

Zero Day Provisioning

Chaining TP-Link ZTP Vulnerabilities to Infiltrate Networks

AUGUST 2026

Stanislav Dashevskiy
Francesco La Spina

 **FORESCOUT**
RESEARCH

VEDERE LABS

Contents

1. Executive Summary	4
2. Why Analyze Zero Touch Provisioning?	6
2.1. TP-Link ZTP	7
3. Main Findings.....	9
4. Attack Scenarios	11
4.1. External Compromise Via a ZTP Race Condition	12
4.2. ZTP MiTM Exploitation in Omada Cloud	14
4.3. Local MiTM Exploitation	15
4.4. Impersonating a Local Omada Controller	16
4.5. Impersonating an Omada Device	16
5. Impact	17
6. Conclusion and Mitigation Recommendations	19
6.1. Disclosure Timeline	20
Part 2: Technical Deep-Dive	21
Omada ZTP Protocols	21
Discovery	21
Adoption	22
FSCT-2025-0003: Predictable Device Serial Numbers Can Be Used for Device Adoption	25

CVE-2025-15544: Device and Site Credentials Are Transmitted Insecurely	26
CVE-2025-15627: Use of Hard-Coded Cryptographic Keys in Omada Protocol Version 1	26
CVE-2025-15628: Compromise of the Chain of Trust	27
CVE-2025-15629: Insufficient Symmetric Key Entropy in Omada Protocol Version 1	29
FSCT-2025-0008: Default Credentials	29
CVE-2025-15630: Race Condition During Device Adoption in Omada Cloud	30
CVE-2025-9289: Cross-Channel Scripting Against Controllers	31
FSCT-2025-0011: Device Enumeration	32
CVE-2025-9290: Insecure Transmission of Credentials, Omada Protocol Version 2	32
CVE-2025-9291: Insufficient Certificate Checks in Omada Protocol Version 2	33
FSCT-2025-0014: Unrestricted File Upload to Omada Controllers	34
CVE-2025-15631: Password Hashes Encrypted with a Hard-Coded Key	34
CVE-2025-9292: CORS Bypass	34
CVE-2025-9293: Improper Certificate Validation	35

1. Executive Summary

For more than three years, Forescout Research – Vedere Labs has reported on the increasing exploitation of network infrastructure devices, such as routers and firewalls. Previous research, including [Sierra:21](#) and [Dray:Break](#), and observed threat actor activity by the larger research community targeting these devices, focused on individual vulnerabilities that enable remote code execution.

However, the growing use of Zero-Touch Provisioning (ZTP) creates opportunities for attacks at a much larger scale. Network vendors offer ZTP ecosystems in which provisioning servers push configurations and updates to client devices, including routers, switches, gateways, and wireless access points. This enables devices to be configured with little or no manual intervention.

In this report, we present 15 new vulnerabilities affecting ZTP in TP-Link Omada, a network device ecosystem for small and medium businesses. Some of these vulnerabilities also affect other TP-Link products and services, including IP cameras, smart home IoT devices, mobile applications, and cloud accounts.

The findings include the compromise of a cryptographic chain of trust through hard-coded keys, sensitive information disclosure, and remote code execution. By combining these vulnerabilities with two [previously disclosed](#) TP-Link vulnerabilities, CVE-2025-7850 and CVE-2025-7851, we demonstrate practical attacks against controllers and client devices that could allow attackers to infiltrate networks.

Beyond the individual vulnerabilities affecting TP-Link products, this research demonstrates how weaknesses in ZTP implementations can enable new attack scenarios at scale. We therefore provide mitigation recommendations for affected organizations and device manufacturers of ZTP-enabled devices.

KEY FINDINGS



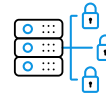
WHAT YOU NEED TO KNOW



15 new vulnerabilities affecting **Zero-Touch Provisioning (ZTP)** in TP-Link's Omada ecosystem



Some vulnerabilities extend to other TP-Link devices: **IP cameras, smart home IoT, apps, cloud ecosystem**



Findings: trust compromise in hard-coded **cryptographic keys**, **sensitive information disclosures**, and **remote code execution**

Two previous CVEs disclosed (CVE-2025-7850, CVE-2025-7851) + these new flaws allow **attackers to infiltrate networks through ZTP controllers and client devices**

4 IMPACT CATEGORIES

- 1 Client-side code execution
- 2 Information disclosure
- 3 Device hijacking and spoofing
- 4 Compromise of encrypted communications

MITIGATION RECOMMENDATIONS



WHAT YOU NEED TO DO



Avoid using the same password across all devices during provisioning



Change device credentials and use strong, unique passwords



Change TP-Link ID credentials and enable MFA



Rotate VPN keys and credentials that may have been exposed



Adopt a **defense-in-depth approach**



Track activity with **intrusion detection and network security monitoring controls**



Avoid MITM attacks: implement 802.1X + NAC, port security, dynamic ARP inspection, wireless client isolation, and network segmentation controls

2. Why Analyze Zero Touch Provisioning?

ZTP allows network administrators to provision network devices, such as switches and routers, remotely. Several network vendors now offer ZTP capabilities under proprietary names and using proprietary protocols.

When a new ZTP-enabled device connects to the network, it identifies a provisioning server, known as a controller, and receives configuration data from it. This may include network settings, credentials, and pending firmware updates. The controller can then continue to manage the enrolled device and apply subsequent configuration changes.

ZTP enables organizations to deploy new equipment rapidly and manage fleets of devices centrally, significantly reducing operational efforts and costs. However, it depends on a secure chain of trust between controllers and managed devices, as well as on securely designed and implemented network protocols. This creates several potential security risks:

- 1. Improved usability can introduce security trade-offs.** Zero-touch deployment may rely on pre-shared secrets or other mechanisms that allow controllers and devices to mutually authenticate. Establishing and maintaining this chain of trust securely can be difficult.
- 2. Centralized management creates a potential single point of compromise.** If an attacker compromises a provisioning server, they may gain access to all the devices it manages, putting entire networks at risk.
- 3. There is no universal set of ZTP protocols.** Vendors commonly design and implement proprietary protocols, increasing the possibility of security weaknesses in their design or implementation.
- 4. ZTP controllers and clients are often highly trusted within the network.** Attacks that compromise ZTP-enabled devices or abuse the provisioning protocols may therefore be difficult to detect. These devices may be covered by firewall exceptions and are often used for network administration, making malicious activity less likely to be identified by network intrusion detection systems. In this sense, abusing ZTP can resemble living off the land within network infrastructure.

Vulnerabilities in ZTP protocols and ZTP-enabled devices can therefore have significant consequences, potentially allowing threat actors to scale attacks across multiple managed devices.

Table 1 summarizes previously disclosed vulnerabilities affecting components used in ZTP environments. None of these vulnerabilities directly affects a ZTP protocol itself. Instead, they affect individual products or components that interact with provisioning and device management systems. This suggests that the security of ZTP protocols and related device-management protocols has received comparatively little public research attention.

Table 1 – Vulnerabilities affecting ZTP and device-management components

CVE	Affected vendor	Vulnerability/Impact	Known exploitation?
CVE-2017-3859	Cisco	Denial of Service	No
CVE-2018-0346	Cisco	Denial of Service	No
CVE-2018-0347	Cisco	Authenticated RCE	No
CVE-2022-30525	Zyxel firewalls	Unauthenticated RCE	Yes, reported by Shadowserver
CVE-2023-22955	AudioCodes VoIP	Missing integrity checks for firmware updates	No
CVE-2023-22956	AudioCodes VoIP	Hard-coded cryptographic keys	No

CVE	Affected vendor	Vulnerability/Impact	Known exploitation?
CVE-2023-22957	AudioCodes VoIP	Hard-coded cryptographic keys	No
CVE-2024-47575	Fortinet FortiManager	RCE	Yes, Mandiant reported that UNC5820 exploited the vulnerability as a zero-day to remotely execute commands, steal FortiGate configurations and credentials, and establish persistence on devices across various industries .

The FortiManager case demonstrates that threat actors are interested in compromising device-management ecosystems, not only individual managed devices. Because many vendors provide similar functionality, we selected one such ecosystem for further analysis.

This report focuses on TP-Link Omada, which is described in the following section. TP-Link also operates other ecosystems, including Festa for networking, VIGI for IP cameras, and Tapo and Kasa for smart home devices. These products use similar underlying protocols and design patterns, and some of the vulnerabilities described in this report also affect them. We discuss the broader impact in Section 5.

2.1. TP-Link ZTP

TP-Link Omada is a ZTP-enabled [ecosystem](#) for routers, switches, gateways, and Wi-Fi access points used by small and medium-sized businesses. The ecosystem supports three types of controllers:

- **Cloud-based controllers**, [provided by TP-Link](#). Figure 1 illustrates a deployment using a cloud-based controller, which can configure devices over the Internet and can be managed through a mobile application.
- **Dedicated hardware** controllers. Devices such as the [OC200](#) can operate as local or remote hardware controllers. Figure 2 illustrates a deployment using an on-premises controller.
- **Software controllers**, which can be installed on a standard computer or virtual machine and deployed locally or remotely.

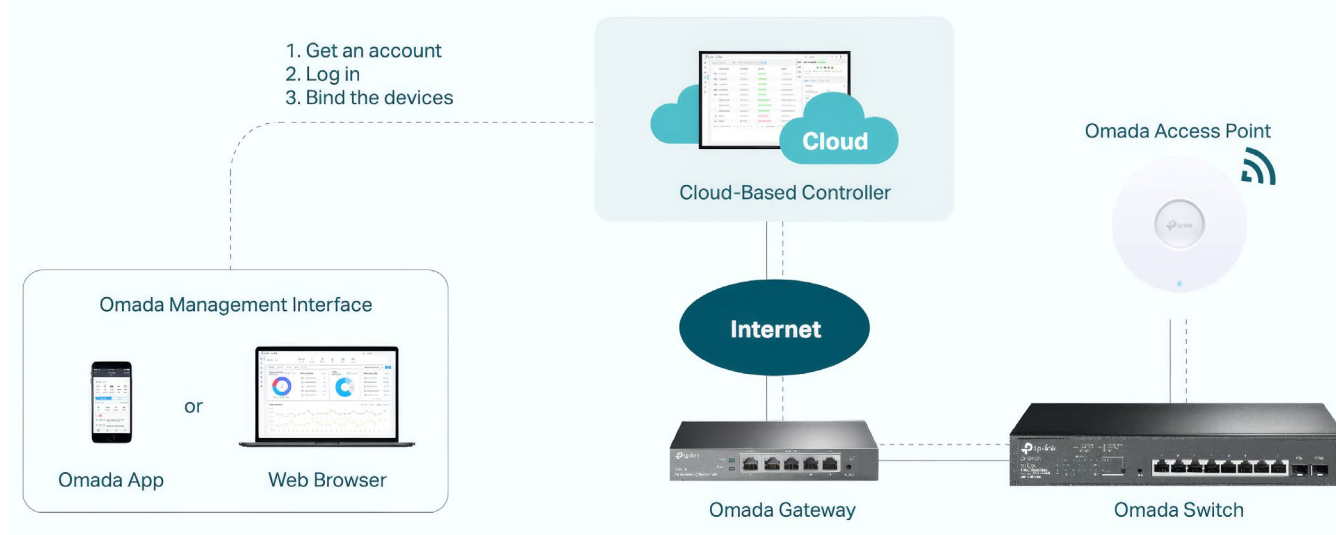


Figure 1 – Omada devices using a cloud-based controller (source: [tplink.com](#))



Figure 2 – Omada devices using a local, on-premises controller (source: tp-link.com)

An organization can operate several controllers, including both on-premises and cloud-based controllers, alongside multiple client devices. The Omada ecosystem is organized hierarchically, into sites, which are effectively groups of devices. Each site can contain multiple switches and access points but only one gateway. Each site uses its own set of credentials: a single username and password shared by all devices within that site. When a controller assigns a device to a site, it configures the device with those site-specific credentials.

