Information security - IV
Prof. Kamakoti V
Department of Computer Science and Engineering
Indian Institute of Technology Madras

Lecture - 57
Introduction to Meltdown Attack

So, one of the primary philosophy that we had been following right from the IS 1 course Information Security 1, 2, 3, and currently the 4 is that the theme has been secure systems engineering rather than looking at security of a particular part in a isolation. What we meant by secure systems engineering is that security cannot be viewed in isolation say at a software layer.

Thus if you say that I am making a secure software there are lot of assumptions that we make before we say the that this is a secure software, or if I say I am making a secure hardware again there are lot of assumptions that somebody is going to write the software would write a secure software.

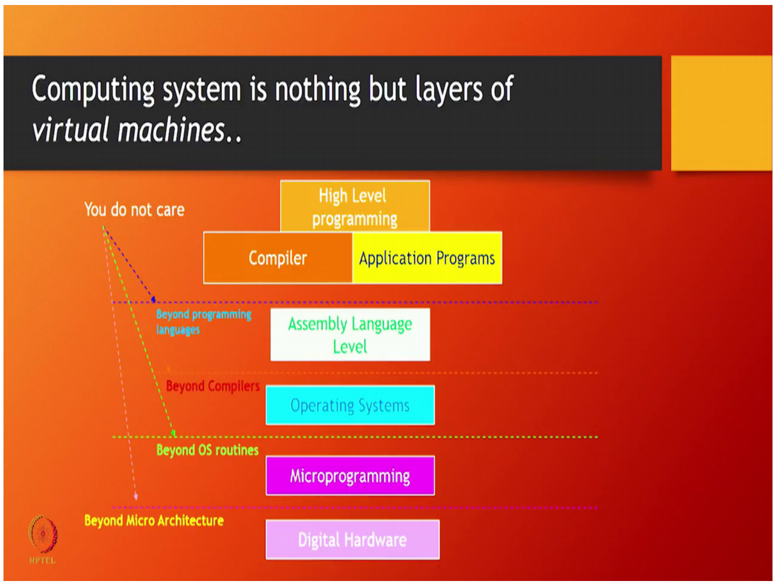
So, security span across all the domains you need to have security at the hardware digital level, otherwise that could be things like site channel attacks where people can say measure for example, I have a secrete key ah, but my secures consumes lot of power when it is processing a 0 when compared to when it is processing a 1, 1; did not know the secrete just keep measuring the power at the underline circuit.

If it consumes lot of power then we says the key that is processing the bit that is processing is 1, if it is consuming less then it can say that the bit it processing is zero. So, now, by this if it starts profiling the power over a period of time then basically it can find out whether what is key like it can be 1 or 0, or 1010.

So, the entire key can be leaked without an actually having access to the key and this is called site channel analysis I do not actually look at the key, but I I study a parameter which varies because of the variations the ones and zeros in the key and by that I could actually go and find what the key is. So, this is the circuit level vulnerability on top of it there is there is a micro architecture, on top of it there is a operating system. Then there are system programs like compiler as a development environments, and on top of it there is an application software.

So, we all these 5 layers one need to have a overall complete outlook before we go and certify a system as a secure systems and this is one philosophy that we have been talking of. So, I will just start of it with one of the slide that I put in information security 1; 3e years before right.

(Refer Slide Time: 02:41)


Computing system is nothing but layers of *virtual machines..*

So, so these are the 5 layers as a digital hardware there is a microprogramming then there is a operating system, then there is a assembly language level or your system programming level, and then there is a high level programming. Now people who work at one level do not bother about the other right.

So, if I am working as a at an application level I do not know which compiler is going to compile which operating system is going to execute, I just write a program and then assume that other things will be taken care of by the other layers. Similarly if I am working at the operating system I really do not know what the hardware is as long as certain interfaces are properly set not understanding what is happening in the other layer being ignorant of the other layer you start writing programs assuming that those layers will do what it is supposed to do which may not be the case. And this ignorance is essentially where vulnerabilities treatment where the security loop holes come up.

