Cloudy With a Chance of Exploits: Assessing Risks in Cloud-Managed IIoT

Roni Gavrilov, Security Researcher / OTORIO

Executive summary

1. Industrial IoT (IIoT) devices are at risk of attacks through cloud-based management platforms provided by the vendors.
2. IIoT devices can be exposed even without being actively configured for cloud use.
3. Breaching IIoT devices can bypass security layers in common Purdue architectures, bridging internal OT networks with the WAN.
4. OTORIO discovered 11 vulnerabilities in cloud-based platforms of three major Industrial Cellular Routers vendors, affecting thousands of industrial environments, and allowing for remote code execution.
5. Vulnerabilities were responsibly disclosed and mitigated by the vendors.
6. Due to the nature of these devices, it is probable that additional vulnerabilities still exist, making IIoT a critical point of failure that needs to be addressed.
7. Users are strongly advised to proactively implement the recommended measures outlined at the end of this document to mitigate potential risks.

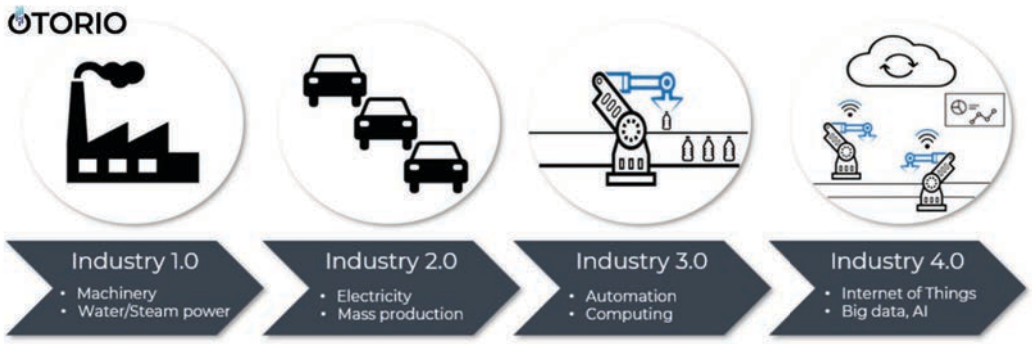
OTORIO

Table of contents

Introduction

Industry 4.0 has revolutionized the industrial landscape, encompassing the integration of advanced technologies into manufacturing and production processes. It involves the convergence of artificial intelligence, robotics, big data analytics, and the Industrial Internet of Things (IoT).

The Industrial IoT plays a pivotal role in enabling real-time connectivity and communication between systems, leading to enhanced productivity and improved efficiency. Industrial IoT devices, such as sensors, meters, controllers, and network devices, are instrumental components within the Industry 4.0 landscape.



Industrial cellular routers and gateways have become one of the most prevalent components in the Industrial IoT landscape. They offer extensive connectivity features and can be seamlessly integrated into existing environments and solutions with minimal modifications.

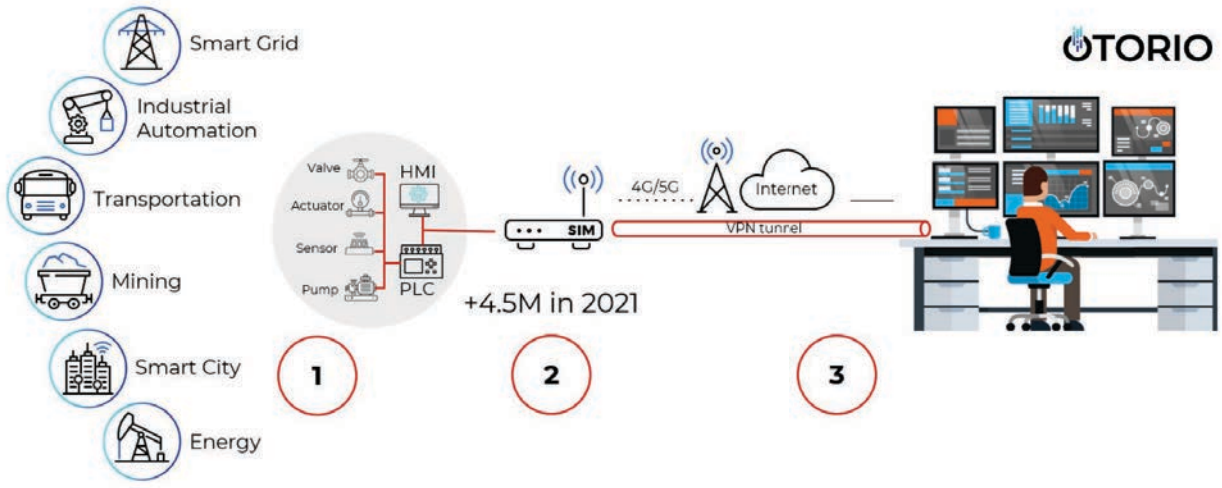
Industrial Cellular routers and gateways

Industrial cellular routers and gateways are essential IIoT devices that provide connectivity for industrial applications, facilitating remote monitoring, control, and data exchange across various industries. These devices are commonly deployed in remote sites or integrated into moving machines, featuring rugged designs to withstand challenging industrial environments.

Industrial cellular routers and gateways commonly offer support for various industrial protocols such as Modbus and DNP3. They also come equipped with a range of functional and security features, including Wi-Fi capabilities, VPN (Virtual Private Network) support, firewall protection, and cloud-based remote management.

In our research, we specifically focused on industrial cellular gateways and their cloud management platforms for three key reasons:

1. These gateways are widely deployed in critical infrastructure sectors, such as substations, water utilities, oil fields, and pipelines, serving as a remote connectivity solution.
2. They enjoy significant popularity, with over 4.5 million cellular gateways shipped globally in 2021 alone*.
3. Successful exploitation of vulnerabilities in these gateways could have severe consequences. Their connection to vulnerable OT (Operational Technology) devices directly impacts availability and process safety. Moreover, the VPN connectivity they provide opens the possibility of propagation to control centers and SCADA (Supervisory Control and Data Acquisition) servers, potentially impacting multiple sites.



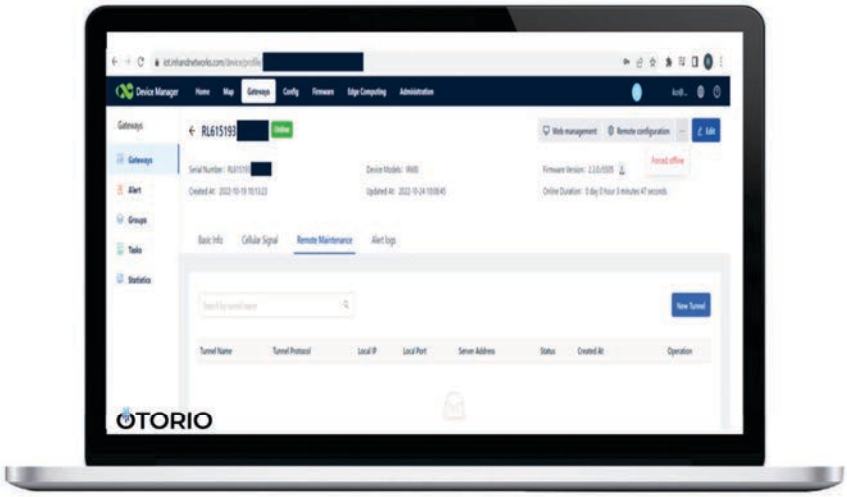
Industrial cellular routers commonly accessible to sensitive networks in a variety of industrial sectors

** https://www.berginsight.com/the-cellular-iot-gateway-market-reached-us-115-billion-in-2021*

Cloud-based management platforms

The remote cloud management of IIoT devices through cloud-based management platforms immediately captured our interest. These platforms, offered as a service by vendors (often as paid services), enable centralized and remote management of devices.

Clients can register their devices under their cloud accounts, facilitating remote monitoring, alerting, and statistical analysis. Moreover, essential operations such as firmware upgrades and reboots can be performed remotely, eliminating the need for physical visits to each device location.