A hypothetical Omada network could include the following components:

- **Controllers:** self-hosted hardware controllers, such as the [OC200](#) or [OC300](#), or cloud-based controllers
- **Routers:** [ER7206](#)
- **Core and distribution switches:** [TL-SX3016F](#), [TL-SG3428XMP](#) or [TL-SG3452P](#)
- **Access points:** [EAP660HD](#), [EAP225-Outdoor](#)

The Technical Deep-Dive section explains the ZTP device-adoption protocols in detail. In brief:

- When a cloud-based controller is used, an administrator can initiate device adoption by entering only the device's serial number in the controller's web interface. The remaining steps occur automatically when the device connects to the Internet.
- When a hardware or software controller is used, the administrator selects a new device from a list in the controller's web interface and initiates adoption. The device appears in this list after the local discovery protocol indicates that it is present and is ready to be adopted.

[Section 5](#) discusses how other TP-Link products, including [VIGI IP cameras](#), use similar protocols and cloud infrastructure.

3. Main Findings

We analyzed the components listed in Table 2. We found few meaningful differences among the relevant binaries across several versions of the hardware, software, and cloud-based controllers and client devices we examined. As a result, the findings affect multiple devices that support the Omada protocols, as well as products from other TP-Link ecosystems, including [VIGI](#), [Festa](#), [Kasa](#), and [Tapo](#).

Table 2 – Analyzed components

Type	Device / Model	Version
Client devices	ER605 router	2.2.6 Build 20240718 Rel.82712
	ER7206 router	2.2.6 Build 20240718 Rel.82712
	VIGI C430 IP camera	V1_2.0.5 Build 241028
	Tapo C200 IP camera	1.1.8 Build 250310 Rel. 73289n
	Kasa KC410S IP camera	2.3.26
Controllers	OC200	5.15.22.40
	Software controller	5.15.8.2
	VIGI VMS	1.8.56_64bits
	Omada Cloud	5.15.22.40
Android mobile applications, analyzed only for CVE-2025-9293 and CVE-2025-15628	Aginet	2.11.23
	Deco	3.9.76
	Festa	1.6.9
	Kasa	3.4.101
	KidShield	1.1.19
	Omada	4.24.13
	Omada Guard	1.0.14
	Tapo	3.11.114
	Tether	4.10.42
	tpCamera	3.2.12
	VIGI	2.7.16
	Wi-Fi Navi	1.4.8
	WiFi Toolkit	1.4.2

We identified the 15 new vulnerabilities listed in Table 3. Their potential impacts fall into four broad categories:

- Client-side code execution through cross-channel scripting
- Disclosure of sensitive information, including passwords and cryptographic keys
- Device hijacking and spoofing
- Compromise of encrypted communications and the underlying chain of trust

Table 3 provides a summary of each issue. The vulnerabilities are described individually in the Technical Deep-Dive section. TP-Link decided not to assign CVE identifiers to FSCT-2025-0003, FSCT-2025-0008, FSCT-2025-0011, or FSCT-2025-0014.

Table 3 – Findings at a glance

Vulnerability ID	Affected products	Description	Potential impact
FSCT-2025-0003	Omada and Festa VPN routers	Devices that have not already been adopted can be adopted using only their serial numbers.	Device hijacking and denial of legitimate adoption.
CVE-2025-15544	Omada hardware, software, and cloud-based controllers, and potentially other product lines	Local and cloud-based controllers transmit site credentials insecurely.	Password disclosure and unauthorized access to devices managed by the same controller.
CVE-2025-15627	Omada hardware and software controllers, and potentially other product lines	Omada controllers contain a hard-coded private key used to protect communications with client devices when version 1 of the Omada protocol is used.	Credential exposure and controller impersonation.
CVE-2025-15628	Omada hardware and software controllers and potentially other product lines	Version 2 of the Omada protocol relies on a hard-coded TLS server certificate and corresponding private key, compromising the chain of trust.	Impersonation of local and cloud controllers, and man-in-the-middle attacks across multiple TP-Link product families.
CVE-2025-15629	Omada hardware and software controllers	The RC4 key used to encrypt communications between clients and controllers in version 1 of the Omada protocol has insufficient entropy and can be predicted.	Decryption of sensitive communications by third parties.
FSCT-2025-0008	Omada hardware, software, and cloud-based controllers and potentially other product lines	During initial device adoption, authentication challenges are signed using default credentials.	Device spoofing and disclosure of sensitive information.
CVE-2025-15630	Omada cloud-based controllers and potentially other product lines	During cloud adoption, an attacker can exploit a race condition by initiating the handshake before the legitimate device while spoofing only its MAC address. This may expose the initial configuration, including administrator credential hashes and other confidential information.	Device spoofing, credential harvesting, information disclosure, and network reconnaissance.
CVE-2025-9289	Omada hardware, software, and cloud-based controllers	The Omada controller web interface is vulnerable to cross-channel scripting because values supplied during device adoption are not properly sanitized.	Arbitrary client-side code execution in an administrator's browser.

Vulnerability ID	Affected products	Description	Potential impact
FSCT-2025-0011	Omada cloud-based controllers	Device serial numbers are sequential and predictable. Attackers can use guessed serial numbers to retrieve device MAC addresses and related information through the Omada Cloud API or web interface.	Device enumeration and support for further device-targeting attacks.
CVE-2025-9290	Omada hardware, software, and cloud-based controllers and potentially other product lines	Version 2 of the Omada protocol does not ensure that salt values used in credential authentication are unique or non-empty, allowing attackers to obtain static credential hashes.	Rainbow-table and offline password-cracking attacks.
CVE-2025-9291	Omada and Festa VPN routers and potentially other devices	Omada client devices perform insufficient Common Name checks when validating controller certificates.	Impersonation of cloud controllers and man-in-the-middle attacks.
FSCT-2025-0014	Omada hardware, software, and cloud-based controllers and potentially other product lines	Local and cloud-based controllers allow arbitrary files to be uploaded and later retrieved through unauthenticated and temporary links.	Unrestricted file hosting and delivery of malicious files into trusted networks.
CVE-2025-15631	Omada and Festa VPN routers and potentially other devices	Unsalted MD5 hashes of device-management passwords are encrypted using a hard-coded key.	Password disclosure and offline credential cracking.
CVE-2025-9292	Omada cloud-based controllers	The controller's Content Security Policy permits connections to CloudFront and Amazon Web Services subdomains.	Data exfiltration through attacker-controlled infrastructure.
CVE-2025-9293	TP-Link Android applications: Tapo 3.11.114 Kasa 3.4.101 Omada 4.24.13 Omada Guard 1.0.14 Tether 4.10.42 Deco 3.9.76 Aginet 2.11.23 tpCamera 3.2.12 WiFi Toolkit 1.4.2 Festa 1.6.9 Wi-Fi Navi 1.4.8 KidShield 1.1.19	The applications perform insufficient certificate validation and rely on the same compromised chain of trust described in CVE-2025-15628.	Man-in-the-middle attacks and interception or manipulation of application traffic.

4. Attack Scenarios

Figure 3 illustrates the network setup used to explain possible attack scenarios involving the newly discovered vulnerabilities. This setup includes:

- Omada-enabled client devices, such as gateways, switches, and access points, deployed on an internal network.
- The TP-Link cloud controller.
- A hardware controller and a software controller deployed on the internal network. These controllers can be accessed from a workstation or through the Omada mobile application.

As recommended by TP-Link, only the controller's web interfaces should be exposed to the Internet, with other ports blocked by a firewall. However, the presence of a firewall does not play a significant role in the attack scenarios presented in this report.

The setup includes three different controllers to illustrate the range of possible deployment models, although a real network would probably only use one or two.

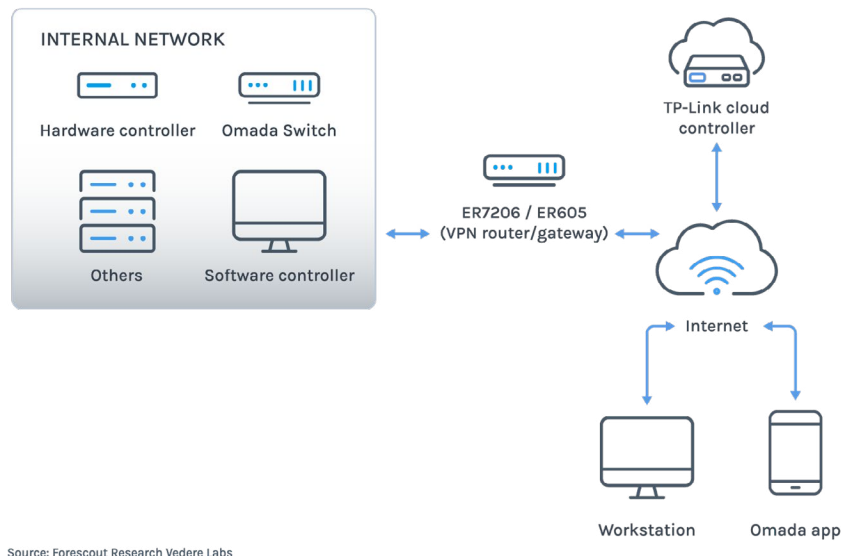
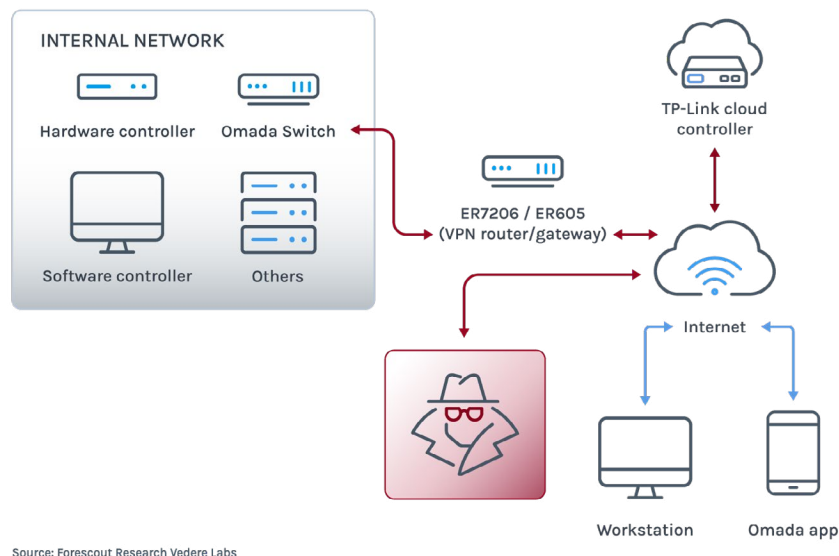


Figure 3 – A typical Omada setup

The following scenarios illustrate how attackers could combine the vulnerabilities we discovered to gain unauthorized access to controllers and client devices. We tested these scenarios in our lab, but other attack paths may be possible depending on the target environment and the attacker's objectives.

