How Many Million BIOSes Would you Like to Infect?

Corey Kallenberg & Xeno Kovah
About us

- We do digital voodoo
- Newly independent as of January 2015
- The only company focused primarily on PC firmware security
This talk has 2 main points

• Because almost no one applies BIOS patches, almost every BIOS in the wild is affected by *at least* one vulnerability, and can be infected

• The high amount of code reuse across UEFI BIOSes means that BIOS infection is automatable and reliable
What’s past is prologue

• Some (mostly-multi-vendor) BIOS vulnerabilities disclosed since 2012:
  • And a bunch from others that didn’t get VU#s
THE INCURSION WALL IS HERE.
• In 2008 ITL disclosed an SMM vulnerability where on some Intel motherboards SMM code called through non-SMRAM function pointer table
  – Low hanging fruit SMM vulnerability!
• How prevalent are low hanging fruit SMM vulnerabilities today?

http://invisiblethingslab.com/resources/bh09usa/Attacking%20Intel%20BIOS.pdf
• But how do you hit what you cannot see?
• Option 1: Reprogram firmware to disable SMRAM protection
  – Disable TSEG
  – Disable SMRRs
• Option 2: Use the power of the dark side
• We did a little RE work to determine which SMM code we could invoke from the OS by writing to port 0xB2
• In this case, function 0xDB05EDCC within SMM can be reached by writing 0x61 to port 0xB2
• Almost every UEFI system we surveyed used this format to record reachable SMM code
• We found a lot of these vulnerabilities
• They were so easy to find, we could write a ~300 line IDAPython script that found so many I stopped counting and (some) vendors stopped emailing me back
• You’re the next contestant on... Is it vulnerable???
• Hint: Hexrays detects the external memory accesses and colors them red.
  – When you see red, bad!
int smi_handler_9d37fe78()
{
    __int64 v0; // rax@1

    LODWORD(v0) = v9CEBEBD38(v9CEBECC8);
    v9CEBEE6C = v0;
    return v0;
}
int smi_handler_9d37fe78()
{
    __int64 v0; // rax@1

    LODWORD(v0) = v9CEBEB38(v9CEBEECC8);
    v9CEBEE6C = v0;
    return v0;
}
int smi_handler_9d37fc18()
{
    __int64 v0; // rax@1
    __int64 v1; // rcx@1
    char v3; // [sp+40h] [bp+18h]@1

    LODWORD(v0) = (*(int (__fastcall **)(char *))(v9CEBED58 + 24i64))(&v3);
    v9CEBEE74 = v0;
    if ( v0 >= 0 )
    {
        LOBYTE(v1) = v3;
        LODWORD(v0) = (*(int (__fastcall **)(__int64))(v9CEBED58 + 64i64))(v1);
        v9CEBEE74 = v0;
    }
    return v0;
}
int smi_handler_9d37fc18()
{
  __int64 v0; // rax@1
  __int64 v1; // rcx@1
  char v3; // [sp+40h] [bp+18h]@1

  LODWORD(v0) = (*(int (__fastcall **) (char *)) (v9CEBEE74 + 24164))(&v3);
  v9CEBEE74 = v0;
  if ( v0 >= 0 )
  {
    LOBYTE(v1) = v3;
    LODWORD(v0) = (*(int (__fastcall **) (__int64)) (v9CEBEE74 + 64164))(v1);
    v9CEBEE74 = v0;
  }
  return v0;
char __fastcall smi_handler_bbb8c660(__int64 a1, __int64 a2)
{
    char v2; // b1@1
    signed __int64 v3; // rcx@1
    unsigned __int8 v4; // dl@10
    __int64 v5; // r8@20
    char result; // al@21
    __int16 v7; // [sp+30h] [bp-28h]@20
    __int16 v8; // [sp+32h] [bp-26h]@20

