Remote Code Execution through Intel CPU Bugs

CPU bugs are like a bullet from behind

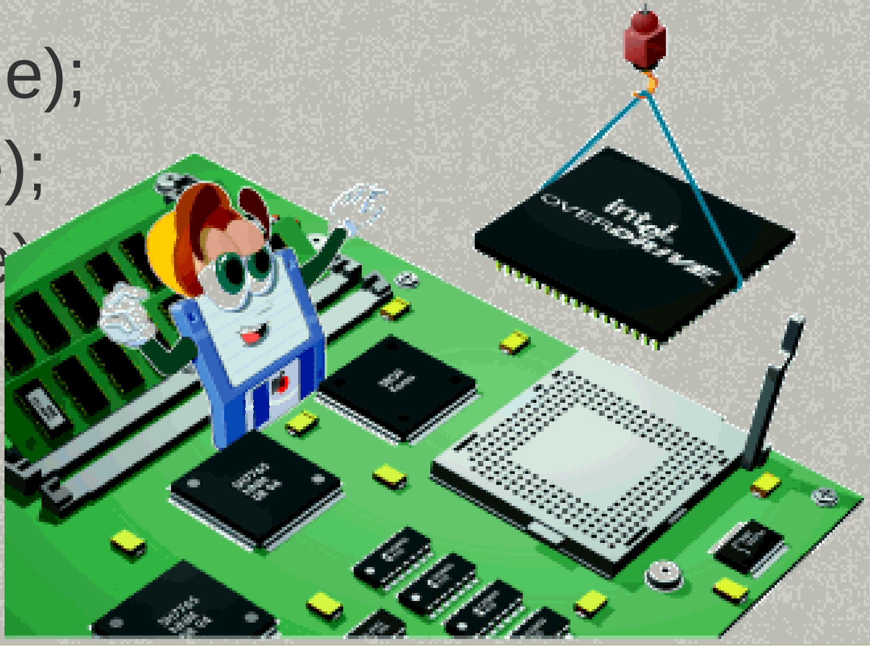
Kris Kaspersky, Alice Chang
Endeavor Security, Inc.

HITBSECCONF2008
27th - 30th October 2008
MALAYSIA
5 Tracks of Hands on Technical Training Sessions
4 Keynote Speakers and over 30 International Experts
Triple-Track Security Conference including new 'HITB Lab'
Capture The Flag 'Live Hacking' Competition
Lock Picking Village by TOOOL.USA
Wireless Village (RFID, Bluetooth, WiFi)
Live DJ set by Electrojunkies
10Mbps INTERNET LINK
VIA METRO ETHERNET!
DEEP KNOWLEDGE
SECURITY CONFERENCE

Who am I?

