

Article

Not peer-reviewed version

Mitigating Man-in-the-Middle Attacks in UEFI Capsule Updates

[Godwin Olaye](#) *

Posted Date: 17 February 2025

doi: 10.20944/preprints202502.1190.v1

Keywords: modern computing systems



Preprints.org is a free multidisciplinary platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This open access article is published under a Creative Commons CC BY 4.0 license, which permit the free download, distribution, and reuse, provided that the author and preprint are cited in any reuse.

Article

Mitigating Man-in-the-Middle Attacks in UEFI Capsule Updates

Godwin Olaoye

Independent Researcher, Nigeria; goolaoye18@student.lautech.edu.ng

Abstract: Unified Extensible Firmware Interface (UEFI) is a critical component in the boot process of modern computing systems, responsible for initializing hardware and loading the operating system. UEFI firmware updates, typically delivered in the form of "capsules," are essential for ensuring that a system operates with the latest security patches, bug fixes, and feature enhancements. However, as these updates are often transmitted over untrusted communication channels, they are vulnerable to Man-in-the-Middle (MitM) attacks. In such attacks, a malicious actor can intercept, modify, or inject harmful code into the firmware update, potentially compromising system security and integrity. This paper investigates the potential security risks associated with MitM attacks on UEFI capsule updates and proposes a set of robust countermeasures. We begin by analyzing the typical attack vectors, focusing on the vulnerabilities inherent in the process of transmitting update data between the system and update servers. Drawing from existing cryptographic protocols, we propose the use of strong encryption, digital signatures, and public-key infrastructures (PKIs) to ensure the authenticity, confidentiality, and integrity of UEFI capsule updates. These measures help verify that updates have not been tampered with during transit, preventing unauthorized code injection by attackers. Furthermore, we explore the role of Secure Boot and Trusted Platform Module (TPM) technologies in providing an additional layer of defense. By using secure boot processes, systems can verify that only authorized firmware updates are executed, thus preventing malicious code from being activated. Similarly, TPM can be employed to securely store cryptographic keys and validation certificates, reducing the risk of key compromise during the update process. We also delve into the concept of certificate pinning, where specific trusted certificate authorities are hardcoded into the UEFI firmware, making it more difficult for attackers to present fraudulent certificates during the update process. This approach significantly reduces the likelihood of MitM attacks exploiting misconfigured or compromised certificate chains. In conclusion, we propose a comprehensive security framework to mitigate the risks of MitM attacks during UEFI capsule updates. This framework combines cryptographic protections with hardware-based security mechanisms to ensure the integrity, authenticity, and confidentiality of firmware updates, fostering a more secure and resilient computing environment.

Keywords: modern computing systems

Introduction

Overview of UEFI (Unified Extensible Firmware Interface) Capsule Update

The Unified Extensible Firmware Interface (UEFI) is a modern firmware standard that defines the interface between an operating system and the platform firmware. UEFI has largely replaced the traditional Basic Input/Output System (BIOS) on most modern computing devices, offering enhanced functionality, security, and flexibility. UEFI provides the foundation for system initialization, hardware configuration, boot management, and OS loading. One of its most important capabilities is its support for firmware updates, which are essential for ensuring the system's continued stability, performance, and security.

A UEFI capsule update refers to a special type of update package used to deliver firmware updates to the system. These updates contain code and data used by the firmware to enhance functionality, fix bugs, improve performance, and address security vulnerabilities. Capsule updates can be delivered via a variety of methods, such as through the operating system, from a network source, or even during the boot process. When the system receives a UEFI capsule update, it verifies and installs the update, replacing or augmenting the existing firmware with the new version.

Brief Explanation of UEFI and its Role in Modern Systems

UEFI serves as a bridge between the hardware and the operating system, offering significant advantages over its predecessor, BIOS. Some key features of UEFI include:

- **Boot Time Optimization:** UEFI allows faster boot times by supporting large hard drives and advanced boot configurations, including the ability to boot from modern storage devices like SSDs.
- **Secure Boot:** This feature ensures that only trusted firmware and operating systems are allowed to boot, protecting against rootkits and other forms of malware.
- **Graphical User Interface:** UEFI supports a more user-friendly interface compared to the text-based BIOS, providing enhanced configuration options and diagnostics for system administrators and users.
- **Extensibility:** UEFI is modular, meaning that additional drivers, applications, and services can be added easily without needing a complete overhaul of the firmware.

In addition to these features, UEFI plays a critical role in maintaining system security and stability by enabling firmware updates. Since UEFI operates at a low level, it is responsible for critical system functions before the operating system loads. As such, any vulnerabilities within UEFI or the firmware could potentially compromise the entire system. This is where UEFI capsule updates become essential—they ensure that the firmware is regularly updated with patches, security fixes, and improvements to maintain optimal performance and prevent potential exploits.

Importance of Capsule Updates for Firmware Management and Security

UEFI capsule updates are a key element of effective firmware management. As with any software component, firmware can contain bugs or vulnerabilities that may be discovered after the system is deployed. Without a mechanism for updating firmware, systems would be left exposed to these security risks. Capsule updates allow manufacturers to address vulnerabilities, fix performance issues, and ensure compatibility with evolving hardware standards.