And this is what we are told at that point of time when we discussed this in IS the information security 1 course we did not have a case study that will span across of all the 5 things. Now the current meltdown that is reported widely in the literature now we can basically see one attack which spans across at least 3 layers namely the operating system the microprogramming and the hardware layer.

So, there is something that is happening to the cache, that something that is happening to out of order execution that is a micro architecture and this is because the operating system has assumed something and so the bottom 3 layers there is an attack and you need to understand everything about every layer large lot of details about every layer, to basically appreciate why meltdown has come up.

So, this is one very important case study that every security engineer should go through. So, that is aware he or she is aware of how things can happen and so hence when you try to propose a secure solution what is it that you need have in your mind this is a very interesting case study and that is why we have introduced this as a part of this IS 4 course.

Every layer has to be a virtual machine; that means that if I am working at the operating system level I do not care what compiler is there what micro architecture is there as long as certain interfaces are satisfied. If I am working at a systems programming level like I am writing a compiler if I am writing a development environment right. So, I do not really bother about the other layers I just bother about the interface I have with the other layers.

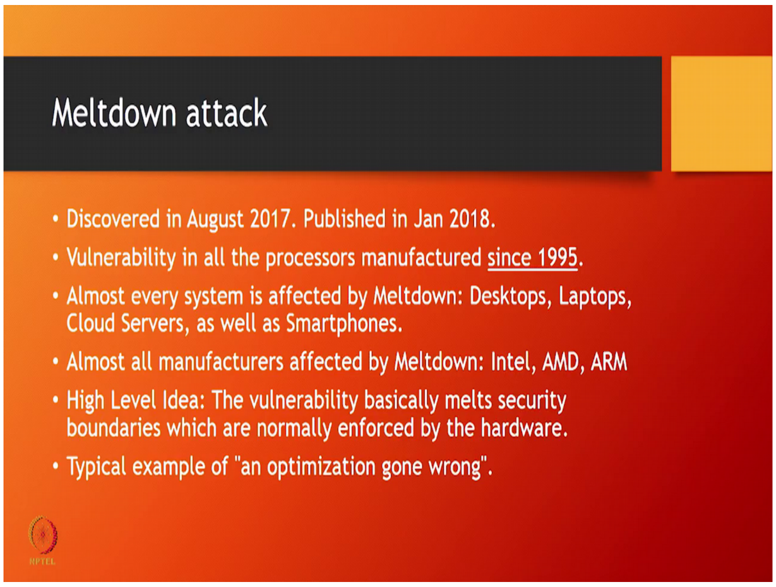
So, how the other layers are implemented it is basically a black box and this is very important because as I given an example in the information security one course. It is very if supposed this type of a virtualization is not there right then even writing an hello world program would be in that I need to you know cover the entire computer science curriculum before people can start writing an hello world program.

For example, you use printf right now printf is there is a device driver involved there is an operating system involved there are there is there is you know the monitor driver the hardware part of the monitor driver is also involved right. So, we need to teach all these things before people can start using printf.

Now all these things are abstracted and given as one hash include s t d i o dot h and then you can start using the printf and then the compiler will borrow code from some where it will it will borrow code from the operating system which in turn can call some device diver which will go and print it out. So, there are lot of things that are involved in the implementation of printf which is completely hidden right and that is an example of a of this virtualization that you are sees seeing in the screen.

Now when we claim to be a security engineer again I repeat then it is not enough that I know each layer or one of this layer in isolation then I say that I will do something secure about that layer that may not work because we need to have a understanding of how that layer interacts with the other layer and all the functionalities a particular layer wants from the other layer how those functionalities are actually implemented. So, this type of an idea if we have then only we can make a secure system and that is something peculiar about secure system engineering and that makes the whole thing much tougher also.