I never really meant to hurt you bad

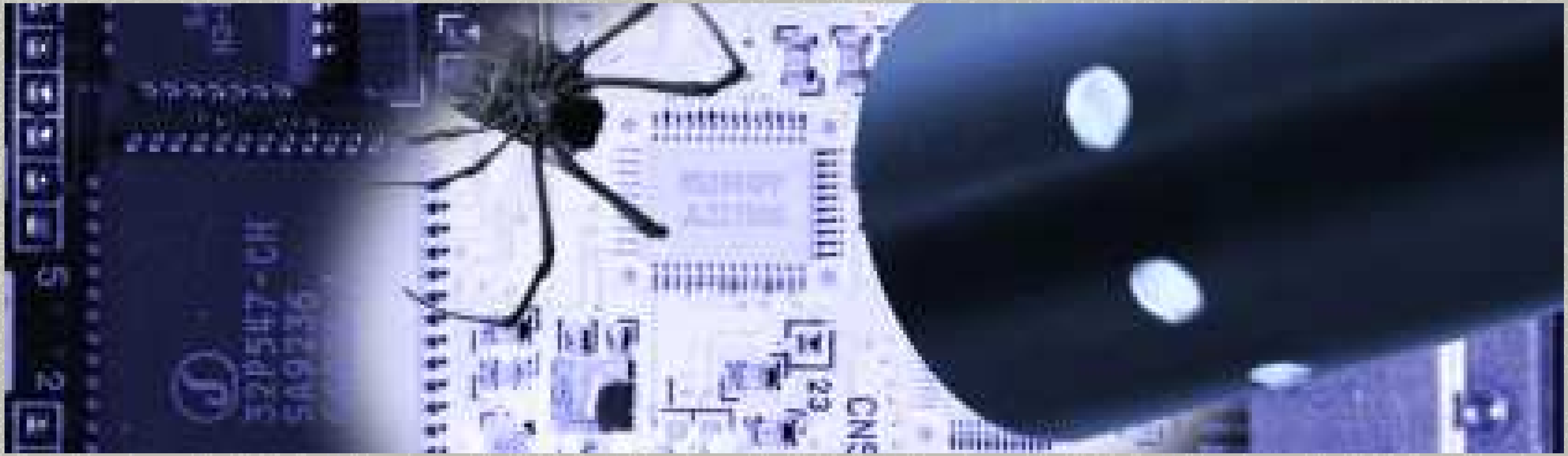
- journalist, reversing as a hobby;
- currently working for:
  - **XAKEP** magazine (www.xakep.ru);
  - **Endeavor Security, Inc** (www.endeavorsecurity.com)
- telephone:
  - +7 (918) 348-92-93 (mobile);
  - +7 (86-140) 5-82-69 (office);
  - +7 (86-140) 5-51-18 (home
- email:
  - poldhiir#gmail.com



The Story

a new age has just begun (continue)

- CPUs always have been buggy!
- Hamarsoft's 86BUGS list:
  - > 100 bugs (last update: Nov 3rd 1994);
  - http://www.xs4all.nl/~feldmann/86bugs.htm;



The Story

a new age has just begun (continue)

- Theo de Raadt:
  - the first to realize CPU bugs are remotely exploitable;
  - **Date: 2007-06-27/From:"Theo de Raadt“/Subj: "Intel Core 2“;**
  - http://marc.info/?l-openbsd-isc&m=118296441702631;
  - OpenBSD is the only OS which fixes CPU bugs by software means;
- Jason Royes (jason.royes#gmail.com) realized:
  - CPU bugs might be exploitable;
  - he researched the treat;
  - he had not published the result;
  - but we were talking about it in the privacy;

The Story

a new age has just begun (continue)

- Selena: underground rootkits writer:
  - July 2007: Interviewed Selena for IT-Spec magazine;
  - according to her, rootkit writers have begun to use CPU bugs for remote attacks;
  - didn't believe her, asked for sample - but she didn't give me anything;
  - started to look for CPU-bug based malware & eventually found it;
  - Intel errata helped to understand the meaning of malicious code;

The Story

a new age has just begun (continue)

- AV industry seems to be aware of it, but nobody is doing anything:
  - fact: malware that uses CPU bugs really does exist;
  - nobody can catch it, since nobody knows how it works or how it looks;
  - actually, it looks innocent!

Why CPU bugs?

they're not what they seem to be!

- Are CPU bugs a real threat?
  - YES!
  - but - don't overestimate it;
  - not apocalypse, just a new threat;
  - malware writers were first, became entrenched;
  - AV vendors will follow, no doubt!
  - users'll be told to upgrade BIOS/buy newer CPUs;

Why CPU bugs?

they're not what they seem to be! (continue)

- Newer CPUs have fewer bugs, look at errata history:
  - Intel Core 2 Duo: ~ 119 bugs;
  - Intel Core 2 Extreme - 128 bugs;
  - Intel Core 2 Extreme Quad-Core - 124 bugs;
  - Intel Core 2 Duo/Extreme Processors on 45-nm - 56 bugs;
  - Intel Core 2 Extreme/ Core 2 Quad on 45-nm - 59 bugs;

Why CPU bugs?

they're not what they seem to be! (continue)

- Intel is working on the problem, fixing old bugs, discovering new ones and fixing them as well;
- what's about AMD and other vendors?!
  - honestly, I just don't know, I'm not AMD fan;
  - don't own a lot of AMD CPU to play with;
  - AMD is also vulnerable, but less popular
  - it's like Opera, Fire Fox and Internet Explorer;
    - IE is more popular, thus undergoes more attacks;
    - Firefox is not bug-free but has fewer attacks…
    - … and Opera even less so.