The Technical Deep-Dive section describes the vulnerabilities in detail, except for CVE-2025-7850 and CVE-2025-7851 which we previously covered in a [separate blog post](#).

4.1. External Compromise Via a ZTP Race Condition



In this scenario, an attacker outside the internal network attempts to gain control of devices inside it. The attack proceeds as follows:

1. The attacker collects the MAC addresses of Omada devices by enumerating predictable serial numbers (FSCT-2025-0003) with the goal of identifying devices that have not yet been adopted by a cloud controller. Using the corresponding serial numbers, the attacker can retrieve MAC addresses through the Omada API or web dashboard (FSCT-2025-0003 and FSCT-2025-0011). These requests can be made in batches.
2. The attacker impersonates an Omada device that is about to be adopted by an administrator through the cloud controller. By spoofing the device's MAC address, the attacker sends an initial type 3 message on its behalf (see Figure 13). The attacker can resend this message every 60 seconds to exploit the race condition described in CVE-2025-15630.

The objective is for the cloud controller to accept the legitimate device's initial "helloCloud" message and then accept the next expected message from the attacker. In our testing, this attack had a high success rate. We estimate that targeting 1,000 different MAC addresses would require approximately 17 requests per second.

3. After the cloud controller responds and the spoofed device enters the Adopt phase, the attacker can authenticate to the cloud using the default credentials admin/admin as described in FSCT-2025-0008. See message types 1048576 and 1048577 in Figure 10.

The cloud controller then returns the device configuration, including a cleartext username, an unsalted MD5 hash of the site password (CVE-2025-15544), and potentially other sensitive information, such as VPN keys.

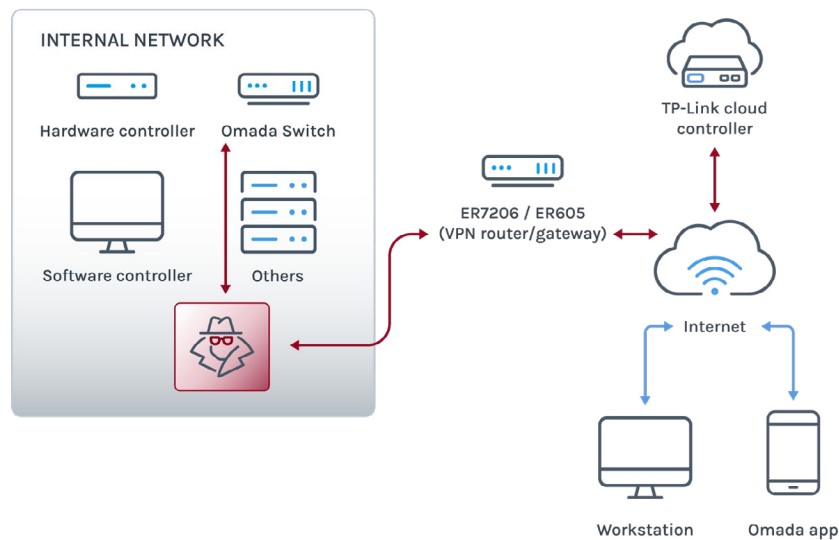
4. The attacker can also inject arbitrary JavaScript into the administrator's web interface through the cloud controller by sending message type 1048580 from the spoofed device (CVE-2025-9289; see Figure 10). This can be used to exfiltrate data. For example, the attacker could display a fake login dialog to capture the administrator's cloud controller credentials, while also exploiting CVE-2025-9292 to bypass [CORS](#) restrictions.
5. After obtaining the cloud controller administrator credentials, the attacker can log in and manage devices that have already been adopted by that controller. The attacker can also adopt the legitimate devices originally targeted in the attack.

The attacker may then configure VPN tunnels into the internal network. Once inside, they can attempt to compromise Omada devices using additional vulnerabilities, including the post-authentication OS command injection vulnerability CVE-2025-7850.

The current site credentials for all managed devices are also visible in "Site settings" in the cloud controller interface. These credentials may allow the attacker to authenticate directly to affected Omada devices and conduct further attacks.

If the plaintext site password cannot be obtained through the interface, the attacker still has the unsalted MD5 hash disclosed during adoption. This hash can be used directly for authentication through the Omada protocols, as described in CVE-2025-15544.

4.2. ZTP MiTM Exploitation in Omada Cloud



In this scenario, the attacker is positioned inside the network and can control traffic entering and leaving the Internet-facing gateway, an ER7206. This position could be established by an internal attacker or by compromising another device on the network.

Using CVE-2025-15628, the attacker can obtain the server certificate and corresponding private key used by the local controller, referred to here as the “localhost” certificate, and use them to perform a man-in-the-middle (MiTM) attack.

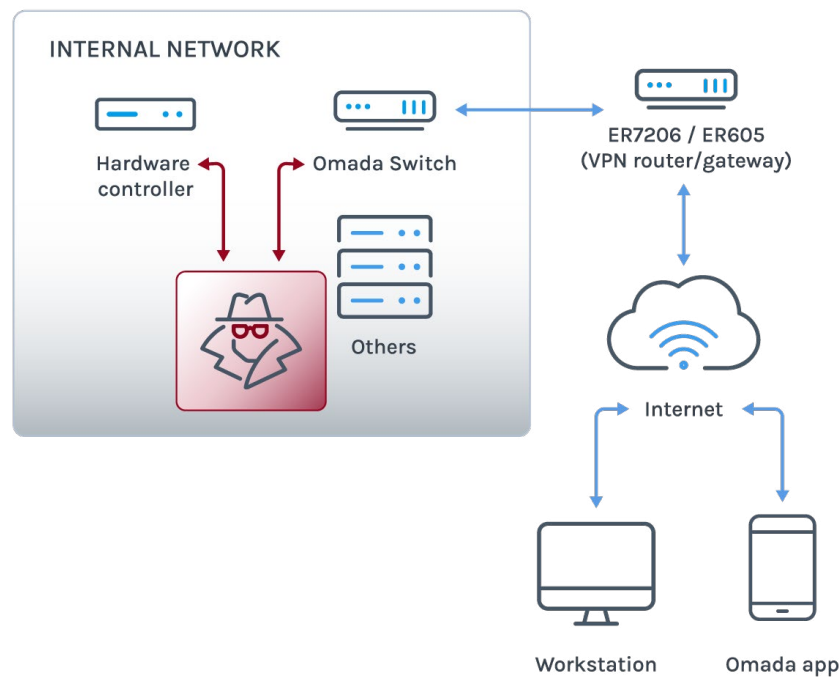
1. When a device sends a “helloCloud” request to the cloud controller, the attacker intercepts it. The corresponding response contains the fully qualified domain name (FQDN) that the device will use to communicate with the controller.

At this stage, the attacker can intercept and modify Discovery protocol traffic because the device does not validate the Common Name (CN) of the TLS certificate and therefore accepts the “localhost” certificate.

2. The attacker replaces the FQDN in the “helloCloud” response with the IP address to which that FQDN resolves. The device continues to communicate with the legitimate cloud controller during the Adoption and Management protocols, while the attacker proxies the traffic between the device and the controller. This is possible because of the flawed certificate-validation logic described in CVE-2025-9291.
3. Once positioned between the device and the cloud controller, the attacker can intercept sensitive information and send malicious device configurations. They may also exploit additional vulnerabilities, such as CVE-2025-7850.
4. The attacker can also exploit vulnerabilities in the controller, including CVE-2025-9289 to inject arbitrary JavaScript into the cloud administrator’s web interface, and CVE-2025-9292 to exfiltrate data.

[Section 4.1](#) describes additional actions the attacker may be able to perform at this stage.

4.3. Local MiTM Exploitation



Source: Forescout Research Vedere Labs

In this scenario, the attacker is positioned in the local network as in the previous scenario, but does not need to control the existing traffic path between a device and a controller.

1. The attacker receives the local Discover request message sent by a victim Omada device (see Figure 8). Because this is a UDP broadcast, any device on the local network can receive it.
2. The attacker forwards the message directly to the local controller, optionally modifying values such as the IP and MAC addresses.
3. The local controller sends the Discover response to the attacker. The attacker forwards the response to the victim device, replacing the controller's IP address with the attacker's own address.

The rogue device then relays messages between the victim and the controller, modifying them as necessary (CVE-2025-15628). At this stage, the attacker can intercept or infer credentials, for example, by exploiting CVE-2025-15544 or CVE-2025-9290.

4. The attacker can then pursue outcomes similar to those described in steps 3 and 4 of [Section 4.2](#).

This scenario assumes that the target Omada device has not yet been adopted by the controller, or that the attacker can interrupt the communication between the victim device and the controller. In the latter case, an adopted Omada device will begin sending and receiving local Discovery protocol messages after a timeout of several minutes.

Even when a cloud controller is in use, the victim device will still send the UDP Discover request.

These conditions are not required if the attacker can establish a full MiTM position and proxy traffic between the victim device and its controller.

4.4. Impersonating a Local Omada Controller

In this scenario, the attacker is positioned on the local network and deploys a rogue device that impersonates a local controller.

1. The attacker responds to the broadcast Discover message and presents the rogue device as a local controller.
2. The attacker can decrypt messages exchanged through the Adoption and Management protocols by exploiting CVE-2025-15628. Alternatively, the attacker can downgrade the protocol version and exploit CVE-2025-15627.
3. If the device has not yet been configured, the attacker can authenticate to it using the default credentials. Otherwise, the attacker can obtain or derive the device credentials in several ways:
 - a. Default credentials (FSCT-2025-0008)
 - b. Rainbow-table attacks (CVE-2025-9290)
 - c. Passive traffic analysis exploiting additional cryptographic weaknesses (CVE-2025-15629 and CVE-2025-15544)
4. If the rogue controller successfully authenticates to the victim device, the attacker can execute OS commands remotely as root by exploiting CVE-2025-7850.

This scenario is similar to the one described in [Section 4.3](#), except that the rogue device acts as a local controller rather than an intermediary between the victim device and the local controller.

For the attack to succeed, the victim device must either not yet have been adopted by a controller, or lose its connection to the controller. In the latter case, the device will begin broadcasting Discover requests that the rogue controller can answer.

In local environments where no controller is present, Omada devices continue to broadcast Discovery protocol messages, allowing the rogue controller to respond whenever an adoption attempt occurs.

