The Ostrich Project

Brandon Creighton (cstone) & Jim Rennie

ToorCon San Diego, 2011

intro / here to talk about soho/end-user embedded devices

About Us

Brandon Creighton: I am a hacker.

- Security researcher at Veracode (static analysis)
- I also build hardware gadgets sometimes
- cstone@pobox.com

Jim Rennie: I am an attorney.

- 3 years as a Public Defender in Las Vegas
- 1+ year doing Internet privacy compliance & policy work in San Francisco

The Devices

SOHO (**S**mall **O**ffice, **HO**me) boxes: networked, single-purpose utilities

- Routers/switches
- Media players (Roku, Dreambox)
- CPE equipment (cable/DSL modems)
- VoIP adapters.... and more

- which devices are we talking about?
- what soho means
- networked, single-purpose utilities; easy to configure and use
- might say “routers” occasionally

Why?

Ubiquity

Widespread use in homes and offices:

- NAT + wi-fi
- VoIP calls
- Streaming media (pirated or not)
- Printers/scanners/97-in-one devices

used the term SOHO before, but that’s mainly an industry term that means “small”
in reality, these things are useful -- so they’re everywhere (e.g. cubicles with few ethernet ports)

Why?

Accessibility

- Easier to understand than Siemens PLCs
- Hardware samples easy to obtain
- No firmware integrity protection
- Often the only Internet-facing device on a network

if you're an attacker, they're worth paying attention to.
in the case of NAT devices, often the way in; they're commonly in privileged network positions; if you're inside a network, they're an often-ignored target (which brings me to the reason behind the name).

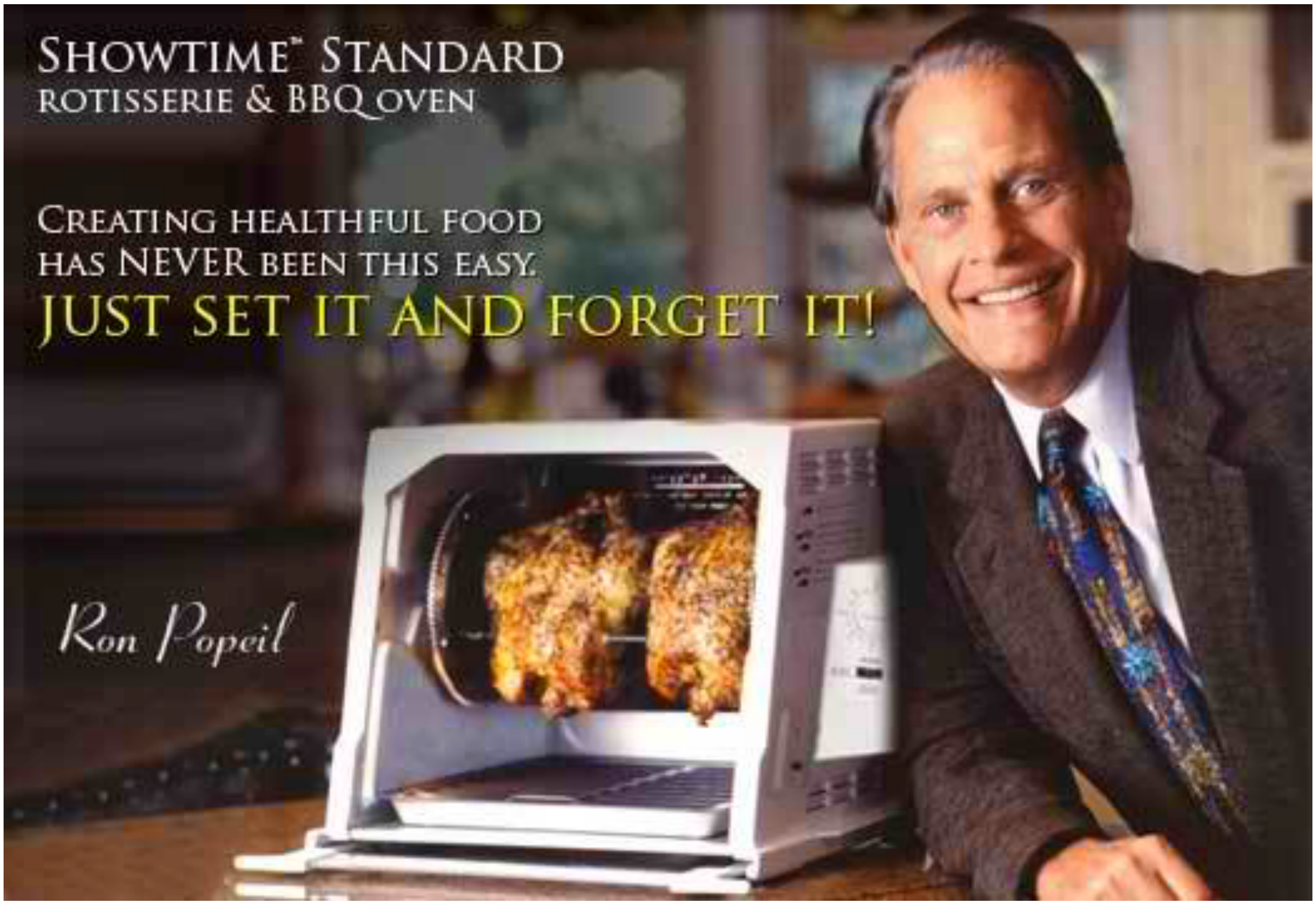
Why?

