SAND2022-10384

SANDIA REPORT
SAND20XX-XXXX
Printed Click to enter a date

Sandia National Laboratories

Cyber and Physical Security Analysis of GSI and Noventum Application for IoT Communications

Jimenez, Yesid
Khalafalla, Aya
Summers, Adam
Onunkwo, Ifeoma
Chavez, Adrian

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico
87185 and Livermore,
California 94550

Sandia National Laboratories
U.S. DEPARTMENT OF ENERGY NNSA

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@osti.gov
Online ordering: http://www.osti.gov/scitech

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Rd
Alexandria, VA 22312

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.gov
Online order: https://classic.ntis.gov/help/order-methods/

ABSTRACT

We present our findings of the red team exercise conducted on the device and application developed by Guardian Sensors, Inc. (GSI) and Noventum. The app is used for situational awareness and control of photovoltaics (PV) and microgrid energy systems. The assessments performed are practical case scenarios that assess the risks and vulnerabilities posed by the app through targeted activities that could be engaged by an adversary. The assessment team's results and recommendations are provided to inform on and mitigate the identified weaknesses to improve secure user authentication, connections, and communications. The recommendations in this report are not intended to be a security panacea but will add the desired defense-in-depth layer to securing communication of such interconnected systems.

This page left blank

CONTENTS

Abstract 3
Executive Summary 6
Acronyms And Terms 8
1. Target Device Overview And Laboratory Environment 9
2. Assessments 11
2.1. WEB APPLICATIONS ASSESSMENTS 11
2.1.1. HIGH SEVERITY FINDINGS 11
2.1.2. MEDIUM SEVERITY FINDINGS 16
2.1.3. LOW SEVERITY FINDINGS 17
2.2. SOFTWARE CHECKS 20
2.2.1. FINDINGS AND OBSERVATIONS 20
2.3. DEVICE PENETRATION TESTING 21
2.3.1. FINDINGS AND OBSERVATIONS 22
3. Summary 30
Appendix B. Web Application Checks 31
Distribution 33

LIST OF FIGURES

Figure 1: Device login portal authentication page 9
Figure 2: Solar array status shown on the index.php page 9
Figure 3: Physical testbed for red teaming 10
Figure 4: Device login portal authentication page 11
Figure 5: Using curl, an unauthenticated user can view restricted content on index.php 11
Figure 6: Using curl with a POST request parameters allows an unauthenticated user to change the status of the solar arrays 12
Figure 7: Login.php PHP redirect flaw causing authentication bypass 12
Figure 8: Unauthorized access to .git directory 13
Figure 9: GitTools creates directories for each commit with source code in each directory 14
Figure 10: Internal directory access showing credentials in plaintext 15
Figure 11:Browsable web directories and pages 16
Figure 12: Cross-site scripting proof of concept 17
Figure 13: XSS Attack displays the PHPSession ID 17
Figure 14: Unauthenticated users can directly access the PHPINFO page at /info.php 18
Figure 15: Test page located at /guardian.php 19
Figure 16: Nmap operating system and services detection 22
Figure 17: Transport layer security TLS 1.3 23
Figure 18: Using the openssl s_client command to check for weak ciphers 23
Figure 19: Interruptions to the application and device 24
Figure 20: ARP poisoning using man-in-the-middle attack 24
Figure 21: Driftnet launched but interception was not successful 25
Figure 22: FIN scan to check firewall 25
Figure 23: Firewall security not detected behind the solarguardian.mgtsciences domain 26
Figure 24: Wireshark capture of duplicated pcap files that was replayed 27
Figure 25: Replayed packets disrupting connections to the web application 27

EXECUTIVE SUMMARY

Our team completed a targeted adversary-based security assessment of the web application and device developed by Guardian Sensors, Inc. (GSI) and Noventum. The evaluation included a vulnerability assessment to identify flaws and security weaknesses as well as penetration testing to exploit the vulnerabilities that compromise information and device security. The assessment activities were performed on an isolated network at the Distributed Energy Technology Laboratory at Sandia National Laboratories (SNL) in Albuquerque, New Mexico.

The target system is a web application and device that is used for situational awareness and control of energy systems. To organize our approach and activities, the assessment team combined practices from multiple sources - Sandia's Information Design Assurance Red Team (IDART™)[1], OWASP Top 10[2], best cyber security practices, and collective expertise regarding web applications.

The assessment team conducted security testing on the current version of the software and device from numerous perspectives including reconnaissance, which identified accessible HTTP and HTTPS ports and service and application version detection. It was observed that the web application uses PHP running on an Apache HTTPD server to operate a TLS 1.3 web server with OpenSSL. Our team also identified several accessible directories that revealed the source code of the application and allowed us to bypass authentication. An in-depth analysis revealed critical issues related to authentication bypass and the methodology used for authentication. To authenticate users, a simple login mechanism is provided by the application, with support for only one user. However, to support the goals of confidentiality and integrity, our team recommends updating the application to support multiple users and using a backend database to manage passwords. To better assure the quality of the code, software checklists to reduce the risks of deploying insecure software was used. The applicable checklists were verified, and the results are presented herein.

Each assessment in this report describes the findings and provides recommendations for addressing the identified vulnerabilities. The following table summarizes our findings ranked by severity. The scoring rubric used to categorize these vulnerabilities in the results section were taken from MITRE's Common Vulnerability and Exposure (CVE) and the NIST's Common Weakness Enumeration (CWE) rankings. Our team ***highly recommends*** addressing the "High" and "Medium" severity vulnerabilities before deploying the application in a production environment.

Severity	Vulnerability	Description	Recommendation
High	Authentication Bypass	Non-authenticated users can view and edit solar array status using curl. This vulnerability is present because the login.php code does not kill the session after it redirects a user. To view the status of the solar arrays: curl -k --include https://<IP>/index.php To change the status of the solar arrays to ON: curl -d "onButton=On" -k --include -X POST https://<IP>/index.php	In the login.php code, add exit() or die() to kill the session after a redirect
High	Browsable .Git Directory	The .git directory is accessible to non-authenticated users and reveals project source code and credentials. https://<IP>/.git	Restrict access to .git and/or disable directory browsing

[1] https://idart.sandia.gov
[2] https://owasp.org/

Severity	Vulnerability	Description	Recommendation
High	Credentials stored in plain-text	Login.php is accessible through the publicly accessible .git directory. Login.php uses a basic string comparison to validate the password which is stored in clear text.	If string comparison must be used, hash and salt passwords. Using a database for authentication is recommended
High	Weak Architecture Design	Based on information from Login.php, the application only supports one user and one password. This is a weak design assuming multiple users will be using the application.	Redesign the web application and use a backed database to support multiple users, authentication, and logging
Medium	Browsable Web Directories and Pages	Several directories are accessible including: /.git /info.php /guardian.php /panels/	Restrict access to directories to limit information disclosure
Medium	Reflected XSS Vulnerabilities	The web application is vulnerable to reflected cross site scripting	Implement HTML encoding and input validation
Low	PHPINFO Page Accessible	Unauthenticated users can access /info.php. Attackers can use this page to scrape information about the application – in this case, PHP and Apache versions, file locations, and settings	Restrict access to info.php
Low	Test Page Accessible	/guardian.php is accessible with no authentication and seems to be a test page for the main index.php. Changes in the index.php page do not reflect back to the guardian.php page	Remove page or restrict access

ACRONYMS AND TERMS

Abbreviation	Definition
CVE	Common Vulnerability Exposure
CVSS	Common Vulnerability Scoring System
DER	Distributed Energy Resource
DOS	Denial of Service
DDOS	Distributed Denial of Service
MITM	Man-In-The-Middle
PV	Photovoltaic

1. TARGET DEVICE OVERVIEW AND LABORATORY ENVIRONMENT

The device provided to our assessment team by GSI and Noventum can be connected to any internal or external network. Users logging into the device with their credentials are first authenticated to their web page before accessing the web portal that simulates the startup and shutdown of a "grid". The user's login page depicted in **Error! Reference source not found.** uses basic authentication. Per our customer (GSI and Noventum), the web page provides read and write permissions to files that represent physical devices that are meant to be on the back of a solar panel. GSI and Noventum plan on extending this feature to communicate with real solar panel devices.

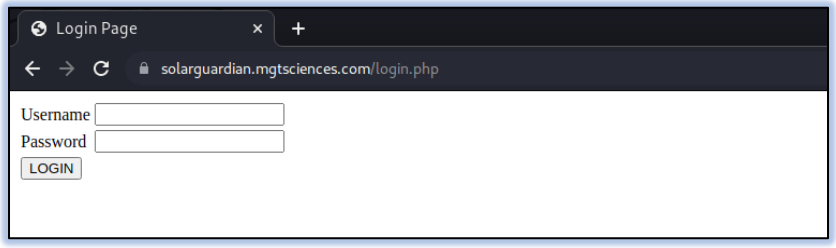

Figure 1: Device login portal authentication page

As shown in **Error! Reference source not found.**, the solar array status can be displayed to the user in either a safe or unsafe status with three off and on buttons respectively. The off button displays the status as safe while the on button means that the status of the solar array is unsafe.

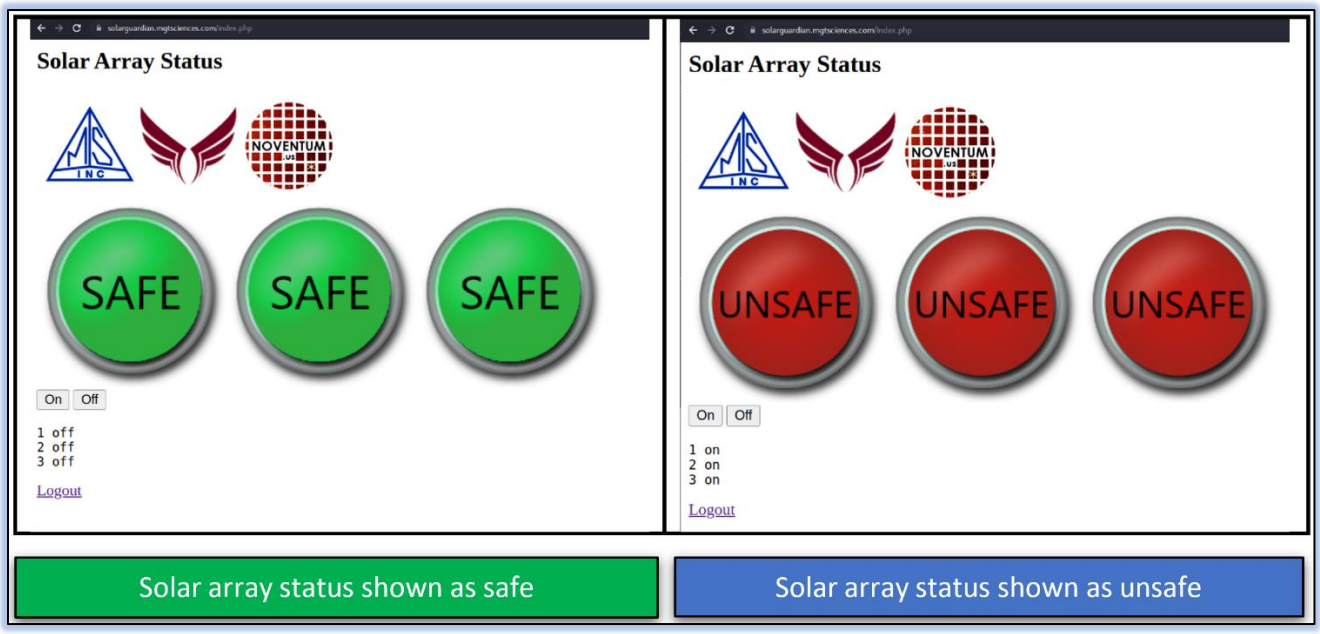

Figure 2: Solar array status shown on the index.php page

The experiments were conducted on an isolated and controlled network environment. The network was created with the device, a network hub that connects devices in a network, and a machine installed with Kali Linux. Kali[3] is an open-source Linux operating system based on Debian that is equipped with security and analysis tools for identifying and exploiting vulnerabilities. Burp Suite Professional [4], an application web security testing software and other open-source tools were

[3] https://www.kali.org/
[4] https://portswigger.net/burp/pro

also installed on the Kali Linux machine. The test network environment as described is shown in **Error! Reference source not found.**.

