How to set up the $NoAH$ infrastructure?

Contents

Chapter 1

Introduction

Within the *NoAH* architecture we can identify the *NoAH* core and external components. The *NoAH* core ia a distributed farm of honeypots. Inside we have both low- and high-interaction honeypots. Low-interaction honeypots serve as front-end to high-interaction ones and try to offload them from uninteresting traffic, like port scanning activities. High-interaction honeypots are instrumented machines that run virtual machines as a containment environment. We use *Argos* as our main containment environment. The components outside the *NoAH* core are honey@home and funneling/tunneling. Both of them aim at empowering people to participate to *NoAH*. Honey@home is a lightweight tool that listens an unused IP address and interacts with the *NoAH* core. All traffic directed to honey@home client is forwarded to the *NoAH* core, and processed by core honeypots. When an attack is detected, the information gathered by *Argos* is passed to the signature generator.

In this document we discuss the installation processes of all the components. The purpose of Section 2 is to describe how an administrator sets up Honey@home core. In Sections 3 and 4 we explain how to set up *Argos* so it runs on the host, and shares the same network enviroment, i.e., the standard dhcp server for the network, using the bridge interfce. First we will describe how to compile and install the emulator, and then discuss how to setup networking. Finally, Section 5 describes the installation process of the Connection Tracker Framework, a standalone application tracking the state of various network protocols up to the application layer. In combination with the attack detection component *Argos* and the Interface extension it outputs state and history information for any connection reported to carry attack traffic. The Interface will use this information to automatically generate network based signatures.

Chapter 2

How to set up honey@home core

The purpose of this Section is to describe how an administrator set up Honey@home core with easy steps. It is an effort to glue together the various components and provide a step-by-step guide for installing the components in order to support Honey@home clients. The Honey@home core consists of four components: the SSL_server, the modified honeyd, an unmodified MySQL database and the Argos honeypots. The architecture of Honey@home core is displayed at Figure 2.

MySQL database

The MySQL database needs no special configuration. The administrator that wants to setup a Honey@home core has two options. The first one is to setup his own website for registration and his own database to maintain the list of users. For this choice, the scripts for creating the database schemas are provided in the installation tarball. The second option, that is more preferable, is to configure the SSL server to connect to the central Honey@home database. Following this option, he should contact the administrator of the central database so as to get access to the database.

SSL server

The SSL server component is a daemon that handles connection of Honey@home users. It first connects to a MySQL database to verify that connected users are registered. The configuration for database connection information can be modified before installing it. Upon user verification, all received packets from users are sent to the modified honeyd. SSL server can be configured to listen to any port. As it will accept connection from users that communicate through the TOR anonymization system, it is advisable to configure SSL server listening on port 80 in order to avoid conflicts with exit policies of TOR routers. The SSL server needs root privileges in order to run as it injects raw packets to the interface with honeyd. The instructions for building the SSL server can be found at Appendix B.

Modified honeyd

Honeyd is a low-interaction honeypot with many interesting properties. Its role in the Honey@home core is to filter out all uninteresting traffic and forward the interesting traffic to high-interaction honeypots. For that purpose, we have modified the latest version of honeyd to enable connection hand-off. As both modified honeyd and SSL server are lightweight components, they can run in the same physical machine and communicate through the loopback interface. The configuration of honeyd is simple. Additionally, a sample configuration file is provided. This configuration file differs from the original honeyd configuration file in the following line:

```
add foobar1 tcp port 80 "sh scripts/dummy.sh" handoff 139.91.70.40:80
```