Forgotten

- Users tend to ignore them unless they're broken
- Network operators/ISPs don't always know about them
- Vendors rarely add features; no auto-update!

users "set it and forget it"; plug them in and let them run, sometimes with initial setup. it's fine as long as it's working. (e.g. 802.11 wifi password setup)
netops/isps/IT don't know about them (unlike, e.g. core routers)
vendors don't care about adding features to a single-purpose device after it's sold; no sec updates!

Sounds Like Fun



if you're like me, you'd say: that sounds like fun. is there any fun to be had on a large scale? just how widespread are these devices, and how vulnerable are they? Let's take a look at what other people have found to see if we can get some idea.

Sounds Like Fun

But how bad is it?

next i’m going to talk about a few data points out there

CSRF Vulns

- Example: CVE-2007-3574
    - Public PoC adds new admin account, opens up console to the WAN
    - Nearly zero user-visible impact (router reboot)
- References: gnucitizen, full-disclosure, RSnake (http://ha.ckers.org/blog)

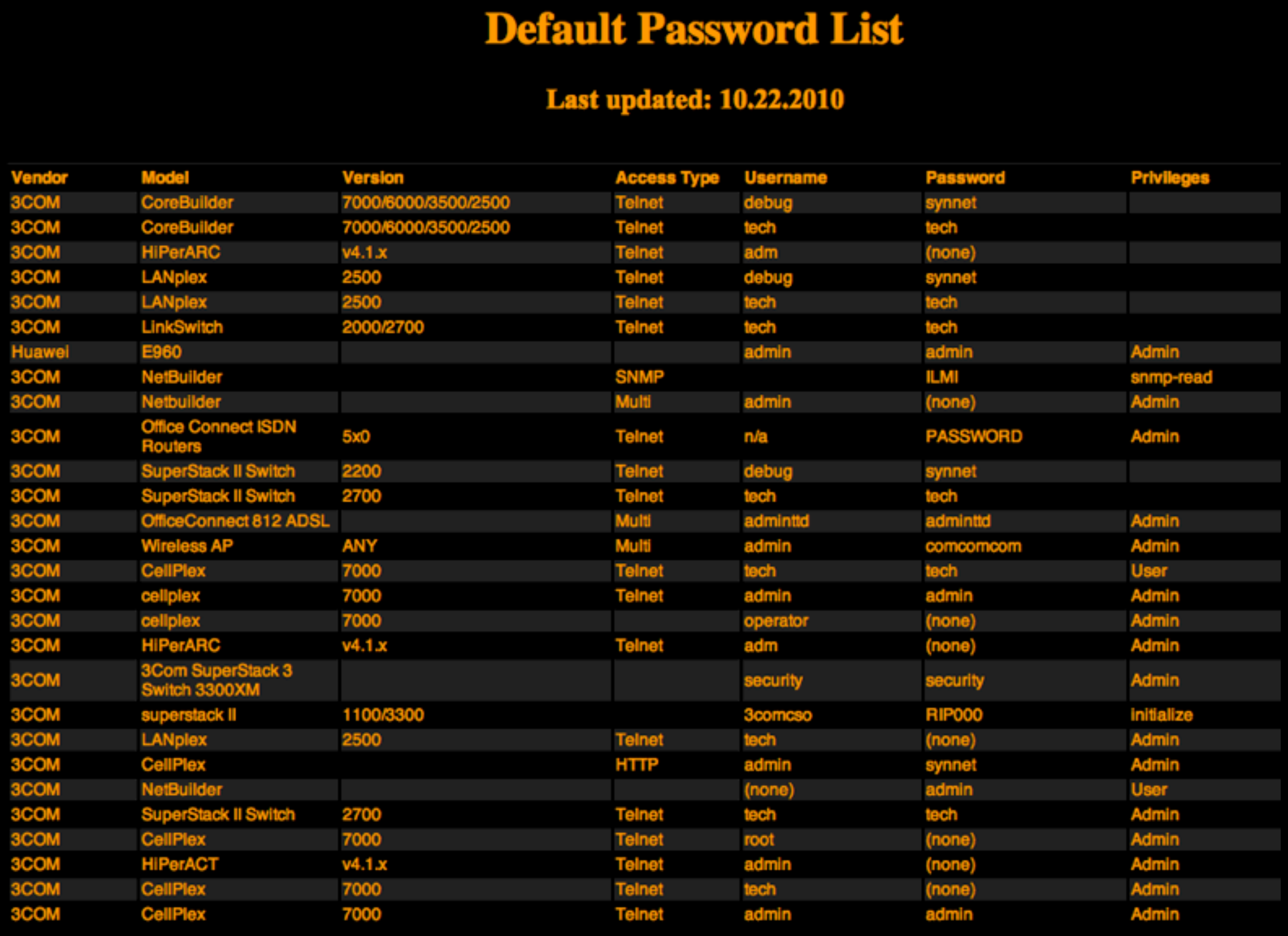
Wednesday, October 26, 2011

Since admin consoles for these devices often aren’t very complicated, it’s often possible to attack them using basic CSRF.

DNS Rebinding

- Attacker-controlled site uses JS + short DNS TTLs (or multiple A records) to reach machines on the browser's LAN
- Good References: Heffner (BHUSA2010), Kaminsky (many!)
- Result: access to device admin consoles
- OpenDNS created fixmylinksys.com

