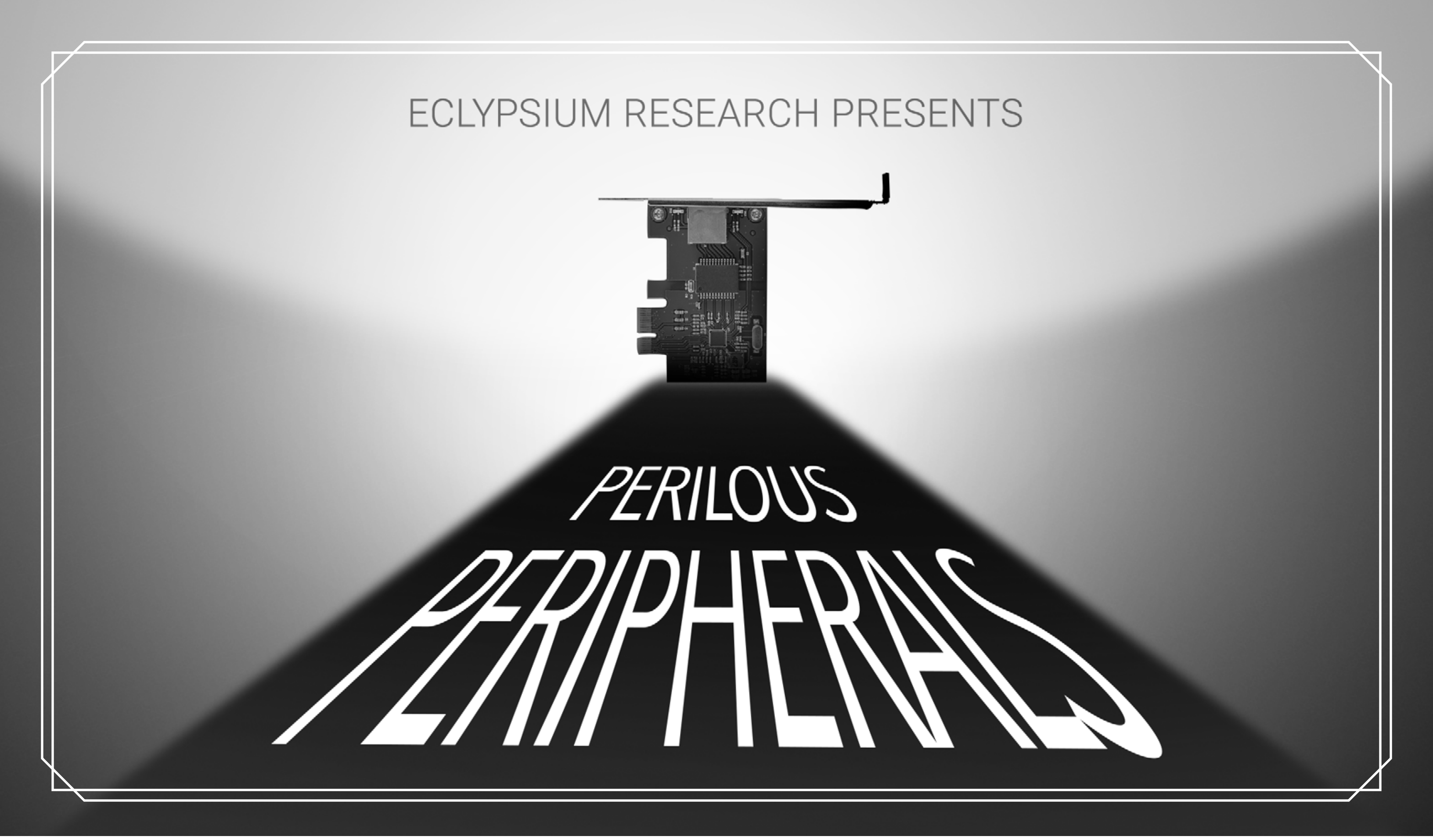


PERILOUS PERIPHERALS: THE HIDDEN DANGERS INSIDE WINDOWS & LINUX COMPUTERS

Unsigned firmware in WiFi adapters, USB hubs, trackpads, laptop cameras and network interface cards provides multiple pathways for malicious attackers to compromise laptops and servers

In new research, Eclypsium found unsigned firmware in WiFi adapters, USB hubs, trackpads, and cameras used in computers from Lenovo, Dell, HP and other major manufacturers. We then demonstrated a successful attack on a server via a network interface card with unsigned firmware used by each of the big three server manufacturers. Once firmware on any of these components is infected using the issues we describe, the malware stays undetected by any software security controls. Despite previous in-the-wild attacks, peripheral manufacturers have been slow to adopt the practice of signing firmware, leaving millions of Windows and Linux systems at risk of firmware attacks that can exfiltrate data, disrupt operations and deliver ransomware.

