SO HOpelessly Broken: The Implications of Pervasive Vulnerabilities in SOHO Router Products.

Jacob Holcomb
Associate Security Analyst
Independent Security Evaluators
Speaker Information

• **Who?** Jacob Holcomb  
  Twitter: @rootHak42  
  Blog: http://infosec42.blogspot.com

• **What?** Security Analyst @ ISE

• **Why?** I <3 exploiting computer code
Why is this information relevant to you?

- Everyone in the audience is a consumer of SOHO networking equipment.

- **100%** of routers we evaluated were vulnerable to exploitation.
Acknowledgements

• **Independent Security Evaluators**
  – Jacob Thompson, Alex Morrow, Stephen Bono, and Kedy Liu

• **Paul Asadoorian – PaulDotCom**
  – SANS Webcast: Hacking Embedded Systems (No Axe Required)

• **Craig Heffner -** [http://www.devttys0.com/](http://www.devttys0.com/)
  – Great resource for embedded device hacking
• Independent Security Evaluators
  – Exploiting SOHO Routers -
    http://securityevaluators.com/content/case-studies/routers/
    soho_router_hacks.jsp
  – Exploiting SOHO Router Services -
    http://securityevaluators.com/content/case-studies/routers/
    soho_service_hacks.jsp
Topics

• What are SOHO devices
• Players in the market
• Router Technology
• Testing Methodology
• Exploit Research and Development
• Mitigations
Holy hole in the router, Batman!

1. CVE-2013-0126: Cross-Site Request Forgery
2. CVE-2013-2644: FTP Directory Traversal
3. CVE-2013-2645: Cross-Site Request Forgery
4. CVE-2013-2646: Denial of Service
5. CVE-2013-3064: Unvalidated URL Redirect
6. CVE-2013-3065: DOM Cross-Site Scripting
7. CVE-2013-3066: Information Disclosure
8. CVE-2013-3067: Cross-Site Scripting
9. CVE-2013-3068: Cross-Site Request Forgery
10. CVE-2013-3069: Cross-Site Request Forgery
11. CVE-2013-3070: Information Disclosure
12. CVE-2013-3071: Authentication Bypass
13. CVE-2013-3072: Unauthenticated Hardware Linking
14. CVE-2013-3073: SMB Symlink Traversal
15. CVE-2013-3074: Media Server Denial of Service
16. CVE-2013-3083: Cross-Site Request Forgery
17. CVE-2013-3084: Cross-Site Scripting
18. CVE-2013-3085: Authentication Bypass
19. CVE-2013-3086: Cross-Site Request Forgery
20. CVE-2013-3087: Cross-Site Scripting
21. CVE-2013-3088: Authentication Bypass
22. CVE-2013-3089: Cross-Site Request Forgery
23. CVE-2013-3090: Cross-Site Scripting
24. CVE-2013-3091: Authentication Bypass
25. CVE-2013-3092: Failure to Validate HTTP Authorization Header
26. CVE-2013-3095: Cross-Site Request Forgery
27. CVE-2013-3096: Unauthenticated Hardware Linking
28. CVE-2013-3097: Cross-Site Scripting
29. CVE-2013-4654: SMB Symlink Traversal
30. CVE-2013-4655: SMB Symlink Traversal
31. CVE-2013-4656: SMB Symlink Traversal
32. CVE-2013-4657: SMB Symlink Traversal
33. CVE-2013-4658: SMB Symlink Traversal
34. CVE-2013-4659: Multiple Buffer Overflows
35. CVE-2013-3365: Multiple Command Injection
36. CVE-2013-3366: Backdoor
37. CVE-2013-3367: Backdoor
38. CVE-2013-3516: Cross-Site Request Forgery/Token Bypass
39. CVE-2013-3517: Cross-Site Scripting
40. CVE-2013-3093: Cross-Site Request Forgery
41. CVE-2013-3094: Persistent Code Execution
42. CVE-2013-3098: Cross-Site Request Forgery
43. CVE-2013-3099: Unvalidated URL Redirect
44. CVE-2013-3100: Multiple Buffer Overflows
45. CVE-2013-3101: Cross-Site Scripting
46. CVE-2013-4855: Symlink Traversal
47. CVE-2013-4856: Information Disclosure
48. CVE-2013-4857: File Inclusion
49. CVE-2013-4848: Cross-Site Request Forgery
50. CVE-2013-4913: Improper File-system permissions
51. CVE-2013-4914: Improper File-system permissions
52. CVE-2013-4915: Improper File-system permissions
53. CVE-2013-4916: Improper File-system permissions
54. CVE-2013-4917: Improper File-system permissions
55. CVE-2013-4918: Insecure Cryptographic Storage
56. CVE-2013-4919: Insecure Cryptographic Storage
Subject Background