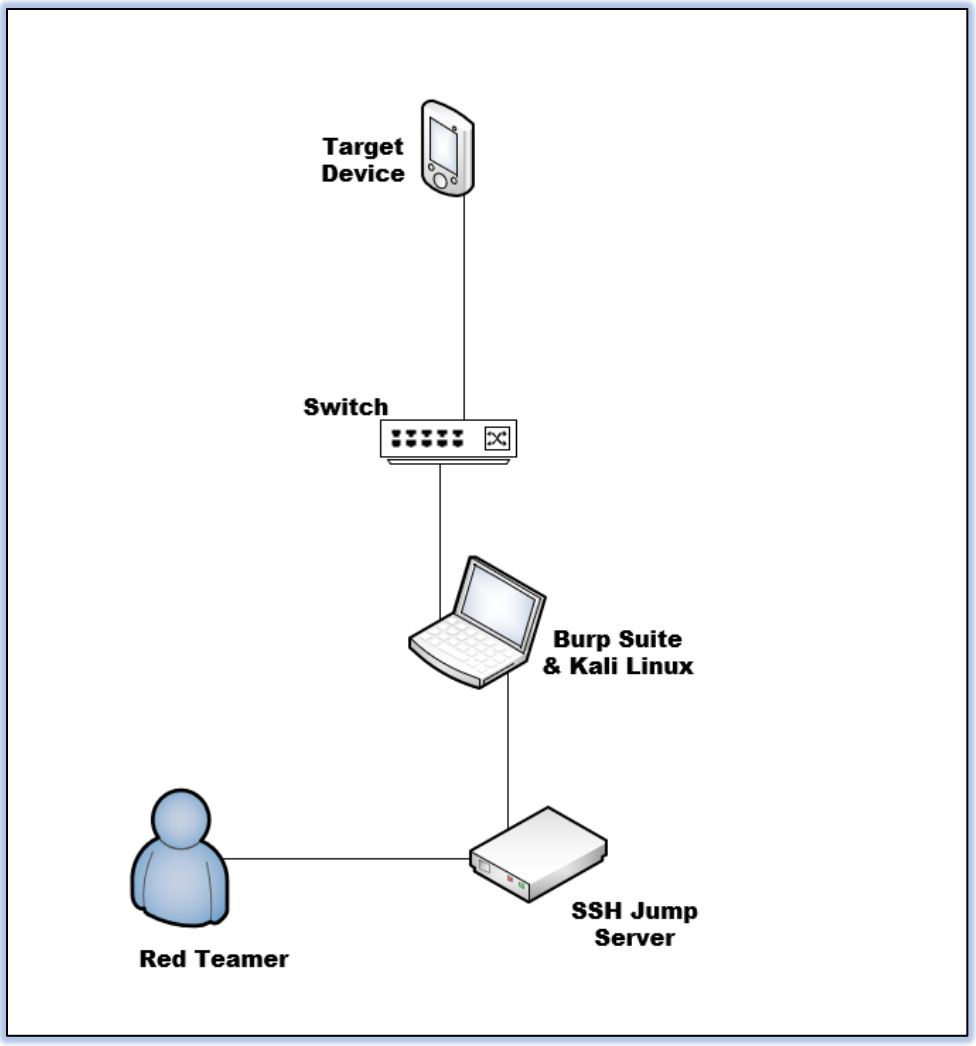

Figure 3: Physical testbed for red teaming

2. ASSESSMENTS

The following tests were performed to assess the security of the device when exposed to complex environments.

2.1. Web Applications Assessments

2.1.1. High Severity Findings

2.1.1.1. Authentication Bypass

The application allows unauthenticated users to view AND change the settings of the on and off buttons that control the solar array. By design, the application requires users to authenticate before they can view or edit the status of the solar arrays as shown in Figure 4. However, using information gathered from reconnaissance, the team was able to identify a weakness in the PHP logic that allows users using curl to view the contents of index.php before logging in as a normal or administrative user.

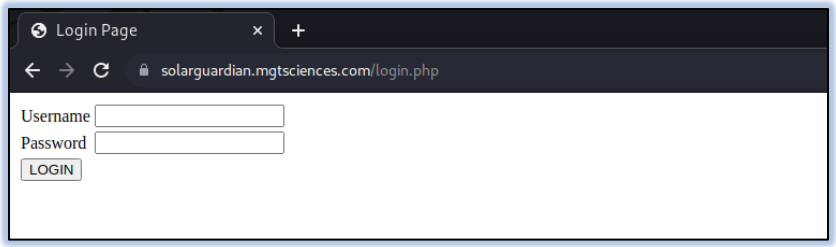

Figure 4: Device login portal authentication page

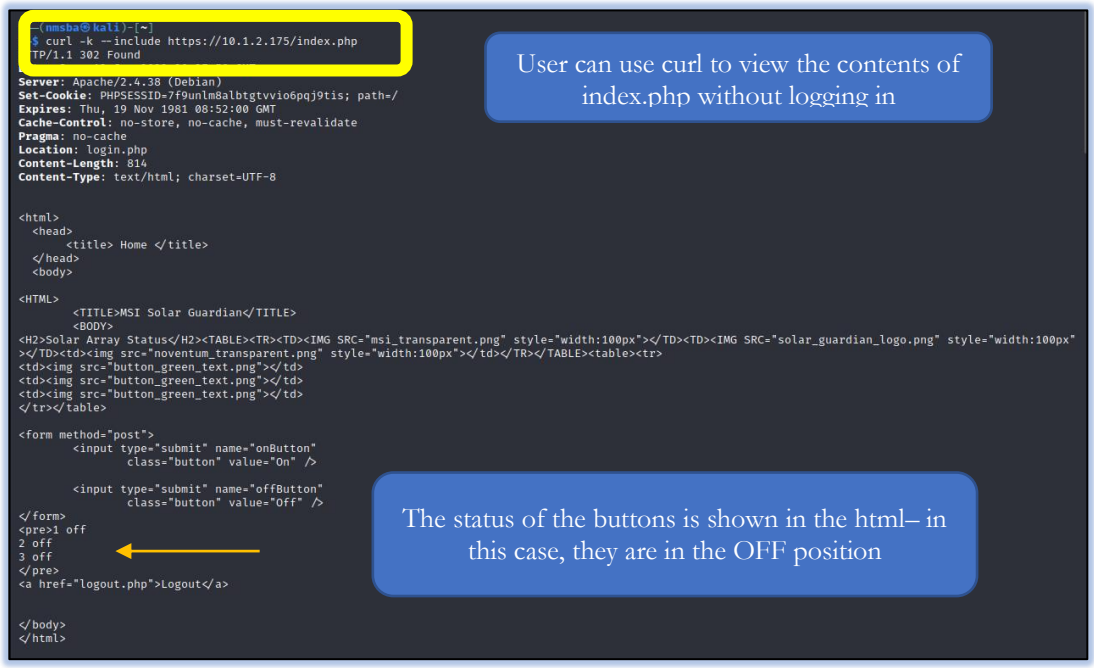

Figure 5: Using curl, an unauthenticated user can view restricted content on index.php

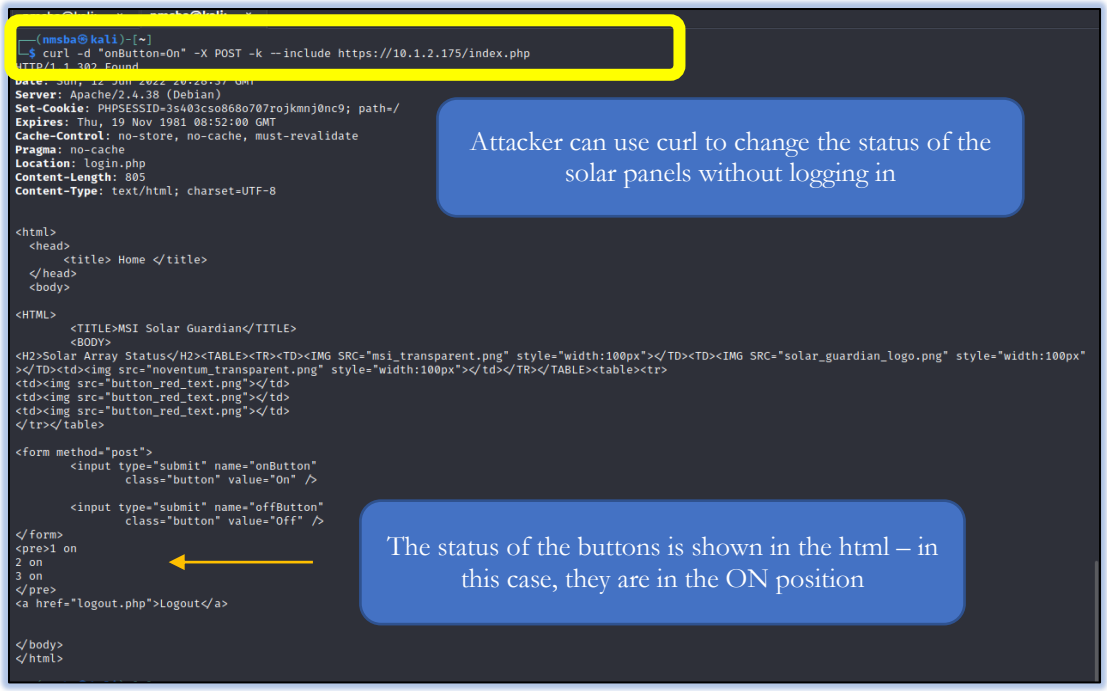

Figure 6: Using curl with a POST request parameters allows an unauthenticated user to change the status of the solar arrays

The authentication bypass vulnerability results are due to an error in the PHP code for the login.php page. Our team then took advantage of a browsable .git directory on the web application to view the source code of the application. Viewing the login.php source code, our team noted a code design vulnerability in the PHP redirect that allowed users to bypass authentication. The code uses a relocation header and an if condition to redirect a user to the index.php page. The code sample in Figure 7 has a redirection flaw. While this is effective with browsers, tools that do not automatically obey a location header (such as curl) will see the remainder of the HTTP response, which in this case includes the restricted solar panel content.