4.5. Impersonating an Omada Device

In this scenario, the attacker is positioned on the local network and deploys a rogue device that impersonates an Omada client device. Because of the way the Omada protocols operate, the attacker has different capabilities from those described in the previous scenarios.

1. The attacker sends a rogue Discover request that mimics a legitimate device expected to be present on the network. The system administrator must accept this request through the controller's web interface, meaning the attack depends on human interaction.

Once the admin accepts the request, the rogue device can proceed with the adoption process.

2. Because of the way the version 2 Adoption protocol works (see Figure 10), the controller authenticates to the rogue device. The attacker can exploit CVE-2025-9290 to derive the site credentials used during the controller's authentication attempt.

An easier method of obtaining the site credentials is to wait for the relevant Management protocol message, as described in CVE-2025-15544. The attacker can then use the captured credentials to access other Omada devices within the same site.

3. Because the controller first attempts to authenticate using the default credentials, the rogue device can accept them and present itself as an unconfigured device. The rogue device can then authenticate to the controller using the same default credentials (FSCT-2025-0008).
4. Once this process is complete, the rogue device can inject malicious JavaScript into the controller's web interface by exploiting CVE-2025-9289. The attacker can also use the newly obtained credentials to access other Omada devices within the same site.

Local controllers use a different Content Security Policy from the cloud controller, so CVE-2025-9292 cannot be exploited in the same way. However, the CSP used by local controllers allows connections to several [Mapbox](#) hosts.

We did not test this possibility, but an attacker may be able to exfiltrate data from a local controller through the Mapbox API using an attacker-controlled paid account.

5. Impact

TP-Link is a major vendor of small office/home (SOHO) networking devices and other IoT equipment. In 2022, the company stated that its products were “[available in more than 170 countries and regions, serving more than 1.7 billion users](#)”.

Not all of these users are affected by the vulnerabilities described in this report. The following data points help indicate the potential scale of exposure.

TP-Link's website provides [examples](#) of Omada devices used in large residential developments, industrial complexes, offices, warehouses and other environments. Figure 4 shows the network topology of an [Omada deployment in an industrial estate](#) comprising nine OC200 controllers, eight Omada routers and switches, and nearly 150 Omada access points.

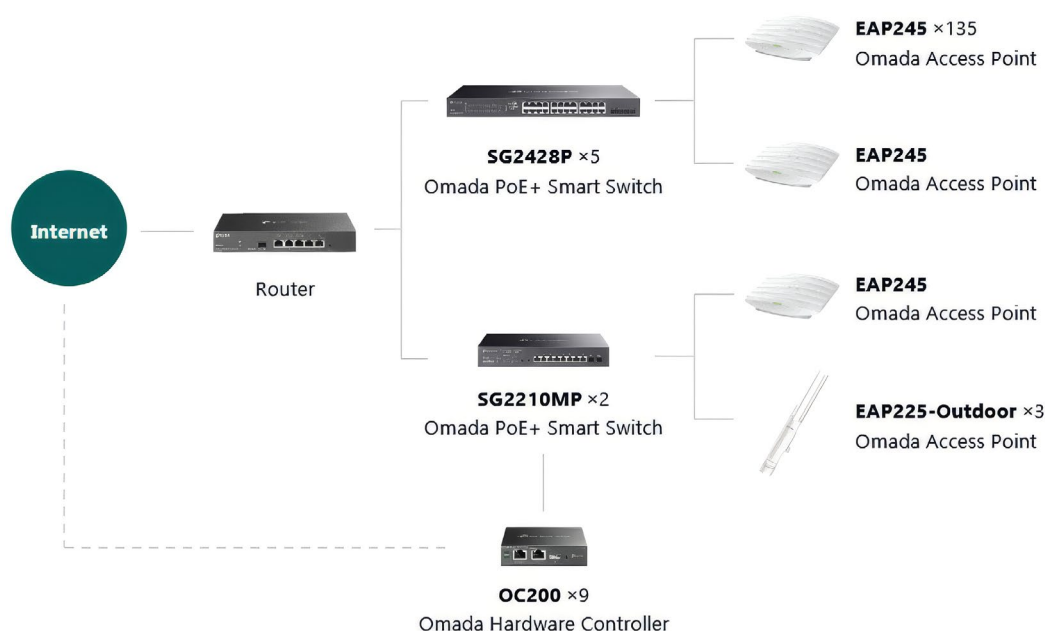


Figure 4 – Omada deployment in an industrial estate (source: [tp-link.com](#))

This deployment is similar to the setup described in the previous section, meaning that several of the attack scenarios presented there may be feasible in an environment of this kind. As Omada becomes a [more prominent TP-Link business product line](#), replacing solutions such as [JetStream](#) and [Pharos](#), we expect the number of similar deployments to increase.

We currently observe more than 1,800 [Omada controllers accessible online](#). These appear to be exceptions, however, because such deployments are not intended to be exposed directly to the Internet.

Mobile application downloads provide another indication of the scale of the Omada ecosystem. The [TP-Link Omada](#) and [Omada Guard](#) applications have collectively been downloaded **1.1 million times** from Google Play.

Other TP-Link mobile applications affected by CVE-2025-15628 and CVE-2025-9293, listed in Table 3, have been **downloaded more than 70 million times** from Google Play. Based on these figures, we estimate that the affected applications may therefore be associated with several million active user accounts.

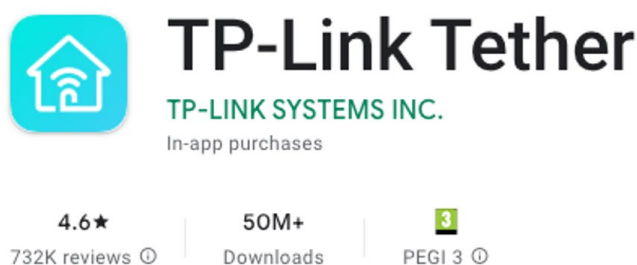


Figure 5 – One of the most downloaded applications affected by CVE-2025-9293 (source: Google Play Store)

Many of these applications can be used with the same TP-Link cloud account. This indicates that multiple TP-Link product ecosystems share elements of the same cloud identity and infrastructure. They also rely on the compromised chain of trust described in CVE-2025-15628 and CVE-2025-9293, potentially allowing attackers to impersonate application back-end services or intercept communications between applications and devices.

The vulnerabilities extend across product lines because the Omada protocols build on existing TP-Link device protocols and share many of the same design decisions. For example, [Festa routers](#) use similar provisioning protocols and are affected by several of the same vulnerabilities, as shown in Table 3.

Beyond Omada and the affected mobile applications, CVE-2025-15628 also affects the [TP-Link VIGI physical surveillance ecosystem](#). Figure 6 shows the network topology of a [VIGI deployment at a financial institution in India](#).

In deployments using a local Video Management System, which operates in a manner similar to a local Omada controller, an attacker may be able to impersonate the controller, intercept surveillance communications, or manipulate data transmitted between the controller and managed devices. Depending on the affected functionality, this could include interference with video feeds.

The use of shared cloud accounts across TP-Link ecosystems may also increase the impact of account compromise. For example, compromising an Omada cloud account, as described in [Section 4.1](#), could potentially provide access to the associated [VIGI Cloud Video Management System](#) account, allowing the compromised Omada account to serve as an entry point into a separate deployment.

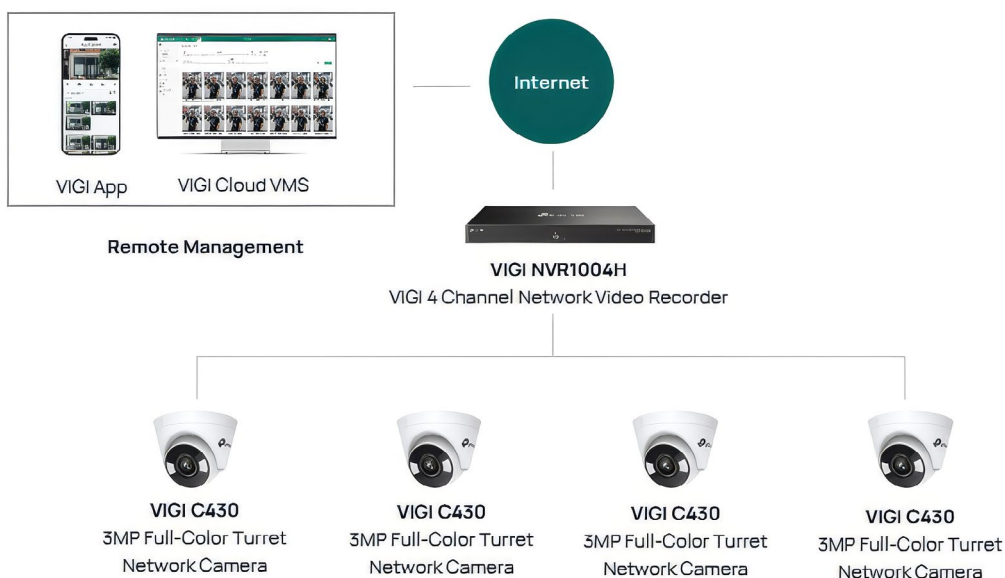


Figure 6 – An example of a real-world VIGI deployment at a financial institution in India (from tp-link.com)

6. Conclusion and Mitigation Recommendations

This research shows that ZTP implementations can contain significant security flaws that threat actors may exploit to conduct attacks at scale. As more vendors implement ZTP and more organizations adopt it, these risks require greater attention.

The potential impact is amplified by the number of affected products and the use of shared cloud infrastructure, protocols, and design patterns across multiple TP-Link product lines.

Mitigating the vulnerabilities described in this report requires organizations to patch affected devices and update all affected software, including controllers and mobile applications.

CVEs	Official advisory
CVE-2025-9289, CVE-2025-9290	https://support.omadanetworks.com/us/document/114950/
CVE-2025-9292, CVE-2025-9293	https://www.tp-link.com/us/support/faq/4969/
CVE-2025-9291, CVE-2025-15544, CVE-2025-15627, CVE-2025-15628, CVE-2025-15629, CVE-2025-15630, CVE-2025-15631	https://support.omadanetworks.com/us/document/130626/

In addition to applying patches and software updates, we recommend that organizations:

- Avoid using the same password across all devices during provisioning.
- Change device credentials and use strong, unique passwords.
- Change TP-Link ID credentials and enable multifactor authentication where available.
- Rotate VPN keys and credentials that may have been exposed.
- Reduce the risk of local MiTM attacks by implementing appropriate controls, such as 802.1X + NAC, port security, Dynamic ARP inspection, wireless client isolation, and network segmentation.
- Adopt a defense-in-depth approach.
- Monitor network activity using intrusion detection and other network security monitoring controls.