(Refer Slide Time: 07:33)



Meltdown attack

- Discovered in August 2017. Published in Jan 2018.
- Vulnerability in all the processors manufactured since 1995.
- Almost every system is affected by Meltdown: Desktops, Laptops, Cloud Servers, as well as Smartphones.
- Almost all manufacturers affected by Meltdown: Intel, AMD, ARM
- High Level Idea: The vulnerability basically melts security boundaries which are normally enforced by the hardware.
- Typical example of "an optimization gone wrong".

Now let us go a little bit and say what is the, this meltdown attack meltdown attack is nothing but any system any operating system works in two modes. One is the user mode, another is the superviser mode or the operating system mode as a user you know what is a process? Process is nothing, but a programming execution if you have hello world dot C hello world dot C is called a source code or source program when I compile it I get some a dot out and that is an executable program.

The moment I say a dot out and press enter then it becomes a process. So, a process is nothing, but a program in execution every process will have two parts one is the user part and another is the operating system part why should I process, when it is in execution have two parts two parts in the sense that the process is basically exists in the memory.

So, the memory itself is divided into two parts one part basically it will be assigned for the program, the other part is basically assigned for the operating system of the kernel. Why I need a kernel is just see that in the subsequent slide but when a process is executing it will have its own part, and it will also have a kernel part. Now similarly another process is executing it will have a its own part and it will also have its kernel part.

The way the operating system is built is that that the user cannot access the kernel part or without proper authentication in some sense it cannot even touch or view the kernel part right. Now what has happened here is that because of some because of certain issues in the micro architecture definition and also in the circuit implementation we can basically go there is a way by which the user can bypass the security mechanism and go and access the kernel memory.

So, when the architecture is working at user mode the processor is working in the user mode ah. So, if you have done the information security 2 course the processor can work at 4 privileged levels namely privilege level 0, 1, 2, and 3. What I mean by the processor is working at the user mode it is working at privileged level 3 which is the least powerful privilege ok, so it is the user mode.

So, it does not have lot of access into different things it cannot execute some instructions privilege instructions it cannot execute. So, it is just a user process that is running there by definition no memory that is at privilege level 0, 1, and 2 should be could be accessed by a privilege level 3 process. Now what this basically essentially means that if a at a higher privileged level meaning p l 3 higher means numerically higher that you know very less powerful privilege level. The security mechanisms that have been defined to block a p l privileged level 3 core from accessing a memory of privileged level 0 is bridged right.

And the reason for that is some problem with the micro architecture right. So, there so the OS is built based on that assumption that the user part of the memory and the operating system part of the memory are isolated and logically isolated, and there are security mechanisms which will ensure proper access of the user space to the kernel space, but that security mechanism getting bridged essentially has crossed the security vulnerability. So, this is in just what is the meltdown attack.

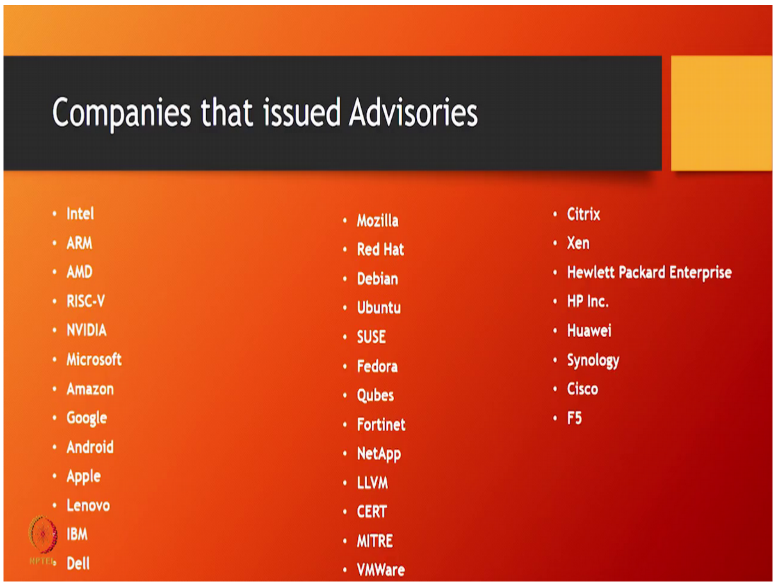
Now this meltdown attack has been discovered in August 2017 and it is published in January 2018 right, and the this has crossed the vulnerability in all the processes that have been manufactured since 1995 right, so that is that is the impact of this meltdown attack. And it involves almost every system like desktop, laptop, cloud servers as well as you know smart phones wherever there was an operating system and wherever there was a distinction between a user processor and the operating system process operating system the entire meltdown has an impact on that.