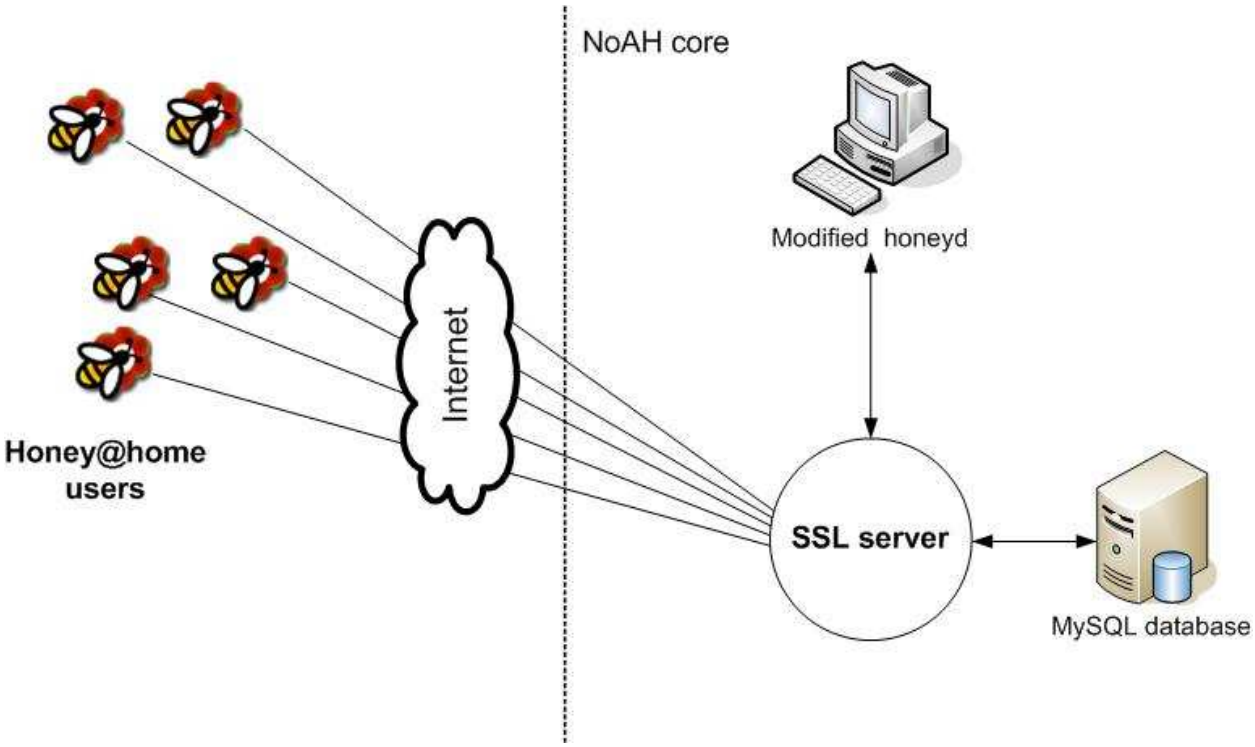

Figure 2.1: Honey@home core overview

This line instructs the honeyd to hand-off all the incoming connections for port 80 to the honeypot listening on IP address 139.91.70.40 and port 80. The administrator only has to change the IP addresses of her honeypots. Multiple hand-off instructions can exist on a single configuration file to hand-off different services to different high-interaction honeypots. By default the hand-off will take place only after honeyd has established a TCP connection. The modified honeyd emulates the whole 10.0.0.0/8. A script for bootstrapping honeyd is included in the modified honeyd distribution, under the name start.sh.

Argos honeypots

The high-interaction honeypot used in the NoAH project is Argos. Argos can accurately detect when a vulnerable application is exploited. As Argos is a heavyweight process, it is recommended to run it in a separate physical machine. Argos does not need any special configuration. Services that will be running on the operating system emulated by Argos is a local decision. More information about installing Argos can be found at Sections 3 and 4.

Chapter 3

How to set up an OS image to work with *Argos*?

In this section we will explain how to set up *Argos* so it runs on the host. We will describe how to compile and install the emulator.

The compilation and installation process of *Argos* is exactly the same as that of *Qemu*, so please consult the official *Qemu* website `http://fabrice.bellard.free.fr/qemu/` in the case of problems.

3.1 Requirements

To set up an operating system to use with *Argos*, you will need the following:

- The *Argos* package from the official website (`https://gforge.cs.vu.nl/projects/argos/`),
- A CD/DVD or a CD/DVD ISO image including the OS you want to install,
- You might need network connectivity (Section 4),
- Recent Linux kernel with support for bridge and tun,
- The SDL development libraries and headers. Binary packages are available for most Linux distributions. If one is not available for your system you download and build the library yourself. (`http://www.libsdl.org`).