InHand Networks 'Device Manager' cloud platform

The convenience and efficiency offered by cloud-based management platforms for IIoT devices are undeniable. However, it is crucial to recognize that they also introduce potential risks to industrial environments.

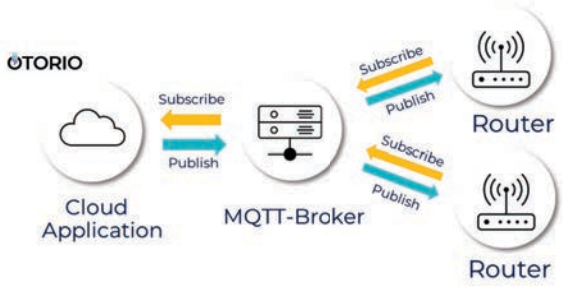
Industrial IoT cloud-based management risks

Many vendors offer IIoT solutions, including industrial cellular routers, along with their own cloud-based management platforms for remote device management. However, this common architecture introduces a significant risk. Targeting a single vendor platform could potentially put thousands of devices and industrial networks at risk simultaneously.

To gain a deeper understanding of the potential impact, complexity, and feasibility of such a scenario, we conducted research on three leading vendors. The results revealed concerning findings, including unexpected risk to devices that were not even actively configured to utilize cloud features.

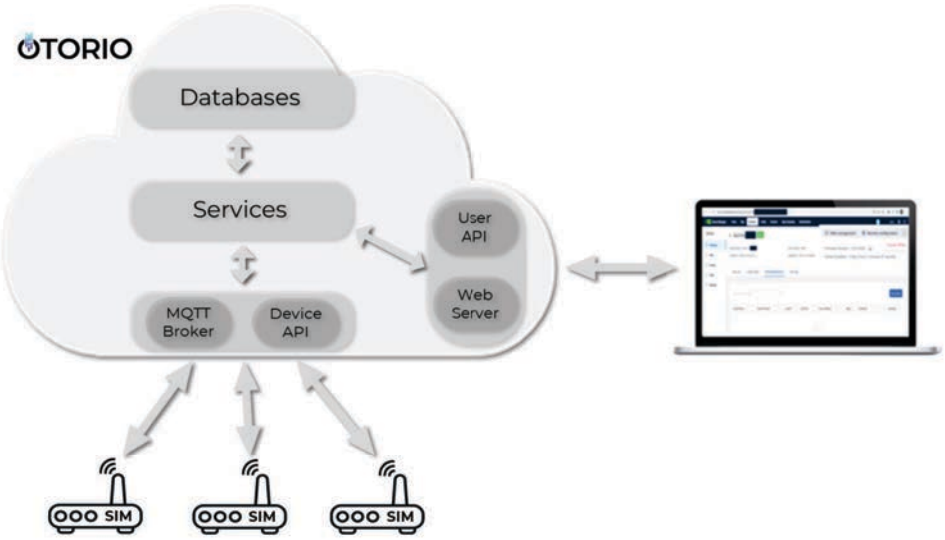
We specifically targeted cloud-to-IIoT device remote code execution attacks by assessing device connectivity to the cloud and exposed interfaces, aiming to reach internal OT infrastructures remotely.

The typical connectivity to these platforms relies on machine-to-machine (M2M) protocols like MQTT for device-cloud communication, coupled with web interfaces for user management. MQTT operates on a publish-subscribe model, where the broker manages topics and devices can subscribe to receive published information.



MQTT communication of industrial cellular routers

Additionally, a specialized device API is commonly used for initialization communication with the cloud platform, along with user API and web interface for management of the devices.



Cloud management platforms connectivity

Upon analyzing our findings, we identified critical issues in three key areas: the asset registration process, security configurations, and external APIs & web interfaces. These findings highlight the risk posed to IIoT devices and the need for safer deployment and hardening of their management platforms.

Asset registration process

The asset registration process serves as the initial stage for remotely managing cloud-connected IIoT devices. To facilitate remote device management via the cloud, device owners are required to register their devices on the cloud platform, linking a specific device to their cloud platform account.

In our analysis of device registration processes in various industrial IoT cloud management solutions, we have observed a concerning trend. Typically, the registration of a cloud-connected device to a user's account requires just two, often

‘non-secret’ identifiers, commonly found on the device itself, such as the MAC address, serial number, or IMEI.

Identifiers required for device registration

Devices are often preconfigured to connect to the cloud server by default, eliminating the need for additional actions or device approval to register it with a specific account. This default configuration poses a vulnerability, **allowing devices to be taken over even when users have not actively configured any cloud features.**

Once an adversary successfully registers an IIoT device, they gain comprehensive control over it, leveraging the cloud platform's features to execute commands and access VPN capabilities.

Unregistered device takeover

The process of taking over unregistered devices typically involves three steps:

1. Collecting the necessary identifiers (such as serial number, MAC address, or IMEI)

2. Registering the device to the attacker's account through the standard registration process
3. Utilizing cloud features to execute commands on the compromised device remotely.

During our investigation, we identified several methods by which the required information could be obtained and subsequently used for remote code execution.

Collection of identifiers over the internet

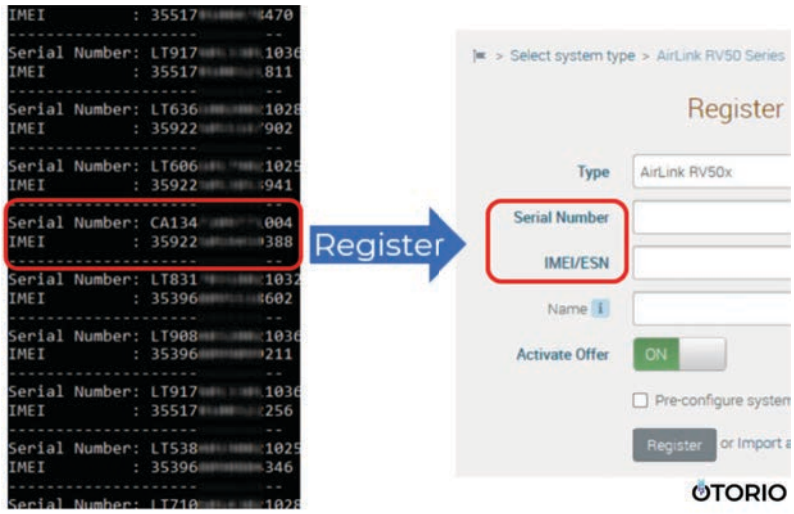
Attackers can acquire the identifiers used for registration through various means over the internet. Here are a few examples of how this can be accomplished:

1. Exposed SNMP service (shodan.io)

By utilizing the search engine such as Shodan.io and leveraging its API, we developed a Python script capable of searching for Sierra Wireless industrial cellular routers and gateways with exposed SNMP services on the internet.