And so this is this is the major characteristics issue and almost all manufacturers all the major manufacturers processor manufacturers across the globe name to name some Intel MD, they have been affected by this type of a meltdown concept. To some of the high level idea is that the vulnerability basically has meltdown melts the security boundaries which are normally enforced by the hardware right. So, between privileged level 3, and privileged level 0 there is a there is there is a particular boundary that has been established by definition by the hardware and that has meltdown.

So, basically giving a wave for a privileged level 3 process to access privileged level 0 code, so and this is cause because of some optimization issue. So, why did it come up because there is in the architecture there are in the micro architecture that micro architecture is defined well, but we try to optimize and get lot more performance and when this type of a an optimization happened the that optimization has gone wrong. And there something wrong with that optimization which has caused this there something went wrong because of that optimization that has caused this particular vulnerability.

So, with this very quick introduction we will now go into some details of this meltdown attack. I just wanted to this is the last minute addition into this course and we wanted to do this because the attack has come and it is very important that people who are taking this and this is something which is involving the operating system and the hardware. And so this course is the real act place for us to basically give you the complete details of this type.

(Refer Slide Time: 14:07)

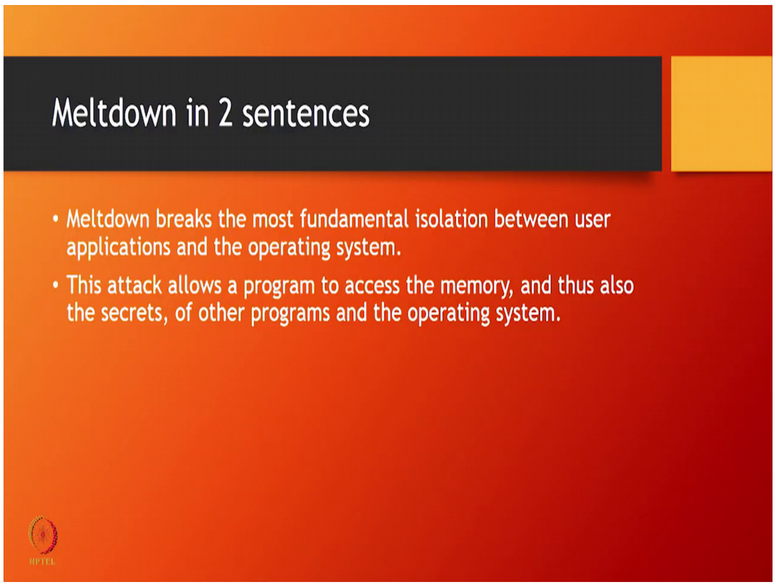


Companies that issued Advisories

- Intel
- ARM
- AMD
- RISC-V
- NVIDIA
- Microsoft
- Amazon
- Google
- Android
- Apple
- Lenovo
- IBM
- Dell
- Mozilla
- Red Hat
- Debian
- Ubuntu
- SUSE
- Fedora
- Qubes
- Fortinet
- NetApp
- LLVM
- CERT
- MITRE
- VMWare
- Citrix
- Xen
- Hewlett Packard Enterprise
- HP Inc.
- Huawei
- Synology
- Cisco
- F5

So, these are the companies that have issued the advisories and that we have more than 3 dozens of companies here, who are actually issued advisories based on this meltdown attack. This is just to give you the impact of this particular vulnerability that has existed in the hardware.