Default Passwords



Default Password List

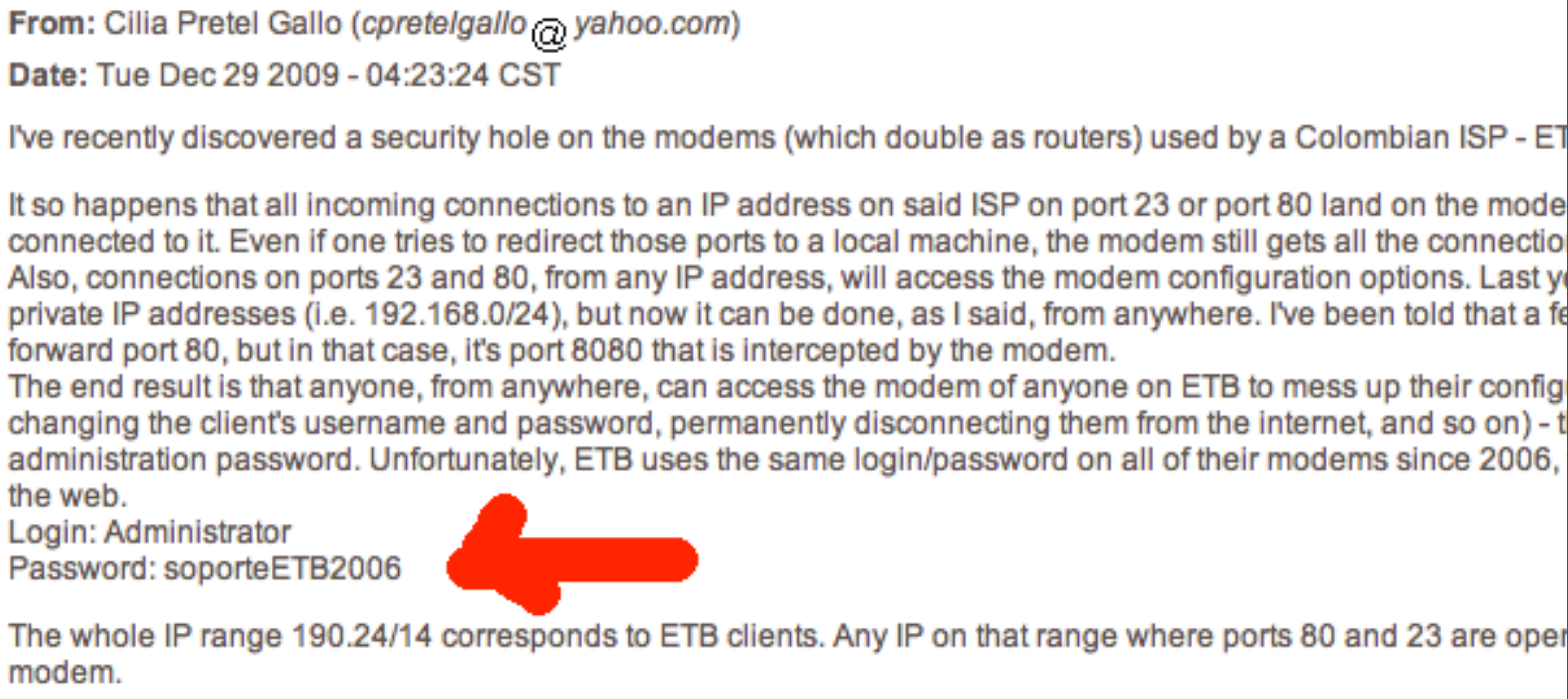
Last updated: 10.22.2010

Vendor	Model	Version	Access Type	Username	Password	Privileges
3COM	CoreBuilder	7000/6000/3500/2500	Telnet	debug	synnet	
3COM	CoreBuilder	7000/6000/3500/2500	Telnet	tech	tech	
3COM	HiPerARC	v4.1.x	Telnet	adm	(none)	
3COM	LANplex	2500	Telnet	debug	synnet	
3COM	LANplex	2500	Telnet	tech	tech	
3COM	LinkSwitch	2000/2700	Telnet	tech	tech	
Huawei	E960			admin	admin	Admin
3COM	NetBuilder		SNMP		ILMI	snmp-read
3COM	Netbuilder		Multi	admin	(none)	Admin
3COM	Office Connect ISDN Routers	5x0	Telnet	n/a	PASSWORD	Admin
3COM	SuperStack II Switch	2200	Telnet	debug	synnet	
3COM	SuperStack II Switch	2700	Telnet	tech	tech	
3COM	OfficeConnect 812 ADSL		Multi	adminttd	adminttd	Admin
3COM	Wireless AP	ANY	Multi	admin	comcomcom	Admin
3COM	CellPlex	7000	Telnet	tech	tech	User
3COM	cellplex	7000	Telnet	admin	admin	Admin
3COM	cellplex	7000		operator	(none)	Admin
3COM	HiPerARC	v4.1.x	Telnet	adm	(none)	Admin
3COM	3Com SuperStack 3 Switch 3300XM			security	security	Admin
3COM	superstack II	1100/3300		3comcso	RIP000	initialize
3COM	LANplex	2500	Telnet	tech	(none)	Admin
3COM	CellPlex		HTTP	admin	synnet	Admin
3COM	NetBuilder			(none)	admin	User
3COM	SuperStack II Switch	2700	Telnet	tech	tech	Admin
3COM	CellPlex	7000	Telnet	root	(none)	Admin
3COM	HiPerACT	v4.1.x	Telnet	admin	(none)	Admin
3COM	CellPlex	7000	Telnet	tech	(none)	Admin
3COM	CellPlex	7000	Telnet	admin	admin	Admin

Default Passwords

- Extremely common: admin/admin