We recommend that vendors implementing ZTP protocols:

- Apply secure-by-design principles and regularly test protocols, software, and platforms for security weaknesses, including through vulnerability disclosure and bug bounty programs.
- Apply foundational security principles such as least privilege and separation of duties. Avoid reusing the same intermediate certificates across multiple product lines and consider using separate certificate authorities where appropriate.
- Validate all relevant certificate properties correctly.
- Never embed certificate private keys in software. Where device-specific keys must be stored locally, protect them using hardware-backed mechanisms such as Trusted Platform Modules.
- Do not hard-code cryptographic keys or passwords or rely on security through obscurity to protect them.
- Avoid default credentials.
- Use cryptographically secure password-hashing algorithms with appropriate salts and work factors.
- Use unique salts for password hashes to prevent precomputed attacks such as rainbow tables.
- Use widely reviewed standard protocols and cryptographic algorithms. For example, use HMAC-SHA256 for signing authentication challenges and maintain the ability to replace algorithms when they no longer provide adequate security.

6.1. Disclosure Timeline

Public disclosure of these vulnerabilities took significantly longer than the industry-standard 90-day period. Although we reported all the issues at the same time, they were addressed at different stages over a total period of 426 days. The disclosure timeline was as follows:

- **Day 0:** We disclosed all the vulnerabilities described in this report to TP-Link in June 2025.
- **Day 18:** TP-Link confirmed the issues and initially estimated that remediation would take 120 days.
- **Day 25:** TP-Link informed us that 120 days might not be sufficient to address all the reported issues.
- **Day 45:** We informed TP-Link that a presentation covering CVE-2025-7850 and CVE-2025-7851 had been accepted for [Hack.lu](#).
- **Day 64:** TP-Link informed us that the issues would be addressed in batches:
 - Several minor issues had already been patched.
 - Three issues were expected to be fixed by the end of September 2025, **119 days after disclosure**.
 - Two issues were expected to be fixed by October 21, 2025, **141 days after disclosure**.
 - Seven issues, representing the core weaknesses in the Omada protocols, were not expected to be fixed until the end of Q1 2026, **301 days after disclosure**.
 - One issue was **not considered a vulnerability and would not be fixed**.
- **Day 140:** We published a blog post about **CVE-2025-7850** and **CVE-2025-7851** and TP-Link published the corresponding advisories.
- **Day 199:** TP-Link reported that patches for two additional vulnerabilities had been completed and were undergoing testing, while patches for two more were nearly complete. TP-Link also determined that another issue was not considered a vulnerability and would not be patched.
- **Day 234:** TP-Link published advisories for **CVE-2025-9290** and **CVE-2025-9289**.
- **Day 240:** TP-Link informed us that addressing **FSCT-2025-0003** and **FSCT-2025-0011** would require system-wide changes to **device-level manufacturing processes, packaging, factory workflows, and channel and distributor logistics**. TP-Link also indicated that remediation of the remaining issues might not be completed before **Q3 2026, at least 394 days after disclosure**.
- **Day 256:** TP-Link published advisories for CVE-2025-9292 and CVE-2025-9293.
- **Day 389:** TP-Link informed us that it had assigned low CVSS scores to **FSCT-2025-0003** and **FSCT-2025-0011** and will not issue the CVE identifiers. The remaining CVE identifiers had been reserved.
- **Day 426:** We published this report and TP-Link published the remaining advisories.

Part 2: Technical Deep-Dive

Omada ZTP Protocols

Omada ZTP uses several network protocols that rely on UDP and TCP for transport and TLS for data encryption. These protocols use multiple default ports, a full list of which is available in [TP-Link's online documentation](#).

Each protocol message consists of a JSON document preceded by a 4-byte payload length encoded in network byte order. Figure 7 shows an example message containing several elements, of which “header” and “body” are always present. The full message format is described in our [previous research](#).

```
1 {
2   "header" : {
3     "version" : "1.2.0",
4     "mac" : "[REDACTED]",
5     "type" : 2,
6     "device" : "gateway",
7     "error":0,
8     "dest": "[REDACTED]",
9     "verCap" : 3
10  },
11  "body" : {
12    "deviceInfo" : {
13      "model" : "ER7206",
14      "modelVer" : "2.20",
15      "hwVer" : "ER7206 v2.20",
16      "fwVer" : "[REDACTED]",
17      "time" : "0 days 03:43:05",
18      "ip" : "192.168.0.1",
19      "fac" : "false"
20    },
21    "deviceMisc" : {
22      "extraPortNum" : {
23        "extraPort" : 0,
24        "lteWan" : 0,
25        "usbLteWan" : 1
26      },
27      "portNum" : 6
28    },
29    "controller" : {
30      "id" : "[REDACTED]"
31    }
32  }
33 }
```

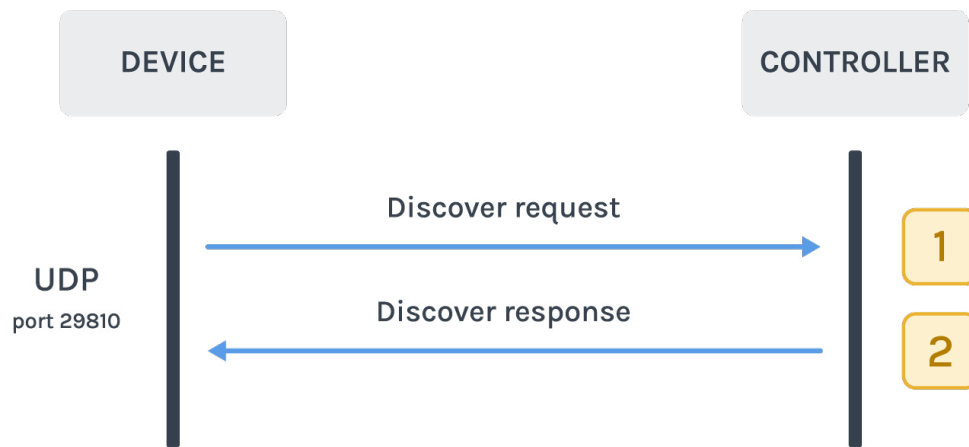
Figure 7 – Example JSON message

Some protocols are divided into phases. For example, device onboarding involves two protocols: Discovery, which has a single phase, and Adoption, which includes Adopt, Manage, Reset, Prelink, Rebuild and several other phases.

There is no public documentation explaining these protocols and phases or providing standard names for them. The following sections explain the protocols to the extent necessary to understand the technical details of the vulnerabilities discussed in this report.

Discovery

Figure 8 illustrates the local Discovery protocol used with on-premises controllers. In this process, a client device attempts to locate a controller on the local network. The device sends a UDP broadcast message to port 29810. In this and the following diagrams, message types contained in the JSON payloads are shown in yellow rectangles.



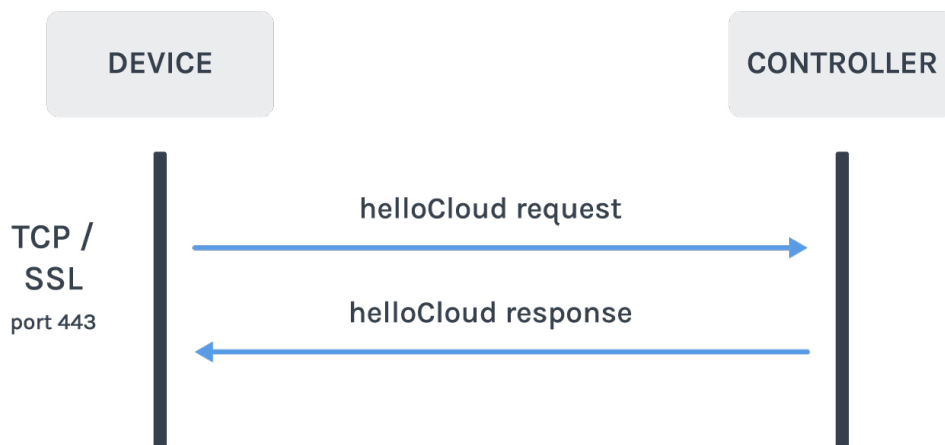
Source: Forescout Research Vedere Labs

Figure 8 – Local Discovery protocol

The client device sends a Discover request, JSON message type 1, containing basic information such as its MAC address and model. The controller responds with a Discover response, type 2, containing similar information, including the controller ID, its IP address, and the port to be used for the Adoption protocol if it differs from the default. Both messages are transmitted in cleartext.

The Discovery protocol used with cloud-based controllers operates differently, as shown in Figure 9. It uses SSL over TCP. In this case, the device sends a Discover request, known as helloCloud, directly to a domain belonging to Omada Cloud.

The message format also differs, as shown in Figure 12, and the payload closely resembles the Tapo device protocol previously [researched by Imre Rad](#). Based on this and other similarities, we believe that at least some parts of the Omada protocols evolved from, or alongside, the TP-Link Smart Home device protocol used by [Tapo](#) and [Kasa](#) devices.



Source: Forescout Research Vedere Labs

Figure 9 – Cloud Discovery protocol

Adoption

After completing the Discovery protocol, the device attempts to communicate with the newly discovered controller through the Adoption protocol. From this point onward, the communication flow is the same for local and cloud-based controllers, although they use different default ports and TLS server certificates.

Figure 10 shows the message flow between a device and a controller during the Adopt phase and part of the Manage phase of the Adoption protocol. The most relevant details are described below.

The client device initiates a connection with the controller by sending a message containing basic information, such as its device type and MAC address. The controller responds with a random authentication challenge `randomKeyForDeviceVerify`, in message type 1048576.

After receiving this challenge, the device sends a two-round hash of its credentials, using the challenge as a salt during the second hashing round. It also sends its own challenge to the controller, `randomKeyForSystemVerify`, in message type 1048577. The controller then uses that challenge to authenticate to the device in the same way, in message type 1048578.

If both parties authenticate successfully, the protocol enters the Adopt phase. During this phase, the controller configures the device remotely through message types 1048585, 1048580, and 1048581.

In summary, the device reports its current configuration to the controller and receives:

- a primary configuration containing basic network settings;
- a list of optional modules to enable; and
- new credentials for future authentication to the device.

These are the site credentials referred to earlier in the report.



Source: Forescout Research Vedere Labs

Figure 10 – Adopt and Manage phases of the Adoption protocol, version 2

Once this process is complete, the controller has adopted the device, and the protocol enters the Manage phase. During this phase, the controller can send additional configuration data, including settings for optional modules such as [WireGuard VPN](#), through message types 1048586, 256, 4096, and others.

The process described above is the version 2 Adoption protocol. A version 1 variant also exists and operates differently.

The software and hardware controllers we tested do not support version 1. However, any client device can be forced to use this variant by negotiating version 1 in the Discover response message, shown as message type 2 in Figure 8.