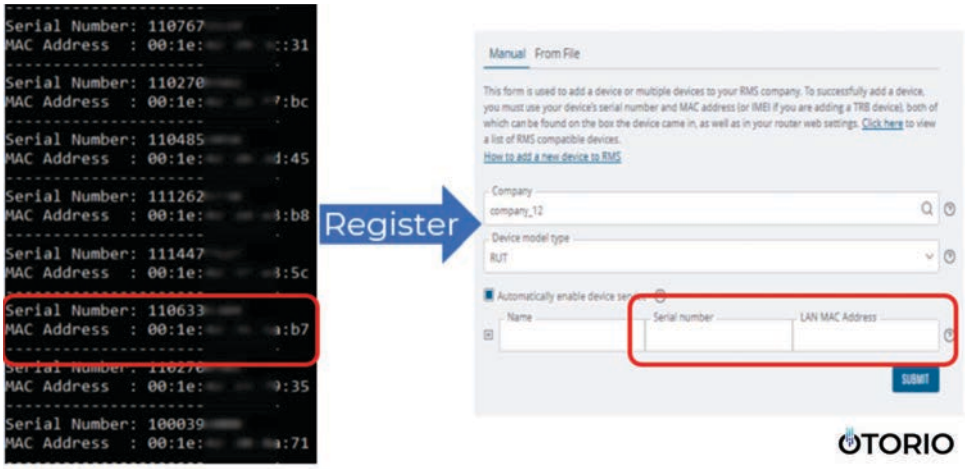
```
queries = ["RV50 port:161",
           "RV55 port:161"]
api = Shodan('[redacted]')

for query in queries:
    page = 1
    while True:
        ans = api.search(query=query, page=page)
        total = ans['total']
        print("Number of results: " + str(total))
        results = ans['matches']
        for result in results:
            try:
                ip_address = result['ip_str']
                query_res = get(ip_address,
                        ['1.3.6.1.4.1.20542.9.1.1.1.1154.0',
                         '1.3.6.1.4.1.20542.9.1.1.2.10.0',
                         '1.3.6.1.4.1.20542.9.1.1.6.5026.0'],
                         hlapi.CommunityData('public'))
                serial = query_res.get('1.3.6.1.4.1.20542.9.1.1.1.1154.0', None)
                imei = query_res.get('1.3.6.1.4.1.20542.9.1.1.2.10.0', None)
                print("----------------------------")
                print("Serial Number: {}".format(serial))
                print("IMEI         : {}".format(imei))
            except Exception as e:
                pass
        if len(results) == 100:
            page += 1
        else:
            break
```


Using our script, we were able to find relevant IP addresses and extract the serial number and IMEI identifiers using SNMP. By querying the serial number and IMEI object identifiers in SNMP, we accessed the registration identifiers for thousands of devices on Sierra Wireless AirVantage cloud-based management platform.



Using shodan.io and SNMP queries for registration of Sierra Wireless IIoT devices

Similar to our approach with Sierra Wireless devices, we applied a comparable method to Teltonika industrial cellular routers and gateways. By querying the SNMP and focusing on the serial number and MAC address identifiers, we uncovered thousands of serial number and MAC address pairs associated with Teltonika Networks IIoT devices that could be leveraged for registration within the RMS cloud-based management platform.



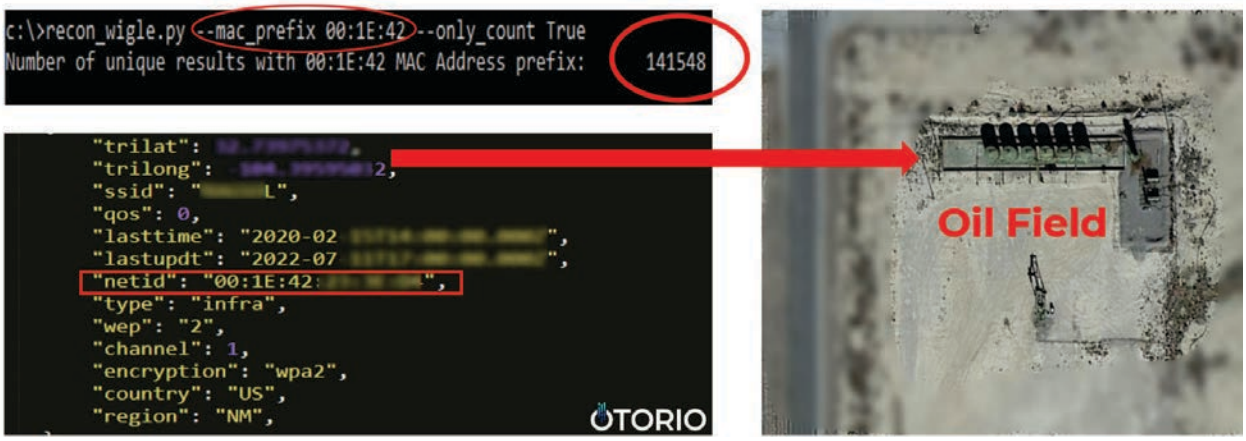
Same issue affects Teltonika Networks cloud platform

2. Publicly available wireless identifiers (WiGLE)

WiGLE™ (wigle.net) is an open and freely accessible platform that houses an extensive database of Wi-Fi network information worldwide, with over one billion unique records. The platform offers both an API and a web interface, providing users with various filtering capabilities.

Basic filtering options include network name (SSID) or network address (BSSID), while advanced filtering, accessible through the REST API, offers additional filtering options such as location, encryption type, and MAC prefix.

Given that Wi-Fi is a prevalent feature in industrial cellular routers and gateways, we leveraged the WiGLE API to filter results based on the MAC prefix of vendors like Teltonika. This search produced over 141,000 results, including notable findings such as oil fields.

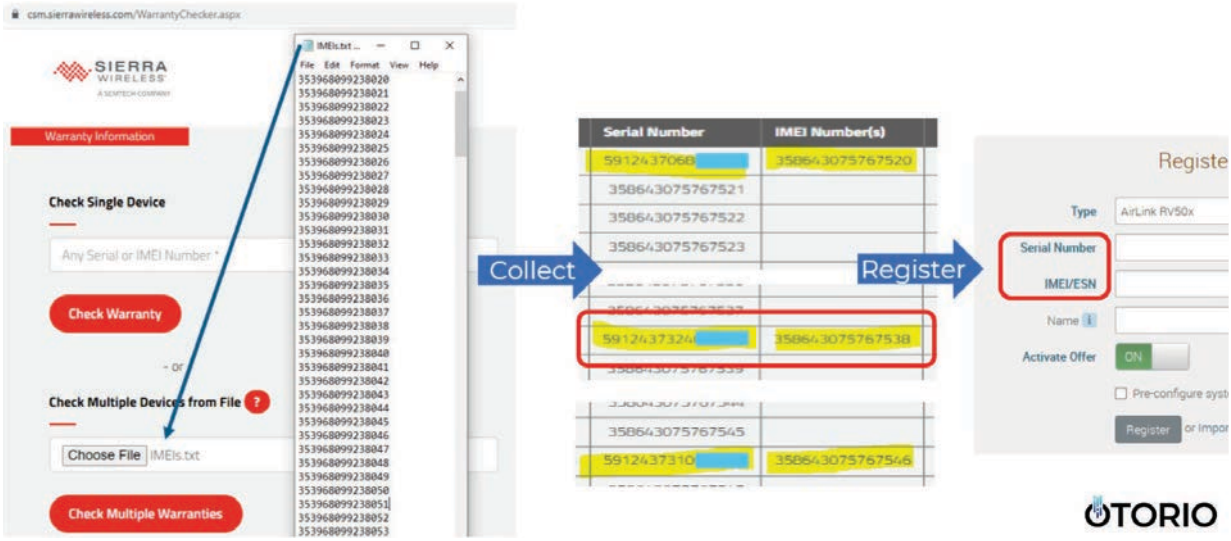


Reconnaissance of sensitive facilities over WiGLE API

With the knowledge that Teltonika Networks RMS cloud-based management platform utilizes serial number and MAC address pairs for device registration, an attacker who possesses the MAC address can employ brute-force techniques to determine the corresponding serial number and subsequently register the device to their own account.

3. Information exposure by vendor

The Sierra Wireless warranty check website provides a functionality to verify the warranty status of routers by entering either the serial number or IMEI. However, a significant issue exists within this system, as inputting a valid identifier inadvertently exposes the corresponding counterpart. This flaw was exploited by utilizing the website's capability to upload a file containing a list of identifiers for batch checking. By uploading a list of IMEIs in ascending order, starting with our lab router's IMEI, the corresponding serial numbers were obtained for the valid IMEIs present in the list.