(Refer Slide Time: 14:23)



Meltdown in 2 sentences

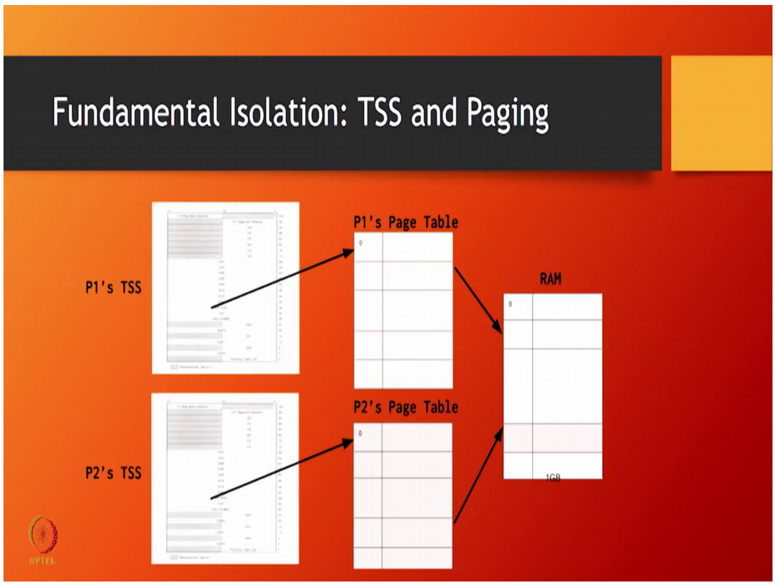
- Meltdown breaks the most fundamental isolation between user applications and the operating system.
- This attack allows a program to access the memory, and thus also the secrets, of other programs and the operating system.

So, in two sentences if I want to summarize a meltdown actually meltdown breaks the most fundamental point that there is an isolation between user applications and the operating system. How is this isolation achieved? Operating system assumes that this isolation is there and the entire security that is built on top of the operating system is essentially based on the availability of this isolation. And how does the hardware ensure this isolation to the operating system if you take the Intel hardware there are 4 privileged levels privilege 0, 1, 2, 3.

The privilege 0 is the very powerful thing that is where the kernel executes; privilege 3 is the least powerful thing where the user programming executes. Now privilege level 3 code or process cannot go and touch a privilege level 0 memory, this is the basic isolation that they are done and that is what is now broken. So, something at a higher privileged level numerically a high privileged level and go and access something at a lower privilege level, so that summarize the first statement that we have put on the slide meltdown breaks the most fundamental isolation between user applications and the operating system.

So, what does it mean? This attack actually allows a program to access the memory and thus also the secrets of other programs and the operating system which it is not supposed to do which is not true in terms of the actual user development manual is the processor company has given to you right. It clearly states that privileged level 3 cannot go and touch anything about privilege 0, but it is now becoming possible right, now why it became possible what all and what is the consequence of that being possible we will just see in the next set of slides.

(Refer Slide Time: 16:16)



Fundamental Isolation: TSS and Paging

Now, I want you to go and look at the IS 2 course the information security 2 course, where we have spend lot of time trying to explain how task switching happens? That is there is a process there is a programming execution, there is something called the context of the process. What is the context of the process? I have said in the information security 2 course that when a when so there are 10 processes, that are lined up to execute. The operating system will execute one by one it will give some, so it will assign say some 5 units of time for each process

So, first 5 unit it will execute process one, then it will pull it out then execute process 2 pull it out, execute process 3 pull it out and go on, so this is called a round robin scheduling. So, every process will have a field that it is executing and that is how every process goes to completion right, so this is the basic scheduling algorithm of the operating system.

Now when one I say process one, is executing then it is pulled out. What you mean by pulling out? I need to save some the context of that process, so that I have to restart it again and when I pull out a process and then restart it should start exactly at the point where it left right, so that is very very important. So, the round robin algorithm works because lot of things work, because the something called a context and that context need to be saved at different times right.