THE PROBLEM OF UNSIGNED FIRMWARE

It has been five years since the Equation Group's HDD implants were found in the wild, and introduced the industry to the power of firmware hacking and the underlying dangers posed by unsigned firmware in peripheral devices. And while there have been pockets of progress in recent years, our research shows that much of the industry continues to turn a blind eye to the risks of unsigned firmware. In four separate new pieces of research we found unsigned firmware in WiFi adapters, USB hubs, trackpads, and cameras in a variety of enterprise devices.

Unfortunately, these issues can be devastating to the security and operation of devices, and more often than not, are very difficult to fix. Disruption to components such as network cards, drives, and other peripherals can completely disable the device or provide attackers with ways to steal data, deliver ransomware and hide from security. This new research from Eclypsium shows that these weaknesses are widespread across components in laptops and servers, offering multiple pathways for malicious attacks.

FIRMWARE: THE BRAIN WITHIN VIRTUALLY EVERY COMPONENT

When it comes to security, most of the attention goes to the most visible components of a system such as the operating system and the applications. But in response to the growing number of threats, many organizations have begun to add firmware to their vulnerability management and threat prevention models.

In many cases these efforts are limited to the system firmware—the UEFI or BIOS resident on the main board of a device. However, virtually every component within a device has its own firmware and its own potential for risk, including network adapters, graphics cards, USB devices, cameras, touchpads and trackpads, and more. You can refer to our online 'Know Your Own Device' resource for an overview of some of the most common firmware-enabled components within devices today.

In virtually every case these components are governed by firmware. The firmware may be burned into the integrated circuit of the device itself, or the component may have its own flash memory where firmware is stored. In other cases firmware may be dynamically provided by the operating system at boot time. Regardless of how the firmware is stored, this firmware can act like a miniature computer that governs the low-level behavior of that particular component. And as we will see, this code is often very susceptible to attack in everything from laptops to servers to network devices.

THE RISK OF UNSIGNED FIRMWARE

The problem is that peripheral devices often lack the same security best practices that we take for granted in operating systems and in other more visible components, like the UEFI or BIOS. Specifically, many peripheral devices do not verify that firmware is properly signed with a high quality public/private key before running the code. This means that these components have no way to validate that the firmware loaded by the device is authentic and should be trusted. An attacker could simply insert a malicious or vulnerable firmware image, which the component would blindly trust and run.

Additionally, as our previous Screwed Drivers research and recent live-off-the-land attacks have demonstrated, vulnerable drivers can be used to bypass protections and enable ransomware to attack without interference. Many components can be updated without the need for special privileges, leading to a very simple and powerful scenario for an attack:

1. An attacker gains access to a device via any method, such as malware delivered via email or a malicious website, or an evil maid attack. With basic user privileges, the attacker/malware could write malicious firmware to a vulnerable component.
2. If the component doesn't require the firmware to be properly signed, the attacker's code is loaded and run by the component.
3. The attacker can then use the unique functionality and privileges of that component to further an attack.

The ways an attacker could abuse firmware would naturally vary from component to component. For example, malicious firmware on a network adapter could allow an attacker to sniff, copy, redirect, or alter traffic leading to a loss of data, man-in-the-middle and other attacks. PCI-based devices could enable Direct Memory Access (DMA) attacks that could easily steal data or take full control over the victim system. Cameras could be used to capture data from the user's environment, while a compromised hard drive could allow the attacker to hide code and tools without being seen by the operating system. However, the overall issue remains the same. If a component doesn't require signed firmware, an attacker can easily gain control over the component, typically without the need for special privileges.

NEW RESEARCH: EXAMPLES OF INSECURE FIRMWARE IN PERIPHERALS

1. Touchpad and TrackPoint Firmware in Lenovo Laptops

Eclypsium researchers recently analyzed a Lenovo ThinkPad X1 Carbon 6th Gen laptop. While we used this particular device for our analysis, we believe many other models and even other vendors would have the same issues. Specifically, Lenovo uses Synaptics as the ODM for their Touchpads, and other manufacturers that use these components could be similarly vulnerable.