3.2 Compiling *Argos*

In order to compile *Argos* perform the following steps:

1. Extract the souce code package `argos-0.2.3.tar.gz`:

   ```
   /download/directory$ tar zxvf argos-0.2.3.tar.gz
   ```

2. Configure *Argos* (and so *Qemu*):

   ```
   /download/directory/argos-0.2.3$ ./configure --target-list=‘‘i386-softmmu‘‘
   ```

3. Build *Argos* by running `make`

```
/download/directory/argos-0.2.3$ make
```

4. Become root and install *Argos*:

```
/download/directory/argos-0.2.3$ make install
```

3.3 Creating a virtual hard disk image

Qemu provides the utility `qemu-img` for this purpose. You can create a disk image using *Qemu*'s copy-on-write format with the command:

```
qemu-img create -f qcow myimage.img mysize
```

where `myimage.img` is the disk image filename and `mysize` is its size in kilobytes. You can add an `M` suffix to give the size in megabytes and a `G` suffix for gigabytes.

3.4 Installing a guest OS

Because you do not need the *dynamic tainted analysis (DTA)* feature of argos when installing a gues OS, you should use *Qemu* instead. To install Qemu download the latest archive from the project page[0] and unzip it. Type `./configure`, `make` and `make install`. Qemu is now located in `/usr/local`. Building Qemu may not work with the Gnu compiler *gcc* version 4.xx. Instead use 3.xx. E.g. change the symbolic link `gcc` in `/usr/bin` from `gcc-4.1` to `gcc-3.4`.

KQemu accelerator

For installation, you could use the Qemu accelerator *KQemu* to speed up this process. It can be downloaded at the project page of Qemu[0]. Unzip the archive, run `./configure`, `make` and `make install`. This will install KQemu into `/usr/local`. To run the kernel module `kqemu` the following commands can be used:

1. `/bin/mknod /dev/kqemu c 250 0`
2. `/bin/chmod 666 /dev/kqemu`
3. `/sbin/modprobe kqemu`

The Qemu version has to support the accelerator module. You can check this with the output of the configure script in the directory of Qemu. In the output it should say `kqemu support yes`.

Install a guest OS from CD/DVD ROM or image file

To install a guest operating system from a cd/dvd or an image the following steps are necessary:

1. `qemu-img create myOS.img 5GB`
2. `qemu -hda myOS.img -cdrom /dev/cdrom or myImage.iso -boot d`

See 3.3 for details on the first command. The second command starts Qemu with `myOS.img` as local harddisk. Because we want to mount `myImage.iso` from where we install the guest operating system, we use the `-cdrom` option. The `-boot d` option tells Qemu not to boot from hda but from the cdrom.
The cdrom option can also be used to mount software into the guest OS. If you want to create an image of a directory you can use

```
mkisofs -D -iso-level 3 -joliet-long -l -o directory.img -no-iso-translate
                        -allow-multidot -U directory
```

The content of the `directory` is now stored in the image `directory.img`. This image can be mounted with Qemu: `qemu -hda myOS.img -cdrom directory.img`. Now the operating system in `myOS.img` can access `directory.img` as a cd-rom drive.

Providing more memory for the Guest OS

If you want to use more memory for the guest OS use the following commands:

- `umount /dev/shm`
- `mount -t tmpfs -o size=528m none /dev/shm`

The last command allows us to set the memory to 512 megabytes by starting Qemu with `qemu -m 512 ...`. We have set the size to 528 mb because the tmpfs should always be slightly bigger than the memory size used for the guest OS.

3.4.1 OS specifics

All OSes *Argos* works with physical addresses. Unfortunatelly, this means that it cannot track virtual memory, and that you will have to disable virtual memory at the guest OS. In Linux, do not create (or activate) a swap partition during installation. In Windows you can disable paging after the installation completes.

All Windows OSes A Windows operating system installed using the previous versions of *Argos* (earlier than 0.2.0), cannot be booted using the latest version (0.2.0). The reason behind this being that versions 0.2.0 and later emulated and different IDE driver, and as a result Windows installations based on older versions do not contain the appropriate driver.