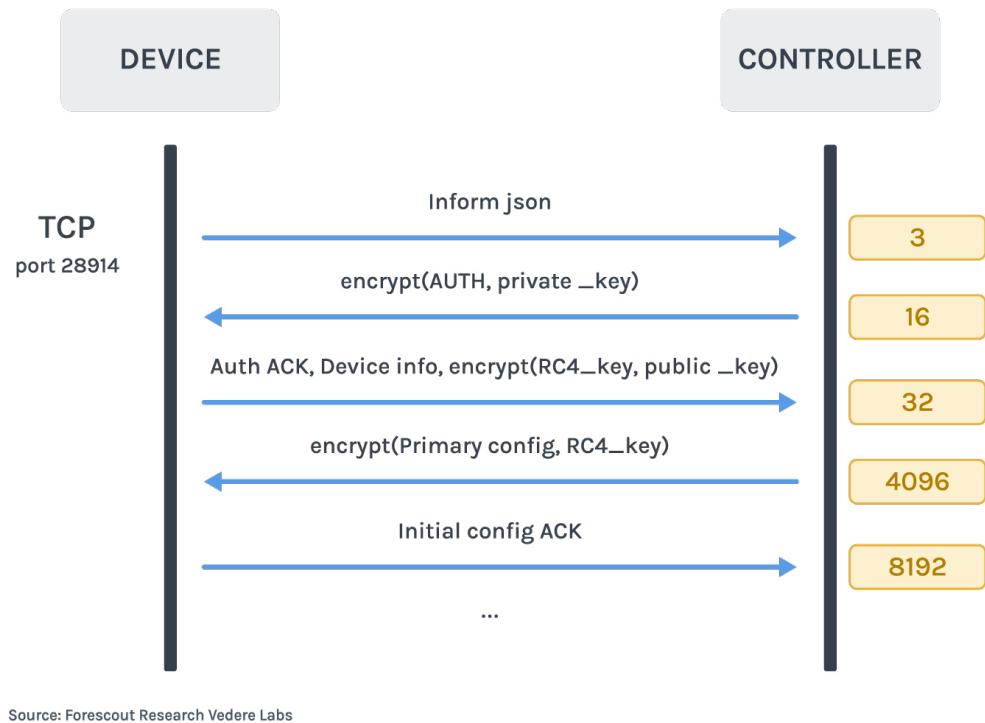


Figure 11 – Adopt phase of the Adoption protocol, version 1

Figure 11 illustrates the Adopt phase in version 1. This variant does not use SSL. Instead, it relies on pre-shared cryptographic keys to protect data payloads.

Another difference is that the controller authenticates to the device by encrypting partially hashed credentials with a pre-shared private key in message type 16. The remaining data is transmitted in cleartext.

The device extracts the authentication hash by decrypting it with the corresponding public key. If authentication succeeds, the device generates a symmetric encryption key for protecting data payloads in both directions. A new key is generated for each connection.

The controller retrieves this symmetric key by decrypting the payload with the same private key, in message type 4096.

```

1 {
2   "id" : 1,
3   "method" : "helloCloud",
4   "params" : {
5     "alias" : "ER605",
6     "authCode" : "[REDACTED]",
7     "cloudUserName" : "",
8     "controllerVersion" : "",
9     "deviceHwVer" : "2.0",
10    "deviceId" : "[REDACTED]",
11    "deviceMac" : "[REDACTED]",
12    "deviceModel" : "ER605",
13    "deviceName" : "ER605",
14    "deviceType" : "SMBROUTER",
15    "fwId" : "",
16    "fwVer" : "2.2.6 Build 20240718 Rel.82712",
17    "hwId" : "[REDACTED]",
18    "oemId" : "[REDACTED]",
19    "tcspVer" : "1.2"
20  }
21 }
22
23 {
24   "error_code" : 0,
25   "id" : 1,
26   "result" : {
27     "cachedSvr" : "n-euw1-device-omada.tplinkcloud.com:443",
28     "illegalType" : 0,
29     "validTimeOnDevice" : 86400
30   }
31 }

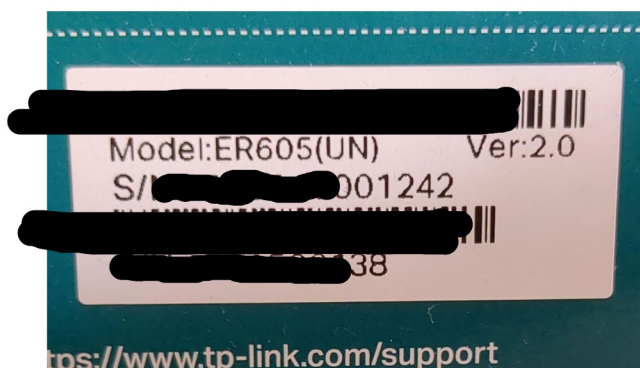
```

Figure 12 – An example of the “helloCloud” message and the corresponding response

FSCT-2025-0003: Predictable Device Serial Numbers Can Be Used for Device Adoption

Devices can be adopted via Omada Cloud using only their serial numbers, provided they have not already been adopted. A device’s serial number is printed on its packaging and on a label attached to the device, and attackers can also exploit FSCT-2025-0011 to identify valid serial numbers.

An attacker who does not possess the device but knows its serial number may be able to adopt it before the legitimate owner because no proof of ownership is required. This could allow the attacker to hijack the device and prevent the legitimate owner from adopting it.



CVE-2025-15544: Device and Site Credentials Are Transmitted Insecurely

Cloud-based and local controllers transmit device and site credentials insecurely to newly provisioned devices. Attackers who can intercept network traffic or spoof a device may gain unauthorized access to multiple devices managed by the same controller.

During the `helloCloud` and device-adoption phases, including message type 1048581, the new site username is transmitted in cleartext, while the password is represented as an unsalted MD5 hash. The following is an example message sent from a controller to a device:

```
{
  "header" : {
    "error" : 0,
    "mac" : "00-DE-AD-BE-EF-00",
    "type" : 1048581,
    "version" : "2.2.0"
  },
  "body" : {
    // omitted for brevity
    "userAccount" : {
      "curPassword" : "21232F297A57A5A743894A0E4A801FC3",
      "curUsername" : "admin",
      "newPassword" : "A601EA74981ADF5A62C23DDA0E0D64CB",
      "newUsername" : "site_admin"
    }
  }
}
```

With the cleartext username and the MD5 hash of the password, attackers can attempt to recover the password using rainbow-table, dictionary, or brute-force attacks and then reuse the credentials to access other devices.

However, recovering the plaintext password may not be necessary. The captured username and password hash contain the information required to authenticate through the Omada protocols. See CVE-2025-9290 for further details. Credentials may differ between sites and devices.

CVE-2025-15627: Use of Hard-Coded Cryptographic Keys in Omada Protocol Version 1

Omada controllers contain a hard-coded private key used to protect communications with Omada client devices when version 1 of the Omada protocol is used. Attackers can retrieve this key and use it to compromise those communications.

Several phases of Omada protocol version 1, referred to below as v1, operate over unencrypted TCP connections.

When communication between a client device and controller enters the Adoption phase, the controller authenticates to the device using message type 16. The authentication value `AUTH` (see CVE-2025-15544) is

encrypted with a private key hard-coded in the controller software.

The device decrypts `AUTH` using the corresponding public key embedded in its firmware, verifies it, and generates an RC4 key to protect subsequent traffic. It then encrypts the RC4 key with the same public key and sends it to the controller in message type 32.

The controller decrypts the RC4 key using the corresponding hard-coded private key and then uses it to encrypt and decrypt subsequent communications. See the [Omada ZTP Protocols](#) section for additional context.

The hard-coded public key can be found in the device firmware library `libecs.so`:

```
-----BEGIN RSA PUBLIC KEY-----
MIGJAoGBAJy11IdwtkMDSV7oQLy/FXmeshgN+t73sjfQItvU7y15Y9s4K+0F5RQL
moBcdRHvG41bQLccIn2EWbzMtcnpjvXvfHjpyE4aTqZzHy0g9i4ypTM7XysCwYnqV
ete9J4KRQJA1seSrX7sGKQH0keUVX9PCEzivH3WIR9EE0rNxoj7HAgMBAAE=
-----END RSA PUBLIC KEY-----
```

The corresponding hard-coded private key is embedded in the controller software package `ecsp-common-1.1.127.jar`:

```
package com.tplink.smb.ecsp.common.util.encrypt;
import java.nio.charset.StandardCharsets;

public class GlobalConfig {

    private static final byte[] ENCRYPTED_NETTY_RSA_PRIVATE_KEY = {
        -38, 0, 52
        // the rest of the key is omitted
    };

    public static final String NETTY_RSA_PRIVATE_KEY = decryptRSAPrivateKey();

    private static String decryptRSAPrivateKey() {
        return new String(TEAUtils.decrypt(ENCRYPTED_NETTY_RSA_PRIVATE_KEY), StandardCharsets.UTF_8);
    }

    private GlobalConfig() {
    }
}
```

The vulnerability arises because Omada protocol v1 relies on the same hard-coded public and private key pair to protect sensitive communications. Attackers who retrieve this key pair can:

- Decrypt RC4 session keys and the communications protected by them.
- Recover `AUTH` values in the form `username + uppercase(md5sum(password))` for offline cracking or pass-the-hash attacks (see CVE-2025-15544).
- Impersonate controllers and perform man-in-the-middle attacks.

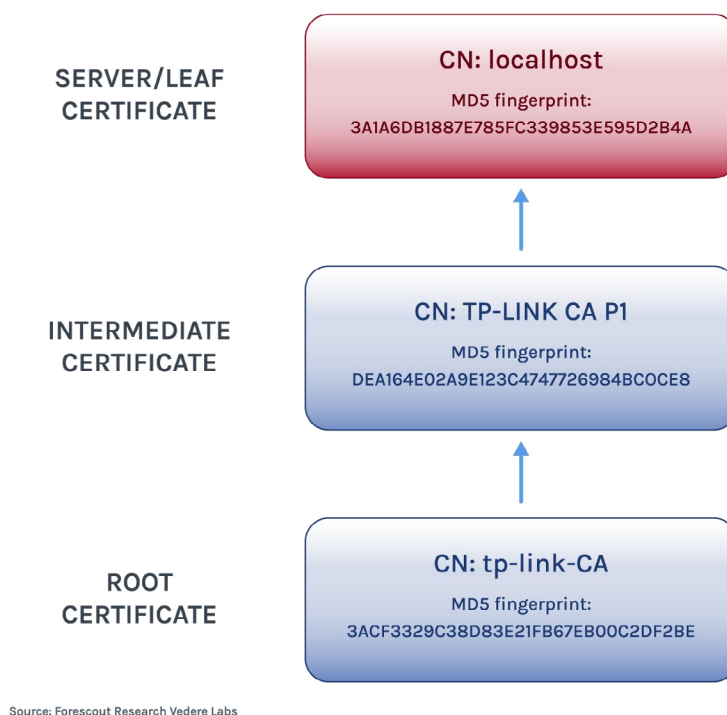
CVE-2025-15628: Compromise of the Chain of Trust

Version 2 of the Omada protocol is vulnerable to man-in-the-middle attacks because it relies on a hard-coded TLS server certificate and corresponding private key. An attacker can use these to impersonate local hardware and software controllers, as well as cloud controllers.

The same issue also affects other TP-Link protocols, including VIGI and Festa, and extends across multiple device families.

Software and hardware controllers include a hard-coded TLS certificate and corresponding private key used to establish the chain of trust with client devices and encrypt communications in version 2 of the Omada protocol.

The following diagram illustrates the chain of trust used for SSL/TLS connections between clients and controllers in version 2 of the Omada protocol:



The root certificate is self-signed and serves as the root certificate authority. It is stored in the trusted keystore of client devices under the filename `2048_newroot.cer` and is used to validate the intermediate certificate.