Within the device, we specifically focused on the following firmware:

- Touchpad Firmware: pr2812761-tm3288-011-0808.img
- TrackPoint Firmware: PSG5E5_RANKA_fv06.bin

We discovered that the Touchpad and TrackPoint use insecure firmware update mechanisms. Specifically, cryptographic signature verification was not required at the device level before firmware updates were applied. This lack of control made it possible to modify the firmware images through software to run arbitrary malicious code within these components.

Lenovo has indicated that the ODM does not have a mechanism to fix this in the current generation of the product.

2. HP Wide Vision FHD Camera Firmware in HP Laptops

Eclypsium researched the firmware updates distributed by HP for the HP Wide Vision FHD camera found in the HP Spectre x360 Convertible 13-ap0xxx laptop. We found that the firmware update was unencrypted and lacked authenticity checks. We also found that the firmware could be modified to alter USB descriptors using the HP-provided update tool.

HP Wide Vision FHD cameras are USB camera modules produced by SunplusIT. HP support provided a firmware update (sp93170.exe) for the camera as used on the HP Spectre x360 Convertible 13-ap0xxx laptop. This firmware updater included SunplusIT's Windows-based firmware update tool along with the firmware image.

The firmware image does not include any form of cryptographic signature or other authenticity information. The Windows-based firmware update tool accepts firmware files that have been modified to adjust USB descriptor contents. This ability to modify USB descriptors can be leveraged to disable the device or cause it to be identified as a different type of USB device. Once additional details of the processor architecture are discovered, the camera module behavior can be altered to be malicious by implementing a USB HID device such as a Rubber Ducky.

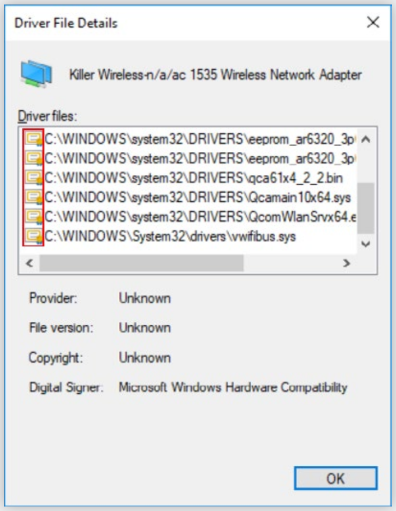
We confirmed this vulnerability through modifying USB descriptors on a device that was updated with the tool. Of particular note, the SunplusIT firmware updater **can successfully update a device even as a normal user**. Firmware updates should require Administrator access.

HP has indicated that they are working on a firmware update and that upcoming camera generations will have signed firmware in future models.

HP has published a security bulletin with the mitigation located in the HP Security Bulletin Archive.

3. WiFi Adapter on Dell XPS Laptop

Eclypsium researchers also demonstrated the ability to modify the firmware of the WiFi adapter on a Dell XPS 15 9560 laptop running Windows 10. The adapter in question was a Killer Wireless-n/a/ac 1535. Windows 10 will check to confirm that the drivers are correctly signed, and a small certificate icon is displayed next to the driver when viewed in the device manager, as shown below.

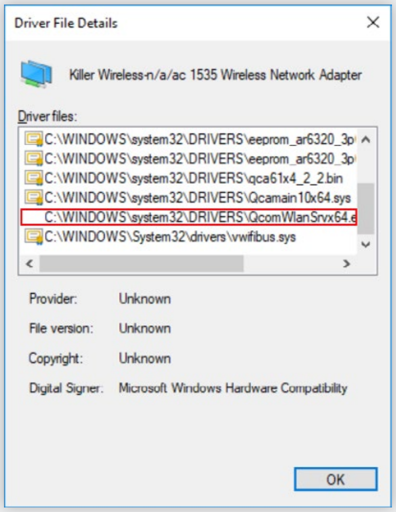

Driver File Details

Killer Wireless-n/a/ac 1535 Wireless Network Adapter

Driver files:
C:\WINDOWS\system32\DRIVERS\eeprom_ar6320_3p
C:\WINDOWS\system32\DRIVERS\eeprom_ar6320_3p
C:\WINDOWS\system32\DRIVERS\qca61x4_2_2.bin
C:\WINDOWS\system32\DRIVERS\Qcamain10x64.sys
C:\WINDOWS\system32\DRIVERS\QcomWlanSrvx64.e
C:\WINDOWS\System32\drivers\vwifibus.sys

Provider:	Unknown
File version:	Unknown
Copyright:	Unknown
Digital Signer:	Microsoft Windows Hardware Compatibility

When we modify the firmware image for the WiFi adapter, we can see that the icon goes away.


