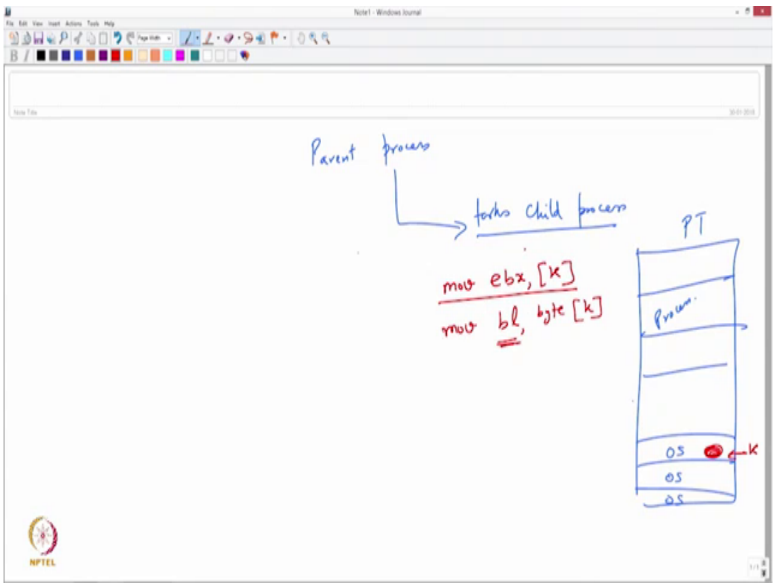
Information security - IV
Prof. V. Kamakoti
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture - 60
Recovering from Exception

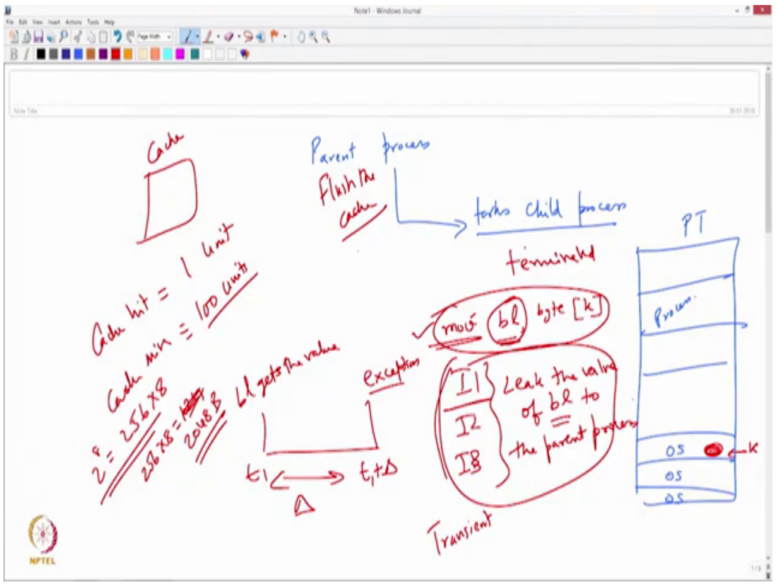
(Refer Slide Time: 00:16)



So, now we will see how the attack can happen, let us say there is a parent process which actually forks a child process. Now this is the pace table for the entire system so these are all processed pages, and these are all OS pages. Now what will this child process to, this child process will access some part of this OS page ok. Let us say it is accessing 1 byte of this OS page, so I can say that move into the register e b x, then I say some, let us say this is ks location or I can even say instead of doing this I want only 1 byte, I can say move b l b l is in Intel’s 8 bit register and then say byte k ok

So, from this k’s location become obviously this is going to give us an exception, but that exception will be detected only after this b l, basically gets a value right. So, what is this subsequent instruction, it will do, it will just you know. So, now after this exception has come right, so this one; so after this exception has come for this after the exception has come that there are going to be some instructions 1 2 3.

(Refer Slide Time: 02:11)



That are following all that these instructions have done will get annulled right so, but let us say there are some instructions here these three these set of instructions will leak the value of this bl to the parent process.