    v2 = vEFF01040;
    vEFF01040 |= 0x30u;
    v3 = 3149860880164;
    qword_BBB8BDCF8 = 3149860880164;
    if ( v1D2 == -5200 || v1D2 == -5549 )
    {
        LOBYTE(a2) = vF803A;
        vBB29C788(&qword_BBB8BDCF8, a2);
        if ( vF803A )
        {
            vBB24A1C0();
            vBB2893C0();
            vBB27B380();
            if ( v1C5 )
                vBB2893C8(432164);
        }
    }
}
void __fastcall smi_handler_da0889e8(__int64 a1, __int64 a2)
{
    __int64 *v2; // rdx@2

    if ( *(__QWORD *)a2 == 0x90164 )
    {
        v2 = &qword_DA087B78[145];
        switch ( vD8AD8024 + 0x80000000 )
        {
        case 0u:
            vD8AD801C = sub_DA088E40(vD8AD8018, (__int64)&qword_DA087B78[145]);
            break;
        case 1u:
            sub_DA088E50(vD8AD8018, vD8AD801C);
            break;
        case 2u:
            sub_DA088584();
            break;
        }
void __fastcall smi_handler_da0889e8(__int64 a1, __int64 a2)
{
    __int64 *v2; // rdx@2

    if ( *(__QWORD *)a2 == 0x90164 )
    {
        v2 = &qword_DA087B78[145];
        switch ( vD8AD8024 + 0x80000000 )
        {
            case 0u:
                vD8AD801C = readmsr_wrapper(vD8AD8018, (__int64)&qword_DA087B78[145]);
                break;
            case 1u:
                wrmsr_wrapper(vD8AD8018, vD8AD801C);
                break;
            case 2u:
                break;
            case 3u:
                break;
            case 4u:
                break;
        }
    }
}
Looking at Acer in IDA
Vendor Response

• Many vendors didn’t reply to our emails and/or claimed they weren’t vulnerable
  – They are vulnerable
• Dell responded and is pushing patches for all of our disclosures
• Lenovo also responded and is releasing patches
What’s possible once you’ve broken into BIOS/SMM?
Hello my friends. Welcome to my home in the Deep Dark
Is it safe to use Tails on a compromised system?

Tails runs independently from the operating system installed on the computer. So, if the computer has only been compromised by software, running from inside your regular operating system (virus, trojan, etc.), then it is safe to use Tails. This is true as long as Tails itself has been installed using a trusted system.

If the computer has been compromised by someone having physical access to it and who installed untrusted pieces of hardware, then it might not be safe to use Tails.

• Tails says that because it runs independent of the operating system, if you have previously been compromised by software means (not physical access), you should be safe...
• Exploit delivered remotely on target Windows 10 system.
  – No physical access is necessary
  – All you need is a remote cmd.exe with admin access
• Exploit bypasses BIOS flash protection and reprograms the portion of the flash associated with System Management Mode
• Malware that was delivered remotely to the main OS (Windows 10) waits in the background and runs in System Management Mode
• It waits for your secrets to be revealed
If you are practicing OPSEC, perhaps you have a private email and private key that you only access from the “secure” Tails so to avoid having confidential communications compromised.
• Using this style of OPSEC, the password for your key should never be entered on your normal operating system (Win10 in this case).

• Since we are in Tails, we are okay though...
• Hence all of your confidential communications should remain shielded from any malware that was delivered to your Win10 installation

• Using our malware, this isn’t the case...
• Runs independent of any operating system you put on the platform

• Has access to all of memory

• Can steal all of your secrets no matter what “secure operating system” you are using
**Starting Wiping the memory, press Control-C to abort earlier**

Help: "/usr/bin/sdmem -h"

Tails also attempts to erase memory to scrub any residual secrets that may be exposed to the main operating system
Our malware still has access to it, as we store the secrets to non-volatile storage so we can exfiltrate at earliest convenience.