Driver File Details

Killer Wireless-n/a/ac 1535 Wireless Network Adapter

Driver files:
C:\WINDOWS\system32\DRIVERS\eeprom_ar6320_3p
C:\WINDOWS\system32\DRIVERS\eeprom_ar6320_3p
C:\WINDOWS\system32\DRIVERS\qca61x4_2_2.bin
C:\WINDOWS\system32\DRIVERS\Qcamain10x64.sys
C:\WINDOWS\system32\DRIVERS\QcomWlanSrvx64.e
C:\WINDOWS\System32\drivers\vwifibus.sys

Provider:	Unknown
File version:	Unknown
Copyright:	Unknown
Digital Signer:	Microsoft Windows Hardware Compatibility

As we can see, the certificate icon is no longer displayed. And although the icon is missing, we were able to modify the firmware in a potentially malicious way, and the driver still successfully loads it into the device.

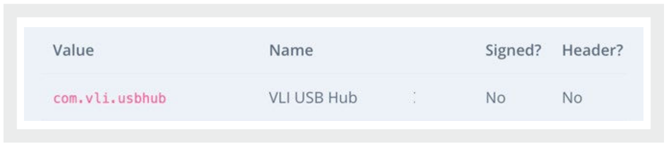
We reported this issue to both Qualcomm, who provides the chipset and driver for the Killer Wireless card and to Microsoft, who checks that such drivers are signed. Qualcomm responded that their chipset is subordinate to the processor, and that the software running on the CPU is expected to take responsibility for validating firmware. They stated that there was no plan to add signature verification for these chips. However, Microsoft responded that it was up to the device vendor to verify firmware that is loaded into the device.

This creates an interesting question of whose responsibility it should be to enforce the use of a valid signed driver and firmware. Given that a privileged attacker could easily replace driver files and bypass any hypothetical checks, the driver seems like a poor candidate. This leaves us with the device and the operating system. Although between these two options the responsibility remains unclear and as we have seen often goes unaddressed altogether.

4. USB Hub Firmware

As another example of the widespread nature of unsigned firmware, we looked to the Linux Vendor Firmware Service, a secure portal that allows hardware vendors to upload firmware updates. From this resource we can focus specifically on update protocols and easily review which are signed and which are not. While we can see that some of the update protocols are related to transport, many others are protocols used for the actual update process.

For example, VLI USB Hub firmware is unsigned.



Value	Name	Signed?	Header?
com.vli.usbhub	VLI USB Hub	No	No

FIRMWARE ATTACK DEMONSTRATION

Next, let's take a look at how unsigned firmware could be abused as part of a real-world attack. As we will see, an attacker who gains control over a peripheral component can not only use the component's functionality for malicious purposes, they can also potentially gain new privileges and even control over the entire system.

In this demonstration, we are attacking unsigned firmware in a network interface card (NIC) chipset. A malicious attack on a NIC can have a profound impact on the server, compromising the operating system remotely, providing a remote backdoor, snooping and exfiltrating raw network traffic and bypassing operating system firewalls to extract data or deliver ransomware. Such an attack could disconnect a server from a network upon a signal, disrupting connectivity for an entire data center.

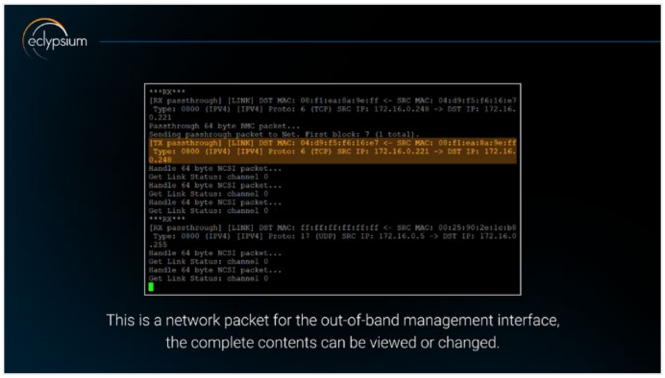
Multiple researchers have highlighted the dangers of unsigned firmware in NICs such as Arrigo Triulzi's Project Maux presentations and "Can you still trust your network card?" by Loïc Duflot and others.

The Broadcom BCM5719 chipset in the NIC we attacked is commonly used in current-generation servers from multiple manufacturers, and is known to not perform signature verification on firmware uploaded from the host. There are multiple public projects working on reverse engineering and reimplementing the firmware for this NIC, such as https://github.com/meklort/bcm5719-fw and https://github.com/hlandau/ortega, which lowers the bar for attackers.