Attack vectors

malware is coming, are you ready to receive?

- Remote exploitable bugs:
  - executing arbitrary code exploiting bug-free apps (Ring-3) or operation systems (Ring-0);
  - a system hung-up/freezing/reboot/data damaging/undefined behavior;
- Local exploitable bugs (helpers):
  - executing arbitrary code exploiting buggy apps, bypassing OS/compiler protections;
  - a system hung-up/freezing/reboot/data damaging/undefined behavior;

Attack vectors

malware is coming, are you ready to receive? (continue)

- **Anti-reversing tricks (AV developer's nightmare)**:
  - code does something it's not supposed to do: something unexpected;
  - errata helps understand the meaning of the code;
  - but - who reads errata?
  - Rustock.C: first rootkit to use PCI/ISA bridge info as crypto-key;
    - Malware works only on the infected PC, not on reverser PC;
    - *that's pretty brilliant!*
  - CPU bug based malware is the same, but much stronger & better;
    - you can't debug the code, designed for buggy CPU, on a bug-free CPU;
- **Moral: read errata!**
  - very important to read errata, especially for OS and driver writers;
  - programmers should consider CPU bugs when writing software;

Understanding the nature of the attack

face the enemy, meet the reality of CPU bug based malware

- **OS independence**:
  - the code depends on the target OS!
    - uses system calls/APIs as usual;
  - but any OS is vulnerable when the target is the CPU and NOT the OS!
    - CPU allows attackers to inject shell code and passes control to it;
    - OS is just a host;
- **Software protections/patch impotence**:
  - CPU bugs are exploitable via bug-free apps/OS;
  - any app processing data in a certain way, is potentially exploitable;
  - the bug is triggered not when data is received, but when processed;
  - there is no signature, nothing to detect on the wire or software level;

Cache coherence issue (remote attack):

are you brave enough to risk everything you only have?

- **Ultimate goal: inject shell-code into OS kernel. It's possible!**
  - two or more cores process the same data (p.e. decode video stream);
  - cache controller bugs allow data to be written to physical memory at arbitrary location;
  - usually we get a crash (including data loss, file system damage);
  - knowing the nature of the bug, an attacker can write data to ANY physical memory cell;
  - physical addresses are a backyard of OS, but they are predictable enough for attack;
- **What does a hacker need in order to attack?**
  - any app or driver, processing the same data with two or more cores;
    - TCP/IP driver assembles TCP packets, sharing data between cores;
    - possible to attack buggy CPU via packet storm sending to bug-free OS;
- **Moderate goal: cause a crash. It's easy!**

Cache coherence issue (remote attack):

are you brave enough to risk everything you only have? (continue)

- Associated errata:
  - lots of errata related to cache coherence flaws;
  - Intel probably hasn't found, disclosed, and documented all bugs;
  - that's why I have decided not to make the code pub. available;
  - not really my code, just ripped off the particular malware;
  - code will be published as soon as Intel has fixed the bugs;

Cache coherence issue (remote attack):

are you brave enough to risk everything you only have? (continue)

- AI39. 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:
  - when request for data from core 1 results in a L1 cache miss, the request is sent to the L2 cache. if this request hits a modified line in the L1 data cache of core 2, certain internal conditions may cause incorrect data to be returned to the core 1;

Cache coherence issue (remote attack):

are you brave enough to risk everything you only have? (continue)

- AI43. Concurrent Multiprocessor Writes to Non-dirty Page May Result in Unpredictable Behavior:
    - When a logical processor writes to a non-dirty page, and another logical processor either writes to the same non-dirty page or explicitly sets the dirty bit in the corresponding page table entry, complex interaction with internal processor activity may cause unpredictable system behavior;

Cache coherence issue (remote attack): are you brave enough to risk everything you only have? (continue)

- AI79. REP Store Instructions in a Specific Situation May Cause the Processor to Hang:
  - During a series of REP (repeat) store instructions, a store may try to dispatch to memory prior to the actual completion of the instruction. This behavior depends on the execution order of the instructions, the timing of a speculative jump, and the timing of an uncacheable memory store. All types of REP store instructions are affected by this erratum;