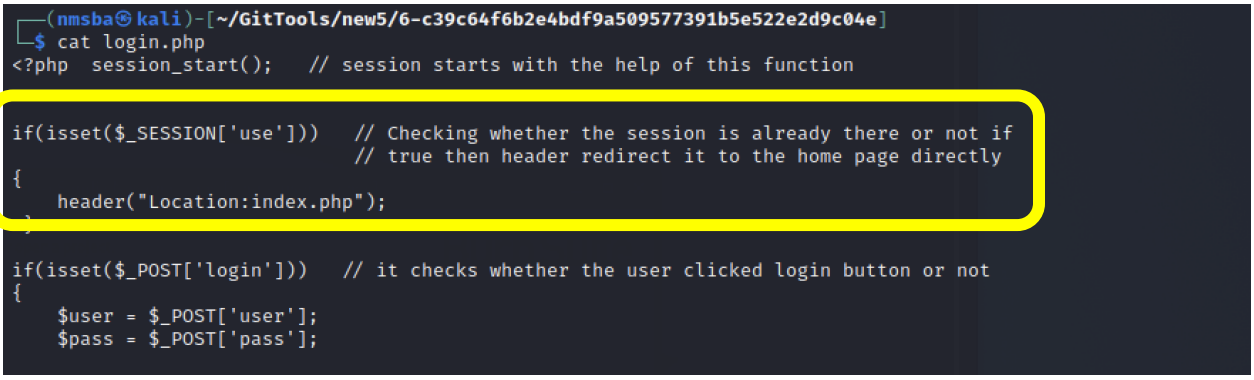

Figure 7: Login.php PHP redirect flaw causing authentication bypass

Recommendation

An effective way to fix this vulnerability is to add die() or exit() after the Location header for PHP to stop processing the rest of the page. This will resolve the authentication bypass vulnerability that currently allows any user with access to the device to arbitrarily view and change the availability of the solar panels.

2.1.1.2. Browsable .Git Directory

During the team's reconnaissance phase, we identified a browsable .git repository using Nmap and GoBuster. This directory is accessible to any unauthenticated user and reveals the source code, commits, and versions of the web application. Our team utilized an opensource tool, GitTools, to access information in the .git directory. This access allowed us to extract sensitive information including plaintext usernames and passwords, application source code, and developer names.

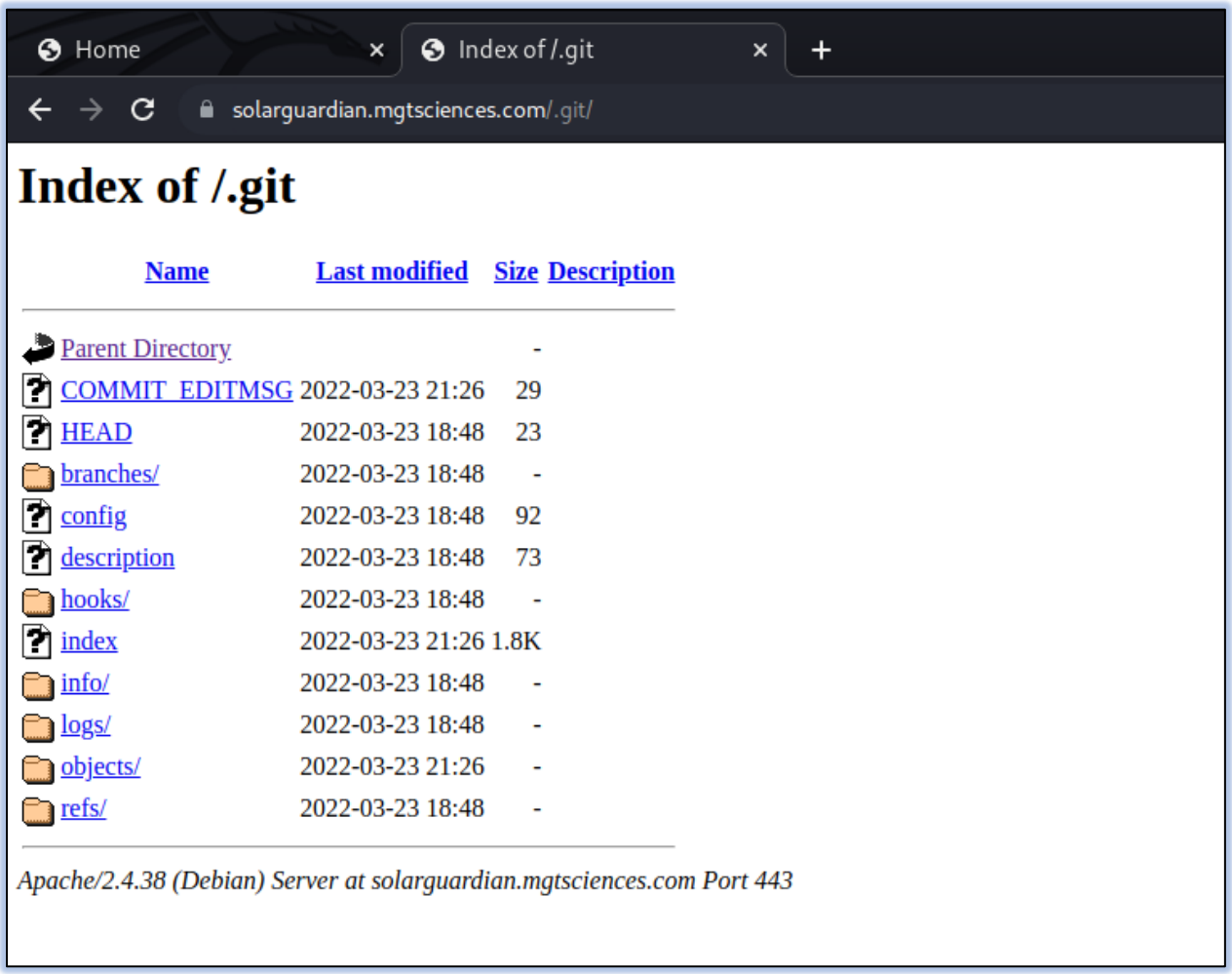

Figure 8: Unauthorized access to .git directory

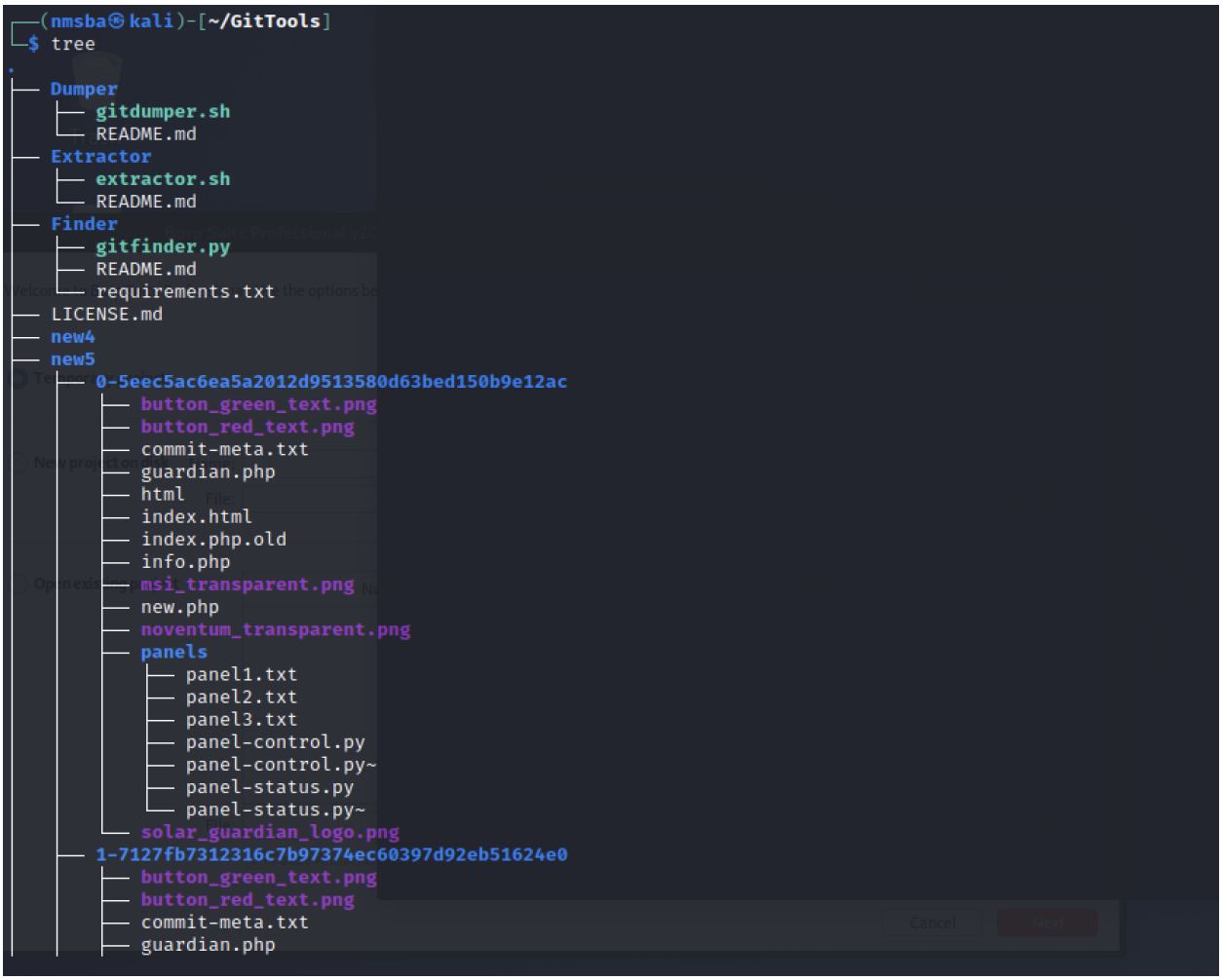

Figure 9: GitTools creates directories for each commit with source code in each directory

Recommendation

Restrict access to the .git directory.

2.1.1.3. Credentials Stored in Plain-Text

Using the information gleaned from the browsable .git directory, our team recovered plaintext usernames and passwords that were being used for serial authentication to the solar panel portal. These credentials are available to unauthorized users with access to the device's network and would give them full control of the application.