- Users aren't always incentivized to change them: "but I already have a password on the wi-fi network!"
- Vendors should be ashamed of themselves

Example: ETB


From: Cilia Pretel Gallo (*cpretelgallo* @ *yahoo.com*)
Date: Tue Dec 29 2009 - 04:23:24 CST

I've recently discovered a security hole on the modems (which double as routers) used by a Colombian ISP - ET

It so happens that all incoming connections to an IP address on said ISP on port 23 or port 80 land on the mode
connected to it. Even if one tries to redirect those ports to a local machine, the modem still gets all the connectio
Also, connections on ports 23 and 80, from any IP address, will access the modem configuration options. Last y
private IP addresses (i.e. 192.168.0/24), but now it can be done, as I said, from anywhere. I've been told that a fe
forward port 80, but in that case, it's port 8080 that is intercepted by the modem.
The end result is that anyone, from anywhere, can access the modem of anyone on ETB to mess up their config
changing the client's username and password, permanently disconnecting them from the internet, and so on) - t
administration password. Unfortunately, ETB uses the same login/password on all of their modems since 2006,
the web.
Login: Administrator
Password: soporteETB2006

The whole IP range 190.24/14 corresponds to ETB clients. Any IP on that range where ports 80 and 23 are oper
modem.

in this case, someone posted an ISP-wide default password to f-d in (XXX). Turned out to affect lots of types of devices

you'd expect this sort of thing to get fixed right away, but...