• **What are SOHO network devices?**
  – Networking equipment used in small networks
  – Supplemental equipment (e.g., enterprise networks)

• **Who uses SOHO networking devices?**
  – Small Businesses
  – Home Users
  – Large Enterprises
Players in the SOHO Market

- **Vendors**
  - Linksys, Belkin, Netgear, ASUS, Actiontec, D-Link, TP-Link, TRENDnet

- **Consumers**
  - Ma and Pa (Home Users)
  - KWIK-E Mart (Small Businesses)
  - Large Enterprises
Evaluated SOHO Products

- **ASUS**: RT-AC66U and RT-N56U
- **TRENDnet**: TEW-812DRU
- **TP-LINK**: TL-WDR4300 and TL-1043ND
- **Linksys**: EA6500 and WRT310Nv2
- **Netgear**: WNR3500 and WNDR4700
- **Belkin**: N900, N300, and F5D8236-4v2
- **D-Link**: DIR-865L
- **Verizon Actiontec**: MI424WR-GEN3I
Why did we choose these routers?

- Popular brands
- Popular models
- New router technology
Is this a Router or a Millennium Falcon?

• 21st Century SOHO Router Technology
  – Ability to stream digital content
  – Ability to backup networked computers
  – Network Attached Storage (NAS)
  – Network Printing
  – Cloud Ready file access
Security Risks

- Larger attack surface
- Insecure by default
- Assumption of security on the (wireless) LAN
- Poor security design and implementation
Testing Methodology

- Information Gathering
- Scanning and Enumeration
- Gaining Access
- Maintaining Access
Information Gathering

• Administration Settings
  – Default credentials
  – Management interface

• WLAN Settings
  – SSID and wireless encryption

• Network Service Settings
  – DHCP, DNS, SNMP, UPnP, SMB, FTP, etc.
Scanning and Enumeration Cont.

**Port Scan**

**TCP:** `nmap -sS -Pn -sV -p T:1-65535 192.168.1.1`

```
root@Hak42:/# nmap -sS -Pn -sV -p T:1-65535 192.168.1.1
```

Starting Nmap 6.25 ( http://nmap.org ) at 2013-07-28 18:25 EDT
Nmap scan report for Wireless_Broadband_Router.InfoSec42 (192.168.1.1)
Host is up (0.0053s latency).
Not shown: 65524 closed ports

<table>
<thead>
<tr>
<th>PORT</th>
<th>STATE</th>
<th>SERVICE</th>
<th>VERSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>23/tcp open</td>
<td>tcpwrapped</td>
<td>Verizon FIOS Actiontec</td>
<td>http config</td>
</tr>
<tr>
<td>80/tcp open</td>
<td>http</td>
<td>Verizon FIOS Actiontec</td>
<td>http config</td>
</tr>
<tr>
<td>992/tcp open</td>
<td>ssl/tcpwrapped</td>
<td>Actiontec TR069 remote access</td>
<td></td>
</tr>
<tr>
<td>2555/tcp open</td>
<td>unknown</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2556/tcp open</td>
<td>unknown</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4567/tcp open</td>
<td>http</td>
<td>Verizon FIOS Actiontec</td>
<td>http config</td>
</tr>
<tr>
<td>8008/tcp open</td>
<td>unknown</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8443/tcp open</td>
<td>ssl/http</td>
<td>Verizon FIOS Actiontec</td>
<td>http config</td>
</tr>
</tbody>
</table>

**UDP:** `nmap -sU -Pn -p U:1-65535 X.X.X.X`

```
root@Hak42:/# nmap -sU -Pn -p U:1-65535 X.X.X.X
```

**Banner Grab**

**Netcat:** `nc -nv <X.X.X.X> <port>`

```
root@Hak42:/# nc -nv 192.168.1.1 8080
(UNKNOWN) [192.168.1.1] 8080 (http-alt) open
GET / HTTP/1.1
HTTP/1.1 200 OK
Content-Type: text/html
Set-Cookie: rg_cookie_session_id=1476875494; path=/;
Cache-Control: no-cache,no-store
Pragma: no-cache
Expires: Sun, 28 Jul 2013 22:33:39 GMT
Date: Sun, 28 Jul 2013 22:33:39 GMT
Accept-Ranges: bytes
Connection: close

<! [Login] </HTML> <META HTTP-EQU="NO-CACHE"> <META HTTP-EQU="PRAGMA" CONTENT="NO-CACHE" > <META HTTP-EQU="VIEW" CONTENT="NO-CACHE" >