Windows 2000 When installing Windows 2000 you will have to enable the following option `-win2k-hack`. This is to overcome a Windows 2000 bug during installation, which causes a disk full problem. When the installation completes you do not need this option any more.

Windows XP Some Windows XP versions install correctly but a security error when booting: *A problem is preventing Windows from accurately checking the license for this computer. Error code: 0x800703e6.* The only solution for now is to install Windows XP Service Pack 2. It is also possible that the installation procedure might freeze, so we strongly recommend that you also use the `-win2k-hack` even when installing Windows XP.

3.5 Post-installation

After installing and configuring the guest OS to your liking remove the `-cdrom` and `-boot d` options when running *Qemu* or *Argos*.

We recommend that you use the `-snapshot` option, which forces *Qemu* and *Argos* to open the disk image as read-only, writing all changes to temporary files. This way you do not risk corrupting the disk image.

Summarizing, you can start *Argos* with the following command:

```
argos -hda myimage.img -m NNN -snapshot -localtime
```

Chapter 4

Argos Honeypot Network Setup

In this section we explain how to set up networking in *Argos*. However, if you come across problems, please consult the following webpages: "*Qemu* - Debian - Linux - TUN/TAP - network bridge" (`http://compsoc.dur.ac.uk/~djw/qemu.html`) and `http://www.friedhoff.org/fscaps.html#Qemu`.

4.1 Requirements

To set up the network for *Argos* you will need ethernet bridging and the tun/tap driver enabled on your Linux box. This requires that you are running the right Linux kernel, and that you have the Ethernet bridge management utilities installed.

What you need is:

- A 2.4 or 2.6 Linux kernel (`http://www.kernel.org`),
- The bridge-utils package (`http://sourceforge.net/project/showfiles.php?group_id=26089`)

In the rare case that you only wish to enable outgoing connections from the *Argos* guest, you could set up iptables and use NAT to forward connections from the guest OS to the Internet, but not the other way around. In this case you will not need the bridge utilities, but you will still need tun/tap support for your Linux kernel.

4.1.1 Linux kernel configuration

Enable the "Universal TUN/TAP device driver" in your Linux configuration. In 2.6 kernels it is located under `Device Drivers ---> Network Device Support`.

Enable the "802.1d Ethernet Bridging" option in your Linux configuration. In 2.6 kernels it is located under `Networking ---> Networking Options`.

If you would also like to firewall the Ethernet bridge there are some extra options available under `Networking ---> Networking Options ---> Network packet filtering ---> Bridge: Netfilter Configuration`.

Now you will need to recompile your kernel, and modules, and reboot your machine.

4.1.2 Bridge utilities

Binary packages for bridge utilities are provided by most Linux distributions. If it is not the case in your system, you will have to download the source from the link provided above and compile it yourself.

4.1.3 Network packet filtering and iptables

To use iptables you will need to enalbe network packet filtering in the Linux kernel, as well as have the iptables package installed in your system. To enable Network Packet Filtering in 2.6 kernels enable the options under `Networking ---> Networking Options--->Network Packet Filtering--->IP: Netfilter Configuration`.

You will need to recompile your kernel and modules.

Again, if you are unlucky enough and no iptable package is available on your system you can get it from `http://www.netfilter.org/`.

4.2 Configuration

4.2.1 *Argos* setup

First copy the file `argos-ifup` found in the *Argos* source package into `/etc`. This shell script is run to configure every virtual network interface of *Argos*. The default script shown bellow uses `sudo` (run a program as root) to add the virtual interface (`$1`) to a default Ethernet bridge (`br0`). It then enables the virtual interface.

`argos-ifup:`

```
#!/bin/sh
sudo /sbin/brctl addif br0 $1
sudo /sbin/ifconfig $1 0.0.0.0 up
```

To enable a user to run the script using `sudo` and without providing a password, you can run `visudo` as root and add the following line:

```
username ALL=(ALL) NOPASSWD: /sbin/ifconfig, /sbin/brctl
```

If you do not have `sudo` installed, or just want to run *Argos* as root, simply remove `sudo` from the above script.