– So even if you were to use Tails in offline mode, to try to avoid exfiltration of secrets, you can still be owned.
Is it safe to use Tails on a compromised system?

Tails runs independently from the operating system installed on the computer. So, if the computer has only been compromised by software, running from inside your regular operating system (virus, trojan, etc.), then it is safe to use Tails. This is true as long as Tails itself has been installed using a trusted system.

If the computer has been compromised by someone having physical access to it and who installed untrusted pieces of hardware, then it might not be safe to use Tails.

• Time to rethink this...
All fall before a LightEater

• The US Air Force made the “Lightweight Portable Security” (LPS) Live CD\(^1\) with much the same purpose as Tails:

• “LPS differs from traditional operating systems in that it isn't continually patched. LPS is designed to run from read-only media and without any persistent storage. Any malware that might infect a computer can only run within that session.”

• “LPS-Public turns an untrusted system (such as a home computer) into a trusted network client. No trace of work activity (or malware) can be written to the local computer. Simply plug in your USB smart card reader to access CAC- and PIV-restricted US government websites.”

• Attackers that infect BIOS will always win against non-persistent OSes, because they can persist across reboots, and live in OS-independent SMM

\(^1\)http://www.spi.dod.mil/lipose.htm
TELL EVERYONE THEIR COMPUTERS ARE ARCHITECTURALLY INSECURE AT THE LOWEST LEVELS AND NOBODY BATS AN EYE

STEAL ONE GPG KEY FROM MEMORY IN TAILS AND EVERYONE LOSES THEIR MINDS
Where’s the architectural flaw?

• The fact that SMM can read/write everyone’s memory is an x86 architectural vulnerability

• No security system (virtualization, live CDs, normal OSes) is secure until this is fixed
  – We’ll come back later to how we intend to fix it
This talk has 2 main points

• Because almost no one applies BIOS patches, almost every BIOS in the wild is affected by *at least* one vulnerability, and can be infected.

• The high amount of code reuse across UEFI BIOSes means that BIOS infection is automatable and reliable.
Further Tales from the Deep Dark

• I’m going to explain why infecting BIOSes is a lot easier than you may have realized
Infection Decision Tree

1. **Want to infect BIOS**
   - **UEFITool FTW?**
     - **Yes**
       - Publicly defeated sanity checks?
         - No/Don’t Know
           - Find hook points
         - Yes
           - No-op checks
             - BIOS Infected
     - **No**
       - **UEFITool FTW?**
         - **Yes**
           - **UEFITool FTW?**
             - **Yes**
               - **UEFITool FTW?**
                 - BIOS Infected
             - **No**
               - **UEFITool FTW?**
                 - BIOS Infected
             - **UEFITool FTW?**
               - BIOS Infected
             - **UEFITool FTW?**
               - BIOS Infected
Infection Decision Tree

- Want to infect BIOS
  - UEFITool FTW?
    - Yes
      - MSI Demo
    - No
      - Publicly defeated sanity checks?
        - Yes
          - Insert hooks
        - No/Don’t Know
          - No-op checks
  - No
  - Publicly defeated sanity checks?
    - Yes
      - Insert hooks
    - No/Don’t Know
      - No-op checks

BIOS Infected
“UEFITool FTW” Infection

• As done on the MSI
• Use Nikolaj Schlej’s excellent UEFITool\(^1\) to replace the module you care about with one that includes malicious functionality

\(^1\)https://github.com/LongSoft/UEFITool
Infection Decision Tree

1. Want to infect BIOS
2. UEFI Tool FTW?
   - Yes
     - No-op checks
     - BIOS Infected
   - No
     - Publicly defeated sanity checks?
       - Yes
         - Insert hooks
         - BIOS Infected
       - No/Don’t Know
         - Find hook points
         - BIOS Infected