Servers have a baseboard management controller (BMC) for out-of-band management of the device and most will have the ability to configure the BMC to share the NIC with the host system. In this arrangement, you have a single physical network connection that, to outside devices, looks like two different network adapters each with their own MAC address. By design, the host software (even at the kernel level) does not see any of the BMC traffic. However, if we load our own firmware into the NIC in a system where the BMC is configured to share the NIC with the host, we can then gain access to traffic that we couldn't touch from the host alone.

Using this approach, we can inspect the contents of BMC network packets, provide those contents to malware running on the host, or even modify BMC traffic on the fly. This could also be used to block alerts sent from the BMC to a central logging server, selectively redirect them to a different server, copy and send traffic to a remote location for analysis, as well as make outgoing network connections to a remote command and control server directly from the NIC itself without the host or BMC being aware that any of this is happening.

Worse still, because the card is connected to the PCI bus, an attacker could use a DMA attack to take complete control of the server. As documented in our previous research, DMA attacks enable a potential attacker to read and write memory off a victim system directly, bypassing the main CPU and OS. By overwriting memory, attackers can gain control over kernel execution to perform virtually any manner of malicious activity. Thus an attacker could use the weakness in the network card to gain full control over the kernel of the entire server.



```
***RX***
[RX passthrough] [LINK] DST MAC: 08:f1:ea:8a:9e:ff <- SRC MAC: 04:d9:f5:f6:16:e7
 Type: 0800 (IPV4) [IPV4] Proto: 6 (TCP) SRC IP: 172.16.0.248 -> DST IP: 172.16.
0.221
Passthrough 64 byte BMC packet...
Sending passhrough packet to Net. First block: 7 (1 total).
[TX passthrough] [LINK] DST MAC: 04:d9:f5:f6:16:e7 <- SRC MAC: 08:f1:ea:8a:9e:ff
 Type: 0800 (IPV4) [IPV4] Proto: 6 (TCP) SRC IP: 172.16.0.221 -> DST IP: 172.16.
0.248
Handle 64 byte NCSI packet...
Get Link Status: channel 0
Handle 64 byte NCSI packet...
Get Link Status: channel 0
Handle 64 byte NCSI packet...
Get Link Status: channel 0
***RX***
[RX passthrough] [LINK] DST MAC: ff:ff:ff:ff:ff:ff <- SRC MAC: 00:25:90:2e:1c:b8
 Type: 0800 (IPV4) [IPV4] Proto: 17 (UDP) SRC IP: 172.16.0.5 -> DST IP: 172.16.0
.255
Handle 64 byte NCSI packet...
Get Link Status: channel 0
Handle 64 byte NCSI packet...
Get Link Status: channel 0
```

This is a network packet for the out-of-band management interface, the complete contents can be viewed or changed.

AN INDUSTRY-WIDE FIRMWARE SECURITY PROBLEM

The issue of unsigned firmware in peripheral devices is a widespread problem affecting a broad spectrum of brands and their ODM suppliers. After the disclosure of the Equation Group's drive implants, many HDD and SSD vendors made changes to ensure their components would only accept valid firmware. However, many of the other peripheral components have yet to follow suit. And given that component suppliers often work with a variety of device vendors, a single vulnerable component can easily end up in multiple products.

Just as importantly, these issues apply to virtually all classes of Windows and Linux devices, from laptops to servers. Apple performs signature verification on all files in a driver package, including firmware, each time before they are loaded into the device, to mitigate this type of attack. In contrast, Windows and Linux only perform this type of verification when the package is initially installed. Ultimately, the device itself needs to perform the signature verification before allowing the firmware update rather than depending on the operating system to perform this task.

Unfortunately, the problems posed by unsigned firmware are not easy to fix. If the component wasn't designed to check for signed firmware, it often can't be fixed with a firmware update. In many cases, the underlying problem in a device or product line can't be fixed at all, meaning that all of the devices in that product line will continue to be vulnerable throughout their lifetime.

CONCLUSION

Unsigned firmware in peripheral devices remains a highly overlooked aspect of cybersecurity. Depending on the capabilities of the component, unsigned firmware can lead to the loss of data, integrity, and privacy, and can allow attackers to gain privileges and hide from traditional security controls. Given the widespread nature of unsigned firmware, enterprises should scan their devices for any vulnerable components, and should assess the firmware posture of new devices during procurement. If you would like to learn more about this or any issues related to firmware security, please contact us research@eclypsium.com.