4.2.2 Host setup

Before starting *Argos* you need to have an Ethernet bridge up and running at the host. Every distribution has different specifics on setting up an Ethernet bridge at start-up. Manual configuration assuming your network interface is `eth0` can be done as follows (replace `XXX.XXX.XXX.XXX` with your own IP and `XXX.XXX.XXX.1` with your gateway):

```
ifconfig eth0 down
brctl addbr br0
brctl addif br0 eth0
ifconfig eth0 0.0.0.0 promisc up
ifconfig br0 XXX.XXX.XXX.XXX up
route add default gw XXX.XXX.XXX.1 dev br0
```

When using ethernet bridging, `eth0` (or any other interface you are using to access the Internet) cannot be directly used any more. The bridge interface is used instead, as shown above.

In the same spirit, dhcp can also be used to configure `br0`.

More information on ethernet bridge can be found on `http://en.wikipedia.org/wiki/Ethernet`.

4.2.3 Host setup without bridging (iptables)

First replace the content of the `argos-ifup` script with the following:

```
#!/bin/sh
sudo /sbin/ifconfig $1 172.20.0.1 up
```

Running the script from A as root will enable NAT, and forwarding at the host so that the guest OS can access the Internet.

In the shell script A set variable `EXTIF` to the network interface that is used to access the network, and variable `INTIF` to the tun/tap interface (`tap0` unless explicitly changed).

4.2.4 Guest setup

Configure the guest OS as you would normally do. You can use both dhcp or a static IP address.

Note that in case you use iptables and not bridging you should statically assign the IP address `172.20.0.2` (has to correspond with the address used in `argos-ifup`).

The MAC address of the virtual interface used by the guest is fixed. If you need to change it you can do that by supplying `-nic,macaddr=00:11:22:33:44:55 -net tap` when running *Argos*.

If you wish to have more than one interface you will have to run *Argos* with the following options for each virtual interface: `-net nic -net tap`.

Chapter 5

Connection Tracker Framework Setup and Integration with Argos

The Connection Tracker Framework is a standalone application tracking the state of various network protocols up to the application layer. Basically, its tracks the state and history of connections to a single machine identified by its IP address. But in combination with the attack detection component Argos and the Interface extension it outputs state and history information for any connection reported to carry attack traffic. Finally, the Interface will use this information to automatically generate network based signatures. Figure 5 shows how these components are connected to each other.