Example: ETB

- Late 2010: still vulnerable devices! Why?
  - Lack of automated update infrastructure?
  - Perceived low risk?
  - Nobody involved reads full-disclosure?
  - Nobody exploiting them? (!)

perceived low risk -- lots of routers won't let you bounce off of them

Example: Dreambox

- Linux-based DVR/media player
- Default password: root (no password)
- Accessible by telnet

In Practice: Malware

- Zlob (TROJ_ZLOB.CC*): Windows trojan modifies router settings
- Primarily attacks Linksys/D-Link routers with a pre-defined list of passwords
- If successful, rewrites DNS servers to rogue IP addresses

I found one example of a Windows trojan family that tries to attack routers accessible by the host machine. It uses a pre-defined dictionary of simple passwords ("admin/admin", etc.); if it works, it redirects all DNS queries to somewhere else, presumably to better offer you the highest-quality herbal viagra pills or something similar

But How Bad Is it?

- I wanted to find out
- Idea: internet-wide survey for specific public-facing services (telnet, tftp, http); passive fingerprinting for versions
- net-wide surveys common for some classes of flaws, particularly DNS and SMTP (Men & Mice, djb, The Measurement Factory, others)

mention surveys

But How Bad Is it?

- Reality: hard to do precise version fingerprinting

the reality is that it’s hard to do version fingerprinting on a lot of these devices that matter; you need to enter commands.
and often, to enter commands, you need to log in.

But How Bad Is it?

- Reality: hard to do precise version fingerprinting
- So: can I try logging in?

Trying 1-2 default passwords for a given device would let me verify that the extent of the issue in the real world.
- even without commands
- it turns out, there can be some problems with logging into other people’s devices..