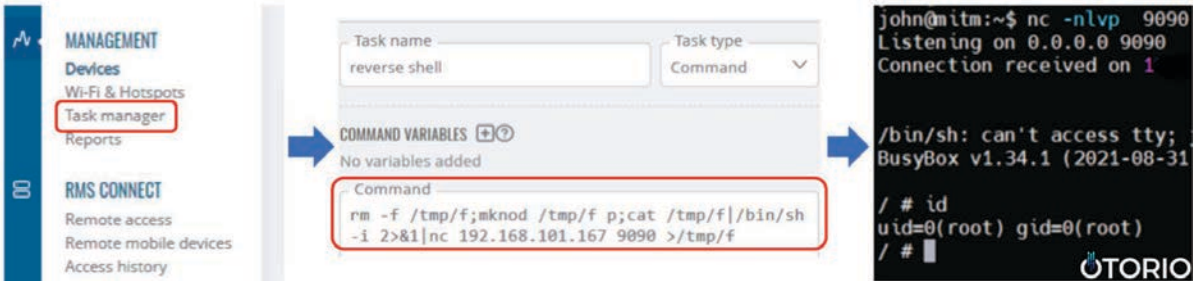
Mass registration of devices based on information disclosure issue

By employing this technique, we successfully acquired 'IMEI-serial number' pairs for nearly all Sierra Wireless industrial cellular routers. Given that these precise identifiers are the required identifiers for device registration on the Sierra Wireless "AirVantage" cloud platform, an attacker could potentially seize control of all unregistered routers connected to the cloud. We responsibly disclosed this security vulnerability (CVE-2023-31280) to the Sierra Wireless security team, who promptly addressed and resolved the issue.

Device takeover to remote-code-execution (RCE)

Ultimately, an attacker who manages to take over a device will likely aim to achieve remote code execution (RCE) on the device. This would grant the attacker full control of the device and enable them to spread throughout the network to which the device is connected.

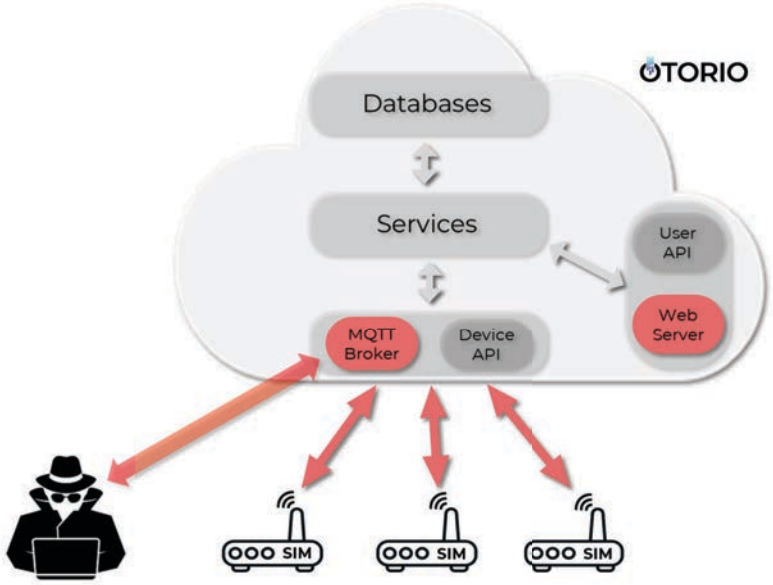
To better understand how an attacker could translate a device takeover into RCE, let's take a closer look at Teltonika Networks RMS platform. The platform includes a 'Task Manager' feature that allows users to create tasks and execute commands on remote devices. An attacker could exploit this feature to create a 'reverse shell' and execute it on the devices he took over.



Remote code execution using "Task Manager" feature

Security configurations

Expanding our analysis to identify potential vulnerabilities in various cloud components that could jeopardize the security of the devices, we've inspected the web interface and MQTT broker, paying close attention to any weak security configurations that could be exploited by users or attackers.



We identified configuration issues in both the Teltonika Networks cloud platform web server and the InHand Networks MQTT broker. To highlight the implications of weak security configurations, we will focus on the specific cloud to device remote code execution use case, based on our findings in the InHand Networks 'Device Manager' cloud platform. Further details regarding the Teltonika Networks platform will be discussed later.

Offered by one of the prominent vendors of industrial cellular gateways, 'Device Manager' cloud-based management platform is used for managing InHand Networks InRouter family of devices. A few misconfigurations on the MQTT broker, along with another exploit in the InRouter firmware and a weak registration mechanism, were able to achieve remote code execution on all cloud-managed routers directly from the cloud.

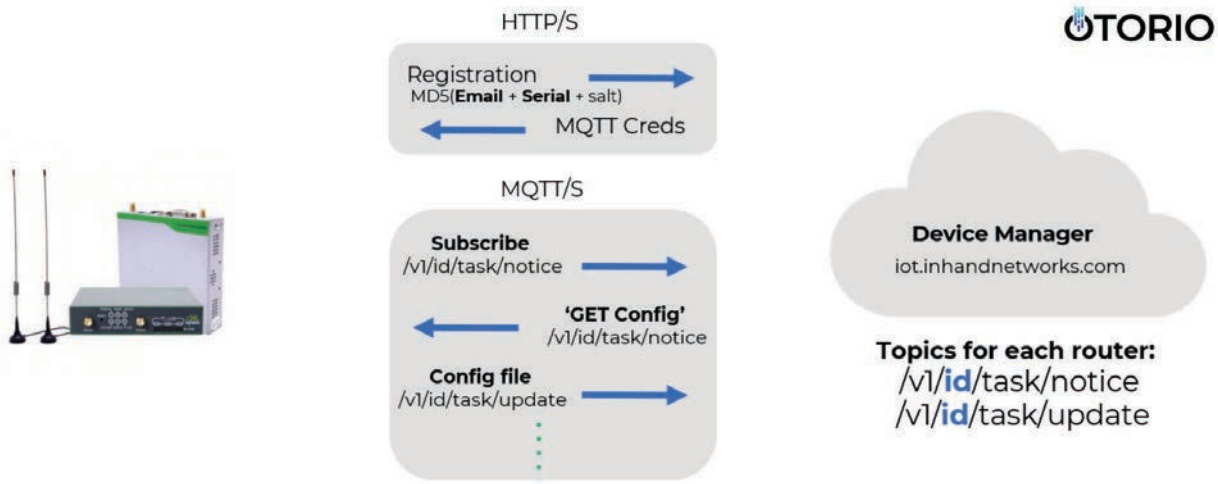
InHand Networks “Device Manager” platform

During the router's boot process, a weak registration occurs with the "Device Manager" platform, where the router's serial number is sent along a "Device Manager" cloud account. If both the account and serial number are valid, the "Device Manager" assigns the device to the account and sends MQTT credentials. The device then establishes an immediate connection to the "Device Manager" using these MQTT credentials.

The "Device Manager" MQTT Broker maintains two primary MQTT topics for each cloud-connected router:

1. *v1/**id**/task/notice* - for receiving tasks from the "Device Manager"
2. *v1/**id**/task/update* - for publishing task results

The 'id' in the topic name (also named ‘client_id’) is a 24-character hexadecimal string that distinguishes between the topic names of different routers and changes for each new session.



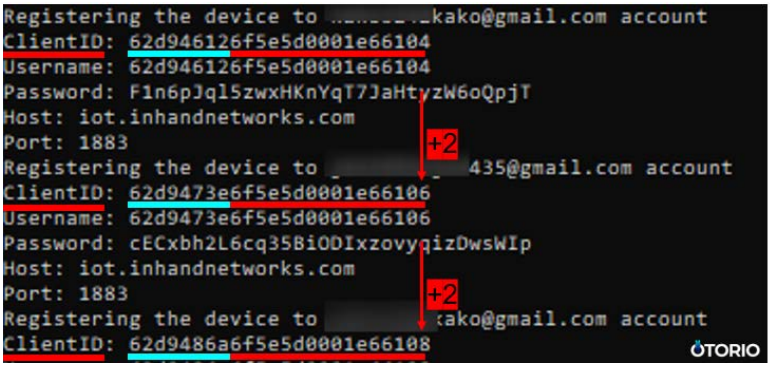
InRouter device connection to “Device Manager” cloud

Upon connecting with MQTT, the "Device Manager" promptly sends the 'Get Config' task, and the devices respond by publishing their configuration files.