The intermediate certificate is signed by the root certificate and stored in the trusted keystore of the controllers. It is used to sign and validate server certificates, also known as end-entity certificates.

The server, or leaf, certificate is used by the controller to establish SSL/TLS connections with client devices, prove the controller's identity, and encrypt communications.

The current implementation has several weaknesses:

1. All three certificates are hard-coded. Devices accept SSL/TLS connections from any endpoint that presents a server certificate signed by the intermediate certificate in this chain of trust.
2. For local controllers, client devices verify only that the server certificate presented by a controller is signed by the intermediate certificate, which is itself signed by the root certificate. Because no additional identity check is required, attackers can use the hard-coded `localhost` certificate and corresponding private key to establish SSL/TLS connections with client devices, while impersonating a controller.

The same weakness can also be used to perform man-in-the-middle attacks by positioning a rogue device between clients and controllers.

3. When connecting to cloud controllers, client devices perform an additional certificate check by validating the Common Name (CN). Because the CN of the hard-coded server certificate is `localhost`, this check would normally prevent attackers from using it to impersonate a cloud controller. **However, this validation can be bypassed, as described in CVE-2025-9291.** As a result, the same hard-coded certificate can also be used to impersonate cloud controllers.
4. The same chain of trust is used across multiple TP-Link device families and protocols, including VIGI for IP cameras, Omada, Festa, and the mobile application used for zero-touch provisioning of IP cameras and access points. It may also be used by additional product families.

We also identified a second localhost certificate with the MD5 fingerprint `6F082C38E42BBC5C8B876FC8F66751F1`. Because it is signed by the same root and intermediate certificates, it can be used for the same attacks.

CVE-2025-15629: Insufficient Symmetric Key Entropy in Omada Protocol Version 1

The symmetric RC4 key used to encrypt sensitive communications between clients and controllers in version 1 of the Omada protocol has insufficient entropy and can be predicted by attackers.

See CVE-2025-15627 for additional context on when this RC4 key is generated and used.

The RC4 key is generated using the following algorithm:

```
shasum(timestamp + "@" + reverse(controller_ip) + ":" + controller_port + mac)
```

Here, `timestamp` is the current timestamp generated by the client, and `controller_ip` is the controller's IP address, `controller_port` is the controller port, and `mac` is intended to be the controller's MAC address.

The following example shows these values concatenated before the `shasum()` function is applied:

```
Wed Oct 30 06:29:44 2024@100.0.168.192:29814@00:00:00:00:00:00
```

The controller's MAC address is known to the attacker because, for example, the Discovery phase is unencrypted. If the controller does not provide its actual MAC address, the value is set to `00:00:00:00:00:00`. The attacker also knows the controller's IP address and the port.

The timestamp is therefore the only value unknown to the attacker. Because this value has limited entropy, an attacker can guess the RC4 key and use it to decrypt the protected communications without significant computational effort.

FSCT-2025-0008: Default Credentials

During the initial device-adoption phase, authentication challenges are signed using default credentials. Authentication to the cloud uses a challenge-response mechanism, in which both the device and the cloud exchange and sign random values.

For a new device, the default username `admin`, concatenated with the MD5 hash of the default password `admin`, is used to sign these challenges.

This weakness also enables other attack scenarios, including those described in CVE-2025-15630 and CVE-2025-9290.

CVE-2025-15630: Race Condition During Device Adoption in Omada Cloud

During device adoption through Omada Cloud, an attacker can exploit a race condition by initiating the handshake before the legitimate device while spoofing only its MAC address. This allows the attacker to obtain the device’s initial configuration, which contains administrator credential hashes and potentially other confidential information.

After completing the initial `helloCloud` phase, in which the device informs the cloud that it is online and receives basic information about the nearest cloud service, the device initiates the adoption handshake with the appropriate regional service.

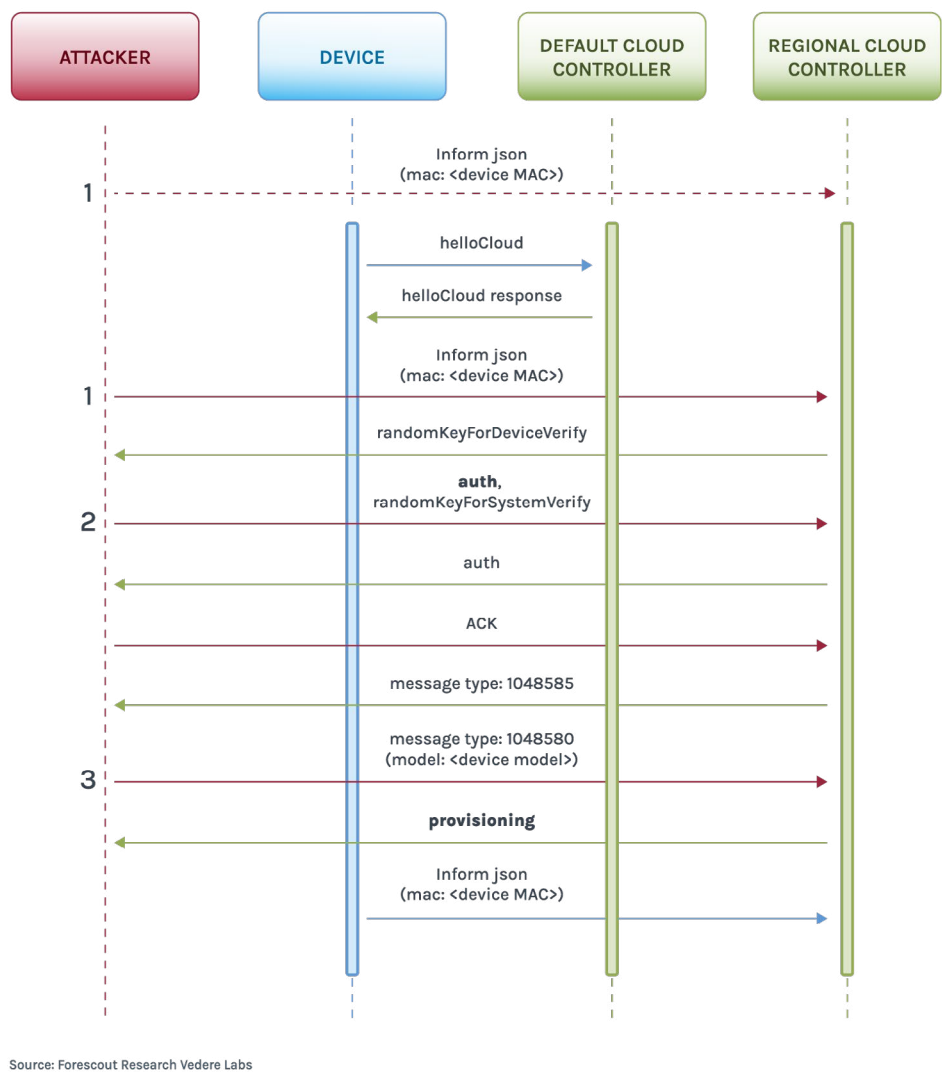


Figure 13 – Sequence diagram of the race condition attack

A time window exists between the initial `helloCloud` phase and the adoption phase that an attacker can exploit, as shown in Figure 13:

1. The attacker initiates the adoption phase before the legitimate device by sending only the target device's MAC address in the initial inform message, type 3. The attacker must send this request to the correct regional cloud service until a challenge is received.
2. The attacker signs the challenge using the default credentials described in FSCT-2025-0008.
3. The attacker then sends message type 1048580 containing the device model and related details, causing the cloud service to return the device's initial configuration.

The device model can also be inferred from the serial number and information available through the cloud API, as described in FSCT-2025-0011.

CVE-2025-9289: Cross-Channel Scripting Against Controllers

The Omada controller web interface is vulnerable to cross-channel scripting because values supplied during the adoption phase of the Omada protocol are not properly sanitized.

A device can inject arbitrary JavaScript when it sends device information to the Omada Cloud or a local controller in message type 1048580. Parameters such as the firmware version can contain JavaScript tags.

A combination of weaknesses allows this code to execute in the administrator's browser. The controller's CSP permits the use of `eval()`, and the version of jQuery used by the web interface can execute scripts passed to the `html()` function.

The Omada controller web interface uses this function to update the page dynamically with information received from devices.

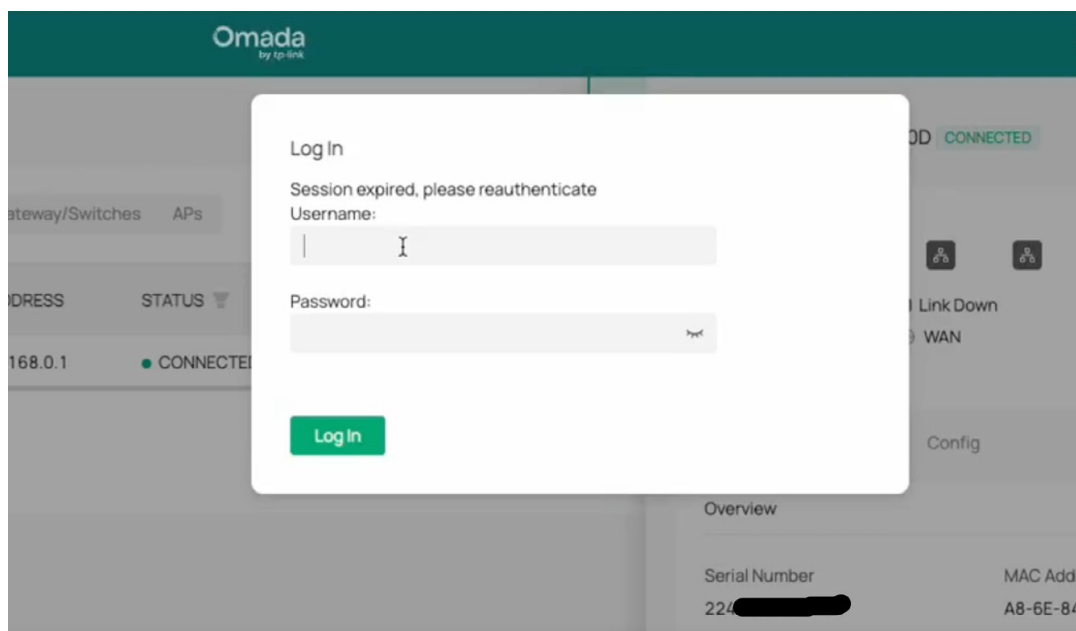


Figure 14 – Fake login pop-up injected by exploiting this vulnerability

FSCT-2025-0011: Device Enumeration

Device serial numbers are sequential and predictable. By guessing valid serial numbers, an attacker can retrieve large numbers of device MAC addresses and related information, such as the device models, through the Omada Cloud web API or user interface.

Device models can also be inferred directly from serial number patterns, for example, serial numbers beginning with 22460J500 appear to correspond to ER605 devices, while those beginning with 224608100 appear to correspond to ER7206 devices.