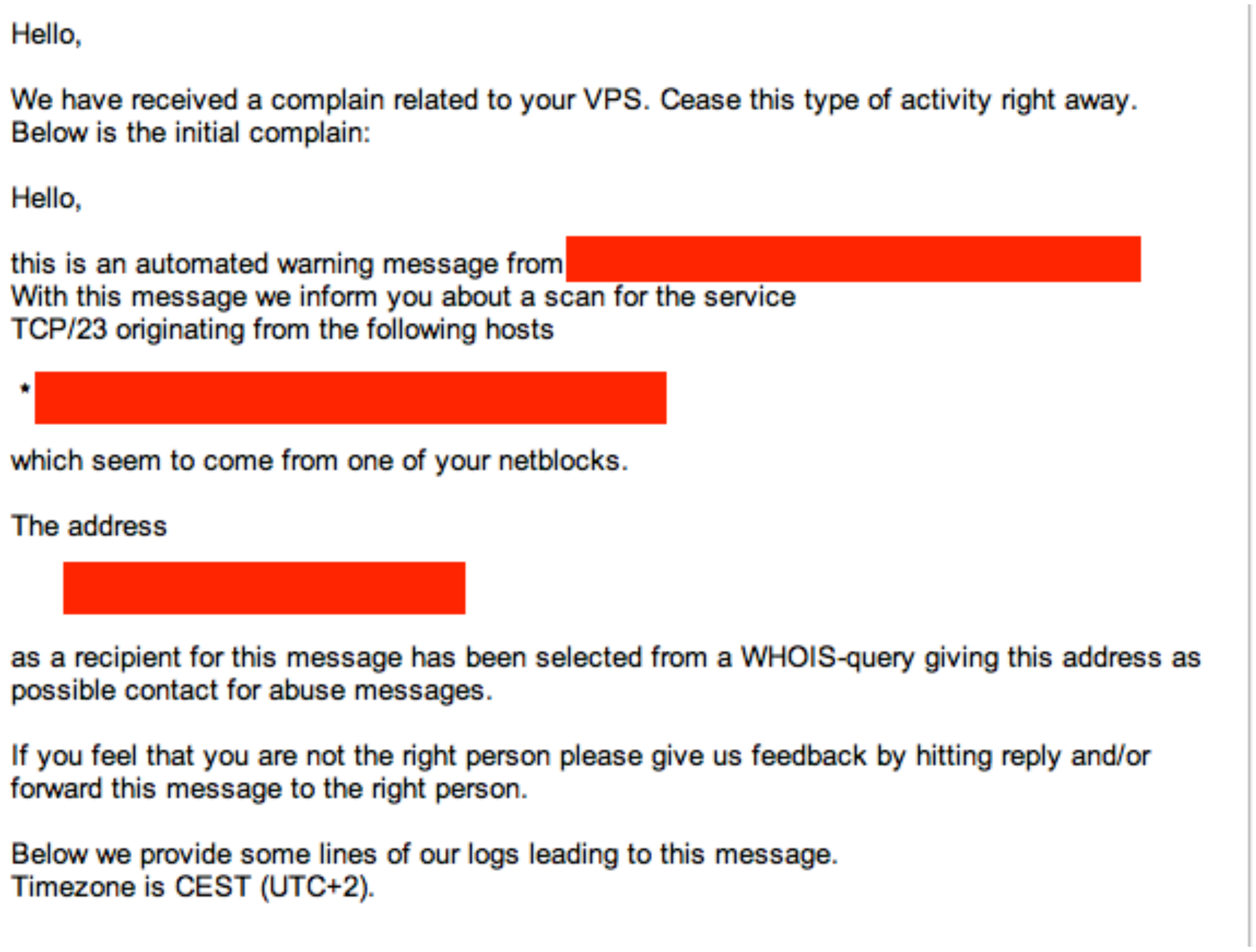
Jim Rennie

Surveys: not so good

- Can’t scan and get good results (if you’re in the US or scanning US machines)
- Logistical problems anyway:

it turns out that in 2011, there are still people who will send you or your ISP an email if you send one of their IPs a *single* SYN packet.

Busted



Hello,

We have received a complain related to your VPS. Cease this type of activity right away.
Below is the initial complain:

Hello,

this is an automated warning message from [redacted]
With this message we inform you about a scan for the service
TCP/23 originating from the following hosts

* [redacted]

which seem to come from one of your netblocks.

The address

[redacted]

as a recipient for this message has been selected from a WHOIS-query giving this address as possible contact for abuse messages.

If you feel that you are not the right person please give us feedback by hitting reply and/or forward this message to the right person.

Below we provide some lines of our logs leading to this message.
Timezone is CEST (UTC+2).

it's not really about getting sued, but more about explaining that "i'm not hacking, i'm doing security research"

Controversial Opinion

- There is an unknown amount of vulnerable, accessible devices out there
- There is little to no pressure on vendors to improve security on these devices
- Result: Vendors stick their heads in the sand

What Now?

- Even though quantifying the problem is hard, the problems are there
- What should we do about it?

What Now?

- Even though quantifying the problem is hard, the problems are there
- What should we do about it?

I started building rootkits.

why rootkits? because they’re fun

Goals

- Transparent to end-users (inject into existing firmware)
- Network traffic interception and manipulation
- Access to any secrets stored on the device
- Remote control

i think an ideal rootkit for this class of devices would be transparent to end users; this means that if there's someone there who bothers to look at the version string, it'll be the same. it means little to no downtime or degraded performance. intercept as well as manipulate packets. and i want it all to be controlled remotely

Others' Work

- Enterprise router work
  - Sebastian Muniz wrote a PoC rootkit for IOS devices in 2008
  - Graeme Neilson wrote PoC rootkits for several enterprise routers (CSW2011): http://www.aurasoftwaresecurity.co.nz/Publications/wtrc.pdf

Before I talk about what I did, I'd like to briefly mention some other research; after all, I'm not the first person to consider adding a trojan to an embedded device.

within the security community: cisco ios work

Others' Work

- Broader hacking community
    - Third-party firmware: DD-WRT, Tomato, RockBox, et al.
    - http://www.devttys0.com/: Awesome blog on reversing/testing routers
    - Etherpuppet: userland sniffer proxy for Linux-based embedded routers

there's also a large community of people who make and use third-party firmware on home devices; these are guys and gals dedicated to improving the functionality of these devices. they're really smart; you can learn a lot about how some devices work based on what they've done