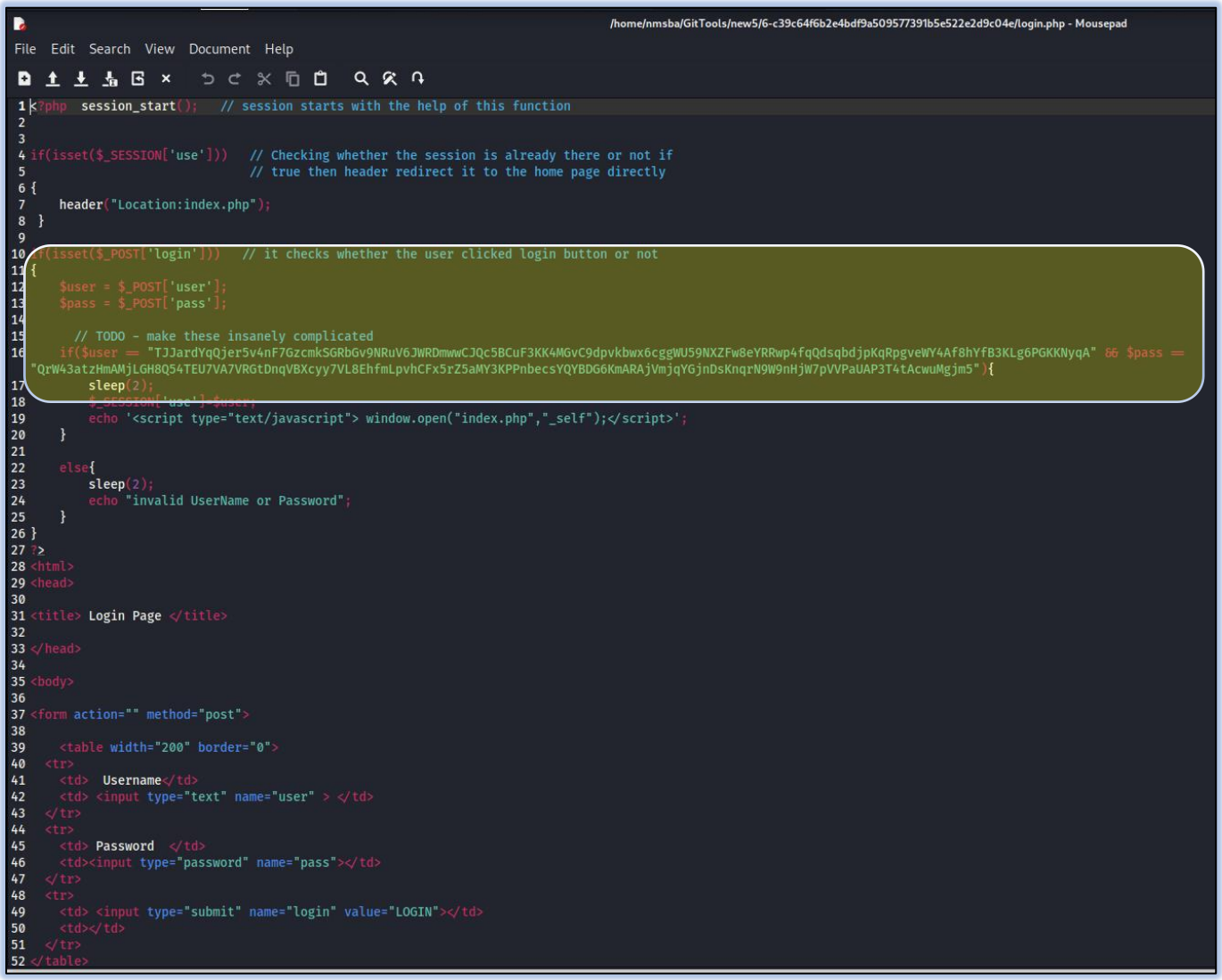

Figure 10: Internal directory access showing credentials in plaintext

Recommendation

If string comparison must be used, hash and salt passwords. However, we recommend using a database for authentication.

2.1.1.4. Weak Architecture Design

During an analysis of the application, our team noted that the application only supports a login feature for one user as shown in Figure 10. This design is inherently insecure and does not support cybersecurity principles including principle of least privilege, role-based access, role separation to mention a few.

Recommendation

If multiple users are to be using the system, our team recommends redesigning the application to support multiple users, with role-based access including one administrator and other non-privileged users. Our team also recommends using a backend database to support user and credential management.

2.1.2. Medium Severity Findings

2.1.2.1. Browsable Web Directories and Pages

Using two open-source tools Dirbuster and Gobuster, our team identified several browsable directories that include sensitive information about the application as show in Figure 11. These directories are accessible to unauthenticated users who have access to the device's network. Notable directories include

- /.git/ - Reveals the entire source code of the application and its prior versions. Usernames, passwords, and developer names were also identified in this directory.
- /panels/ - Reveals the python-based source code of the application.
- /Guardian.php – Reveals to what seems to be a test page that provides an unauthenticated user information about the structure of the application. Our team performed tests to confirm if changes in the main /index.php page reflected in the guardian.php and they did not.
- /info.php – Reveals information about the application's configuration including software versions. This information can be used by an attacker to leverage further attacks.

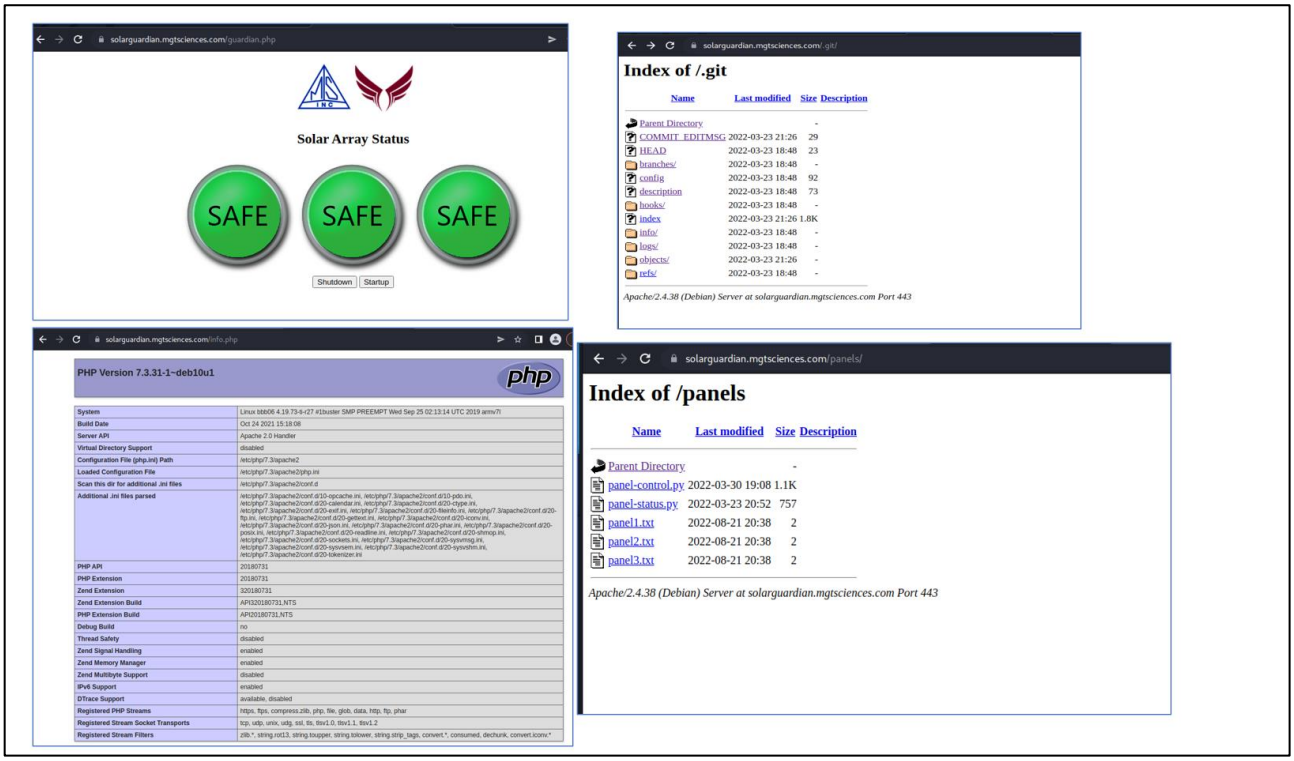

Figure 11:Browsable web directories and pages

Recommendation

Restrict unauthorized access to sensitive pages.

2.1.2.2. Reflected Cross-Site Scripting

Our team utilized BurpSuite to scan the application for common web application vulnerabilities. BurpSuite identified a reflected cross site scripting (XSS) vulnerability in the application as shown in Figures 12 and 13.
Reflected cross-site scripting vulnerabilities arise when data is copied from a request and echoed into the application's immediate response in an unsafe way. An attacker can use the vulnerability to construct a request which, if issued by another application user, will cause JavaScript code supplied by the attacker to execute within the user's browser in the context of that user's session with the application. The attacker-supplied code can perform a wide variety of actions, such as stealing the victim's session token or login credentials, performing arbitrary actions on the victim's behalf, and logging their keystrokes.[5]

Figure 12: Cross-site scripting proof of concept

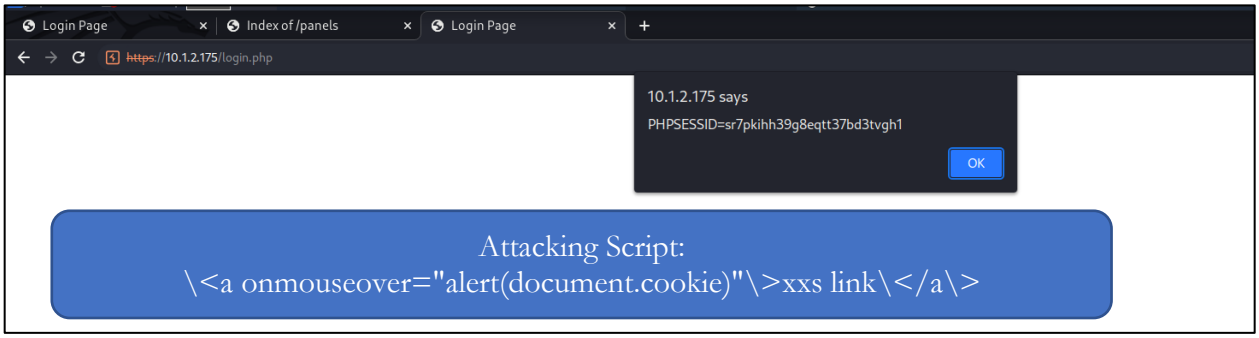

Figure 13: XSS Attack displays the PHPSession ID

Recommendation

Enforce input validation and HTML encode user input.

2.1.3. Low Severity Findings

2.1.3.1. PHPINFO Page Accessible

During the reconnaissance phase of the assessment, our team identified a PHPINFO page located at /info.php that is accessible to unauthenticated users (Figure 14). This page is a default PHP page that provides information about the device's unique Apache and PHP configurations including versions, file locations, and IP addresses. While this vulnerability is not directly exploitable, it provides attackers with valuable information to launch further attacks.

[5] https://portswigger.net/kb/issues/00200300_cross-site-scripting-reflected

In this case, the /info.php page revealed information about Vulnerable and outdated components as shown by the currently installed PHP version 7.3.31-1 while the latest is 8.1 and the currently installed Apache version is 2.4.38 while the latest is 2.4.46.