The dead end (remote/local attack):

new malware breaks down the system, countdown to extinction

- Errata:
  - AX52. Short Nested Loops that Span Multiple 16-Byte Boundaries May Cause a Machine Check Execution or a System Hang:
    - Under a rare set of timing conditions and address alignment of instructions in a short nested loop sequence, software that contains multiple conditional jump instructions and spans multiple 16-bye boundaries, may cause a machine check exception or a system hang;
- Possible to exploit this bug via JIT-compilers (Java);

Helpers (local attack):

this is real CPU bug-based malware

- **Example**:
  - we have a buggy app causing integer overflow:

  -
    ```
    foo(char *p, int len)
    {
      char buf[XXL];
      …
      if (len > XXL) return -1;
      memcpy(buf, p, len);
      return 0;
    }
    ```
  - this is exploitable:
    - "int" means "signed int", while memcpy() takes "size_t" (unsigned int);
    - if len < 0, memcpy() copies ((unsigned int) len) bytes to buf[XXL];
    - if sizeof(len) == sizeof(DWORD), minimal exploitable len is 8000000h;
    - 8000000h is too much, we'll get a crash (access violation exception);

Helpers (local attack):

this is real CPU bug-based malware (continue)

- possible to overwrite SEH to pass control to shell-code;
- impossible to exploit this bug if there is no SEH (Linux) or if SafeSEH is used (XP +);
- CPU bugs can help to perform the attack!
    - under certain conditions, CPU interrupts REP MOVS before ECX turns to zero;
    - as a result: memcpy() overwrites only return address, does not touch memory outside stack;

Helpers (local attack):

this is real CPU bug-based malware (continue)

- AI30. (E)CX May Get Incorrectly Updated When Performing Fast String REP MOVS or Fast String REP STOS with Large Data Structures:
  - When performing fast string REP MOVS or REP STOS commands with data structures [(E)CX*DataSize] larger than the supported address size structure (64K for 16-bit address size and 4G for 32-bit address size), some addresses may be processed more than once. After an amount of data greater or equal to the address size structure has been processed, external events (such as interrupts) will cause the (E)CX registers to be incremented by a value that corresponds to 64K bytes for 16-bit address size and 4G bytes for 32-bit address size;

Helpers (local attack):

this is real CPU bug-based malware (continue)

- AI37. REP CMPS/SCAS Operations May Terminate Early in 64-bit Mode when RCX >= 0X100000000.
  - REP CMPS (Compare String) and SCAS (Scan String) instructions in 64-bit mode may terminate before the count in RCX reaches zero if the initial value of RCX is greater than or equal to 0X100000000;
- AI58. CMPSB, LODSB, or SCASB in 64-bit Mode with Count Greater or Equal to 248 May Terminate Early.
  - In 64-bit mode CMPSB, LODSB, or SCASB executed with a repeat prefix and count greater than or equal to 248 may terminate early. Early termination may result in one of the following:
    - The last iteration not being executed;
    - Signaling of a canonical limit fault (#GP) on the last iteration;

Helpers (local attack):

this is real CPU bug-based malware (continue)

- AI101. (E)CX May Get Incorrectly Updated when Performing Fast String REP STOS with Large Data Structures:
  - When performing fast string REP STOS commands with data structures [(E)CX*DataSize] larger than the supported address size structure (64K for 16-bit address size and 4G for 32-bit address size), some addresses may be processed more than once. After an amount of data greater than or equal to the address size structure has been processed, external events (such as interrupts) will cause the (E)CX registers to be incremented by a value that corresponds to 64K bytes for 16-bit address size and 5G bytes for 32-bit address size;

Anti-reversing bugs (local):

CPU bugs based malware ready for full-scale war!

- What does the following code do?