From a security perspective, UEFI capsule updates are essential for safeguarding the system's integrity. UEFI provides a trusted execution environment, but this environment can be compromised if malicious code is allowed to run at the firmware level. Regular capsule updates ensure that known vulnerabilities are patched and that new security measures are incorporated to protect against emerging threats. Capsule updates also help prevent the installation of malicious firmware by ensuring that only authenticated updates from trusted sources are accepted.

In a world where cyber threats are continuously evolving, timely and secure firmware updates are a cornerstone of maintaining system resilience. By facilitating secure, reliable updates, UEFI capsule updates help protect the firmware against exploitation and ensure that the system continues to function as intended. Without effective capsule update mechanisms, systems could become more susceptible to attacks that compromise the firmware, leading to potentially devastating security breaches.

Man-in-the-Middle (MITM) Attacks

Definition and Implications of MITM Attacks

A **Man-in-the-Middle (MITM) attack** is a type of cyber attack where a malicious actor secretly intercepts and potentially alters the communication between two parties without their knowledge. In such an attack, the attacker places themselves between the sender and the receiver, gaining control over the data being exchanged. This position allows the attacker to eavesdrop on the communication, modify the data, inject malicious content, or impersonate one of the legitimate parties to deceive the other.

MITM attacks exploit vulnerabilities in communication channels, particularly when data is transmitted over insecure or unencrypted networks. This type of attack can have serious implications for data confidentiality, integrity, and authenticity. MITM attackers can access sensitive information, alter legitimate communications, or inject harmful software, which could lead to unauthorized access to systems, data breaches, or even the complete compromise of the affected devices.

For example, in an HTTPS session, a MITM attacker might intercept the traffic between a client (e.g., a computer) and a server. They can then alter the data sent by the client or the server, potentially stealing sensitive data such as passwords, injecting malicious code, or manipulating the data flow. MITM attacks can be devastating if the attacker gains control over critical communications, particularly in high-stakes environments like financial transactions, software updates, or sensitive business communications.

How MITM Can Compromise Capsule Update Integrity

When applied to the context of **UEFI capsule updates**, a MITM attack can have dire consequences for both the integrity and authenticity of the firmware update process. The capsule update process involves the transmission of sensitive firmware data from a trusted source (such as a server or system manufacturer) to the target device (e.g., a computer). If this communication is intercepted or altered by an attacker, the results can be catastrophic, undermining both the system's security and its functionality.

Here's how a MITM attack can compromise capsule update integrity:

1. **Interception and Data Manipulation:** During the update process, if an attacker is positioned in the communication channel between the update server and the target device, they can intercept and alter the capsule data. For example, the attacker might modify the firmware being transmitted, replacing legitimate update files with malicious ones. This could lead to the installation of compromised or malicious firmware that includes backdoors, rootkits, or other malicious payloads designed to compromise the device.
2. **Injection of Malicious Code:** An attacker might inject harmful code into the capsule update during transmission. Even if the capsule is encrypted, a sufficiently advanced attacker might exploit weak points in the encryption or attempt to break the encryption to insert malicious payloads. Once the update is received and executed, this code could be activated, compromising the system's integrity.
3. **Impersonating the Update Server:** If the attacker can spoof or impersonate the trusted update server, they could trick the target system into accepting a fraudulent or malicious capsule update. This could occur if the system is not using proper authentication mechanisms, such as certificate validation, to verify the identity of the update source. In such a scenario, the attacker might present themselves as a legitimate update provider, fooling the system into accepting a dangerous firmware update.
4. **Loss of Data Integrity:** Capsule updates often require the verification of data integrity to ensure that the update has not been tampered with during transmission. If an attacker alters or corrupts the update data, the system may fail to validate the capsule correctly. This could lead to a corrupted firmware installation, potentially rendering the system unstable or inoperable. Worse, if the system does not detect the tampering, the malicious or corrupted update could proceed without any indication that something is wrong.
5. **Bypassing Security Measures:** Many modern firmware updates come with security mechanisms such as cryptographic signatures, hashing, and secure boot features to prevent

tampering. However, a sophisticated MITM attacker may target vulnerabilities in the validation process itself. For instance, the attacker may attempt to break the cryptographic signing mechanism or supply a counterfeit certificate that the system mistakenly accepts as legitimate. By bypassing these security checks, the attacker can replace the intended update with a malicious one.

In essence, a successful MITM attack on a UEFI capsule update can result in the installation of rogue firmware, the theft of sensitive system data, or even the total compromise of the system's security. To mitigate such risks, it is crucial to implement strong encryption, authentication, and validation mechanisms throughout the capsule update process. This ensures that updates are only accepted if they come from trusted sources and remain intact during transmission. Without these protections, the UEFI capsule update process remains a vulnerable target for attackers.

Understanding UEFI Capsule Updates

What is a UEFI Capsule Update?