Ostrich Overview

- Ostrich aims to be a portable, extensible framework for building embedded rootkits
- Written in C, initial release here
- Two-layer arch: packet manipulation (PML) and command-and-control (OCTRL)
- Separation of machine-dependent from machine-independent code

so here i'm presenting ostrich, a portable rootkit framework. i'm releasing two alpha-quality ports today to github, as well as documentation, and i'll be working on it more in the coming weeks and months.
portability is achieved by having a common set of commands for both

Functionality Goals

- Manipulate packets passing through routers
  - Programmatically and interactively (active MITM)
  - Divert packets for analysis elsewhere
  - Flexible filters with accessible state

But what exactly can ostrich do? When I started this, I wrote up a long list of things I thought it'd be fun to do. I wanted to be able to rewrite packets moving across routers both automatically (e.g. zlob/sslstrip); or interactively (e.g. through an active MITM attack). i wanted remote sniffing.

PML Architecture

- PML is a simple bytecode VM for manipulating packets across interfaces in a device
- Evades complex rulesets
- Not the first packet VM: BPF (used in *BSD, including OS X) is used to match packets today

I decided to accomplish all of the above by using a bytecode VM. This allowed me to avoid having to design a custom set of rules. Many of these devices, of course, have their own internal firewall rules; but they're not all compatible, and I decided that trying to unify them all wouldn't be worth the effort. By pushing some logic to the client, I was able to make the code fairly simple and small as well, which is important for embedded devices. I was inspired by the utility of BPF, which is a kernel-level mechanism for filtering packets. if you use libpcap filters inside tcpdump or wireshark, you're probably using BPF.

PML Architecture

- A PML program processes every packet that transits the device
    - Command packets may be excluded
    - PML program itself chooses whether to drop the packet; if you want to MITM, simply send elsewhere and rewrite

Ostrich devices can have zero or one PML programs currently defined. When defined, they're run for every packet that transits the device. In the case of packets to or from the command layer, or packets originated from PML itself, this can be skipped. The program itself chooses whether or not to drop the packet.

PML Machine

- Operands:

```
A: accumulator     (32-bit)
X: index/GP        (32-bit)
Y: index/GP        (32-bit)
M[]: memory store, seeded with data
P[]: packet
PC: program counter
channels: (address,port,type) pairs
```

- No built-in stack or function calling
  - but jumps save PC in Y; you can do it if you really want to
- Like BPF, PML is a Harvard architecture

This slide is a general overview of the data available inside PML. There are three general purpose registers, each 32 bits wide. There's a variable-length memory area called M that persists between packets. There's P, which is access to the packet itself. And there's PC, which is a program counter.

There's no instruction-implemented stack; but the jump functions will save the current PC in Y, so you can use state in M and implement methods if you want.

Both PML and the control layer use the concept of channels to store information about address to send data to. For example, if you'd like to send data as a packet out to a remote host from within PML, you define a channel using the control layer.

Like BPF, PML is a Harvard architecture. This means that code and memory are separate; you're not allowed to jump to code, and you're not allowed to treat code as data. Programs may be modified on the fly; but that's a function of the control layer.

PML Instructions

- Basics: `MOV`, `MOVS` (for special values: lengths of P/M, header offsets, PC, etc.)
- Data munging: `INSERT`, `DELETE`, `COPY`, `FIND`
- Arithmetic: +, -, &, |, ^, <<, >>
- Jumps: `JMP`, `J{GT,LT,GE,LE,EQ,SET}`
- Packet diversion: `DIVERT`
- Misc.: `SETFLAG`, `CHECKSUM`, `EXIT`

Here’s the instruction set. As you can see, you can move among the various data areas; you can send packets out with DIVERT; you can search for strings with FIND; you can jump around conditionally, back and forth; and you can even checksum regions of code.

Sniffing UDP packets

```
Label   PC    Instruction

start:
         0    MOVS A, IP4TLH_OFF #  A <- off of transport layer
         6    JEQ  0, doexit     #  if A == 0, jump to exit
        12    MOVW A, 0          #  clear A
        18    MOVS X, IPH_OFF    #  X <- offset of IP4 header
        24    MOVB A, P[X+9]     #  A[0] <- P[X+9] (protocol byte)
        30    MOVW X, 17         #  X <- 17 (protocol UDP)
        36    JEQ  X, doexit     # if A == X, jump to divert
doexit:
              MOVS X, IPH_OFF    # X <- offset of IP4 header
        42    DIVERT P[X], 1, #ffffffff  # divert pkt to channel 1
```

Summary: BPF vs. PML

- BPF: no backwards JMP
- BPF only lets you truncate packets
- Arithmetic/bitwise logic
- One scratch register (X)
- Utility instructions for common functions (e.g. ip hdr calc)

- PML is turing-complete
- PML lets you modify, expand, contract packets
- Ditto
- X and Y
- Yes, and more (checksuming, string searching)

Control (OCTRL)

- PoC quality at the moment; pretty basic
- Not very stealthy
  - Watches for commands to a specific UDP (host,port) tuple containing a specific preshared cookie; commands are unencrypted; so APTs, stay away!