So, what will happen is, and when it will leak between the time, where there is an exception and between the time where there is an exception and between the time, where the value is stored into b l and then here the exception is going to rise. So, say let us at time t 1 b l gets the value at time t 1 plus delta this is where exception is raised. The moment exception is raised, whatever I 1 I 2 I 3 is done is gone. So, between this time interval t, this delta time interval, these set of instructions whatever is following this should somehow leak the value of bl to the parent process right.

Now, what will happen is, once an exception comes here, essentially the child will be terminated. So, once this exception is raised child will be terminated, at that point the parent process will get the control right, and it will the parent process will not do anything about this termination let it go, but it has got the value of bl. So, then what this parent process, will keep on forking multiple child process and in each time, it can get 1 byte or whatever byte and it will get leak information and then essentially the entire kernel dump can be brought out.

So, this is the basic methodology that is followed in meltdown. The reason again is that there is out of order execution. If it was in order execution 1 after another, this instruction would have come, this, once this instruction finishes then only if I say, I 1 can execute, I do not see those, you know if you are seen, if you remember in the previous session we have seen 1 blue bus right, which is feeding into all the execution unit there was no out of order execution at all.

And obviously, I 1 I 2 I 3 will never execute before this move finishes, but we want optima, we want performance and so what we have done, we have allowed out of order (Refer Time: 04:50) and that is a micro architectural decision that we have taken, and because of that what happens, before I even realize that this is an unauthorized access, there are other instructions which will basically execute, assuming this is a correct operation. This is also called speculative execution. Since I speculate that this move will work correctly and I go and execute another instruction.

Because of that what happens some data, these instructions will have access to that b l which is the confidential data. And then now there is the intelligence come here, how I leaked that b l to the parent process, because once the exception is raised, all the things that we have done that I have read b l, but that as a I 1 or I 2 or I 3 I know the value of b l, but the moment that exception happens at this instruction, all these gets automatically annulled by the hardware. So, in that small time window I need to leak the value of b l to this right.

So, now the problem is that the operating system believed that nobody can touch the OS memory at all right. Now because of the optimization of the micro architecture level, it did not know about the micro optimization with the micro architecture level, because of the optimization that we have done at the micro architecture level, what has happened. There is a small window of time where this data could be leaked to another instruction, it is not just it has gone. Now, so till now so that is that is the you know the break.

Now, some other instructions got it so what, because once the exception is raised everything will be erased, but then those instructions can do something intelligent to basically send back the value to the parent process, and that is where the challenge, the third challenge comes ok. So, we had an operating system assumption that got violated because of a micro architecture driven optimization. Now that micro architecture even thought right ok, it is fine if an exception, when they were actually framing this out of order execution. Do you think they would not have taught about? Yes, they would have taught about it, completely, saying that anyway, let it know, let it get the value of e b x e b l here b l here let the other instructions, anyway when the exception is raised everything is going to annul what is issue. These instructions actually exploit a circuit level phenomena namely it is a cache organization which is a circuit level phenomena to basically leak the information

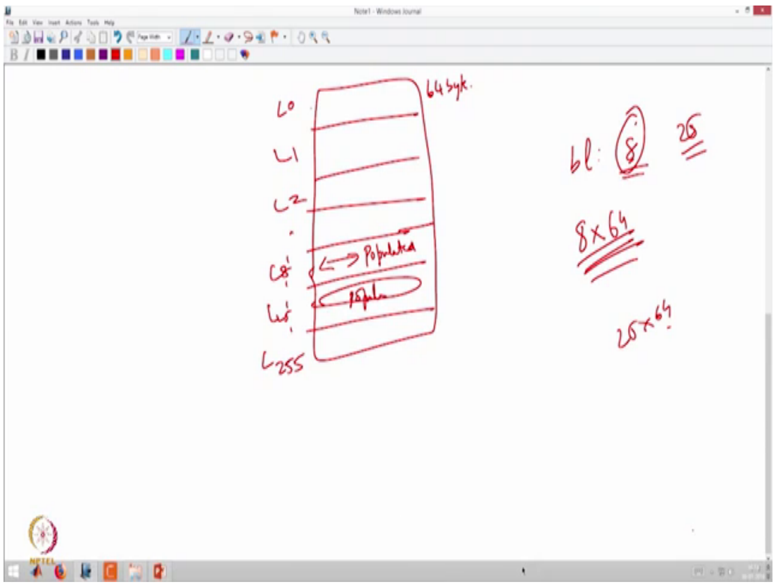
So, there is a side channel, it is called basically a side channel, we will we will see how it is going to happen. So, that is why we have been telling that there is an operating system concept involved that got broken, there is an micro architectural level concept involved it got broken, and then finally, there is a circuit level concept which also enabled this whole attack. So, this is three layers in their operating system, the micro architecture and the circuit level jointly trying to get this melt down in place. So, this is our attack