Vulnerabilities in “Device Manager” platform

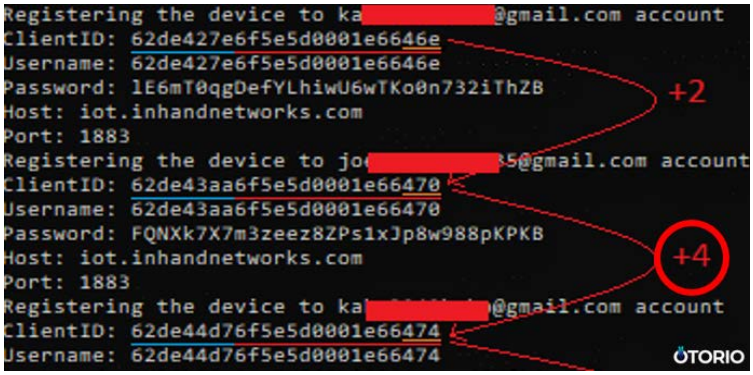
Use of Insufficiently Random Values (CVE-2023-22601)

To delve deeper into the registration process, we implemented a Python script to perform repeated registrations at regular intervals. Through data analysis, we uncovered an interesting pattern: the first 8 hexadecimal characters in the registration data corresponded to the timestamp of each registration, while the second part of the data incremented by 2 with each new registration.



ClientID incrementation

Next, we conducted additional registrations with a 5-minute interval between each attempt. During this analysis, we observed that the second part of the device's ID increased by 4 between the second and third registration. This finding indicated the presence of a real device connected to the cloud. With a time window of 300 seconds (5 minutes) for the first part of the device's ID and the known value of 472 for the second part, we narrowed down the possibilities, making IDs for other devices easily predictable.



Double incrementation - additional device connected to the cloud

Improper access control (CVE-2023-22600)

After attempting to subscribe to all 300 available options for the 'v1/id/task/update' MQTT topic name, which the router utilizes to publish task results, we managed to hit the correct id after 175 attempts. This enabled us to access and retrieve the device's configuration file. Encouraged by this breakthrough, we further validated our capabilities by successfully modifying the hostname.

175 attempts leading to successful configuration leak and modification

OS command injection (CVE-2023-22598)

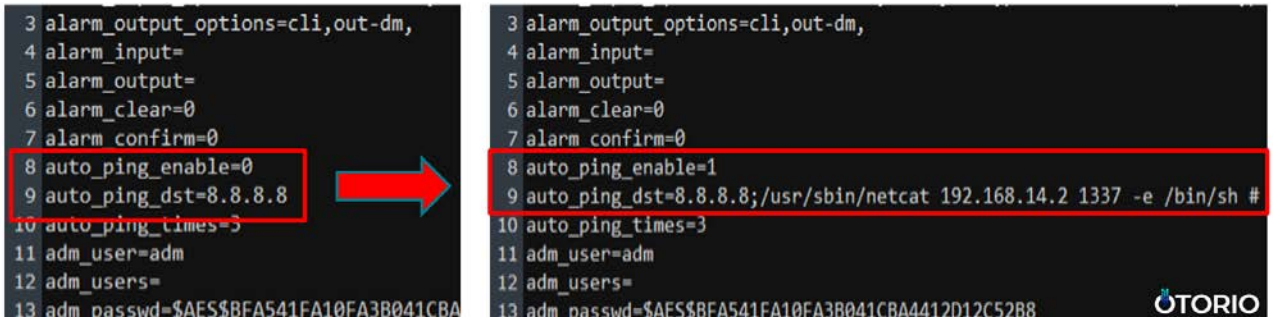
To achieve 'remote code execution,' we proceeded with an in-depth exploration of the InRouter firmware, specifically focusing on the functions responsible for parsing the device configurations. After extracting and thoroughly examining the firmware, we made a significant discovery. Within the code, we identified a critical 'command injection' vulnerability. This flaw resided in the function associated with handling the 'auto_ping' parameter of the device configuration.

Upon closer inspection, we found that when the 'auto_ping_enable' parameter is set to '1', the ping_action_start function is triggered. This function concatenates the string from the 'auto_ping_dst' parameter with a ping command, enabling the device to perform internet connectivity checks.

```
void ping_action_start(void)

{

  [....]
  pcVar1 = (char *)nvram_default_get("auto_ping_dst","8.8.8.8");                    (1)
  strncpy(acStack280,pcVar1,0x80);                                                  (2)
  [....]
  snprintf(command_line,0x80,"echo \"ping-host=%s\r\" > %s",acStack280,"/tmp/ping_result.txt");
  system(command_line);
  snprintf(command_line,0x80,"echo \"ping-size=%d\r\" >> %s",iVar3,"/tmp/ping_result.txt");   (3)
  system(command_line);
  snprintf(command_line,0x80,"ping -c %d -s %d %s >> %s",iVar2,iVar3,acStack280,"/tmp/ping_result.txt");   (4)
  system(command_line);
  return;
```

Vulnerable ping_action_start function

Exploiting the lack of IP address input validation in the 'auto_ping_dst' parameter, we appended a 'reverse shell' command, leading to remote code execution with root privileges. This exploit involved modifying the relevant parameters and transmitting the updated configuration using the 'SET config' task.



Exploitation of OS command injection

Exploiting routers managed over “Device Manager” platform

By chaining these vulnerabilities together, we were able to target a specific device using its serial number or randomly target devices connected to the cloud. However, to ensure the safety of real environments, our focus was solely on demonstrating an attack against our own device, exclusively targeting its specific serial number.

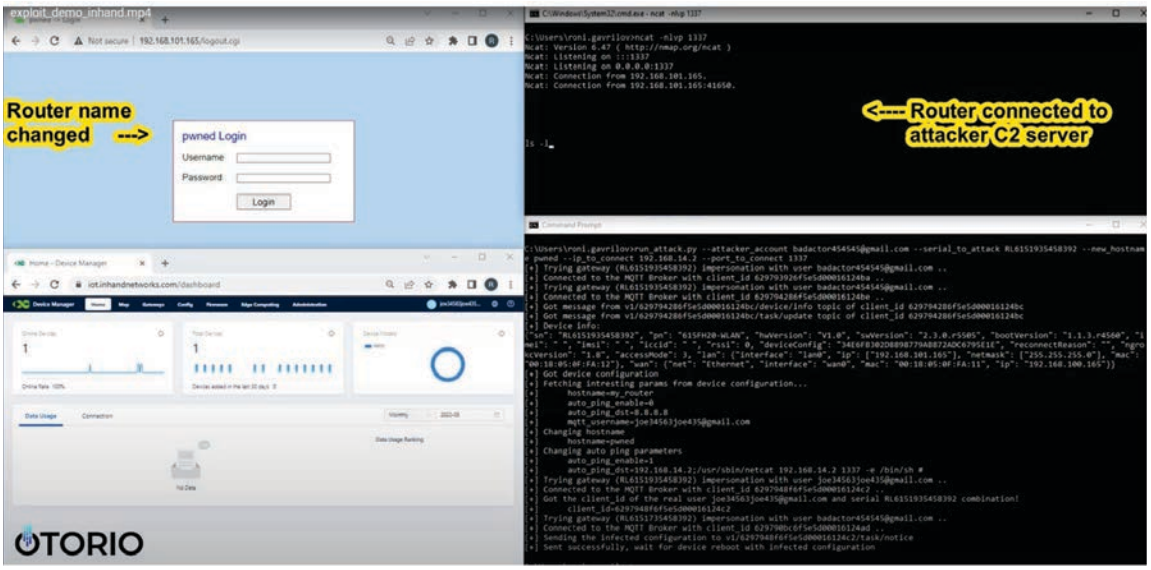
Taking advantage of the flawed registration mechanism, we developed a script that impersonates our router and initiates registration with the "Device Manager" using the router's identifiers. Throughout this process, we closely monitored the response

from the "Device Manager," with particular attention to analyzing the second part of the 'id' parameter.