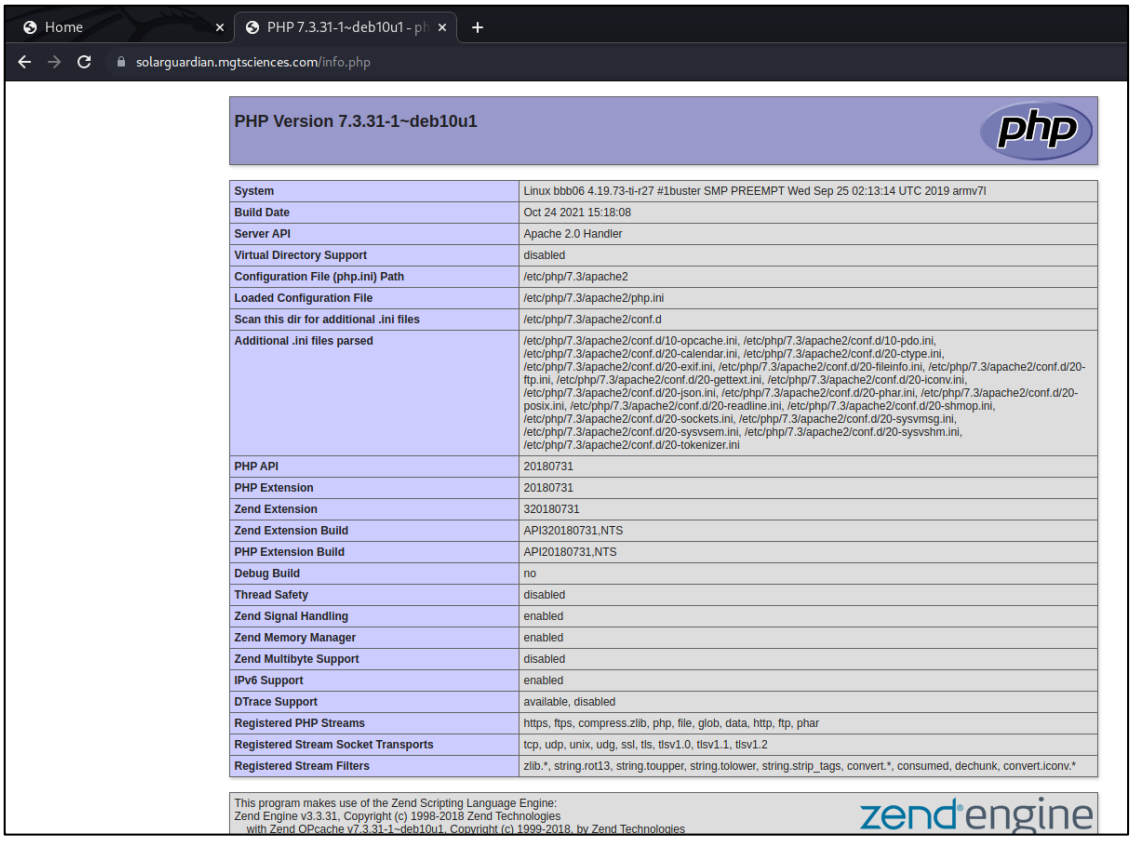

Figure 14: Unauthenticated users can directly access the PHPINFO page at /info.php

Recommendation

Disable access to the PHPINFO page located at /info.php

2.1.3.2. Test Page Available

During the team's reconnaissance phase of testing, we identified a page that appears to be identical to the main status page located at /guardian.php (Figure 15). Our team performed tests to ensure that changes made to the main index.php page were NOT reflected in the guardian.php. We believe that this is a page that was used for testing. While this is not directly exploitable, this page is accessible to unauthenticated users and provides attackers an ability to learn more about the application's structure.

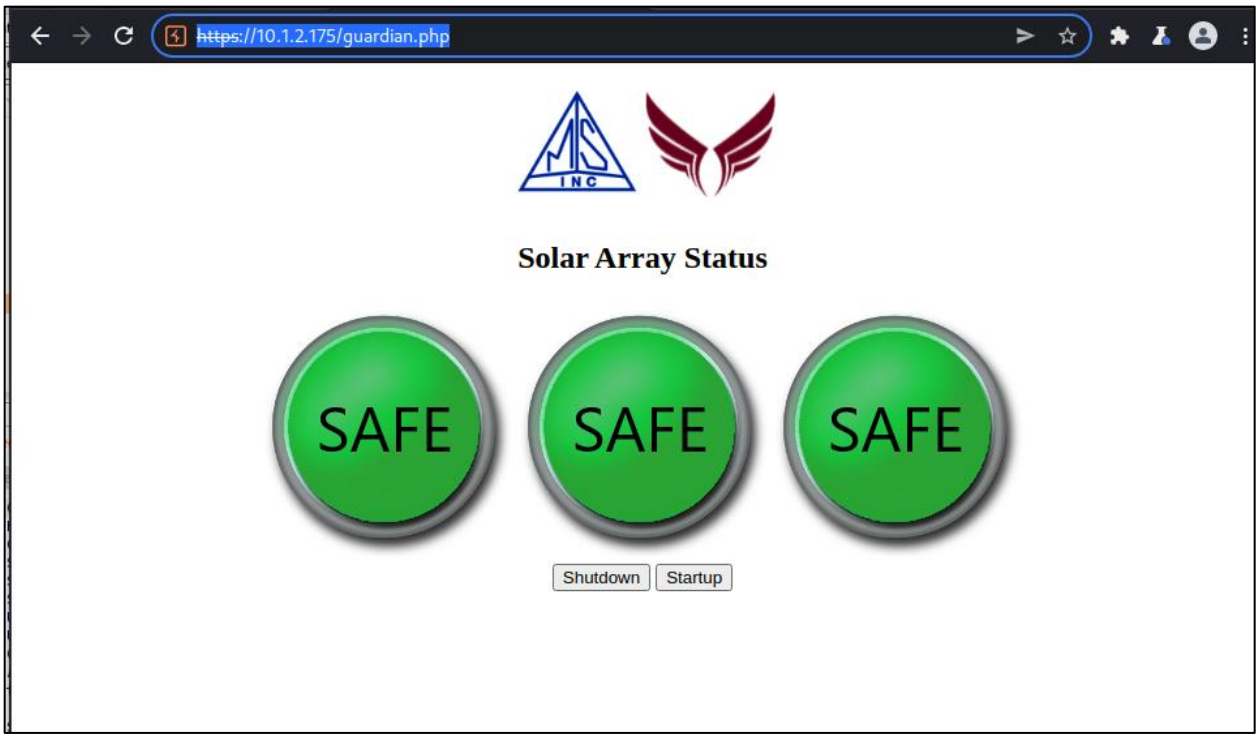

Figure 15: Test page located at /guardian.php

Recommendation

Disable access to guardian.php

2.2. Software Checks

To minimize the risk of deploying insecure software, the assessment team went through the following software checklists. While the applicable findings are outlined below based on the team's access of the device and application, the team strongly suggests that the other checklists to improve the deployment of the product should be tested.

Category	Checklist	Findings
Operating System	Is the app using a formally verifiable operating system?	Apache/2.4.38 (Debian)
Compiler	Is the app using a formally verifiable compiler?	N/A
CPU	Is the app using a formally verifiable CUP core?	N/A
Memory	Is the app using a memory safe language?	N/A
Type-Safe Language	Is the app using a type-safe language?	N/A
Formal Language	Are the programs written in a formal or safe language?	Yes (HTML, PHP)

2.2.1. Findings and Observations

The team determined that the Apache HTTP Server in use is version 2.4.38. This server is vulnerable to CVE-2019-10097(CVSS) which leads to a stack buffer overflow attack. This is a CVE with a high severity score (CVSSv3 7.2). A version upgrade has been determined to mitigate this vulnerability. The server is also vulnerable to CVE-2019-0215 which also has a high severity score (CVSSv3 7.4). This vulnerability leading to privilege escalation, was discovered in November 2018 and there is no current known patch. The team determined that the software uses specified formal programming languages. At the time of testing, it could not be determined if the application uses a type-safe or memory-safe languages, verifiable complier, or verifiable CPU core.

The Apache HTTP Server Project and Apache Software Foundation announced in June, 2022[6] that operating an Apache HTTP Server requires a "version 2.4.43 or newer to operate a TLS 1.3 web server with OpenSSL 1.1.1".

Recommendations

Software updates are not pushed to Linux systems. Patching the system by the developers and including a reminder for upgrading to newer versions to fix vulnerabilities enables the correction of critical security weaknesses. The assessment team recommends upgrading the Apache HTTP server to a more secure version and applying security and hardening best practices for better performance and security.

Also, the checklists not verifiable by the security team should be verified by the GSI and Noventum team to better assure the security of the software.

[6] https://httpd.apache.org/

2.3. Device Penetration Testing

Category	Checklist	Potential Tools	Findings
Reconnaissance	Involves both active and passive information gathering about the target system	Nmap, OpenVAS, Wireshark, Nessus, Metasploit	HTTP, HTTPS ports and accessible directories with source code were identified
Interruption	Involves obstruction to communication and rendering the system unavailable to legitimate users	Hping, Metasploit	Flood attacks to consume resources on the devices showed initial disruption to communication
Interception	Involves altering communication between two or more users or entities	Ettercap, Metasploit, Burpsuite	Curl commands to extract the index.php page and maliciously control the device was identified. ARP cache poisoning for possible eavesdropping was shown to be effective on the network
Packet Replay	Involves maliciously replaying or repeating data transmissions	Tcpdump, Tcpreplay	Tcpdump and Tcpreplay was used to capture raw network data, dumped to a pcap file, and replayed on the network causing a denial-of-service attack
Firewall	Involves identifying vulnerabilities that does not restrict ingress or egress traffic	Nmap, Hping, Hping2, Netcat	It appears there are no firewall restrictions to filter network traffic. No WAF to restrict incoming or outgoing traffic was observed to be implemented

2.3.1. Findings and Observations

A.1.1.1 Reconnaissance

Our team's penetration testing of the device began with Nmap scans on the network. Nmap was used to scan ports, fingerprint the OS, and enumerate services on endpoints to help the assessment team understand the target attack surface. Ports 80 (HTTP) and 443 (HTTPS) were identified to be open as shown in Figure 16. HTTP is an insecure protocol, but the team discovered that the HTTP requests are redirected to the secure version of the protocol which is HTTPS.

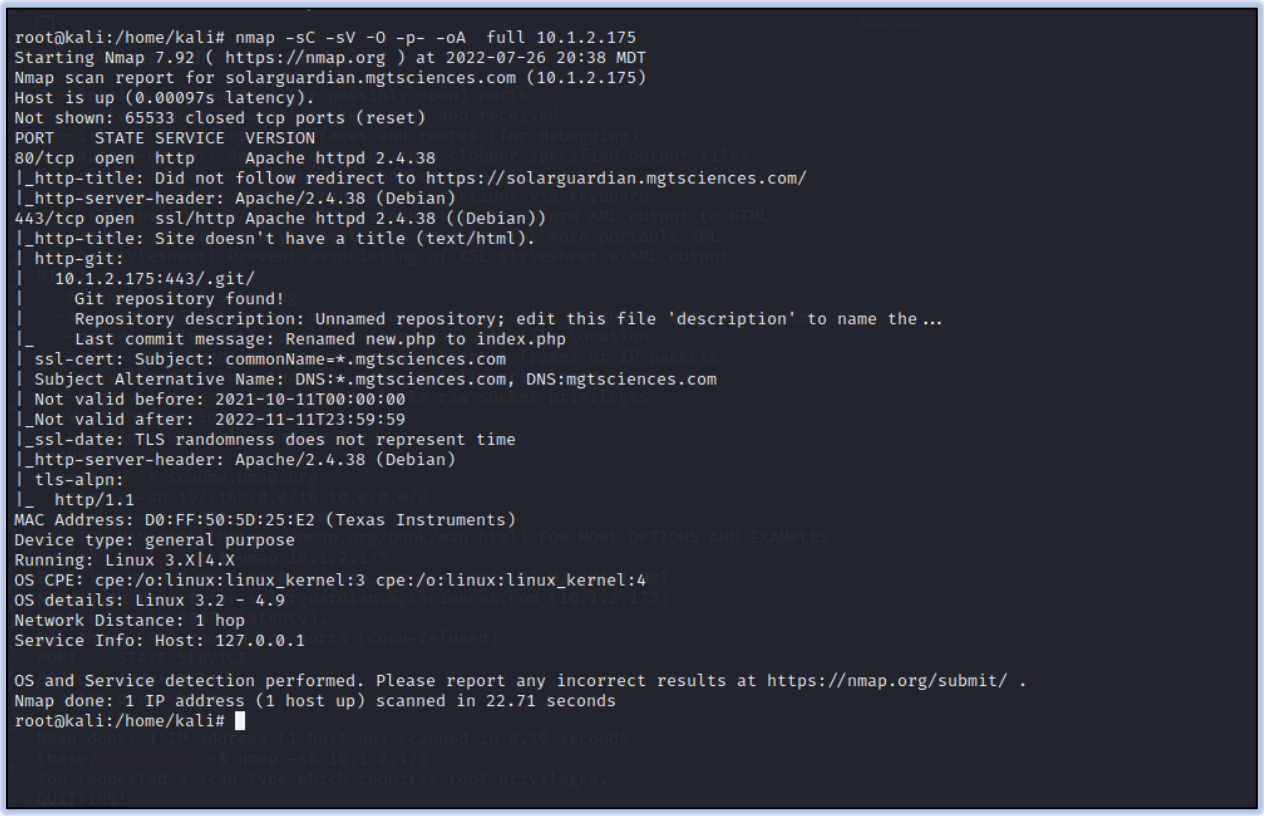

Figure 16: Nmap operating system and services detection

Wireshark was used to sniff the network traffic. As shown in Figure 17, it was observed that the device is using the recommended transport layer version (TLS v1.3) for providing network communication security.