Now, what needs to be explained now is, what are these three, how these three, two or three or whatever how are these instructions, we call them the literature currently calls them as transient instructions, how are these three instructions going to leak their value to the parent process. Now this is a basically a, this is a parent child orchestration that needs to happen. Now what will the parent do. Now let us understand. Now we have cache, now I am trying to access memory, if it is a cache hit, I say I get it in 1 unit of time. If it is a cache miss, then I take x units of time, several say probably if depending upon the cache org memory organization can be 100 units of time, so if there is a cache miss.

So, and there are performance counters available in all these architecture which will measure the memory access time, you can say whether there is a cache it or not ok. So, that is very important right. So, what we can do what the child process can do is suppose b l. Suppose have given say so this bl is 8 bits, so 8 is 2 per 8 is 256 correct. I could have 256 cache lines right, 256 blocks.

So, let us say every caches is some bytes say some 32 bytes or 64 bytes, let us say every cache line is 64 bytes, so 64 is 8, so let me say that 256 into 8 is 1024 right 2 2048 right. So, so let us go to the next, I will just explain how the cache is now organized and that will give us lot more insight into right.

(Refer Slide Time: 10:36)



Now, the cache as say 256 lines; so let me say line zero line 1 line two line 255. Suppose in my b l I have read say 8 I go, and so let me say that so I i i know, I now go and access the line and the address 8 into 6 each is 64 bytes cache line. So, I go and access 8 into 64. The moment I access 8 into 64, just I access 18 to 64 then that line elate will now be populated right.

So, as a child process suppose I am reading from some kernel I get the value 8, immediately I will go and just access, so go read or write something into 8 into 64 right into an address which is 8 into 64, then automatically that particular cache line gets populated right. Similarly suppose I got, instead of 8 I got 25 I will go and read 25 into 64, so the l 25 will basically get populated.

So, what I can do is as a parent process, I can execute a simple code which will flush all the cache lines, it is very easy to do that; I can flush all the cache lines. So, when the child starts executing all the cache lines are empty. Now this child process basically reads say 8, so it goes and populates only the eighth line, and then what happens then the exception got raised, everything is annulled, but please note that the fact that there was something updated in the cache cannot be annulled right.

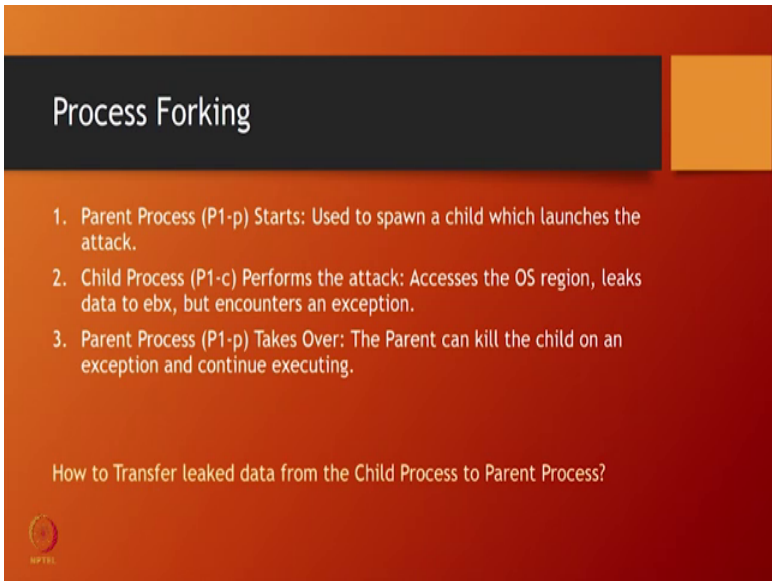
There is something updated in the cache cannot be annulled, so what will happen is that, when I go back to the process main process, the main process will do several things. The main process will start accessing l 0 to l 2551 by 1 it will access, and that it will go and find out if there is some data entered, there is a hit or not right. So l 0 to l, suppose l 8 was there l 0 to l 7 and l 9 to l 2 25, it will always be miss, but l 8 it will find something some access there right.