INDEX	SN CODE	NAME	STATUS	RESULT
1	22[REDACTED]6	A8-6E-84-[REDACTED]	Offline	 Failed to adopt this gateway because a gateway already exists in this site.
2	2[REDACTED]5	A8-6E-84-[REDACTED]	Online	 Failed to adopt this gateway because a gateway already exists in this site.
3	2[REDACTED]4	A8-6E-84-[REDACTED]	Offline	 Failed to adopt this gateway because a gateway already exists in this site.
4	2[REDACTED]7	A8-6E-84-[REDACTED]	Offline	

Figure 15 – Example of sequential serial numbers

CVE-2025-9290: Insecure Transmission of Credentials, Omada Protocol Version 2

Version 2 of the Omada protocol is vulnerable to rainbow-table attacks because the challenge values used when hashing credentials are not required to be unique, random or non-empty.

In version 2 of the Omada protocol, a client authenticates to a controller by sending a double hash of its credentials, combined with a random challenge value. The controller then authenticates to the device in the same way, using a separately generated challenge.

However, neither the client nor the controller verifies that these challenge values are unique or random, and both implementations accept empty values. An attacker can therefore supply repeated or empty challenge values, causing the same static credential hashes to be exchanged. The attacker can collect these hashes for offline cracking and compare them against rainbow tables generated for common credentials.

The protocol does not ensure that either `randomKeyForDeviceVerify` or `randomKeyForSystemVerify` is unique and randomly generated for each request. It also does not verify that either value is non-empty. See the [Omada ZTP Protocols](#) section for further context.

For example:

```
randomKeyForDeviceVerify = ""

AUTH = sha256sum(
    sha256sum(username + md5sum(password))
    + randomKeyForDeviceVerify
)

AUTH = sha256sum(
```

```
    sha256sum(username + md5sum(password))
)
```

When the same challenge value is used, the resulting AUTH value will also be the same for a given username and password.

Unlike version 1 of the protocol, the device authenticates first using the `randomKeyForDeviceVerify` value provided by the controller. A malicious controller can set this value to an empty string and observe the resulting hash.

Attackers can then compare collected hashes against precomputed values for common username and password combinations, such as `admin/admin` and `admin/password1234`, or use them for other forms of offline password cracking.

CVE-2025-9291: Insufficient Certificate Checks in Omada Protocol Version 2

Omada client devices have insufficient Common Name (CN) validation, which attackers can exploit to impersonate a cloud controller.

The `ecs_verifySsl()` function in the `omadad` daemon verifies the certificates used for SSL connections in version 2 of the Omada protocol. It validates the certificate chain, starting from the root certificate embedded in the client, followed by the intermediate certificate embedded in the controller, and finally the server certificate.

The pseudocode below shows some of the checks performed on the server certificate, including validation of the Common Name (CN) beginning at line 10:

```
int64 ecs_verifySsl(unsigned int preverify_ok, void* store_ctx) {
    // ...
01:   if ( !error_depth )
02:   {
03:       subject_name_str = strstr(subject_name_buf, "/CN=");
04:       if ( !subject_name_str )
05:           return 0;
06:       cert_subject_name = subject_name_str + 4;
07:       v11 = strchr(subject_name_str + 4, '/');
08:       if ( v11 )
09:           *v11 = 0;
10:       if ( strstr(global_controller_host, "tplinkcloud.com") && !strstr(cert_subject_name, "tplinkcloud.com") )
11:       {
12:           if ( HIDWORD(qword_5CCB8) )
13:               printf(
14:                   "[ECS][ERROR]%s():%5d @ verify error:CN mismatch(%s), controllerUrl(%s).\n\r",
15:                   "_ecs_verifySsl",
16:                   125LL,
17:                   cert_subject_name,
18:                   global_controller_host);
19:           if ( qword_5CCB8 )
20:           {
21:               // ...
22:               ecs_log(2LL, "[ECS][ERROR]<%=s>%s():%5d @ verify error:CN mismatch(%s), controllerUrl(%s).\n\r");
23:               // ...
24:           }
25:           return 0;
26:       }
27:   }
    // ...
}
```

The check at line 10 works as follows. The controller host name `global_controller_host` is obtained from the `helloCloud` message or through the Discovery phase. If hostname is a subdomain of `tplinkcloud.com`, the Common Name (CN) of the leaf certificate must also contain the string `tplinkcloud.com`.

Because attackers do not possess the private keys associated with the root or intermediate certificates, they cannot generate a new server certificate containing this string in its CN. Under normal circumstances, this check prevents them from impersonating a cloud controller.

However, attackers can retrieve a `localhost` server certificate and corresponding private key from a software or hardware controller and use them to spoof the `helloCloud` response. They can then replace the cloud controller hostname with its IP address, causing the check at line 10 to be skipped.

The client consequently accepts the `localhost` certificate for the SSL/TLS connection, allowing the attacker to position themselves between the client device and the legitimate cloud controller.

FSCT-2025-0014: Unrestricted File Upload to Omada Controllers

Both Omada Cloud and local Omada controllers allow unrestricted file uploads. Uploaded files can later be accessed through unauthenticated links.

Omada Cloud and local controllers allow administrators to update client devices by uploading firmware images through the web interface. When an administrator uploads a firmware update, the controller generates a temporary link that the client device uses to download the file through the Omada protocol. This may be a local URL such as

```
https://192.168.0.2:443/upgrade/files/[omada _ id]/[md5sum _ 2]
```

or a link to an AWS S3 bucket.

This functionality has the following flaws:

1. The controller checks the integrity of the firmware image and rejects invalid files. However, arbitrary files are still uploaded to the controller or S3 bucket and remain accessible through a temporary link.
2. The temporary download links do not require authentication. Anyone who knows the URL can retrieve the uploaded files from the controller or S3 bucket.

Because controllers and associated S3 buckets are trusted sources and may be exempt from firewall restrictions, attackers can exploit these flaws to upload malicious files through the web interface and deliver them into internal networks.

CVE-2025-15631: Password Hashes Encrypted with a Hard-Coded Key

The device administration credentials stored in `/etc/config/accountmgt` are hashed using unsalted MD5 and then encrypted with AES-256 in CBC mode through OpenSSL. The encryption uses the hard-coded key `who are you?`, which is defined in `accountmgt.lua`.

CVE-2025-9292: CORS Bypass

The Content Security Policy (CSP) used by the Omada cloud-based controller allows connections to `*.cloudfront.net` and `*.amazonaws.com`. An attacker can exploit these permitted destinations to bypass CORS restrictions and exfiltrate data through an XSS attack.

The CSP directive `connect-src` allows HTTPS connections to these domains. For example, an attacker can deploy a malicious web server on an AWS EC2 instance and place it behind a CloudFront distribution with a valid and reachable hostname such as:

```
<random string>.cloudfront.net.
```

Because this hostname is permitted by the controller's CSP, injected JavaScript can send sensitive information to the attacker-controlled CloudFront endpoint.

```
██████████ 46.76 - - [12/Jun/2025 10:16:01] "OPTIONS /admin/HelloFrancesco HTTP/1.1" 200 -
Received GET request:
Path: /admin/HelloFrancesco
Headers:
Host: ██████████.cloudfront.net
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:136.0) Gecko/20100101 Firefox/136.0
X-Amz-Cf-Id: QCj1Gaf5TdLi35m9GCxHF7WJx112Aap82yAeiGKuEW6GrxdCqHKmZA==
Connection: Keep-Alive
Accept-Language: en-US,en;q=0.5
Accept: */*
Referer: https://euw1-omada-cloud.tplinkcloud.com/
Te: trailers
X-Forwarded-For: ██████████ 126.59
Via: 2.0 ██████████ fa.cloudfront.net (CloudFront)
Accept-Encoding: br,gzip
x-requested-with: XMLHttpRequest
refresh: manual
origin: https://euw1-omada-cloud.tplinkcloud.com
sec-fetch-dest: empty
sec-fetch-mode: cors
sec-fetch-site: cross-site
priority: u=0
```

```
██████████ 84.94 - - [12/Jun/2025 10:16:01] "GET /admin/HelloFrancesco HTTP/1.1" 200 -
```

Figure 16 – Example of exfiltrated credentials

Content-Security-Policy

```
default-src 'self' https://*.tplinkcloud.com/; script-src 'self' 'unsafe-eval' https://www.paypal.com/ https://www.paypalobjects.com/ https://js.stripe.com 'sha256-7W9UIBaYGIOPt1aQBLegqffUvHbYq6/ZAB+ErjUb40=' 'sha256-VGQ8jNTL2g0e8wPwOgyCQJDqhuRgfV7gRYexcBkBe4Y=' 'sha256-+9dQLByJOrMH7ojZkdfnL0p0S0pZqKwTih7vSa+FMg=' 'sha256-x2jgB1zBLi30IsfY+VNgWjwBGeHPJxOSrzi+ldst6k0=' 'sha256-0AHZXO4clnpdcxqdmASPBEP4JCirtaxIX/mUuL1kzZw=' 'sha256-IfXIPY3+MCPOPb4mrw1Y961+745U3WIDQVcOXdcchSQc=' 'sha256-Vohl58KrXs+2LMTip+0SXk2/JwaZNxGj9hadiPKc2E='; style-src 'self' 'unsafe-inline'; connect-src 'self' https://*.cloudfront.net/ https://*.tplinkcloud.com/ https://*.tplinkcloud.com:8843/ http://*.tplinkcloud.com:8088/ wss://*.tplinknbu.com/ ws://localhost/ wss://*.tplinkcloud.com/ https://*.paypal.com/ https://api.stripe.com https://*.tiles.mapbox.com https://api.mapbox.com/ https://events.mapbox.com https://*.amazonaws.com/ data; frame-src 'self' OmadaSightWebPlayer: vigiwebplayer: https://*.tplinkcloud.com/ https://js.stripe.com https://hooks.stripe.com https://*.paypal.com/ data: blob; img-src 'self' https://*.cloudfront.net/ https://*.tplinkcloud.com/ https://*.amazonaws.com/ https://*.mzstatic.com/ https://play-lh.googleusercontent.com/ https://kcart.alipay.com/ data: blob; child-src blob; worker-src https://*.tplinkcloud.com blob; media-src https://*.tplinkcloud.com/ https://*.cloudfront.net/ https://*.amazonaws.com/ blob; object-src 'self' data: blob;
```

Figure 17 – Vulnerable CSP

CVE-2025-9293: Improper Certificate Validation

TP-Link mobile applications perform insufficient Common Name (CN) validation, which attackers can exploit to conduct man-in-the-middle attacks. The same root and intermediate certificates are used across Omada, Festa, Tapo, Kasa, and other TP-Link ecosystems.

Attackers can retrieve a `localhost` server certificate and corresponding private key from an Omada software or hardware controller, as described in CVE-2025-15628, and use them to impersonate an API back-end server in a man-in-the-middle attack.

© 2026 Forescout Technologies, Inc. All rights reserved. Forescout Technologies, Inc. is a Delaware corporation. A list of our trademarks and patents is available at www.forescout.com/company/legal/intellectual-property-patents-trademarks. Other brands, products or service names may be trademarks or service marks of their respective owners.