A **UEFI Capsule Update** is a package or format used to update the firmware (the UEFI firmware) on a system. These updates are essential for maintaining the system's functionality, security, and performance. Capsule updates contain code, configuration data, and potentially other resources that are required to modify or replace the firmware stored in the system's UEFI. This type of update ensures that the system's firmware can be safely and efficiently updated with new features, bug fixes, or security patches.

UEFI capsule updates are typically packaged and signed to ensure their integrity and authenticity before being applied to the system. The update process is designed to be secure, with mechanisms in place to verify that the update is coming from a trusted source and that the contents of the update have not been altered during transmission.

Firmware Update Format Used in UEFI

UEFI capsule updates use a specific format that is recognized by the UEFI firmware. The format allows the firmware to recognize and validate the update before installing it. The capsule consists of a series of structured components, including:

- **Header Information:** Contains metadata about the update, such as the version of the capsule, the type of update (e.g., firmware, driver), and the update's security properties (e.g., signature).
- **Data Payload:** The actual data to be installed as part of the update. This includes the new firmware or firmware patches and potentially configuration settings.
- **Security Information:** This includes cryptographic signatures, hash values, or certificates that validate the update's integrity and authenticity.

The capsule format is designed to be easily recognized by the UEFI firmware, ensuring that the update process can be carried out with minimal user intervention while ensuring security and integrity.

Communication Channels for UEFI Capsule Updates

UEFI capsule updates are typically communicated over various channels to deliver firmware updates from a trusted source (such as a server or a local storage device) to the system that requires the update. The communication channels used can vary, but some common methods include:

1. **Operating System-Based Updates:** The update process may be initiated from within the operating system. The OS may contact an update server (such as the manufacturer's website or a corporate update server) to download the capsule update. In this case, the system firmware is updated during runtime after the OS loads.

2. **Network-Based Updates:** Capsule updates may be delivered over the network using protocols such as HTTP, HTTPS, or other secure file transfer methods. The system may initiate an update request to a server, receive the update, and then apply it.
3. **BIOS/UEFI-Based Updates:** Some systems offer the ability to update the firmware directly from the UEFI shell or via a bootable USB drive. This method allows for updates to be applied outside of the operating system and is often used for emergency updates or updates in a non-operating system environment.
4. **Pre-Boot Environment:** Updates can also be delivered through a pre-boot environment, such as during system startup when the UEFI environment is active but the operating system has not loaded. This allows the firmware to be updated before the OS is even involved.

Process of Capsule Update

The process of applying a UEFI capsule update generally follows several key steps: 1.

Initiation:

- The update process is initiated either manually by the user or automatically through a system update service. In the case of automatic updates, the update is typically triggered by the system's firmware management or update service. ○ In many cases, the system checks for available updates from a trusted server or source. For example, a computer might reach out to a manufacturer's server to check for firmware updates.

3. Download:

- Once an update is detected, the capsule update file is downloaded from the trusted source. This process can be done over a network connection, via USB, or from another local storage device.
- The update is then stored temporarily in a secure location on the system.

4. Verification:

- Before installing the update, the system will verify the authenticity and integrity of the capsule update. This involves checking cryptographic signatures, hash values, or certificates associated with the update to ensure that it has not been tampered with.
- Secure boot and other integrity verification techniques can be used during this process to confirm that the update package is legitimate and comes from a trusted source.

5. Installation:

- After the update is verified, the system proceeds with the installation of the update. The system may reboot to apply the update, depending on the type of update being installed. In the case of firmware updates, the system is likely to reboot into a UEFI environment to complete the installation.
- During the installation, the system firmware is updated to the new version, which may involve replacing code or patches within the firmware.

6. Post-Installation:

- After installation, the system may perform self-checks to ensure the update has been successfully applied. In some cases, the firmware may be verified once more to confirm that the correct version has been installed.

Security Considerations During Capsule Update

The process of applying UEFI capsule updates must be secure to prevent tampering and ensure that the system is not compromised during the update. Some key security considerations include:

1. **Encryption and Integrity Protection:** Capsule updates should be encrypted to protect the data from being intercepted or altered during transmission. Secure protocols (e.g., HTTPS) should be used for network-based updates to prevent eavesdropping. Additionally, cryptographic hash functions or digital signatures should be employed to verify the integrity of the update package.
2. **Authentication:** The system must authenticate the source of the capsule update before accepting and applying the update. This is often achieved through the use of digital certificates and public-key infrastructure (PKI), ensuring that only updates from trusted sources are accepted.
3. **Secure Boot:** Secure Boot is a UEFI feature that ensures only trusted firmware and operating systems are allowed to run during the boot process. This feature helps prevent unauthorized or malicious updates from being installed by verifying that the firmware and update signatures are legitimate before they are applied.
4. **Rollback Mechanisms:** In case the update process fails or results in an unstable firmware version, rollback mechanisms are crucial. These allow the system to revert to a previous, stable firmware version to ensure system recovery and minimize downtime or damage.
5. **Digital Signatures and Validation:** Before installing an update, the system should validate the update's signature and ensure that the firmware update has not been tampered with. Digital signatures are used to ensure the authenticity of the update and that it has not been modified in transit.
6. **Protection Against Man-in-the-Middle (MITM) Attacks:** Communication channels used to download and apply updates must be secured to prevent MITM attacks, where an attacker might intercept and alter the update data. Encryption, certificate validation, and the use of secure protocols (e.g., TLS/SSL) are essential to mitigate these risks.