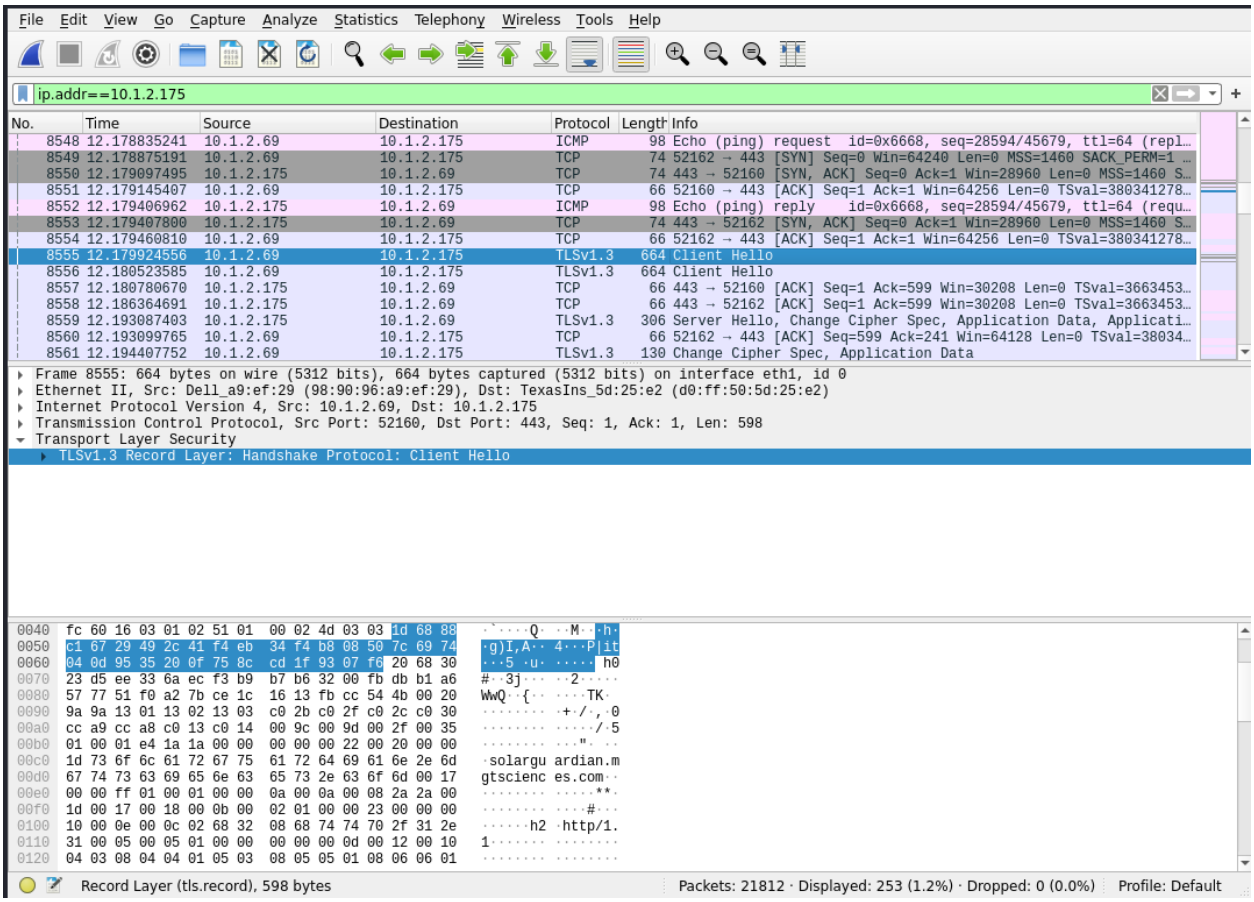

Figure 17: Transport layer security TLS 1.3

An attempt was made to use the weak TLS_PSK_WITH_AES_128_CBC_SHA suite to connect to the device to check if the device supports vulnerable ciphers such as this. This check shown in Figure 18 below, resulted in an error, which is an indication that such weak ciphers aren't supported.

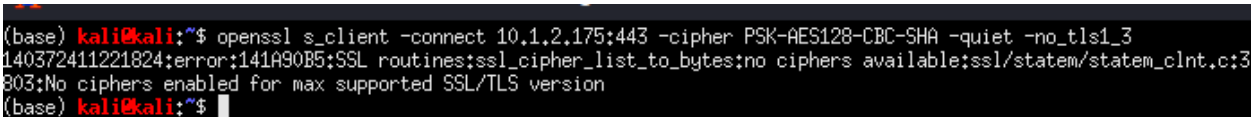

Figure 18: Using the openssl s_client command to check for weak ciphers

A.1.1.2 Interruption

A deluge of data transmissions to render the device unusable to legitimate users was orchestrated. Minimal impact to the application and operations was initially observed as depicted in Figure 19.

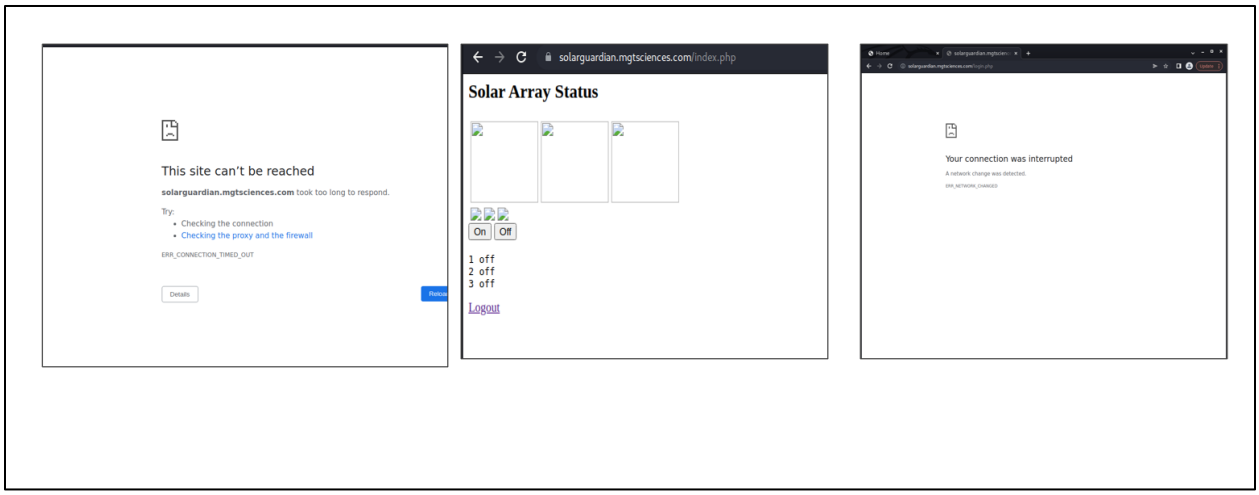

Figure 19: Interruptions to the application and device

It is termed minimal because there was some latency in loading the page during the attack. However, after a brief delay, the login.php page was available to the user to login to the application. At the time of testing, the data communication to the application was not significantly impacted during this specific denial-of-service attack. However, as noted in the packet replay section, replay of packets caused a significant DoS. Also, for this use case, a dedicated attacker can orchestrate a distributed denial of service attack using multiple devices that can significantly impact the system's operation.

Recommendation

Monitor traffic volumes, generate security alerts, and throttle traffic - using tools like Fail2Ban - when high traffic volumes are detected.

A.1.1.3 Interception

ARP Spoofing using man-in-the-middle attacks that allows an attacker to intercept communication was observed. **Error! Reference source not found.** 20 shows the Kali Linux machine used for the attack was able to trick the Solar device and the gateway router that it had the correct MAC addresses for both IP addresses so that the device and router connected to Kali instead of connecting to each other. This means that data packets between the solar device and the gateway were all routed through Kali.

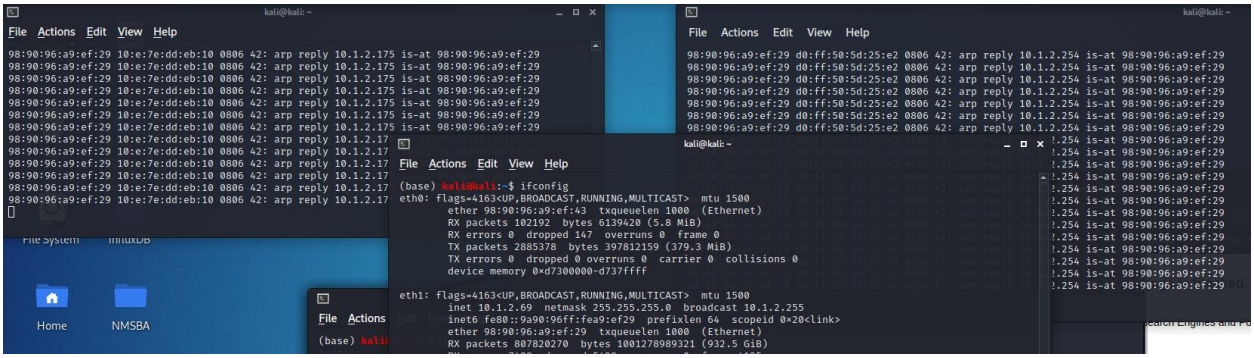

Figure 20: ARP poisoning using man-in-the-middle attack

Recommendation

Use industry best practices for preventing ARP spoofing from attackers who have infiltrated the network. ARP spoof detection and prevention is built into many commercially available network switches and should be used to prevent such attacks.

However, for this test case as shown in Figure 21, the team was not successful in using Driftnet to intercept and capture images during ARP poisoning and MITM attacks. This is because HTTP traffic was not observed due to the port redirect of the application to the more secure HTTPS protocol.