```

`<`
Gaining Access

• **Service Investigation**
  – Analyze web applications
  – Analyze servers (e.g., FTP, SMTP, SMB, HTTP)
  – Source Code Review (Static Code Analysis)
  – Fuzz Network Services (Dynamic Analysis)
Analyzing Web Applications

• **Understand the application**
  – Programming languages used
    • Server side (e.g., PHP, .NET, Python, ASP, Ruby on Rails)
    • Client side (e.g., JavaScript, HTML, JSON, Flash)
  – Protocols and APIs used (e.g., SOAP, REST)
  – Internet Media Type/MIME (e.g., JavaScript, HTML)

• **Toolz**
  – Web proxy (i.e., Burpsuite)
  – Firebug (JavaScript debugger, HTML inspection)
  – Web Crawler
Analyzing Web Applications Cont.

Burpsuite

```html
<html>
  <head>
    <meta charset="text/html; charset=UTF-8" http-equiv="Content-Type">
    <meta content="NO-CACHE" http-equiv="CACHE-CONTROL">
    <meta content="Pragma" http-equiv="Pragma">
    <title>Verizon</title>
  </head>
  <body>
    <h2>302 Moved Temporarily</h2>
    <p>
      <a href="/index.cgi?active%5fpage%5fstr=9074&active%5fpage%5fstr=page%5flog&req%5fmode=1&mimic%5fbutton=">
    </p>
  </body>
</html>
```
Analyzing Servers

• **Authentication**
  – Type (e.g., Password, Certificate)
  – Anonymous access/Weak or no credentials
  – Misconfigurations (e.g., Directory listing, permissions)

• **Encryption**
  – SSL/TLS?
  – SSH (AES, 3DES)?
Static Code Analysis

• If source code is available, **GET IT!**

• **Things to look for:**
  – Logic flaws (e.g., authentication, authorization)
  – Functions not performing bounds-checking
  – Backdoors
Vulnerable code

```c
char ttybuf[15], buf[256];
FILE *ppp_fp;
int i;

system("mkdir -p /tmp/ppp");
sprintf(buf, "echo %s %s" >/tmp/ppo/pap-secrets", nvaram_safe_get("wan_pptp_username"), nvaram_safe_get("wan_pptp_passwd"));
system(buf);

system(buf);

"Code from the TRENDnet TEW-812DRU – network.c"
Fuzzing (Dynamic Analysis)

• What happens if peculiar input is introduced?
  – A{-G42!BBB}}}{{{}/\}/}]]})+═-__-1234d`~((_.)(_.))$
  – AAAAAAAAAAAAAAAAAAAAAAAAAAAAA

• Fuzzers
  – SPIKE: generic_send_tcp X.X.X.X 21 ftp.spk 0 0
  – BED: ./bed.pl -s HTTP -t X.X.X.X -p 80
  – Metasploit Framework
  – Python!
Spike Template (*.spk)
Fuzzing

```
Gimppy@Hak42:/usr/share/spike$ generic_send_tcp 192.168.1.1 8080 http.spk 0 0
Total Number of Strings is 681
Fuzzing
Fuzzing Variable 0:0
Fuzzing Variable 0:1
Variablesize= 5004
Fuzzing Variable 0:2
Variablesize= 5005
Fuzzing Variable 0:3
Variablesize= 21
^C
```
Analyze Fuzzing Results

- **Toolz**
  - Debugger (i.e., GDB)
  - System Call Tracer (i.e., strace)

*Debugging ASUS RT-AC66U exploit*
Gaining Access Cont.

• **Reverse Engineering**
  – Router Binaries

• **Simple RE Toolz and Techniques**
  – Strings
  – Hexdump
  – Grep
  – Open source? Perform static analysis!

• **Exploit Development**
Reverse Engineering Toolz and Techniques

• **Strings:** strings –n <INT> <FILE>

```
Gimpy@Hak42:/ISE/SOHO/TP-LINK/TL-WDR1043ND$ strings -n 10 wr1043nv1_en_3_13
12_up_boot\(120405\)\bin
TP-LINK Technologies
U-Boot 1.1.4 (Mar 31 2012 - 10:40:21)
ag7100_get_ethaddr
`*** failed ***
### ERROR ### Please RESET the board ###
## Warning: gatewayip needed but not set
ARP Retry count exceeded; starting again
%d.%d.%d.%d
bad length %d < %d
```

*TP-Link TL-1043ND Firmware*
Reverse Engineering Toolz and Techniques

- **Grep**: `grep -R <string> *

*Code from the TRENDnet TEW-812DRU*
Exploit Development

• Cross-Site Request Forgery
• Command Injection
• Directory Traversal
• Buffer Overflow
#define: CSRF is an attack that forces an unsuspecting victim into executing web commands that perform unwanted actions on a web application.
Testing for Cross-Site Request Forgery

- Anti-CSRF Tokens?
- HTTP referrer checking?