By addressing these security considerations, UEFI capsule updates can be securely applied to ensure the system remains up-to-date, stable, and protected from potential exploits. Proper security measures during the update process help ensure that the update does not compromise the integrity or safety of the device.

Man-in-the-Middle (MITM) Attacks: Threat Landscape

How MITM Attacks Work

A **Man-in-the-Middle (MITM)** attack occurs when an attacker positions themselves between two parties (such as a user and a server) to intercept, observe, and potentially alter the communication between them without either party knowing. This attack works by intercepting data transmissions, either by exploiting vulnerabilities in the network or by directly gaining access to communication channels.

In the context of **UEFI capsule updates**, MITM attacks can exploit insecure or improperly authenticated update mechanisms. Here's how a typical MITM attack unfolds:

1. **Interception:** The attacker places themselves between the target system (e.g., a computer or device receiving a UEFI update) and the update server. This could be accomplished by compromising a public Wi-Fi network, DNS poisoning, or other methods to redirect traffic to a malicious proxy server that the attacker controls. The attacker gains visibility into the communication between the two parties, enabling them to monitor or intercept the update data.
2. **Modification:** Once the update data has been intercepted, the attacker can modify it before it reaches the target system. This can involve changing the contents of the firmware update (e.g., replacing it with malicious code), injecting harmful payloads, or even tampering with the integrity-checking components (e.g., digital signatures or hash values) that ensure the update is legitimate.
3. **Injection of Malicious Code:** In some cases, the attacker may inject malicious code directly into the firmware update. This could involve adding a backdoor, rootkit, or other types of malware designed to compromise the target system once the update is installed. By altering the update's contents or signature, the attacker can ensure that their malicious code is installed along with the legitimate firmware update, often without detection.

The attack occurs without the knowledge of either party involved in the communication (i.e., the system receiving the update or the update server). As a result, the attacker can gain control over the target device or cause irreparable harm to the system's integrity.

Potential for Malicious Code Injection or Firmware Manipulation

MITM attacks targeting UEFI capsule updates have a high potential for injecting malicious code or manipulating the firmware in ways that go unnoticed. The attacker may modify the firmware update package in the following ways:

1. **Malware Injection:** Malicious payloads can be inserted into the firmware update. These payloads could be viruses, trojans, rootkits, or ransomware, which are installed when the system processes the update. Since firmware operates at such a low level of the system, malware injected into the firmware can be extremely difficult to detect and remove.
2. **Rootkits:** A particularly dangerous form of malware is a rootkit, which can be embedded in the firmware update and provide the attacker with deep system-level access. Rootkits operate at the operating system or kernel level, allowing attackers to maintain persistent access to the system and avoid detection by traditional security tools.
3. **Firmware Manipulation:** The attacker might directly alter the firmware update itself to introduce instability or backdoors. For example, the attacker could modify the firmware code to include malicious instructions or exploits that take advantage of known vulnerabilities in the system's hardware or software components.

By modifying the update, an attacker could potentially manipulate the system's behavior in a way that compromises security, data integrity, or system functionality. The system would unknowingly apply the corrupted update, believing it to be legitimate.

Impact on UEFI Capsule Updates

The implications of MITM attacks on **UEFI capsule updates** can be severe. These updates are designed to improve firmware, fix bugs, and address security vulnerabilities. If the update process is compromised through a MITM attack, it can lead to several negative consequences:

1. **Corruption of Firmware During the Update:** One of the most significant risks is the corruption of the firmware itself. If the attacker successfully alters the capsule update, the system may

receive a broken or modified firmware package. This could render the system inoperable or cause it to malfunction, leading to system crashes, data loss, or even the failure of the update process. In some cases, corrupted firmware may prevent the system from booting entirely.

2. **Unauthorized Modifications and Malware Installation:** As discussed, MITM attackers can inject malicious code such as rootkits or backdoors into the firmware update. Once installed, these malicious modifications can provide attackers with elevated access to the system, allowing them to carry out further attacks. For instance, attackers could exfiltrate sensitive data, disrupt system operations, or even take full control of the device remotely. Such unauthorized modifications can undermine the trustworthiness of the entire update process, causing long-term security risks for users and organizations alike.
3. **Compromise of Security Features:** A successful MITM attack may alter or disable key security mechanisms built into UEFI firmware, such as **Secure Boot** or **TPM (Trusted Platform Module)** functionality. These features are designed to protect the system from unauthorized firmware, so an attacker who compromises them could potentially disable essential security checks, making it easier for future attacks to succeed.
4. **Persistence of Malware:** Since firmware resides at a low level in the system, malware installed via a compromised UEFI update can be extremely persistent. Unlike software-based attacks that can be removed by reinstalling the operating system, firmware-based malware remains even if the OS is reinstalled or the **hard** drive is replaced. This persistence makes MITM attacks on firmware updates especially dangerous.