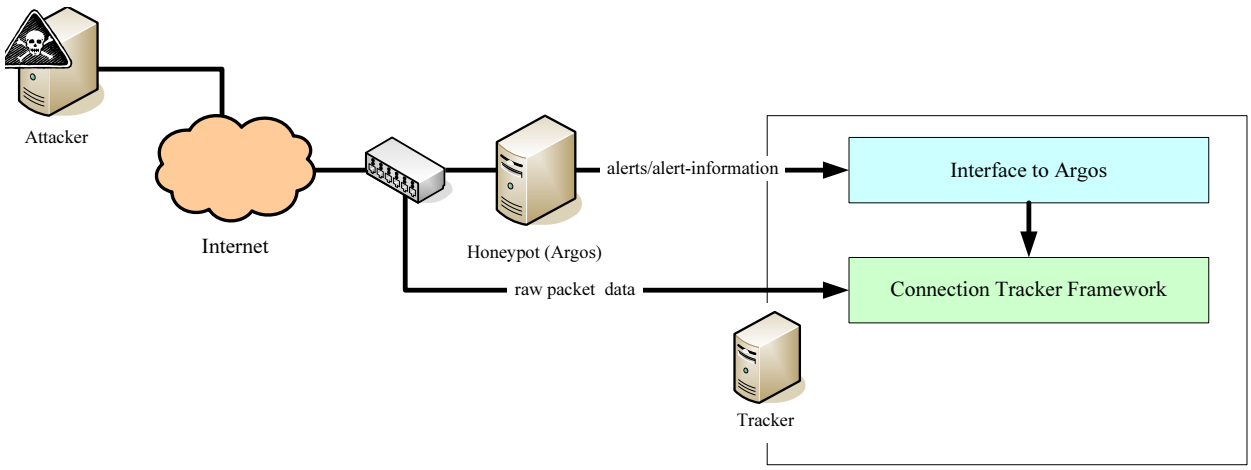

Figure 5.1: Connection Tracker Integration with Argos

Even though Argos and the Interface plus Connection Tracker Framework) can be installed on separate machines, this installation guide describes the necessary steps for an installation on on a single machine. For an installation on different machines, the Connection Tracker must have access to the logfiles created by Argos. If this is ensured, e.g., by using the Network File System (NFS) to export the directory containing these files, and if the machine with the Connection Tracker can listen in the network traffic to Argos (e.g. by using a HUB), the installation procedure is the same.

5.1 The Connection Tracker Framework

5.1.1 Installation

The installation is not yet automated but rather straightforward. The following points have to be done for a proper installation:

1. Make sure you have a version of the *libpcap*[1] installed. The Connection Tracker Framework has been tested with the version 0.8 of libpcap. This library is used for capturing the raw network packets and is used by various other projects in the network domain, e.g. by Wireshark (former Ethereal). To install libpcap under Debian and Ubuntu Linux Distributions follow these steps:

   (a) Download and install the library: `apt-get install libpcap0.8`

   (b) As the Connection Tracker is looking for the library libpcap.so, we have to create a link with this name to the libpcap version we want to use,
   e.g. `ln /usr/lib/libpcap.so.0.8 /usr/lib/libpcap.so`

   The installation of the libpcap will be quite similar on other Linux distributions.

2. Copy the Connection Tracker files into a folder.

3. If you compile the Framework the first time, type `autoreconf -i`

4. Start the configuration script: `./configure`

5. Run the Makefiles: `make`

5.1.2 Configuration

The Connection Tracker is configured by a configuration file named `trackerConfig.xml`. Later versions of the Framework might allow configuration files specified by the user. The configuration file first defines some parameters for the Connection Tracker Framework itself. The number of state tracker threads for instance allows to improve the utilization of multicore-processors or multiprocessor systems. Then the network protocols available as plugins for the framework are declared. A protocol can have other subprotocols attached. Note that the current version of the tracker is able to monitor only one host, i.e. packets are only examined if they belong to a connection with the monitored host as source or destination port.

5.2 Argos and the Interface

5.2.1 Argos

The state tracker is meant to work in conjunction with Argos although it can be run independently and without the interface. To use the Argos and the interface, you need to install Argos first. Please refer to sections 3 and 4 for the corresponding installation instructions. After doing so, Qemu or Argos should be ready to use. Nevertheless, in order to help with the networking setup and the start of Argos, we provide a set of sample scripts:

qemu-ifup A script which automates the networking setup (appendix C.1).

[1] http://sourceforge.net/projects/libpcap/

qemu-ifdown A script which roles back the changes to the network setup (appendix C.2).

start-argos A script which starts Argos (appendix C.3). Starting Qemu works similar.

Please do not forget to configure the guest OS too. The presented script in C.1 explains how it works.

5.2.2 Connection Tracker Interface

The Connection Tracker Interface links the Framework to Argos. When Argos detects an attack the Interface is notified via a socket. The Interface sends a signal to the Connection Tracker and the latter dumps its outputs. Afterwards, the Interface searches the dump and identifies the network packet(s) which have caused Argos to produce an alarm.

Installation

The interface needs the *cargos-lib* which can be downloaded at the project homepage[2]. Because the Interface is an extension to the Connection Tracker Framework, it is contained in a subfolder of the framework. Therefore, runing `autoreconf -i`, `./configure` and `make`.`autoreconf` for the Connection Tracker will invoke the makefile of the interface and create the necessary files for it too.

Configuration

The interface is configured by the file `configI.xml`. This configuration file is used to specify the IP address of the Argos control socket. In the proposed single machine configuration, this IP address corresponds to the IP address of the bridge set up during the installation of Argos (see C for an example on how to set up networking). The .netlog file of Argos will be located where Argos is started. So be aware when setting the path to the netlog file, that Argos will have to be started from the same directory. It is recommended to use a dedicated directory from which Argos is invoked to keep all session related files in a fixed place. The output and the dump file of the Connection Tracker will be located in the Connection Tracker directory.