So, now it will know ok, the value that was read by the child process is 8, so this is how the basic information is leaked. So, let us go back to the previous now. So, this is the parent process what does the parent process da do, it will flush the pipe, flush the cache completely and it will fork a child. What will the child do. It will go and access something in the kernel space and subsequently there will be an instruction, which will, if so it has access to say some 8 it will go and write into the eighth cache line, and then now some exception will come, the registers everything will be erased, but 1 thing what has happened to the cache will remain exactly right.

So, now what, now the parent process, when there is an exception the child actually gets killed, the parent will not do anything fine, it will immediately start accessing all the lines from 0 to 255, it does. First it has flushed everything so nothing will be that, except at that eighth, so that it will take the value, again it will force process, now it will read that b l, now it will write that.

So, byte by byte I can re recover and essentially the entire kernel dump I can get it. So, I can read from k starting from this point I am marking it in green here, staked starting from this point to this point byte I can read every time I will fork a child, I will get the value fork a child, get a value for a child and I can complete the entire stop. So, this is how the meltdown attack has taken place. So, I will just summarize this whole thing with this particular

(Refer Slide Time: 15:27)



Process Forking

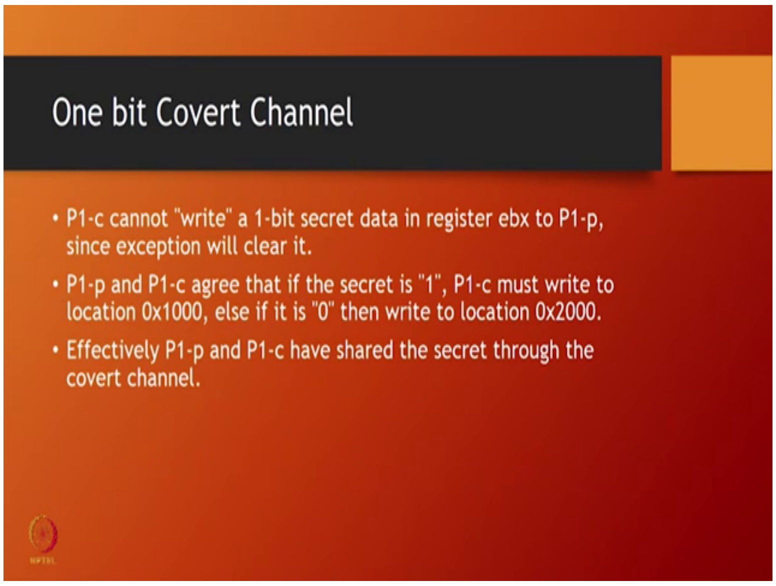
1. Parent Process (P1-p) Starts: Used to spawn a child which launches the attack.
2. Child Process (P1-c) Performs the attack: Accesses the OS region, leaks data to ebx, but encounters an exception.
3. Parent Process (P1-p) Takes Over: The Parent can kill the child on an exception and continue executing.

How to Transfer leaked data from the Child Process to Parent Process?

What happens is the parent process as I told you will spawn a child which launches the attack, the child lances the attack. What will the child do? It will access the voice region and it will leg leak the data to e b x then it will encounter an exception, then what we will do, the parent process will take over, the parent can kill the child on an exception and it will continue executing.

Now the question was that how to transfer the leaked data from the child process to the parent process. So one very interesting thing is, the child process cannot write a 1 bit secret data into register e b x to P 1 to the parent process, since exception will clear it, whatever you do, so when the exception comes whatever the three transient instructions dead that will be cleared.