Round robin algorithm scheduling is one instance where the context need to be saved one process is executing, I want to pull out that process then put a new process in then at some point of time the same process which I have pulled out has to start executing again for that I need to start from the point exactly where I left. So, for me to start at the point where I exactly left I need to save the context of the process. So, the context of the process is the minimum information that is necessary for me to restart the process exactly at the point where it left, and that minimum information is basically stored in what you called as the task state segment TSS.

Now look so what you see here on the slide is there are two task stage segment there are two processes that are executing P 1 is executing, and P 2 is executing. Let us say they are going to execute one after the other, so P 1 executes first and it will executes so, for some 5 units of time then the operating system will pull it out when it pulls out all its context meaning the general purpose registers, it difference stack registers, it difference stack pointers all these things gets stored in the task stage segment and the another process say P 2 its context get loaded.

So, it will start executing when P 2 is pulled out then all its current values of the general purpose stage etcetera, as I as you have seeing on the screen will be saved in this in the P 2 task stage segment. And then the P 1 will be restarted what do you mean by the restarting p 1? The content of all these general purposes registers stack etcetera are will be reloaded into the different registers and then P 1 will restart right so this is how say getting like a round robin scheduling works.

More importantly when you look at the task stage segment there is one entry called C r 3. So, if you are looked at information security 2 course C r 3 if where the page directory resides right. So, paging resides so; that means every process will be assigned a page table of it is own we have described that in the information security 2 course and again I am repeating that.

So, every page every every process will have its own page table. What will be stored in the page table the page table will have for the different pages that are going to be allocated to this process the corresponding entry in the RAM, page table is basically translation from the virtual address to a physical address. So, I will have P 1’s page table, I will have P 2’s page table. What could happen is inside a page table there will be some pages that are assigned for the process. There will be some pages that are actually assigned for the operating system.

Why do I need some assignment for the operating system we will now see shortly. But what happens is what every operating system today does is that there are some pages assigned for the process and some pages for the operating system. Similarly in P 2 there will be some for the process, and some for the operating system. Some pages of the operating system are shared between these two processes right.

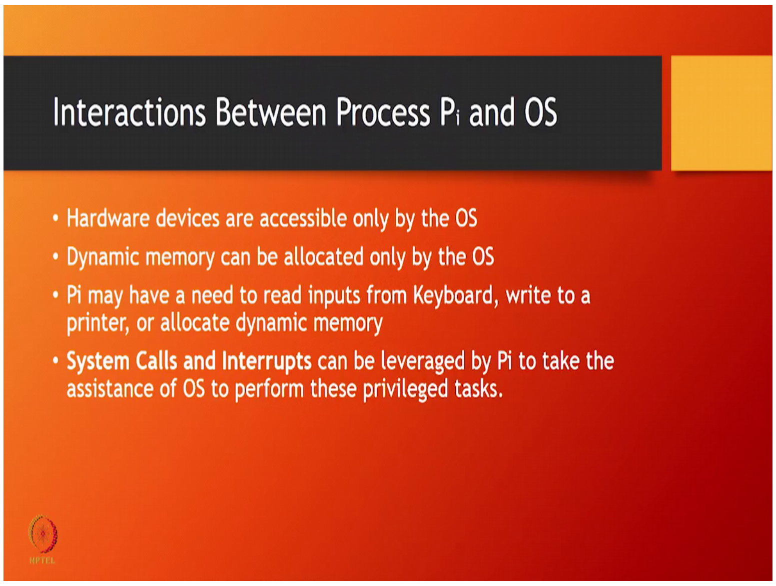
So, this page table will have will point there will some entry which will point to some location in the ram and in the page table for P 2 also the same thing will be pointed. So, though I am changing the page table I can reuse the pages, or I can share the pages between these two processes and which are the pages I will share I will learn or share the user pages are the memory allocated to for the process that I will share the memory that is allocated for the operating system for lot of reasons ok. And so this is this is one very interesting stuff right.