As a result of this operation, the original router is automatically re-registered, leading to an increase of more than 2 in the second part of the 'id' parameter compared to the previous registration. This indicates the presence of another device that has successfully registered with the cloud.

Subsequently, our script systematically subscribes to all 300 possible topics for the re-registered device until it identifies the correct topic. Once the correct topic is located, the "Device Manager" publishes the most recent message sent by the genuine router to that specific topic, which contains the configuration file. To ensure the device matches our expectations, we perform a verification step.

Upon verification, we modify the 'auto_ping' configuration parameter by adding the 'reverse shell' command, and then publish the updated configuration file back to the router using the 'SET config' task. Compromised device automatically reboots and initiates a remote shell with root privileges.

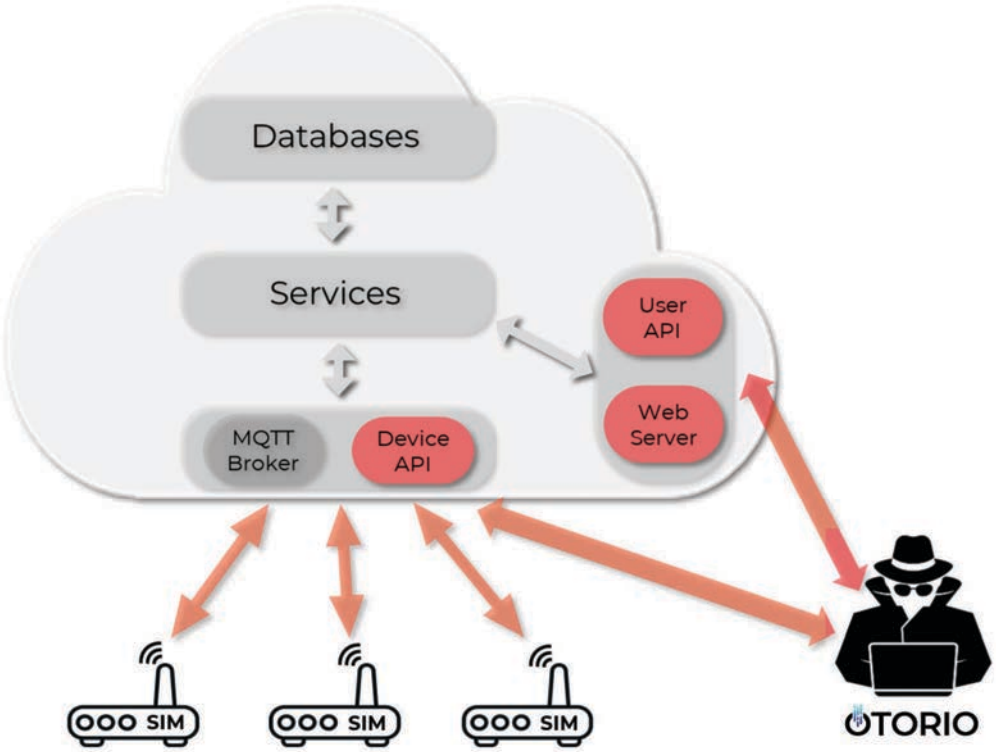


Exploitation demo screenshot

External API and web interfaces

As another potential vector for attacks on the cloud-based platform, we thoroughly examined the external APIs and web interfaces. For demonstration purposes, we focused on Teltonika Networks RMS platform, which is utilized for managing their RUT, TRB, and TCR router families widely deployed in industrial settings.

During our investigation of the 'RMS' platform, we discovered vulnerabilities in both the Device API and the web interface. By exploiting and chaining these vulnerabilities, we successfully executed an account takeover and achieved remote code execution on all the managed routers within these families.

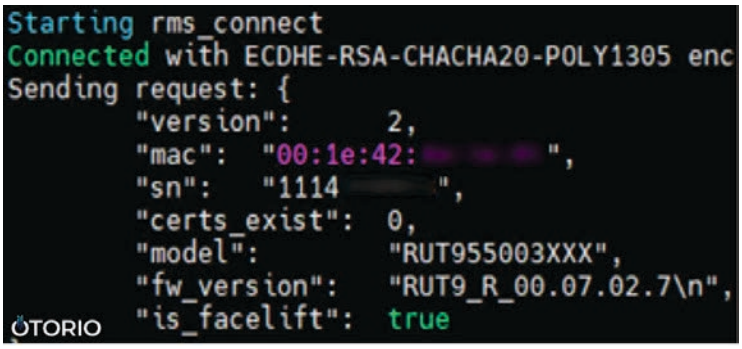


External APIs and web interfaces as additional vectors

Teltonika Networks “RMS” platform

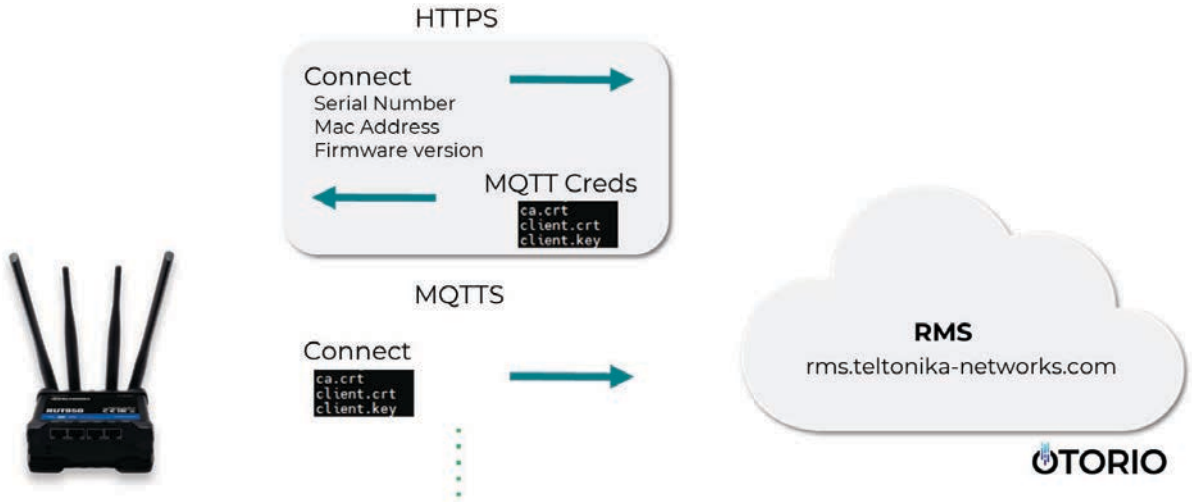
Upon booting, the router initiates a connection to the 'RMS' cloud platform through the rms_connect executable. This connection is established automatically and utilizes HTTPS with hardcoded credentials, a certificate and key. As part of the

connection process, rms_connect transmits various identifiers to the 'RMS' platform, such as the router's MAC address, serial number (SN), firmware version, router model, and other relevant information.



Connection initialization using the device API

The 'RMS' platform verifies the existence of a router by validating the pair of MAC address and serial number provided. If the pair is correct, the 'RMS' platform generates a unique "one-time" certificate and key, which are then sent to the router. Utilizing these credentials, the router establishes a MQTT session with the 'RMS' cloud platform, ensuring secure and authenticated communication between the two entities.



Industrial cellular router connection to the RMS platform

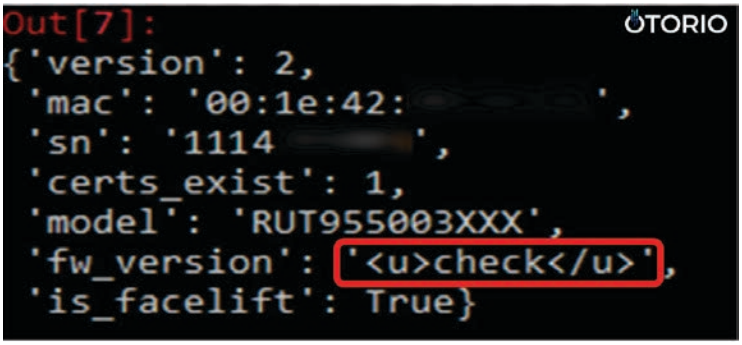
Vulnerabilities in external APIs and web interfaces

Impersonation to ‘RMS’ managed router (CVE-2023-32347)

By establishing a connection to the 'RMS' cloud platform using the hardcoded certificate and key, the attacker can transmit the MAC address and serial number pair, along with other relevant parameters. This action tricks the 'RMS' platform into believing that the attacker is the legitimate router. Consequently, any information intended for the router will be diverted to the attacker instead, granting them unauthorized access to the data and control over the communication between the cloud platform and the router.