As for the control layer of Ostrich, it's not as complicated. In fact, it's a little too basic; I built it in a hurry, and as a result it's not very stealthy or full of features. It merely watches for UDP or TCP packets containing a specific string of text, called a cookie; the data following that cookie is interpreted as commands. Needless to say, this isn't the most safe way of doing things; so if you plan on using this to stay hidden, you should be aware of that.

OCTRL Functions

- Get the version
- Set the current PML program (filter)
- Get/define channel info (address,port,type)
  - only type so far: plaintext UDP
- Get/modify data in M
- Modify command IP, port, cookie

Release

- C code for Ostrich, documentation, and two implementations:
  - userland: Linux-based harness for shuttling data from one interface to a TAP/TUN if, processing with Ostrich
  - WRT150N: Linksys MIPS(el) Linux-based router; diff against GPL source tree
- Scapy client code

Release

- https://github.com/unsynchronized/ostrich
- http://unsynchronized.org/ostrich

Caveats

- In heavy development; some things are broken, and some things will change
  - This isn't ready for prime time
  - (What does "prime time" mean for a router rootkit? Uh oh...)
- No fun talking about something if you can't share it, even if it's broken

Coming

- More platforms!
  - Targeted: Netgear MR814v2 (eCOS), Grandstream HT502
- More utilities!
- More stabilities!

Thanks

- Chris Nelson / far_call / Dan Kaminsky / Aaron Sigel
- Nicole Danos

References

- Zlob trojan family: http://blog.trendmicro.com/new-zlob-rigs-routers/
- DJB surveys (SMTP versions, DNS implementation behavior): http://cr.yp.to/surveys.html
- The Measurement Factory surveys: http://dns.measurement-factory.com/
- Sebastian Muniz's IOS rootkit: http://eusecwest.com/sebastian-muniz-da-ios-rootkit.html
- Dror Shalev gave a Defcon presentation talking about issues is networked embedded devices: http://www.drorshalev.com/dev/upnp/toaster/
- Craig Heffner (BHUSA2010) gave a presentation on the state of DNS rebinding attacks, focusing on SOHO routers: https://media.blackhat.com/bh-us-10/presentations/Heffner/BlackHat-USA-2010-Heffner-How-to-Hack-

References

- CSRF attacks in routers (blog post): http://ha.ckers.org/blog/20080202/csrf-yup-its-real-folks/
- More CSRF (attack example): http://www.gnucitizen.org/blog/persistent-xss-and-csrf-on-wireless-g-adsl-gateway-with-speedbooster-wag54gs/
- Another CSF PoC: http://www.exploit-db.com/exploits/15675/
- SourceSec found several admin-access flaws in D-Link routers: http://www.sourcesec.com/2010/01/09/d-link-routers-one-hack-to-own-them-all/
    - http://www.sourcesec.com/Lab/dlink_hnap_captcha.pdf
- Graeme Neilson's CSW2011 presentation on enterprise router rootkits: http://www.aurasoftwaresecurity.co.nz/Publications/wtrc.pdf

References

- http://www.devttys0.com/ is a fantastic resource on reverse-engineering embedded devices; they've released several small tools (particularly binwalk) for investigating device firmware. There's a blog too!
- Etherpuppet is a userland program that routes traffic from one interface to a TUN/TAP interface; it's designed for use in Linux routers. http://www.secdev.org/projects/etherpuppet/
- BPF: The Berkeley Packet Filter is a kernel-level device that runs a bytecode VM not that different from Ostrich's; your best place to learn about it is in the bpf(7) manpage on your nearest BSD-based system, or here: http://www.freebsd.org/cgi/man.cgi?query=bpf&apropos=0&sektion=0&manpath=FreeBSD+8.2-RELEASE&arch=default&format=html