(Refer Slide Time: 15:58)



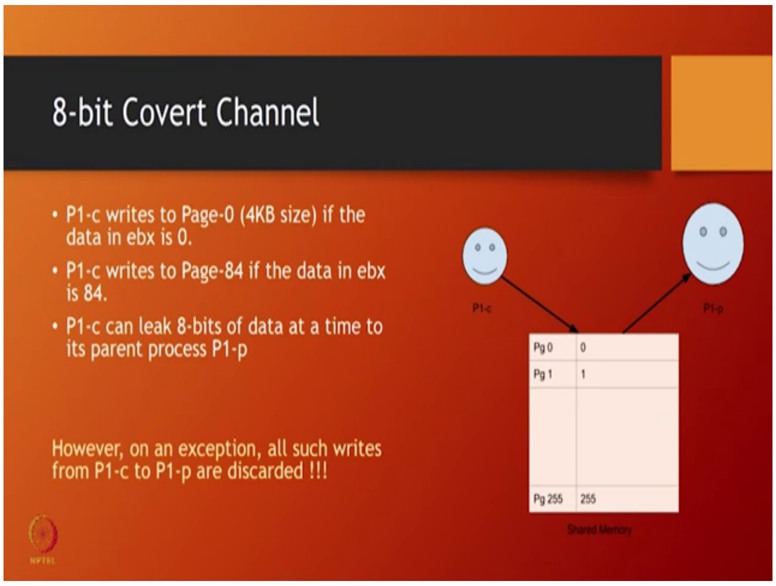
One bit Covert Channel

- P1-c cannot "write" a 1-bit secret data in register ebx to P1-p, since exception will clear it.
- P1-p and P1-c agree that if the secret is "1", P1-c must write to location 0x1000, else if it is "0" then write to location 0x2000.
- Effectively P1-p and P1-c have shared the secret through the covert channel.

So, I the pay child process cannot pass the information to the parent process, basically using a register, because whatever this child process does after that exception. Any instruction the child process executes after that exception, instruction causing the exception that will be annulled.

So, what can P 1 and the P 2 do? P 1 P 1 parent and P 1 child what it can do? they will agree that they, if the secret. So, suppose I am reading bit by bit if the secret is 1, then P 1 must write to location thousand, else it if it is zero it should write some value to a location two thousands, after effectively P 1 P 1 parent and the P 1 child I have shared the secrets through this covered channel, but what will happen is.

(Refer Slide Time: 16:57)



8-bit Covert Channel

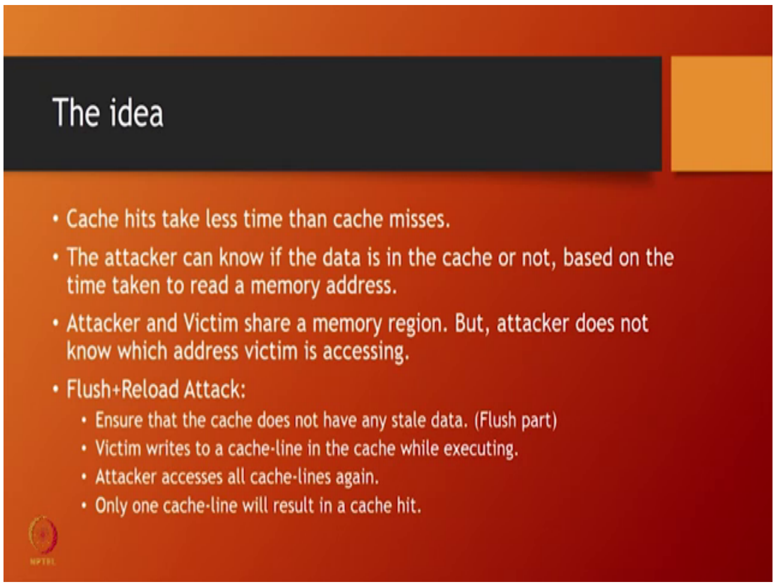
- P1-c writes to Page-0 (4KB size) if the data in ebx is 0.
- P1-c writes to Page-84 if the data in ebx is 84.
- P1-c can leak 8-bits of data at a time to its parent process P1-p

However, on an exception, all such writes from P1-c to P1-p are discarded !!!