So, so we need to understand that there are pages specifically operating system pages that are being shared between two processes. So, let us first understand how does a process basically talk to the OS? Why should the process at all talk to the OS? Let me give you very simple examples already I talked about printf where I need to talk to the monitor. And I as a user will have only give printf I do not know whether it is a some x companies monitor, or y companies monitor. I do not know how pixels, I do not know what resolution are there nothing I know, I just say printf or a I will just give a graphics image, I will just open up a window where you know opening up lot of graphics image.

I as a user or a programmer high level programmer will not understand anything about the arc hardware or architecture or anything of the monitor itself, or the device driver for the monitor. So, the monitor will have some controller, monitor will have some device driver, I do not know anything about anybody ok, so I just I just write printf ok. So, printf basically when you do a printf as a process when I do a printf I need to call the operating system routine and the operating system routine will take the details of what needs to be printed.

Then it will go and contact the corresponding device driver which in turn will contact the corresponding device controller and then it will then go and print. So, there is lot of action that happens after I just put the printf, as far as I am concerned I do not know anything about printf everything is taken care by the operating systems ok.

(Refer Slide Time: 23:48)



Interactions Between Process P_i and OS

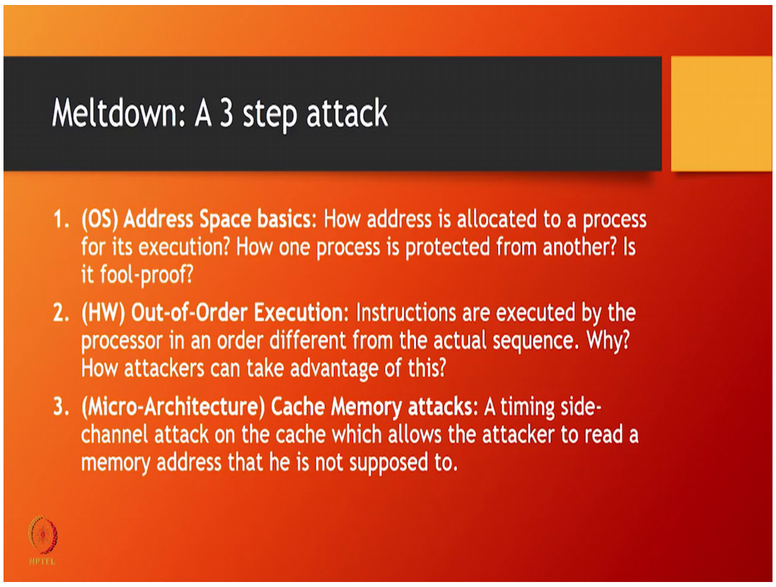
- Hardware devices are accessible only by the OS
- Dynamic memory can be allocated only by the OS
- Pi may have a need to read inputs from Keyboard, write to a printer, or allocate dynamic memory
- **System Calls and Interrupts** can be leveraged by Pi to take the assistance of OS to perform these privileged tasks.

So, so when I execute a printf I need the operating system to be lot more work for me. So, so as a user process I also need access to the operating system, that is why when a page table is allocated for me there will be some memory allocated for me as a process, there will be substantially you know at least equal or larger memory that will be allocated for the operating system to do the work that I am assigning to that operating system fine.

So, I as a processor assigning one very important example which is also of very good relevance here is about malloc right, I as a C code I am asking for some memory right. So, I have who will I ask I should ask the kernel the kernel has to come and give me that memory right, so that is very very important, and that is what that is what is the interaction between the process and the operating system. Typically you can also see even lot of things like I type something on the keyboard, the keyboard has to go through the operating system to the user process.

So, so any interrupt any type of interrupt or any system call that I make from my program or any external interrupt that comes like I am asking scanf and somebody has to the user actually types some input through the keyboard, or mouse etcetera. All these things basically need OS support and that is one of the precise reason why that the page table that we have has both the page table that we have for every process will have some pages for the process and some page for the kernel. So, the a process should always in interact with OS, even for its execution it is not that the process will never interact with OS it has to interact with the OS for many of these things.