HP Demo
Sanity Check Speed Bumps

• Some vendors like HP build in sanity checks
• Descriptions of bypasses can be easily found on the net, and would be developed quietly by anyone who actually cared enough
• We created a 9 byte signature for one HP sanity check by following the steps in a public blog post
  – And 2 variant signatures based on looking at a few models where the signature didn’t fire
  – The 3 signature variants matched 570/789 HP BIOS images
    • Could be improved further, but we’re just making a point
• If signature found, replace the last 2 bytes w/ 0x9090
• Goto previous slide
LightEater on HP

• For a change of pace, let’s see how easy evil-maid / border-guard / interdiction attacks are!
• NIC-agnostic exfiltration of data via Intel Serial-Over-LAN
• Option to “encrypt” data with bitwise rot13 to stop network defenders from creating a “Papa Legba” snort signature :P
A word about AMT SOL

- Unlike past work for low level networking[10-12], we don’t need to know anything about the NIC
- We write to a fake serial port that AMT creates
- AMT magically translates it to Intel’s proprietary SOL protocol (that there’s no wireshark dissector for)

Infection Decision Tree

Want to infect BIOS

UEFITool FTW?

Publicly defeated sanity checks?

Find hook points

No/Don’t Know

No-op checks

Insert hooks

BIOS Infected

Asus Demo
“BIOSkit” Infection

• Sometimes UEFITool doesn’t work, and you don’t care enough to RE why

• Fall back to the generic technique of “hook-and-hop”, just like a normal bootkit
  – Just starting earlier, and more privileged

• You’re more or less guaranteed that there’s an easily targeted, uncompressed, easy-to-hook starting location in the PEI core file
Reminder of how normal bootkits work

Windows Boot Process

**START**

- BIOS
  - Master Boot Record
  - Partition Bootloader
  - `ntldr / bootmgr`
  - OS Loader
  - `winload.exe`

**END**

- NT kernel

Ntldr = 16-bit stub + OS Loader (just binary appended)
Windows Vista splits up `ntldr` into `bootmgr`, `winload.exe` and `winresume.exe`

<table>
<thead>
<tr>
<th>Windows XP</th>
<th>Windows Vista</th>
<th>Processor Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ntldr</td>
<td>bootmgr</td>
<td>Real Mode</td>
</tr>
<tr>
<td>OS Loader</td>
<td>OS Loader</td>
<td>Protected Mode</td>
</tr>
<tr>
<td>-</td>
<td><code>winload.exe</code></td>
<td>Protected Mode</td>
</tr>
<tr>
<td>NT kernel</td>
<td>NT kernel</td>
<td>Protected Mode + Paging</td>
</tr>
</tbody>
</table>
PEI = Pre EFI Initialization
PEIM = PEI Module
IPL = Initial Program Loader
DXE = Driver Execution Environment
SMM = System Management Mode
BDS = Boot Device Select

The UEFI skeleton
(that all vendors just add their own meat to)
Minimal hook paths in UEFI
Minimal hook paths in UEFI
LightEater on ASUS

- Uses hook-and-hop from DXE IPL to SMM
- From SMM attacks Windows 10
- Gets woken up every time a process starts, prints information about the process
Evidence of Scalability of Infection

• We wanted to show that the code an attacker wants to find can easily be identified with *simple and stupid* byte signatures

• Only took a couple days to develop
Example: DXE to BDS transition

- EDK open source code for DXE -> BDS transition
- DxeMain.c
- Equivalent exact assembly found in 6 separate vendors’ BIOSes

```c
//
// Transfer control to the BDS Architectural Protocol
//
gBds->Entry (gBds);
```

```
4C 8B 1D 8A AF 00 00  mov  r11, cs:gBDS
49 8B CB  
41 FF 13  mov  rcx, r11
49 8B CB  
41 FF 13  call  qword ptr [r11]
```

_Yara rule = {4C 8B 1D [4] 49 8B CB 41 FF 13}_{

(yes, I know, I obviously should technically make it register-independent, but I don’t care because it worked well enough as you’ll see in a second :P)
Analysis targets

