So this is made possible because of the signature's which can be computed up for the enclave and also the fact that all the information in an enclave is actually encrypted.

So there are 2 Attestation techniques which are possible one is known as the inter machine and the intra machine. So the intra machine Attestation in a scenario where there are multiple applications running in a system and one application having a specific enclave can prove to another application in the same system that it has this particular enclave. So this is done by the signatures and the second thing about the inter machine Attestation where as we mention before the application could actually prove to a third party over the internet that it has a specific enclave.

This has a several applications your encourage will look at Intel poet that is a prove of elapsed time technique which is quit a technique for block chain, where this feature of Attestation is comes in handy and machines systems can actually proves to some other system on the network that it owns a certain enclave and has performed certain specific work. So a lot of this Attestation is possible due to a register known as the MR enclave, so this MR enclave essentially uses a SHA-256 hash of an internal log that measures the activity done by the enclave.

So the log contains the various aspects like it has signatures of the code, data stack and heap in the enclave. It has relative positions of the pages in that enclave security flag associated and so on. So the Attestation process is will not go into too much detail about this particular process but you could actually look at this paper title innovative technologies for CPO based Attestation and Sealing and this was published in HASP 2015, thank you.