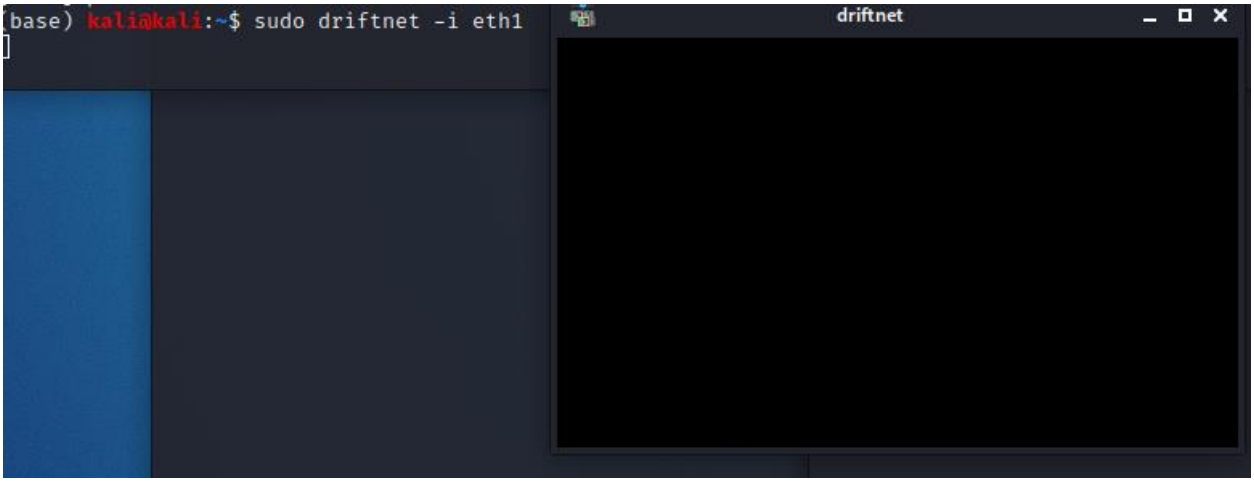

Figure 21: Driftnet launched but interception was not successful

We also had issues using Ettercap to drop or modify packets. Due to time constraints in resolving the graphical display issues being experienced at the time of testing this packet modification testing was not effectively pursued. Although HTTP port 80 was identified as open, the test tool Urlsnarf, was not able to sniff and capture any URL links from HTTP requests because as noted previously, HTTP port 80 traffic is being redirected to HTTPS port 443.

A.1.1.4 Firewall

Tools like Nmap, Hping, and firewalk to better understand the firewall restrictions on the device was carried out. All the scans that were initiated by these tools showed only two open ports. **Error! Reference source not found.** shows the results of an Nmap firewall bypass technique using a FIN scan command.

```
(base) kali@kali:~$ sudo nmap -sF -p1-1000 -T4 10.1.2.175
Starting Nmap 7.92 ( https://nmap.org ) at 2022-08-15 10:31 MDT
Nmap scan report for solarguardian.mgtsciences.com (10.1.2.175)
Host is up (0.0017s latency).
Not shown: 998 closed tcp ports (reset)
PORT    STATE         SERVICE
80/tcp  open|filtered http
443/tcp open|filtered https
MAC Address: D0:FF:50:5D:25:E2 (Texas Instruments)
```

Figure 22: FIN scan to check firewall

The team used the WAFW00F tool in Kali to determine if the there was a web application firewall (WAF) tool being used to monitor ingress and egress traffic for blocking of malicious traffic, software, and files that can infect the device. No WAF was detected as shown in **Error! Reference source not found.**.

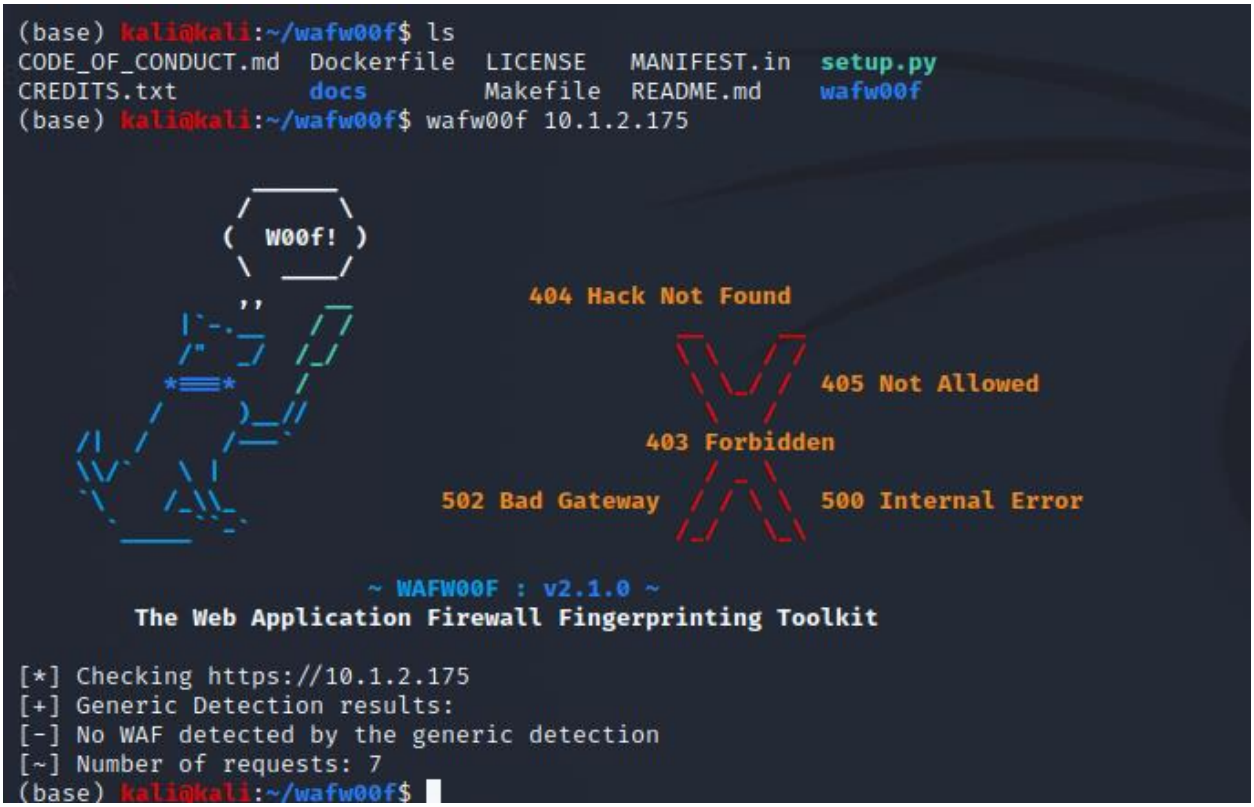

Figure 23: Firewall security not detected behind the solarguardian.mgtsciences domain

Recommendation

Given all the above enumerated and noted vulnerabilities discovered during the web application security testing, the team recommends the use of web application firewalls to prevent the exploitation of the application from common attacks like cross-site scripting (XSS).

A.1.1.5 Packet replay

The team was able to capture live network traffic using Tcpdump. Tcpreplay, was used to modify and replay the captured network traffic in the environment. Figure 24 shows the duplicated packet captures that was retransmitted to the device on Wireshark.

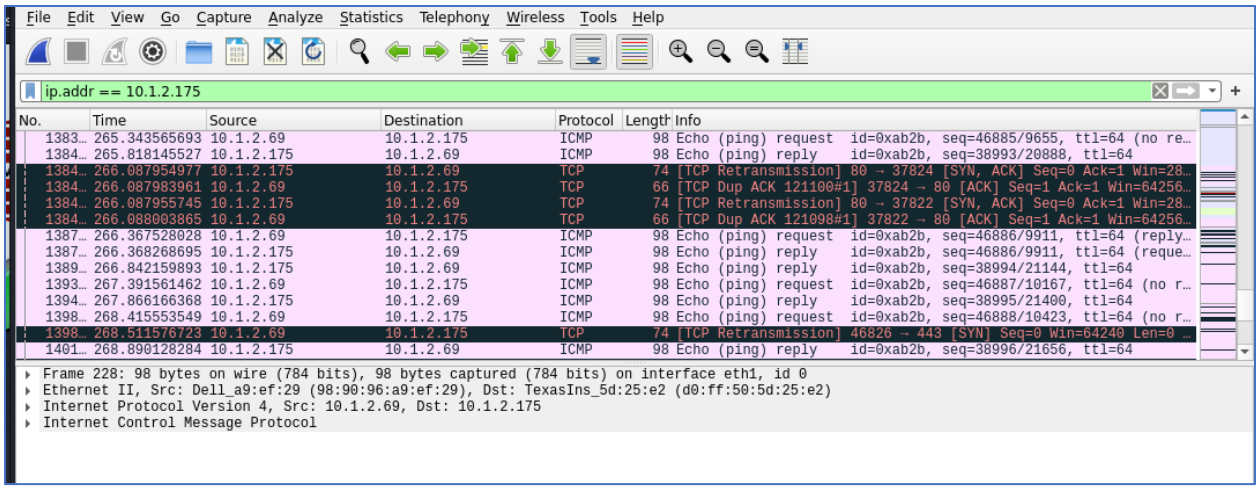

Figure 24: Wireshark capture of duplicated pcap files that was replayed

The replayed settings for Tcpreplay were customized to loop through the pcap file 100 times. This caused a denial-of-service attack on the device. This is shown in Figure 25.

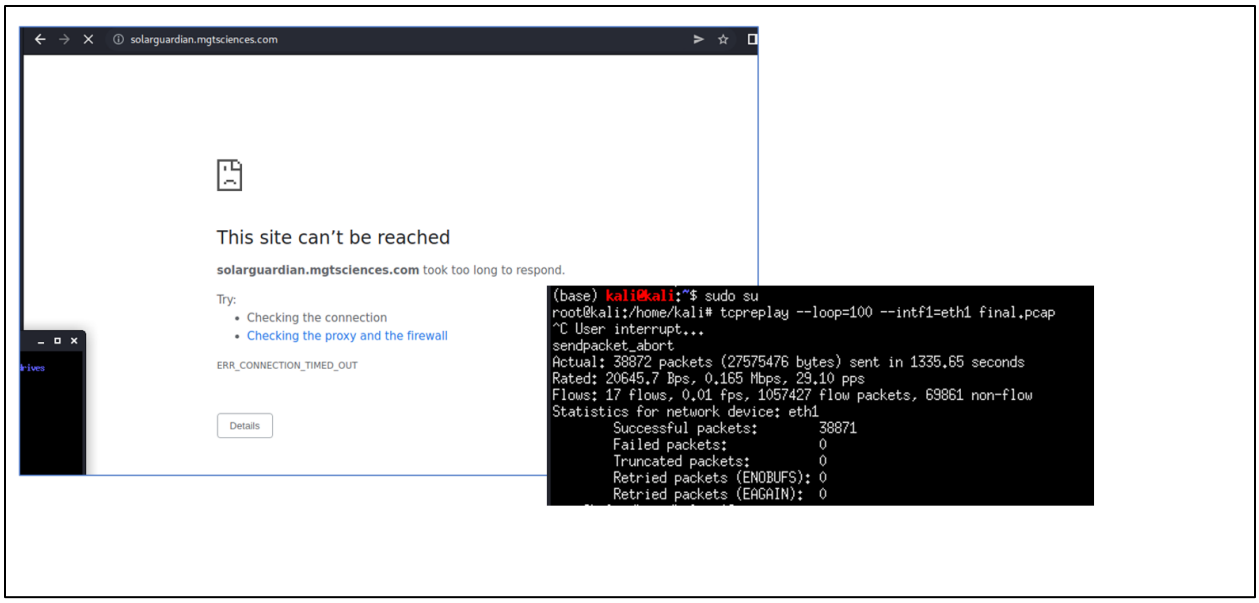

Figure 25: Replayed packets disrupting connections to the web application

Recommendation

To mitigate negative impact on legitimate connections the team recommends using industry best practices to discard duplicated or invalid packets so that the integrity of Given the vulnerabilities discovered by our attack tools during the web application security testing that is enumerated in the next section, the team recommends the use of web application firewalls to prevent the exploitation of the application from common attacks like cross-site scripting (XSS).