Real-World Examples of MITM Attacks

While specific, high-profile MITM attacks targeting UEFI capsule updates are rare, there have been numerous incidents of MITM attacks in other areas of cybersecurity, especially where firmware and update processes are involved. Here are some general examples that illustrate the potential consequences of such attacks:

1. **The ShadowBrokers Leak (2017):** The ShadowBrokers hacker group released a series of leaked NSA hacking tools, including exploits targeting vulnerabilities in various firmware components. While not strictly a MITM attack, these exploits illustrated how attackers could manipulate or take advantage of firmware vulnerabilities to gain remote access. This type of attack could potentially be facilitated by a MITM attacker, allowing for the injection of malicious code into firmware updates.
2. **The ASUS Rog Router Firmware Update Incident (2019):** A notable case of a supply chain attack, where hackers managed to compromise legitimate firmware updates for ASUS routers. The attack didn't rely on MITM directly, but demonstrated the risk of injecting malicious code into legitimate firmware updates. A similar attack could occur in a MITM scenario, with malicious firmware updates being delivered under the guise of an official update.
3. **Stuxnet (2010):** Although primarily a targeted attack on industrial control systems, **Stuxnet** serves as a vivid example of how malicious code injected into firmware can cause severe damage. While it wasn't a MITM attack in the traditional sense, the methodology of tampering with firmware and the system's operation illustrates how devastating it could be if an attacker modified UEFI capsule updates.
4. **MITM Attacks in Public Wi-Fi Networks:** Many MITM attacks are carried out on unencrypted communication channels, particularly in public Wi-Fi environments. If a user or organization is

downloading a firmware update over a public network, an attacker can intercept and manipulate the update if the communication is not properly secured (e.g., if there's no HTTPS or SSL/TLS in place). While there are no widely known examples of MITM attacks specifically targeting UEFI capsule updates over Wi-Fi, this type of attack scenario is realistic, and organizations are encouraged to use encrypted communication for update transmission.

MITM attacks represent a serious threat to the integrity and security of UEFI capsule updates. By intercepting, altering, or injecting malicious content into the firmware update process, attackers can undermine system security, corrupt firmware, and introduce persistent malware. It's crucial for organizations and device manufacturers to implement strong encryption, authentication, and verification measures to prevent MITM attacks and ensure that UEFI capsule updates are secure, trustworthy, and properly executed.

Security Risks in UEFI Capsule Update Process

The **UEFI capsule update process** plays a critical role in maintaining and securing the firmware of modern systems. However, as with any system, vulnerabilities can be present in the update mechanism itself. If these vulnerabilities are not addressed, they can lead to significant security risks, potentially allowing attackers to compromise the firmware update process and inject malicious code into the system.

Vulnerabilities in UEFI Capsule Update Process

1. Lack of Encryption or Authentication During Communication

One of the most significant risks in the UEFI capsule update process is the potential lack of proper encryption or authentication mechanisms during communication. If the update process relies on insecure channels (e.g., HTTP instead of HTTPS) for transmitting firmware data, attackers can intercept, modify, or inject malicious code into the capsule update. Without encryption (like TLS/SSL), an attacker could:

Intercept Update Data: A man-in-the-middle (MITM) attacker could capture the firmware update as it is transmitted, potentially altering its contents. This includes injecting malicious payloads or changing update files to compromise the system during installation.

Perform Downgrade Attacks: If the update is not securely signed or the communication lacks proper authentication, attackers could trick the system into rolling back to an older, vulnerable version of firmware.

To mitigate this risk, the use of **HTTPS** (TLS-encrypted communication) for secure transmission and the inclusion of strong **digital signatures** for update packages are essential.

2. Weaknesses in Software or Firmware Validation

A critical part of the UEFI update process is validating the integrity and authenticity of the capsule update before it is installed. Without proper validation mechanisms, attackers can exploit weak points in the validation process to bypass security checks. Some weaknesses include:

Unencrypted or Weak Signatures: If the update's digital signature is not verified correctly, or if the signature uses weak cryptographic algorithms, it can be bypassed. This allows attackers to present forged updates as legitimate, leading to the installation of malware or backdoors.

Missing Hash Validations: Updates without proper hash checks could allow attackers to modify the firmware update without being detected. For instance, an attacker could alter the update payload, but without the proper cryptographic hash verification, the system may install the corrupted update.

To address these issues, robust validation techniques like **secure hashing algorithms (SHA-256 or better)**, **digital certificates**, and **public-key cryptography** should be used to ensure the authenticity of firmware updates and prevent unauthorized code execution.