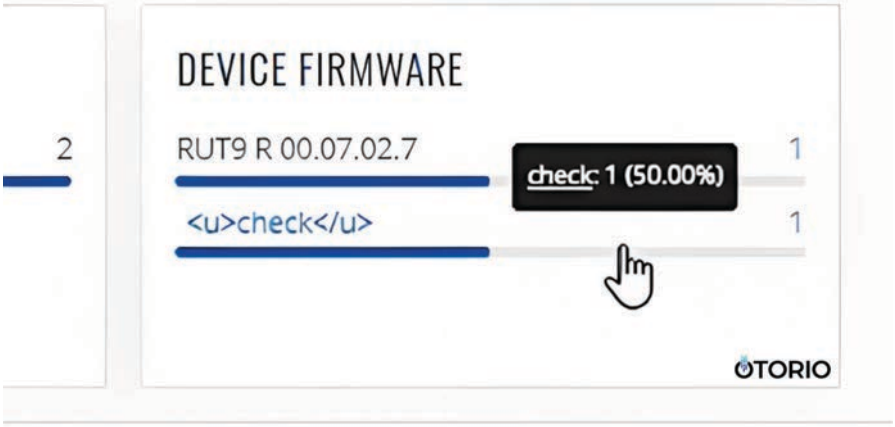
Stored-XSS in ‘RMS’ main page (CVE-2023-2587)

Using the MAC-serial combination of a registered device, attackers can exploit the vulnerability by sending a carefully crafted JSON message. Specifically, they manipulate the fw_version field by injecting an HTML object, triggering a vulnerability within the RMS web interface.



XSS vulnerability affecting RMS web interface

By exploiting a stored cross-site scripting (XSS) vulnerability present in the "DEVICE FIRMWARE" section of the main RMS page, the attacker can execute client-side logic on the targeted administrator's (victim's) browser, just by waiting for the victim to hover their mouse cursor over the middle of the screen. The injected content, as illustrated in the accompanying image, includes an HTML object with an <u> (underline) tag for demonstration purposes.

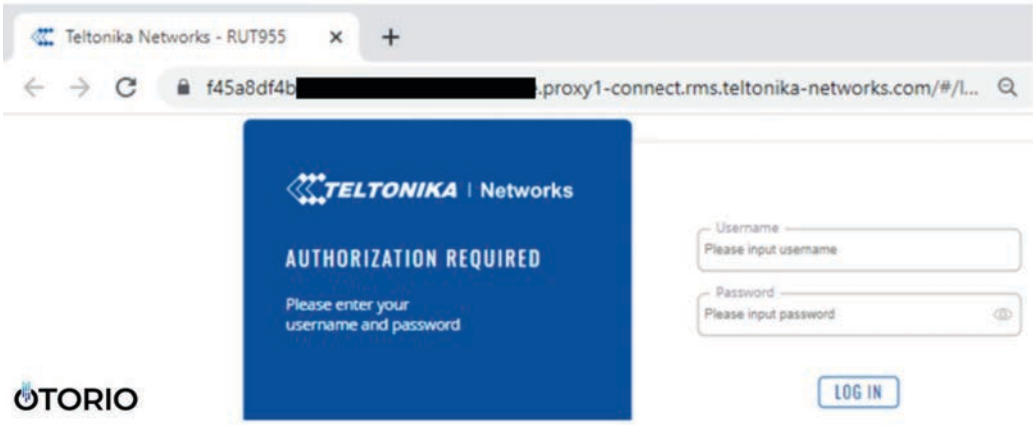


Stored *XSS on mouseover*

The discovered XSS vulnerability presents an opportunity for a malicious actor to extract and leak cookies, potentially leading to an account takeover. However, due to certain limitations in the XSS vulnerability, the exploitation process becomes more challenging. To overcome this obstacle, we successfully chained it with another vulnerability, which significantly simplified achieving an account takeover.

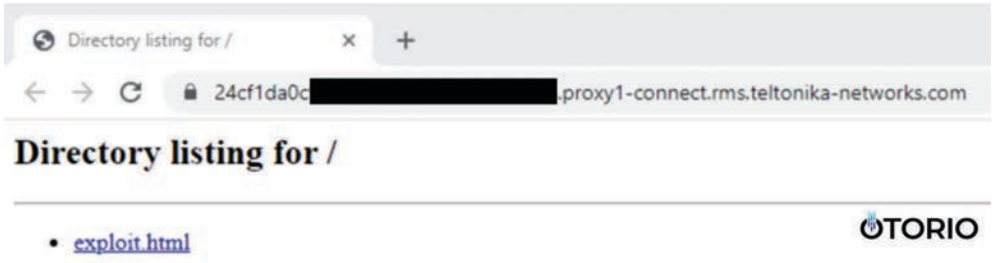
Abusing user-controlled subdomains in teltonika-networks.com (CVE-2023-2588)

The "Device WEB" feature in the RMS, allows users to access managed device's local web management service over the RMS cloud proxy. Requesting for a web proxy to our device, will result in a url in the RMS cloud subdomain leading to our device local web interface:



RMS device web proxy feature

This URL can be shared with others and requires no RMS-level authentication. Since we are in control of our local device, we are also in control of the web pages that are served over this URL. For example, we can redirect the traffic internally to our own web server instead of the local web service:



Proxied local web server forwarding to attackers web service

While the actual web service is hosted locally on the device, the domain scope is defined as part of the RMS domain (**.rms.teltonika-networks.com*) according to RMS server Content Security Policy header:

```
content-security-policy: form-action 'self' rms.teltonika-networks.com *.rms.teltonika-networks.com,
```

Content-security-policy header received from the RMS web interface

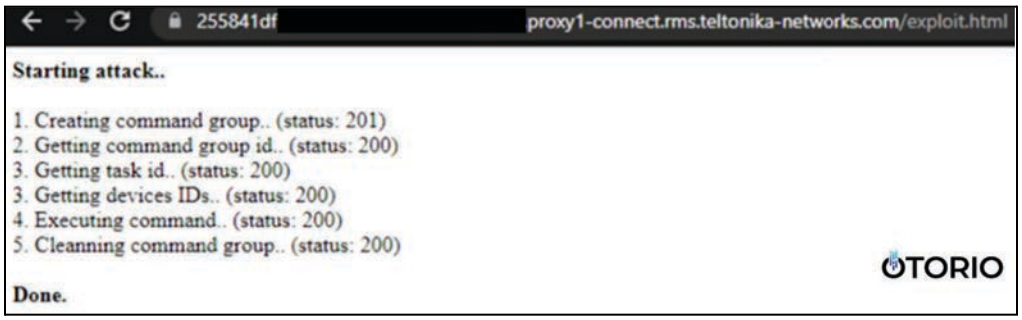
This configuration will cause the user browser to assume the web page provided by the end-device is part of the RMS, including access to the user's cookies, leading to a significant risk originating from a bad security configuration.

Attacker may leverage this vulnerability in different ways, only a few operations are required for initializing it:

1. Acquire his own IIoT device, or Impersonate as one (using previously described vulnerability).
2. Register the device to his account, and ask the cloud platform to provide a proxy link to access the device local web interface

3. Redirect incoming tunneled connections to a custom web server leaking cookies or performing operations on behalf of the user account.