- Created YARA signatures from what the code looked like on 9 systems
- Key for next slides: “1,1,2” = “PEI_TO_DXEIPL variant 1, DXEIPL_TO_DXE variant 1, and DXE_TO_BDS variant 2 matched for this system”
Some Analysis Results

• Teddy Reed graciously provided the data set from his 2014 Infiltrate talk\(^1\)
• 2158 BIOS images spidered from Lenovo, HP, Dell, Gigabyte, & Asrock’s websites
  — Haven’t counted how many individual models yet
• Signature scanning results:
  — PEI_TO_DXEIPL: 3 misses (1 model)
  — DXEIPL_TO_DXE: 0 misses
  — DXE_TO_BDS: 4 misses (2 models)

\(^1\)“Analytics, and Scalability, and UEFI Exploitation (Oh My)” – Teddy Reed
For reading at your leisure
(from Teddy Reed’s data set)
(2158 images, 7 misses)

Lenovo (442 images)
- PEI_TO_DXE: 0 misses
- DXE_TO_BDS: 2 misses

HP (388 images)
- PEI_TO_DXE: 0 misses
- DXE_TO_BDS: 0 misses

Dell (381 images)
- PEI_TO_DXE: 3 misses
- DXE_TO_BDS: 2 misses

Gigabyte (347 images)
- PEI_TO_DXE: 0 misses
- DXE_TO_BDS: 0 misses

Asrock (596 images)
- PEI_TO_DXE: 0 misses
- DXE_TO_BDS: 0 misses
For reading at your leisure
(from a completely different LegbaCore data set)
(1003 images, 5 misses)

<table>
<thead>
<tr>
<th>Company</th>
<th>Total Images</th>
<th>PEI_TO_DXEIPL Misses</th>
<th>DXEIPL_TO_DXE Misses</th>
<th>DXE_TO_BDS Misses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lenovo</td>
<td>213</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>HP</td>
<td>401</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Dell</td>
<td>348</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LG</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Asus</td>
<td>13</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Acer</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Model 1</td>
<td>Model 2</td>
<td>Model 3</td>
<td>Model 4</td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>---------------------------------------------</td>
<td>---------------------------------------------</td>
<td>---------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>HP EliteBook 750 G1 NoteBook PC (ENERGY STAR)</strong></td>
<td><strong>HP EliteBook 755 G2 NoteBook PC (ENERGY STAR)</strong></td>
<td><strong>HP EliteBook 740 G1 NoteBook PC (ENERGY STAR)</strong></td>
<td><strong>HP EliteBook 745 G2 NoteBook PC (ENERGY STAR)</strong></td>
<td></td>
</tr>
<tr>
<td><img src="image1" alt="HP EliteBook 750" /></td>
<td><img src="image2" alt="HP EliteBook 755" /></td>
<td><img src="image3" alt="HP EliteBook 740" /></td>
<td><img src="image4" alt="HP EliteBook 745" /></td>
<td></td>
</tr>
<tr>
<td><img src="image5" alt="HP EliteBook 720" /></td>
<td><img src="image6" alt="HP EliteBook 725" /></td>
<td>![HP ZBook 15 G2 Mobile Workstation (ENERGY STAR)]</td>
<td>![HP ZBook 17 G2 Mobile Workstation (ENERGY STAR)]</td>
<td></td>
</tr>
<tr>
<td><img src="image7" alt="HP EliteBook 840" /></td>
<td><img src="image8" alt="HP EliteBook 850" /></td>
<td>![HP ZBook 14 Mobile Workstation (ENERGY STAR)]</td>
<td><img src="image9" alt="HP EliteBook 820" /></td>
<td></td>
</tr>
<tr>
<td><img src="image10" alt="HP EliteBook 820" /></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image11" alt="HP EliteBook 820" /></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