```html
<h1>Password Reset Configuration</h1>
<h3>Choose one of the questions in the list for each question, then provide an answer. You will have to answer password.</h3>
<h2>Challenge Questions</h2>
<form id="Form1" method="POST" name="PasswordQuestions" style="margin:0" action="">
  <input type="hidden" value="18z2q5m5j7m5v4iufksyioh0e3bycnytr6wdq7dsnns4hfvro" name="1k8lin552kl9o0tc">
  <input type="hidden" value="submit" name="submitted">
  <input type="hidden" value="false" name="isSimpleResetEnabled">
</form>
```
Cross-Site Request Forgery Countermeasures

• Users
  – Logout of web applications
  – Do NOT save credentials in your browser

• Developers
  – Implement Anti-CSRF tokens AND HTTP referrer checking
Command Injection

#define:
Command Injection is a form of attack where operating system specific commands are injected into a vulnerable application for execution.
Testing for Command Injection

• **Survey the application**
  – Look for application features that could call underlying system functionality (e.g., ping, traceroute)
  – Source code? Static analysis!

• **Test Examples**
  – `ifconfig ; cat /etc/passwd` ← Linux
  – `dir | ipconfig` ← Windows/Linux
  – `ls /var/www/`<cmd>` or `$(<cmd>)` ← Linux*  
    *Command substitution
Command Injection – Vulnerable Code

```php
$dig=shell_exec("dig ${_GET['Domain']}");
echo($dig);
?>
```

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats
```

```
Server,x,:/nonexistent:/bin/false messagebus:x:102:106:/var/run/dbus:/bin
/usbmux:/bin/false miredo:x:105:65534:/var/run/miredo:/bin/false ntp:x:1
/run/avahi-daemon:/bin/false pulse:x:109:117:PulseAudio daemon,,:/var/
/bin/false postgres:x:112:123:PostgreSQL administrator,,:/var/lib/postg
/lib/snmp:/bin/false stunnel4:x:116:127:/var/run/stunnel4:/bin/false sta
```

```
gdm:x:120:131:Gnome Display Manager:/var/lib/gdm3:/bin/false Gimpyp:
```
Command Injection Countermeasures

• Developers
  – Avoid calling shell commands when possible
  – If an API does not exist, sanitize user input before passing it to a function that executes system commands.

• Python Example
  – **BAD**: `os.system('ls ' + dir)`
  – **GOOD**: `os.listdir(dir)"
DEMO

- CSRF and Command Injection
CSRF and Command Injection Demo
Directory Traversal

**#define:** Directory Traversal is a form of attack where an attacker can access files and directories outside of the intended directory.
Testing for Directory Traversal

• **Enumerate the application**
  – Are there commands or request parameters that could be used for file-related operations?

• **URL Encoding (Web only)**
  – %2f → /
  – %2e%2e%2f → ../

• **Test Examples**
  – http://infosec2.blogspot.com/DT.php?file=../..../etc/passwd%00
  – symlink / rootfs ← SMB
Directory Traversal– Vulnerable Code

```php
<?php
if ($_GET['file'])
  $file = $_GET['file'];
include('/var/www/'.$file);
?>
```

![Error message on browser](image)
Directory Traversal Countermeasures

• **Developers**
  – Try not to use user input in file system calls
  – Perform path canonicalization (symlinks, . & .. are resolved)
  – Properly configure services
• Directory Traversal
#define: Buffer Overflows occur when a program attempts to write data that exceeds the capacity of a fixed length buffer, and consequently, overwrites adjacent memory.
Testing for Buffer Overflows

- Testing for overflows
  - Dynamic Analysis
  - Static Analysis
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char * argv[]){
    char argument[42];
    if (argc < 2){
        printf("\n[!!!] Please supply a program argument. [!!!]\n\n"); exit(0);
    }

    printf("\n[*] Gimppy's BOF code example\n");
    strcpy(argument, argv[1]);
    printf("[*] You supplied '%s' as your argument!\n", argument);
    printf("[*] Program Completed. \n");
}
```
Buffer Overflow Countermeasures

- **Developers**
  - Don’t use unsafe functions
  - Perform bounds checking
  - Compile with overflow prevention techniques
    - Canary/Stack Cookie
    - safeSEH (Windows)
    - ASLR
    - DEP
• Buffer Overflow
YIKES! What can we do?

• **Consumers**
  – Harden the SOHO device
  – Demand that vendors put more emphasis into securing SOHO networking equipment.

• **Vendors**
  – Design software using Defense in Depth
  – Abide by the principal of least privilege
  – Follow coding best practices
  – Patch management