2.4. Serial Interface Testing

2.4.1. Findings and Observations

B 1.1.1 Reconnaissance

To monitor and analyze the serial port communication, our team physically connected the serial interface to the Kali box using a removable USB. See Figure 26 below.

```
15363859.882547] usb 2-14: new full-speed USB device number 5 using xhci_hcd
15363860.035869] usb 2-14: New USB device found, idVendor=0403, idProduct=6001, bcdDevice= 6.00
15363860.035876] usb 2-14: New USB device strings: Mfr=1, Product=2, SerialNumber=3
15363860.035880] usb 2-14: Product: TTL232R-3V3
15363860.035884] usb 2-14: Manufacturer: FTDI
15363860.035888] usb 2-14: SerialNumber: FTBQFLLX
15363860.038736] ftdi_sio 2-14:1.0: FTDI USB Serial Device converter detected
15363860.038827] usb 2-14: Detected FT232RL
15363860.039325] usb 2-14: FTDI USB Serial Device converter now attached to ttyUSB0
15448759.705957] device eth1 entered promiscuous mode
15449793.887545] systemd[1]: systemd 245.4-3 running in system mode. (+PAM +AUDIT +SELINUX +IMA +APPARM
R +SMACK +SYSVINIT +UTMP +LIBCRYPTSETUP +GCRYPT +GNUTLS +ACL +XZ +LZ4 +SECCOMP +BLKID +ELFUTILS +KMOD +
DN2 -IDN +PCRE2 default-hierarchy=hybrid)
15449793.906689] systemd[1]: Detected architecture x86-64.
15449793.943288] systemd-rc-local-generator[468245]: /etc/rc.local is not marked executable, skipping.
15634979.145486] mce: [Hardware Error]: Machine check events logged
15634979.145493] EDAC sbridge MC0: HANDLING MCE MEMORY ERROR
15634979.145498] EDAC sbridge MC0: CPU 10: Machine Check Event: 0 Bank 9: 8c000045000800c0
15634979.145500] EDAC sbridge MC0: TSC 9830568a891499
15634979.145502] EDAC sbridge MC0: ADDR ab5cfb000
15634979.145505] EDAC sbridge MC0: MISC 908404000400a8c
15634979.145510] EDAC sbridge MC0: PROCESSOR 0:306f2 TIME 1668371077 SOCKET 1 APIC 20
15634979.145535] EDAC MC0: 1 CE memory scrubbing error on CPU_SrcID#1_Ha#0_Chan#0_DIMM#0 (channel:0 pag
:0×ab5cfb offset:0×0 grain:32 syndrome:0×0 -  area:DRAM err_code:0008:00c0 socket:1 ha:0 channel_mask:1
rank:255)
```

Figure 26: Serial port connection

We were able to establish a serial connection using 115200 baud rate connection without authentication to the system console and administrative interface as a debian user. **The device assumed that any attacker who is physically connected to the serial port the first time has the right to make any configurations.** This is shown in Figure 27.

```
(base) kali@kali:~$ cu -l /dev/ttyUSB0 -s 115200
Connected.
```

```
kali@kali: ~
File  Actions  Edit  View  Help
debian@bbb06:/var/www/html$ ls
button_green_text.png  index.php            noventum_transparent.png
button_red_text.png    info.php             panels
guardian.php           login.php            solar_guardian_logo.png
html                   logout.php
index.html             msi_transparent.png
debian@bbb06:/var/www/html$
```

Figure 27: Unauthenticated access to the serial console

Recommendation

Access to the serial console is granted to anyone who can establish a serial communication to the device. There are no polices or controls to limit access either at the admin or user levels. The team strongly recommends that different levels of access control to the serial console be enabled to manage account access. Inactivity timeouts for the serial console should be enabled as well as remote event logging for incident response and forensics.

After logging out of the device, information on the console as seen in Figure 29 showed that the device was using a BeagleBone board. Subsequent connections to the device using the default username and password shown on the screen below in did not yield a successful connection. However, a brute force password attack can be used by a hacker to crack the password.

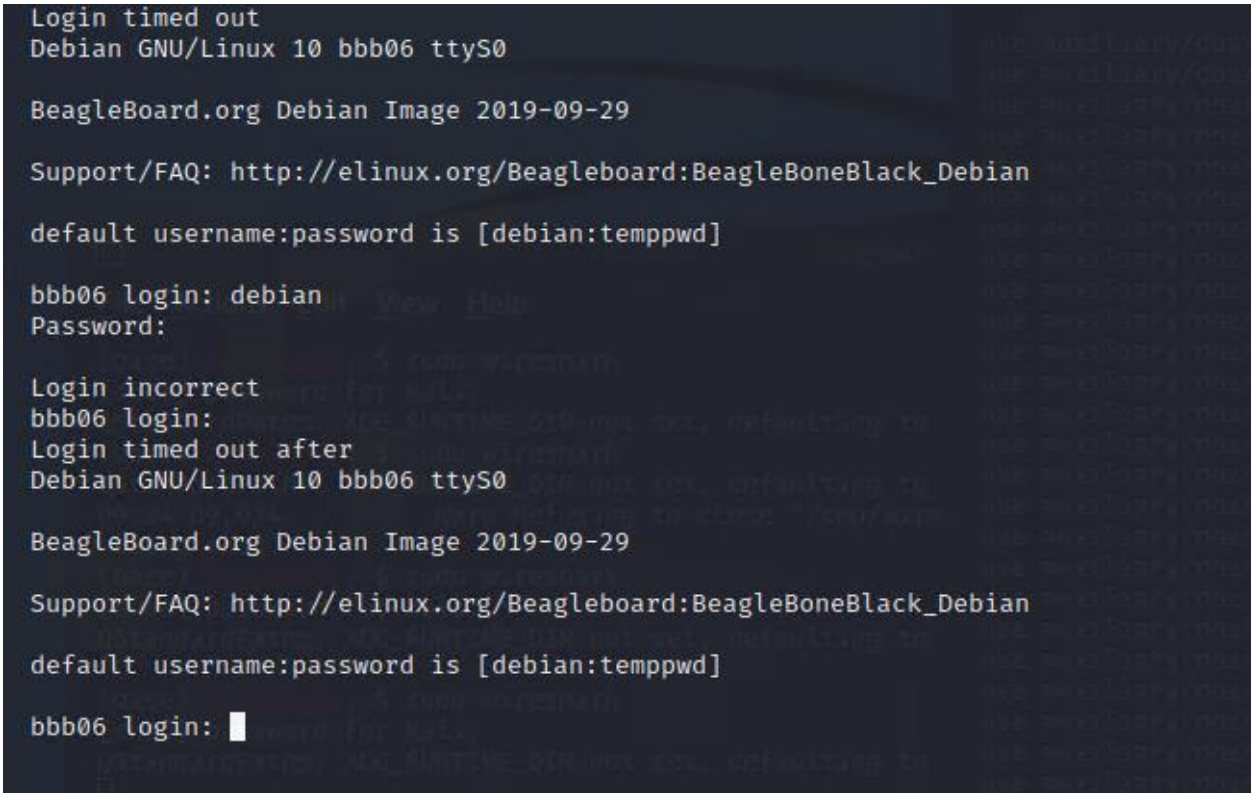

Figure 28: Serial Console login page

Recommendation

Depending on the vendor and version of BeagleBone being run, the team recommends that CVE[7] details be explored to ensure that its vulnerability statistics is low.

[7]https://cve.report/software/codesys/control_for_beaglebone_sl

3. SUMMARY

Our team conducted a cybersecurity evaluation of the application and device with respect to its use for PV systems. We performed software checks, noted key attack scenarios and top 10 OWASP web application security tests that could be used for exploitation. The assessment team recommends that the software checklists not verifiable in this report should be verified by the GSI and Noventum team to better assure the security of the software. In addition to the recommendations provided in Section 2, applying security, and hardening best practices for better performance and security of their device configurations and communications. Although not in-scope for the assessment, the team strongly recommends the use of static analysis tools to search the application's source or binary code to identify vulnerabilities or inconsistencies in the code. Finally, the team recommends a biennial security assessment for a snapshot of the security risk of the application for continuous mitigation

APPENDIX B. WEB APPLICATION CHECKS

Category	Checklist	Potential Tools	Findings
Broken Access Control	Involves exploiting access not properly enforced	Metasploit, Burp Suite	Directories and .PHP files discovered using Dirbuster and Burp Suite
Security Misconfiguration	Involves exploiting security controls that are not securely implemented	Metasploit, Burp Suite	Directories and .PHP files discovered using Dirbuster and Burp Suite
Injection	Involves sending malicious data to either disclose or corrupt data	Burp Suite	Cross-site scripting reflected
Cryptographic Failures	Involves compromising and exfiltrating unencrypted sensitive data	Burp Suite	Username and password is not encoded, clear text submission of password and username. Missing encryption.
Insecure Design	Involves exploiting design and architectural flaws	Burp Suite	Login credentials sent in plain text, access was bypassed using curl commands, .git directory, XSS vulnerability
Vulnerable and Outdated Components	Involves exploiting unnecessary features and components	Burp Suite	The latest PHP version is 8.1, current installed version is 7.3.31-1 The latest Apache version is 2.4.46, current installed version is 2.4.38
Identification and Authentication Failures	Involves compromising credentials for authentication	Burp Suite	Username and password are not encoded, clear text submission of password and username.
Software and Data Integrity Failures	Involves verifying the integrity of software update	OWASP CycloneDX	Out of the scope, OWASP CycloneDX is a lightweight Software Bill of Materials (SBOM) standard designed for use in application security contexts and supply chain component analysis.
Security Logging and Monitoring Failures	Involves verifying that relevant events are logged and stored appropriately	Burp Suite	No data stored as seen from direct access with username and password 16 alpha numeric characters
Server-Side Request Forgery	Involves exploiting user-supplied URL	Burp Suite	N/A

This page left blank

DISTRIBUTION

Email—Internal

Name	Org.	Sandia Email Address
Adam Summers	8813	asummer@sandia.gov
Aya Khalafalla	9366	amkhala@sandia.gov
Ifeoma Onunkwo	9366	ionunkw@sandia.gov
Adrian Chavez	5683	adrchav@sandia.gov
Technical Library	1911	sanddocs@sandia.gov

Email—External

Name	Company Email Address	Company Name
Yesid Jimenez	yjime030@fiu.edu	N/A
Ken Blemel	Ken_Blemel@mgtsciences.com	Guardian Sensors, Inc. (GSI)
Brian Stinar	Brian@noventum.us	Noventum

This page left blank

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.