3. Exploiting Insufficient or Absent Secure Boot Configurations

Secure Boot is a critical feature in UEFI that ensures only trusted and signed firmware is allowed to run during system boot-up. However, if Secure Boot is not properly enabled or is misconfigured, it can be exploited by attackers to install unauthorized firmware updates. o **Bypassing Secure Boot:** Attackers could exploit vulnerabilities in the Secure Boot configuration or in the UEFI environment to disable the secure boot process, allowing them to install unsigned or malicious updates.

- o **Insecure Default Configurations:** Many systems may have insecure default settings, such as insecure boot configurations or incomplete certificate chains, which leave the system vulnerable to attacks during the update process.

Ensuring **proper Secure Boot configuration** and regular checks on the certificate chains used to validate UEFI firmware updates can significantly mitigate these risks.

Exploiting Communication Channels

The communication channels through which UEFI capsule updates are delivered are often a prime target for attackers, as they provide an entry point for injecting malicious code or tampering with updates.

1. Exploiting HTTP or HTTPs Communication

- o **Unencrypted HTTP Traffic:** Using unencrypted communication channels like HTTP poses significant security risks, as data sent over such channels can easily be intercepted and altered by attackers in a MITM attack. Sensitive firmware data, including update packages, could be compromised, and attackers could inject malicious code or modify legitimate firmware updates.
- o **Weak or Misconfigured HTTPS:** Even if HTTPS is used, weaknesses such as improper configuration, expired certificates, or lack of proper certificate validation can leave the system vulnerable to attacks. Attackers can exploit these flaws to launch a **man-in-the-middle (MITM)** attack, impersonate the update server, and push malicious updates to the target system.

The risk is especially high when firmware updates are fetched from publicly accessible servers over HTTP or from unsecured local network connections, which can be vulnerable to sniffing or tampering.

2. Local Communication Channels (USB, Local Network, etc.)

- o **Compromised USB Devices:** If the firmware update is delivered via a USB drive or other local media, there is a risk that the device could be tampered with before it is used. Attackers can replace the legitimate update with a malicious update, which could lead to the installation of compromised firmware.
- o **Local Network Exploits:** Systems within a local network that rely on unsecured communication for firmware updates (e.g., using unsecured file-sharing protocols or local HTTP servers) can be susceptible to internal MITM attacks. An attacker with access to the local network can intercept update requests and deliver altered or malicious firmware.
- o **USB Spoofing:** A particularly dangerous attack involves inserting a compromised USB drive into the target system. This drive may contain malicious firmware updates disguised as

legitimate ones. If the system does not properly validate the update before applying it, the malicious firmware could be installed.

To minimize these risks, systems should ensure that all local communication channels are properly encrypted and verified. **USB drives** used for firmware updates should also be signed, and devices should be secured to prevent unauthorized access.

3. Firmware Rollback Attacks

Many systems allow for firmware rollbacks, where the system reverts to a previous firmware version in case the current update fails. While this is intended as a safety feature, it can be exploited by attackers.

Exploiting Rollback Vulnerabilities: If attackers gain access to the system's firmware rollback mechanism, they could exploit it to force the system to revert to an older, vulnerable firmware version that is easier to exploit. This could also allow them to install outdated firmware that bypasses newer security features. **Insecure Rollback Protection:** Some systems may not properly protect against unauthorized rollback requests or could allow for downgrades even when it is not intended. This opens a door for attackers to deliberately trigger a rollback, enabling them to reinstall a vulnerable firmware version or maliciously alter firmware.

Mitigation: One solution to this risk is to implement **firmware version locks** that prevent unauthorized rollbacks, only allowing updates to progress to newer versions. Additionally, updates should be designed to ensure that rollback features are securely protected and that only authorized personnel can perform firmware downgrades.

4. Unauthorized Update Installation

Attackers can target the update installation process itself by:

- **Spoofing Update Servers:** If a device does not properly authenticate the source of its firmware update (e.g., relying only on an IP address or insecure protocol), attackers could impersonate a legitimate update server. They could provide a malicious update, which would then be installed on the target system, leading to malware infection or system compromise.
- **Bypassing Update Integrity Checks:** In some cases, the system may not adequately check the integrity of the update before installation, allowing an attacker to provide a tampered update that passes the validation checks.
- **Unrestricted Update Permissions:** If the update process is not properly secured, attackers may be able to send unauthorized update requests to the system, allowing them to install malicious firmware without proper authentication or user consent.

To secure the update process, **strong authentication mechanisms** (e.g., certificates, hardware-based security) and **integrity checks** (e.g., hashing, signature verification) must be in place to ensure that only trusted updates are installed.

The **UEFI capsule update process** presents several security risks that can be exploited if proper safeguards are not in place. **Lack of encryption or authentication** during the communication of update data, **weaknesses in firmware validation**, and **vulnerabilities in communication channels** are all potential avenues for attackers to compromise the firmware update process. To mitigate these risks, it is critical to use secure communication channels (e.g., HTTPS), employ robust **digital signatures** for updates, ensure **proper validation** of firmware, and protect the system from unauthorized rollback or update modifications. By addressing these vulnerabilities, organizations can significantly reduce the likelihood of a successful attack on their firmware update mechanisms.