61
Extrapolation to millions

<table>
<thead>
<tr>
<th>Company</th>
<th>4Q14 Shipments</th>
<th>4Q14 Market Share (%)</th>
<th>4Q13 Shipments</th>
<th>4Q13 Market Share (%)</th>
<th>4Q14-4Q13 Growth (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lenovo</td>
<td>16,284.8</td>
<td>19.4</td>
<td>15,153.5</td>
<td>18.3</td>
<td>7.5</td>
</tr>
<tr>
<td>HP</td>
<td>15,769.6</td>
<td>18.8</td>
<td>13,591.3</td>
<td>16.4</td>
<td>16.0</td>
</tr>
<tr>
<td>Dell</td>
<td>10,674.1</td>
<td>12.7</td>
<td>9,810.6</td>
<td>11.8</td>
<td>8.8</td>
</tr>
<tr>
<td>Acer Group</td>
<td>6,786.9</td>
<td>8.1</td>
<td>6,083.4</td>
<td>7.3</td>
<td>11.6</td>
</tr>
<tr>
<td>ASUS</td>
<td>6,259.8</td>
<td>7.5</td>
<td>6,220.2</td>
<td>7.5</td>
<td>0.6</td>
</tr>
<tr>
<td>Others</td>
<td>27,971.5</td>
<td>33.4</td>
<td>32,070.0</td>
<td>38.7</td>
<td>-12.8</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>83,746.7</strong></td>
<td><strong>100.0</strong></td>
<td><strong>82,929.1</strong></td>
<td><strong>100.0</strong></td>
<td><strong>1.0</strong></td>
</tr>
</tbody>
</table>

Notes: Data includes desk-based PCs, notebook PCs, premium ultramobiles and all Windows-based tablets. It excludes Chromebooks and other non-Windows-based tablets. All data is estimated based on a preliminary study. Final estimates will be subject to change.

Source: Gartner (January 2015)

From https://www.gartner.com/newsroom/id/2960125
Extrapolation to millions

• So if almost no one applies BIOS vulnerability patches...
• And if my tiny set of signatures can reliably find hook points and disable sanity checks in the machines HP is currently selling...
• And if HP shipped ~15M PCs in Q4 2014...
• Then we would understand that millions of these BIOSes could be reliably infected, yes?
It was about this time I got really tired of making these slides and manually downloading BIOSes ;)
A little good news before we go

- Were working with vendors like Dell to do security assessments to find and fix issues before they ship on new systems. Lenovo and others are also on the list.
A little good news before we go

• We’re also working with Intel to try to create the first commercial-grade SMM isolation
• Intel has the ability for their hardware virtualization to jail SMM
• We will then work with BIOS vendors to incorporate the technology into shipping systems
• End result will be that even if attackers break into SMM, they can’t read/write arbitrary memory
  – And we could detect attackers through measurements.
Conclusions

The dark dimension’s grip is tight.

Papa Legba sees me through--barely.
This talk has 2 main points

• Because almost no one applies BIOS patches, almost every BIOS in the wild is affected by at least one vulnerability, and can be infected

• The high amount of code reuse across UEFI BIOSes means that BIOS infection is automatable and reliable
What we showed

• All systems we have looked at contain Incursion vulnerabilities that allow breaking in to SMM
• Incursion vulnerabilities can be found programatically
• The LightEater SMM attacker can perform any attack that is available to lesser attackers
  – We showed stealing GPG keys/messages from memory (on MSI), data exfiltration over the network (on HP), Windows kernel rootkit behavior (on Asus)
• Showed how a physical BIOS attack can be done in 2 minutes by an unskilled accomplice (maid/border guard)
• Homogeneity in UEFI BIOSes for the things an attacker cares about. Creating signatures from ~10 BIOSes is sufficient to find matches on thousands of images (which relate to millions of shipped machines)
Conclusions