P1-c
P1-p
Pg 0 0
Pg 1 1
Pg 255 255
Shared Memory

So, this 1 bit by bit is difficult, let us say byte by byte how will I do? P 1 writes to page zero, it is 4 k b size if the data is in e b x is zero, P 1 child rights to say page 84 if the data and e b x is 84. Instead I could have 256 pages I can write to 1 of these pages. So, the P 1 child can leak 8 bits of data at a time to its parent process P 1 p. now all these rights also will get annulled, also states from P 1 c to P and P will get discarded because once that exception comes the hardware will discard all these weights, so again I cannot go back.

(Refer Slide Time: 17:40)

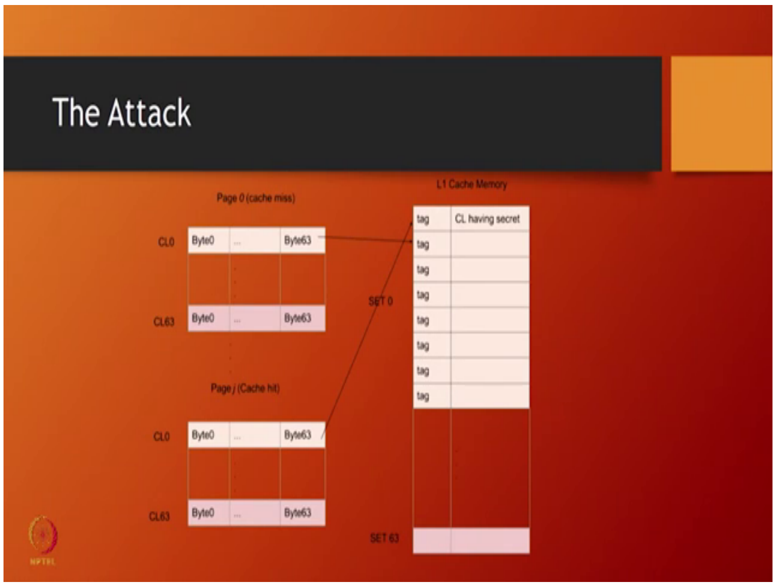


The idea

- Cache hits take less time than cache misses.
- The attacker can know if the data is in the cache or not, based on the time taken to read a memory address.
- Attacker and Victim share a memory region. But, attacker does not know which address victim is accessing.
- Flush+Reload Attack:
  - Ensure that the cache does not have any stale data. (Flush part)
  - Victim writes to a cache-line in the cache while executing.
  - Attacker accesses all cache-lines again.
  - Only one cache-line will result in a cache hit.

So, so the next challenge essentially comes, this is the idea of the cache, so what will the parent process do?

(Refer Slide Time: 17:50)



The Attack

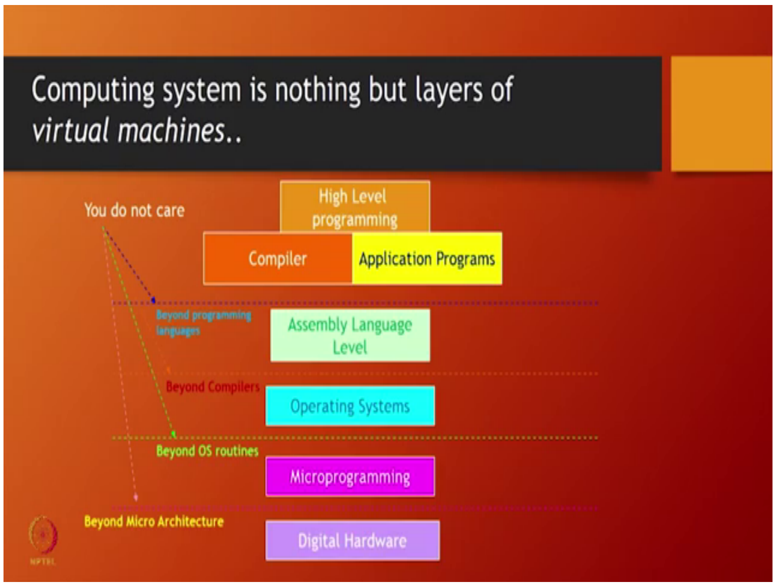
It will first flush everything and give you a fresh cache to this fellow. Now what will the fellow do, when it when it sees a particular value, it will go and access that particular cache, it will not do anything just access that cache. Now after that the exception will come then that parent process again starts and what will the parent process do. So, when I do a fla, this is called a flush and reload attack.