Mitigation Strategies

To protect **UEFI capsule updates** from various security risks, it is essential to adopt a comprehensive set of mitigation strategies. These strategies focus on securing the communication channels, verifying the authenticity and integrity of the updates, and ensuring the firmware remains safe even after installation. Below are some of the key strategies to mitigate potential threats:

Encryption of Communication Channels

Encryption of communication channels is critical to ensuring that firmware updates cannot be intercepted, altered, or injected with malicious content during transmission. • **Implementation of Secure Protocols (HTTPS/TLS):**

- The communication between the system and the update server should always occur over **HTTPS** (Hypertext Transfer Protocol Secure) using **TLS** (Transport Layer Security). This ensures that all data transmitted during the update process is encrypted and protected against interception by attackers, especially during download and installation.
- By using **HTTPS**, the integrity of the communication is protected, preventing attackers from performing **Man-in-the-Middle (MITM)** attacks where they could modify or inject malicious content into the update.
- **Ensuring Integrity and Confidentiality of Update Data in Transit:**
- **TLS/SSL** guarantees the **confidentiality** of the update data, making it unreadable to any attackers intercepting the communication.
- **Integrity** is maintained by the use of encryption, ensuring that any changes made to the data during transit will be detected, preventing attackers from tampering with the update without being noticed.
- Ensuring that the update process is **tamper-proof** also protects against the risk of malicious firmware modifications, ensuring the system installs only the legitimate update.

Digital Signatures and Code Validation

Digital signatures and **cryptographic hashing** are crucial for ensuring the **authenticity** and **integrity** of the firmware update, preventing unauthorized or malicious code from being installed.

- **Use of Digital Signatures to Verify the Authenticity of Capsule Update Files:**
 - Every capsule update file should be **digitally signed** by a trusted authority using a **private key**. When the update is downloaded by the system, the signature is verified using the corresponding **public key**.
 - If the signature does not match, it indicates that the update file has been tampered with or is not authentic, and the update should be rejected.
 - This process ensures that only updates from trusted sources can be installed, preventing attackers from injecting malicious updates that appear to be legitimate.
- **Role of Cryptographic Hashing in Validating the Integrity of the Firmware:**

Cryptographic hash functions (e.g., **SHA-256**) can be used to generate a **checksum** or hash value of the firmware update before and after it is downloaded. The system can compare this hash value with the one provided by the update server.

If the hashes do not match, it indicates that the integrity of the firmware has been compromised, and the system can abort the update installation.

 - Hashing provides an additional layer of security, allowing the system to verify that the firmware file has not been altered or corrupted during transmission.

Secure Boot and Trusted Execution Environments

Secure Boot and **Trusted Execution Environments (TEEs)** are hardware-based security features that help protect the integrity of the firmware and prevent unauthorized updates.

- **Enforcing Secure Boot to Block Unauthorized Updates:**
 - **Secure Boot** ensures that only **trusted** and **signed firmware** can be executed during system startup. The system verifies that the firmware update is signed by a trusted certificate before allowing it to be installed.
 - If an attacker attempts to load malicious firmware or an unsigned update, **Secure Boot** will block its execution, preventing the system from running any unauthorized code during the boot process.
 - This mechanism effectively prevents attackers from replacing or modifying the UEFI firmware with malicious versions during the update process.
- **Implementing Trusted Platform Module (TPM) to Secure the Update Process:**
 - The **Trusted Platform Module (TPM)** is a dedicated hardware chip designed to enhance system security by storing sensitive data like encryption keys and digital certificates.
 - TPM can be used to **securely store** keys used for validating firmware updates, ensuring that only authorized parties can sign and validate updates.
 - **TPM** can also assist in securely generating random numbers, which are crucial for generating **digital signatures** and ensuring that the update process is protected against tampering or spoofing.

Certificate Pinning and Strong Authentication

Certificate pinning and **strong authentication** mechanisms can protect against **MITM attacks** where attackers impersonate legitimate update servers.

- **Enabling Certificate Pinning to Prevent MITM Attacks:**
 - **Certificate pinning** involves hardcoding the **public key** or **certificate** of the update server into the client system. This prevents attackers from impersonating the update server by presenting fake certificates.
 - If an attacker tries to intercept or spoof the update server, they would not be able to produce the expected certificate, thus thwarting any MITM attack.
 - This provides an additional layer of security by ensuring that the system only trusts specific, pre-approved certificates, reducing the risk of attack from compromised or fraudulent update servers.
- **Multi-Factor Authentication Mechanisms for Verifying Update Sources:**
 - **Multi-factor authentication (MFA)** can be implemented to further secure the update process. For example, the system could require multiple forms of validation (e.g., certificate validation, one-time passwords, or biometric authentication) before allowing the update to proceed.
 - MFA prevents unauthorized access to the update mechanism by making it significantly harder for attackers to compromise the system and force a malicious update.