```
.00403854:   BC59384000    mov    esp, 000403859
.00403859:   8F442404      pop    d, [esp] [04]
.0040385D:   33C0          xor    eax, eax
.0040385F:   8B00          mov    eax, [eax]
```

  - looks like it just overwrites the machine instruction @ 00403861h;
  - this actually demonstrates the XMC (Cross-Modifying Code) bug;

Anti-reversing bugs (local):

CPU bugs based malware ready for full-scale war! (continue)

- **AI33. Unsynchronized Cross-Modifying Code Operations Can Cause Unexpected Instruction Execution Results:**
  - The act of one processor, or system bus master, writing data into a currently executing code segment of a second processor with the intent of having the second processor execute that data as code is called cross-modifying code (XMC). XMC that does not force the second processor to execute a synchronizing instruction, prior to execution of the new code, is called unsynchronized XMC. Software using unsynchronized XMC to modify the instruction byte stream of a processor can see unexpected or unpredictable execution behavior from the processor that is executing the modified code;
  - In this case, the phrase "unexpected or unpredictable execution behavior" encompasses the generation of most of the exceptions listed in the Intel Architecture Software Developer's Manual Volume 3: System Programming Guide, including a General Protection Fault (GPF) or other unexpected behaviors. In the event that unpredictable execution causes a GPF, the application executing the unsynchronized XMC operation would be terminated by the operating system;
  - In order to avoid this erratum, programmers should use the XMC synchronization algorithm as detailed in the Intel Architecture Software Developer's Manual Volume 3: System Programming Guide, Section: Handling Self- and Cross-Modifying Code;

Anti-reversing bugs (local):

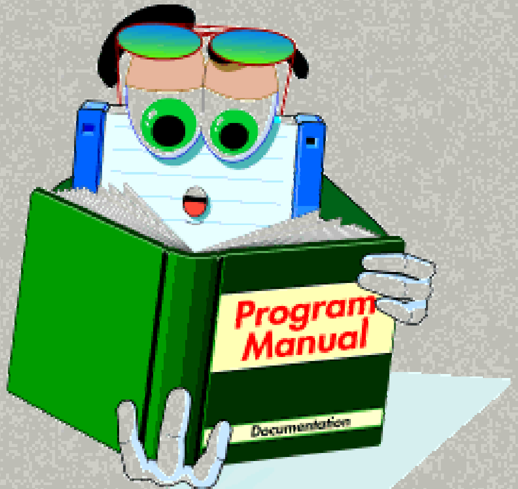
CPU bugs based malware ready for full-scale war! (continue)

- After all - why do we need:
  - **XOR EAX, EAX/ MOVE EAX, [EAX]**?
- Bug is triggered by:
  - an instruction which raises an exception, followed by the modified command;
- Result:
  - behavior of analyzed code is not obvious to those who do not read errata;
- Moral:
  - there are a lot of instructions and situations described in errata, which are different from the manuals;

Bugs in manuals:

CPU bugs turn your data into ashes, then turn ashes into dust!

- Intel manuals contain many errors and misprints:
  - example found by **Bow Sineath**:
  - Intel 64 and IA-32 Architectures Software Developer's Manual Volume 2A: Instruction Set Reference. A-M:
    - 7C cb JL rel8 Jump short if less (SF <> 0F);
  - this should read:
    - 7C cb JL rel8 Jump short if less (**ZF = 0** AND SF <> 0F);



Proof of concept:

the first veil falls, there is no more illusion of safety!

- for some reasons, POCs will not be made public available;

Who protects us?

AV vendors are human just like you, not omniscient gods!

- **Intel has workarounds for almost all bugs:**
  - Intel provides fixed micro-code to major BIOS vendors;
  - who knows which vendor keeps microcode updated?
  - users are in the dark;
- **How to check if you're secure or not?**
  - Intel doesn't provide any test programs;
  - worst of all - some bugs are still unfixed;
- **The computer world is not going to give up:**
  - possible to detect/prevent known CPU-bug based attacks with IDS/IPS;
  - to write the detector, we must know much more information thatn the errata sections disclose;
  - Intel doesn't reveal all details, this is not enough to reproduce a bug;
  - **Endeavor Security, Inc** is working on this, and provides the signature set to IDS/IPS vendors;
    - for more information, visit: **www.endeavorsecurity.com;**

Questions?