(Refer Slide Time: 25:56)



Meltdown: A 3 step attack

1. **(OS) Address Space basics:** How address is allocated to a process for its execution? How one process is protected from another? Is it fool-proof?
2. **(HW) Out-of-Order Execution:** Instructions are executed by the processor in an order different from the actual sequence. Why? How attackers can take advantage of this?
3. **(Micro-Architecture) Cache Memory attacks:** A timing side-channel attack on the cache which allows the attacker to read a memory address that he is not supposed to.

When you want to understand meltdown with this background we have to answer 3 questions. The first question is we need to understand the address space we have talked something about the address space, physical address space, virtual address space etcetera. Now in a process is requesting for memory how address is allocated to a process for its execution? We need to understand some amount of this, and then the next thing is how one process is protected from another and it is free full proof. What do you mean by protecting one process from another? The process is not a human being where I am having so a process who represents the process? The memory represents the process.

So, so there is a process executing what is the identity? What is the trace? The context of the process and the memory that the process execute accesses these are the proof that the process exists right. So, if I want to you know protect one process from a another process basically; that means, I should protect this process x process from accessing any of OS memory, and I should stop the OS process restrict the OS process from accessing any of excess memory right. So, so x and y are two processes and I protect them means I have to protect the memory regions that both of them access right. So, so this is something so we have to understand the OS issue here. Then comes the hardware issue where we are trying to do something called out of order execution and this also we have you know a sort of covered in IS 2 course to a reasonable level.

So, when I have a program it is now a C program that is compiled in the assembling there is an order in which these execute that is called in order execution. So, the assembly language program will have x then next instruction and next instruction. Now by doing this, what will happen one instruction after another instruction are going to execute one after the other.

But suppose bunch of instructions which are not dependent on each other and if I have enough functional units say I have 10 functional unit, 10 addition units, and I have say 100 additions to be performed nothing is depending upon another. Then basically I can take this 100, and with a split spitted across these different units 10 for each for this unit, and then parallelly execute concurrently execute this, when I concurrently execute what will happen, so I have two instructions I 1, I 2; I 2 following I 1 in the in order definition.

Now I will start executing I 1, I also start executing I 2, I 1 can be a slow instruction I 2 can be fast instruction, I 2 can be a bite addition, I 1 can be a floating point division. If that happens I 2 actually finishes before I 1 then that slot can be taken by I 3, and if a I three is very simple instruction I three also can finish before I 1 right or after I 2, I 1 can finish then I 3 can finish.

So, all these possible combination are there now if I 2 actually finishes before I 1, then this is basically called out of order execution. I am not executing exactly in the order is given in the program, but I can actually depending upon the availability of the different units I can go and schedule different instructions and execute then different point of time. So I 2 actually finishes before I 1, so that is what we say it is an out of order execution.

Now out of order execution is done for me to improve this cycles per instruction basically the out of order execution is basically done to improve performance timing performance. So, this optimization essentially has cross the vulnerability if this optimization did not exist probably this vulnerability may not have existed, so there is hardware.

So, what we try we need to understand what is out of order execution off course we will give you a couple of slides to basically re write that, we have done it in the IS 2 course, out of order execution also. Then the third step is off course the micro architecture where in we are still working on the cache issues. So, so one of the important thing is the cache attack or cache memory attack. What is the cache memory attack it is the timing side channel attack on the cache which allows the attacker to read a memory address that he is not supposed to right, so that is also this is basically a micro architecture issue.

So, when we look at meltdown there are 3 major steps. One is the operating system, another is the out of order execution, and another is the micro architecture. In the next session we will talk about address space basics basically the out of content that are covered in information security 2 and information security 3 course. We will try and complete some part of it some relevant part of it from the contrast of meltdown we will we will do it in this section namely address space basics.

Thank you.