Regular Firmware Integrity Checks

To continuously monitor the integrity of firmware and detect any unauthorized modifications after updates have been applied, regular **firmware integrity checks** are necessary. ☺ **Automatic Verification of Firmware Signatures Upon Boot:**

- Each time the system boots, the firmware should be verified against its digital signature and cryptographic hash to ensure that it has not been tampered with since the last successful boot or update.
- This process could involve checking that the firmware's signature matches the expected

signature stored in a secure location (e.g., TPM, secure hardware storage). If the verification fails, the system can halt the boot process and alert the user or administrator.

- **Monitoring Systems for Unauthorized Modifications After Updates:**

- Continuous monitoring tools can be used to track changes to the system firmware after updates. If unauthorized modifications are detected, the system can alert the user or initiate rollback procedures.
- Additionally, implementing **self-healing** mechanisms that automatically restore the original firmware from a trusted source can ensure that any unauthorized modification is quickly mitigated.

Securing **UEFI capsule updates** from potential threats requires a multi-layered approach, combining several techniques that protect the integrity of the update process and the system as a whole. Key strategies include the encryption of communication channels using **HTTPS/TLS**, the use of **digital signatures** and **cryptographic hashing** for update validation, leveraging **Secure Boot** and **TPM** for hardware-based security, and enforcing **certificate pinning** and **strong authentication** to prevent MITM attacks. Furthermore, regular **firmware integrity checks** and monitoring ensure that the system maintains its security over time. Together, these mitigation strategies will significantly reduce the risk of successful attacks on the UEFI update process and ensure the ongoing trustworthiness of firmware installations.

Conclusions

In conclusion, ensuring the security of **UEFI Capsule Updates** is critical to protecting the firmware of modern systems from potential attacks, especially **Man-in-the-Middle (MITM)** attacks. As cyber threats continue to evolve, securing the firmware update process is essential for maintaining the integrity and functionality of systems. This paper has outlined several key **mitigation strategies** that can significantly reduce the risks posed by MITM and other related attacks.

Summary of Key Mitigation Strategies

The following strategies were discussed in the paper to enhance the security of UEFI Capsule updates:

1. **Encryption of Communication Channels:** Implementing secure communication protocols such as **HTTPS/TLS** ensures the integrity and confidentiality of the update data during transmission, preventing interception or tampering.
2. **Digital Signatures and Code Validation:** Using **digital signatures** and **cryptographic hashing** to validate the authenticity and integrity of the capsule update files helps ensure that only legitimate, untampered firmware is installed.
3. **Secure Boot and Trusted Execution Environments:** Enforcing **Secure Boot** prevents unauthorized firmware from being executed, while the **Trusted Platform Module (TPM)** enhances security by securely storing keys and validating updates.
4. **Certificate Pinning and Strong Authentication:** Enabling **certificate pinning** and implementing **multi-factor authentication** for update sources help protect against MITM attacks by ensuring the authenticity of the update server.
5. **Regular Firmware Integrity Checks:** Continuous validation of the firmware's integrity during boot and through monitoring tools helps detect and mitigate unauthorized modifications to the system's firmware.

Importance of Ongoing Research and Improvement

As the cybersecurity landscape evolves, so do the threats targeting firmware and UEFI updates. Attackers continuously develop new techniques to bypass security mechanisms, making it imperative for organizations and developers to remain vigilant and proactive in securing the firmware update process.

Ongoing research into UEFI security and **continuous improvement** of security measures are essential for staying ahead of emerging threats. Security practices that are effective today may not be sufficient tomorrow, which is why it's crucial to keep up with the latest developments in **encryption technologies, authentication protocols, and firmware protection techniques**.

The **collaboration between industry leaders**, security researchers, and system vendors plays a vital role in creating more robust solutions that can address future vulnerabilities and protect systems against evolving attack vectors. Therefore, while the strategies discussed here provide a strong foundation for mitigating MITM attacks in UEFI Capsule updates, **continuous adaptation and vigilance** will be necessary to ensure the long-term security of firmware systems.

In summary, mitigating the risks of MITM attacks in UEFI Capsule updates involves a combination of secure communication, validation techniques, and robust hardware security mechanisms. By adopting these strategies, organizations can better protect their systems from malicious interference, ensuring that the firmware update process remains secure and trustworthy.

References

1. Evangelista, Francesco. "Automatic Extraction of Exploitation Primitives in UEFI." PhD diss., Politecnico di Torino, 2023.
2. Sarvepalli, Vijay. "Securing UEFI: An Underpinning Technology for Computing." (2023): 15.
3. Bulusu, Mallik, and Vincent Zimmer. "White Paper UEFI Plugfest 2015-Challenges for UEFI and the Cloud." (2015).
4. Shaik, Y. (2024). *Securing Firmware updates: Addressing security challenges in UEFI capsule update mechanisms*. Researchgate.
5. Younus Shaik. (2024). *Securing Firmware updates: Addressing security challenges in UEFI capsule update mechanisms*. Researchgate. https://www.researchgate.net/publication/382447021_Securing_Firmware_Updates_Addressing_Security_Challenges_in_UEFI_Capsule_Update_Mechanisms

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.