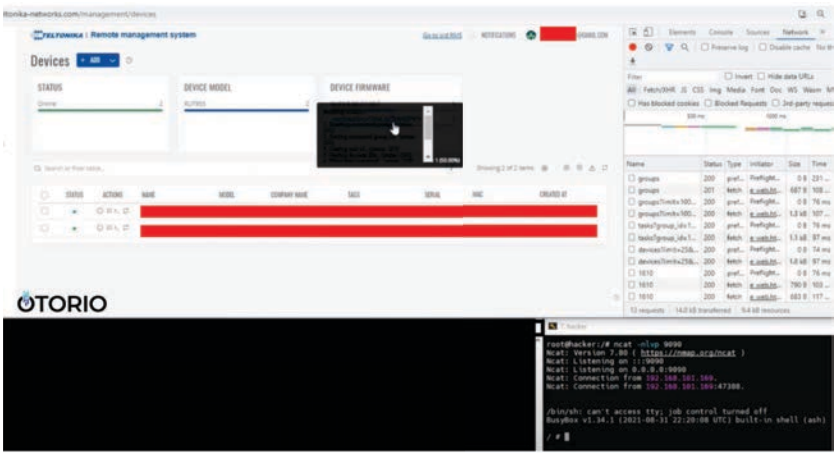
At this phase, any connected RMS user loading the attacker's private proxy web page is exposed to full account takeover.



Exploit executing reverse shell on all connected devices for the targeted account

While this malicious page can be embedded on any website online or sent as part of a phishing campaign, using the stored-XSS vulnerability in the *fw_version* field, an attacker can target specific users, using the HTML <embed> tag to load the exploit page as part of their management panel.

Once the victim moves the mouse over this area, the exploit.html web page will be loaded, creating a reverse shell on all managed routers under this RMS user.



Mouseover to account takeover attack on RMS platform main interface

Summary and Recommendations

The increasing presence of IIoT devices in sensitive environments raises significant concerns. These devices, with their extensive functionalities and direct internet connectivity, are highly susceptible to compromise. Along with the common deployment of these devices without additional layers of security, directly exposes operational processes.

The potential impact on thousands of industrial environments is alarming and should be a concern for both private and public sectors. It highlights the urgent need for effective security measures, proper deployment best practices, and improved communication of risks by vendors.

A key challenge lies in user awareness. Many users may be unaware of the automatic connections established by these devices or may follow simplified deployment architectures outlined in vendor guides. Moreover, the presence of certain built-in security features can create a false sense of security. It is important to recognize that the overall safety of these devices is only as strong as their weakest component. Once compromised, the effectiveness of these security features becomes irrelevant.

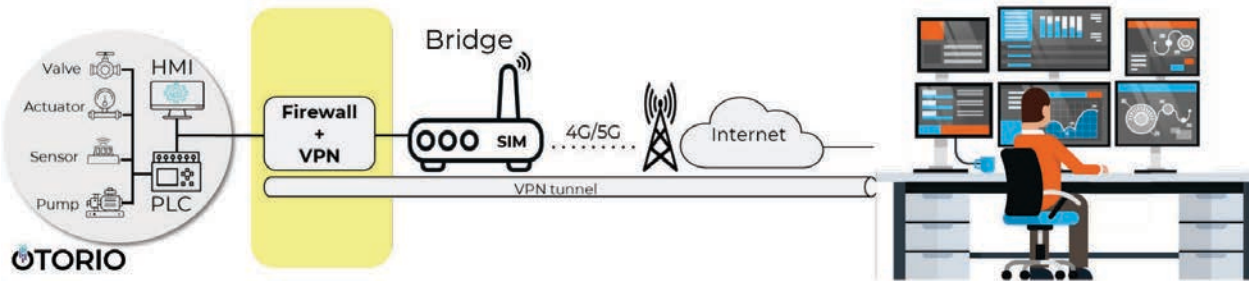
To address these challenges, organizations must prioritize the implementation of robust security measures and making informed decisions regarding device deployment. By taking proactive measures, organizations can effectively safeguard their devices in sensitive industrial environments.

Practical mitigation strategies for users

1. **Disable unused cloud feature:** If you're not actively using the cloud management feature of your industrial cellular routers, disable it to prevent device takeovers and reduce the attack surface.

2. **Register devices:** Before connecting your devices to the internet, register them under your own account in the cloud platform to establish ownership and control, preventing unauthorized access

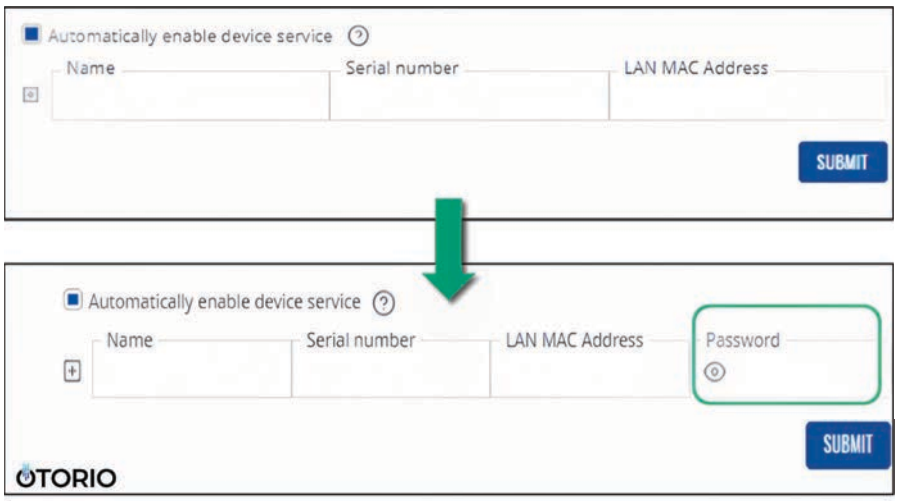
3. **Limiting direct access from IIoT:** Industrial cellular routers' built-in security features like VPN tunnels and firewalls are ineffective once compromised. Adding separate firewall and VPN layers can assist with delimitering and reduce risks from exposed IIoT devices used for remote connectivity.



Suggested hardening for cloud managed IIoT devices connectivity

Practical mitigation strategies for vendors

1. **Avoid usage of weak identifiers:** Use an additional "secret" identifier during device registration and connection establishment for enhanced security.
2. **Enforce initial credential setup:** Default credentials for remote-accessible devices can present a significant problem.
3. **Industrial IoT ≠ IoT:** Implement tailored security measures for IIoT considering the unique requirements. This may involve reducing "high-risk" features upon demand and adding extra layers of authentication, encryption, access control, and monitoring.



Vendor includes additional secret to address weak registration process.

Closing remarks

Proactive measures can effectively mitigate many of the risks discussed in this paper, reducing them to an acceptable level while maintaining functionality of the IIoT devices. We strongly encourage our clients and IIoT users to carefully consider the recommendations outlined above. Should you have any further questions or concerns, we are available to provide assistance and support at:

- General inquiries - info@otorio.com
- Roni Gavrilov, Security Researcher - roni.gavriolv@otorio.com
- Eran Jacob, Security Research Team Leader - eran.jacob@otorio.com

Appendix A - Disclosed vulnerabilities table

Vendor	CVEs	Affected devices	Link
Teltonika Networks	CVE-2023-32347 - CVSS 10.0 CVE-2023-2586 - CVSS 9.0 CVE-2023-2587 - CVSS 8.8 CVE-2023-2588 - CVSS 8.8	RUT devices / Remote Management System platform	https://www.cisa.gov/news-events/ics-advisories/icsa-23-131-08
InHand Networks	CVE-2023-22600 - CVSS 10.0 CVE-2023-22598 - CVSS 7.2 CVE-2023-22599 - CVSS 7.0 CVE-2023-22597 - CVSS 6.5 CVE-2023-2261 - CVSS 5.3	InRouter devices / Device Manager platform	https://www.cisa.gov/uscert/ics/advisories/icsa-23-012-03
Sierra Wireless	CVE-2023-31279 - CVSS 8.1 CVE-2023-31280 - CVSS 5.3	AirLink devices / AirVantage platform	https://www.cisa.gov/news-events/ics-advisories/icsa-23-131-07