• 2 guys + 4 weeks + $2k = Multiple vendors’ BIOSes infected, with multiple infection capabilities
• One hand (purposely) tied behind our backs: Didn’t use special debug hardware. Serial prints only!
• Do you really think that Five Eyes are the only ones who have developed capabilities in this space?
• “Absence of evidence is not evidence of absence”
• *It’s time to start checking your firmware*
  – Stop giving firmware attackers a free pass and indefinite invisibility
• *It’s time to start patching your BIOSes*
  – Demand the capability from your patch management software
• *It’s time to demand better BIOS security from your OEM*
  – We’ll eventually make a name-and-shame list of vendors who are perennially leaving their customers open to BIOS attacks
Pour a 40 on the curb for the PCs we’ve lost…

Toshiba Tecra…
Short circuited during disassembly
Rest in pieces buddy
Contact

- Twitter: @coreykal, @xenokovah, @legbacore
- Email: {corey, xeno}@legbacore.com
- http://legbacore.com/Contact.html for our GPG keys

- As always, go check out OpenSecurityTraining.info for the free classes from Corey and Xeno on x86 assembly & architecture, binary executable formats, stealth malware, and exploits.
- Then go forth and do cool research for us to read about!
GIGABYTE Patented DualBIOS™ (UEFI) Design

GIGABYTE 9 series Ultra Durable™ motherboards feature GIGABYTE DualBIOS™, an exclusive technology from GIGABYTE that protects arguably one of your PC’s most crucial components, the BIOS. GIGABYTE DualBIOS™ means that your motherboard has both a ‘Main BIOS’ and a ‘Backup BIOS’, making users protected from BIOS failure due to virus attack, hardware malfunction, improper OC settings or power failure during the update process.
They are vulnerable to the language of null.
References

[0] Attacking Intel BIOS – Rafal Wojtczuk and Alexander Tereshkin, July 2009
http://invisiblethingslab.com/resources/bh09usa/Attacking%20Intel%20BIOS.pdf
http://www.kb.cert.org/vuls/id/127284 (CERT never posted?!)

http://www.kb.cert.org/vuls/id/912156
http://www.kb.cert.org/vuls/id/255726 (CERT hasn’t posted yet despite request)

[2] All Your Boot Are Belong To Us (MITRE portion) – Kallenberg et al. – Mar. 2014,
delayed from publicly disclosing potential for bricking until HITB at Intel’s request
http://www.kb.cert.org/vuls/id/758382


http://syscan.org/index.php/download/get/6e597f6067493dd581eed737146f3af/new/CoresyScan2014_CoreyKallenberg_SetupforFailureDefeatingSecureBoot.zip
http://www.kb.cert.org/vuls/id/291102 (CERT hasn’t posted yet despite request)
References

http://www.kb.cert.org/vuls/id/766164

https://bromiumlabs.files.wordpress.com/2015/01/attacksonuefi_slides.pdf
http://www.kb.cert.org/vuls/id/533140

https://frab.cccv.de/system/attachments/2565/original/speed_racer_whitepaper.pdf
http://www.kb.cert.org/vuls/id/912156

https://frab.cccv.de/system/attachments/2566/original/venamis_whitepaper.pdf
http://www.kb.cert.org/vuls/id/552286

https://frab.cccv.de/system/attachments/2566/original/venamis_whitepaper.pdf
http://www.kb.cert.org/vuls/id/577140 (CERT hasn’t posted yet despite request)
References

http://clearhatconsulting.com/DeeperDoor
www.alchemistowl.org/arrigo/Papers/Arrigo-Triulzi-CANSEC10-Project-Maux-III.pdf
https://www.youtube.com/watch?v=vILAlhwUgIU (contains leaked classified NSA documents)
[X] See all the related work we’re aware of here:

http://timeglider.com/timeline/5ca2daa6078caaf4
Backup

• “Should you worry when the skullhead is in front of you? Or is it worse because it’s always waiting, where your eyes don’t go?”
  
  — They Might Be Giants