Now, the parent process first ensures that the cache does not have any stale data at all. Now the victim actually writes to a cache line in the cache while executing, now the attacker accesses all the cache lines again and only 1 cache line will result in a cache hit, all the other things would be thing and so we will.

So, this is this is something very interesting and that is where the I will not actually write the data, but I will just get a cache hit there. So, then I know that this is a, this is the exact value, because the line number where the hit has happened is essentially equal to the value I have read, so by this I can get the value.

And so I repeatedly keep spawning child processes and basically get out of them. So, to conclude I think 1 of the basic theory that we have been promoting at IIT Madras on these, we say that security is not an isolated phenomena, it spawns across all layers, as we have been talking of in our first slide of my first I S 1 course.

(Refer Slide Time: 19:26)



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

The entire security spawns across these file errors. Has you see on the screen now and any security vulnerability is not restricted to 1 layer, but it is an act of several layers getting together and trying to get out that vulnerability. All the vulnerabilities that we have seen in the past also has some level of you know misunderstanding between the application and the operating systems right, and some amount of hardware support which was not used to, but this is the first attack which has come up in come up and hopefully we do not have such similar attacks in the future, which has exploited certain micro architecture and circuit level concept.

Of course, at the circuit level there were lot, there is lot of literature, there is a lot of work that is happening on the site channel attacks basically to leak information, but this attack is a different class of its word, where it actually used the cache. It actually used out of order and then it also used the fact that the operating system need to have a single page table, which has the both the OS pages and the process pages, and so that is how this three layers have got together to melt down the, in our security feature.

So, this is a very important eye opener especially for hardware architects, operating system developers, specifically virtualization developers people who rely on what children, which is virtual virtualization, basically the v ms. Those are the people who need to worry about this leaking of secret from 1 virtual machine to another virtual machine. Each virtual machine per says a process and even on network routers where we configure say V P Ns multiple V P Ns are there, and 1 V P N should not talk to another V P N and this isolation we are achieving and these isolations are logical.

So, people have to carefully start looking into those definitions and see how those software has written, and we have to look at whether there are no vulnerabilities at that stage. So, these are something very interesting and from an academic point of view, this a brilliant attack, but you know from a commercial point of view, it is really a big jolt to many. So, with this what we have done in this I S 4 course, is to get you big detail about all the whole we have trying to give you a holistic picture.

So, what we will do in the I S 5 course, that will be in December 2018 or January 2019, we will concentrate more on the hardware side on the digital hardware and micro architecture, and we will be talking about some of the recent hardware developments that have basically helped in trying to get out of many of these security issues. Hopefully we will have a much broader understanding of this type of architectural interventions to ensuring security, and we can give a very nice sum up, finally, end up this course.

Please note that the hardware the digital hardware as you see in the screen is the foundation and if the digital hardware is weak, whatever good things you do in the top can be leaked. So, it is very important that digital hardware and the micro programming level, the micro architecture has to be extremely strong and should have lot of security orientation into it. So, when we decide, when we design a micro architecture, when we design a circuit we should have a lot of security. The thought process of the designer should be no more security oriented.

What is that is something which we need to basically talk about debate and that will be part of our information security 5 courses. I wish you all the best in this course and I hope many of you take the exam and get yourself certified. I also thank the information security education awareness program of the Government of India through the Ministry of Information Technology for actually enabling us to. You know, make these course materials and giving us the platform, where we can basically develop this material.

And of course, the MOOC platform of IIT Madras has enabled us to deliver this course and reach many of you. I am sure it is going to be a very very important exercise for all of us to see that we have where very safe and secure digital world to live in.

Thank you very much.