5.3 Putting the Pieces Together

After the successful installations of all software components you can use the scripts listed in citeappendix:scripts to setup networking and to run either Qemu or Argos. Hint: Use Qemu for the installation of guest operating systems and for the installation of additional software. When Argos is running, the Connection Tracker and the Interface can be started.

[2] https://gforge.cs.vu.nl/projects/argos

Appendix A

Argos host setup without bridging. Script

This script can be also downloaded from https://gforge.cs.vu.nl/docman/view.php/14/12/network%20setup.html

```
#!/bin/sh

PATH=/usr/sbin:/sbin:/usr/local/sbin

IPTABLES=iptables
MODPROBE=modprobe

EXTIF="eth1"
INTIF="tap0"

echo -en "ip_tables, "
$MODPROBE ip_tables

# Load the IPTABLES filtering module - "iptable_filter"
#  - Loaded automatically when filter policies are activated

# Load the stateful connection tracking framework - "ip_conntrack"
#
# The conntrack  module in itself does nothing without other specific
# conntrack modules being loaded afterwards such as the "ip_conntrack_ftp"
# module
#
#  - This module is loaded automatically when MASQ functionality is
#    enabled
#
#  - Loaded manually to clean up kernel auto-loading timing issues
#
echo -en "ip_conntrack, "
$MODPROBE ip_conntrack
```

```
# Load the FTP tracking mechanism for full FTP tracking
#
# Enabled by default -- insert a "#" on the next line to deactivate
#
echo -en "ip_conntrack_ftp, "
$MODPROBE ip_conntrack_ftp

# Load the IRC tracking mechanism for full IRC tracking
#
# Enabled by default -- insert a "#" on the next line to deactivate
#
echo -en "ip_conntrack_irc, "
$MODPROBE ip_conntrack_irc

# Load the general IPTABLES NAT code - "iptable_nat"
#  - Loaded automatically when MASQ functionality is turned on
#
#  - Loaded manually to clean up kernel auto-loading timing issues
#
echo -en "iptable_nat, "
$MODPROBE iptable_nat

# Loads the FTP NAT functionality into the core IPTABLES code
# Required to support non-PASV FTP.
#
# Enabled by default -- insert a "#" on the next line to deactivate
#
echo -en "ip_nat_ftp, "
$MODPROBE ip_nat_ftp

# Load the IRC NAT functionality into the core IPTABLES code
# Required to support NAT of IRC DCC requests
#
# Disabled by default -- remove the "#" on the next line to activate
#
#echo -e "ip_nat_irc"
#$MODPROBE ip_nat_irc

echo "----------------------------------------------------------------------"

# Just to be complete, here is a partial list of some of the other
# IPTABLES kernel modules and their function.  Please note that most
# of these modules (the ipt ones) are automatically loaded by the
# master kernel module for proper operation and don't need to be
# manually loaded.
# ---------------------------------------------------------------------
#
#    ip_nat_snmp_basic - this module allows for proper NATing of some
```

```
#                        SNMP traffic
#
#    iptable_mangle    - this target allows for packets to be
#                        manipulated for things like the TCPMSS
#                        option, etc.
#
# --
#
#    ipt_mark       - this target marks a given packet for future action.
#                     This automatically loads the ipt_MARK module
#
#    ipt_tcpmss     - this target allows to manipulate the TCP MSS
#                     option for braindead remote firewalls.
#                     This automatically loads the ipt_TCPMSS module
#
#    ipt_limit      - this target allows for packets to be limited to
#                     to many hits per sec/min/hr
#
#    ipt_multiport  - this match allows for targets within a range
#                     of port numbers vs. listing each port individually
#
#    ipt_state      - this match allows to catch packets with various
#                     IP and TCP flags set/unset
#
#    ipt_unclean    - this match allows to catch packets that have invalid
#                     IP/TCP flags set
#
#    iptable_filter - this module allows for packets to be DROPped,
#                     REJECTed, or LOGged.  This module automatically
#                     loads the following modules:
#
#                     ipt_LOG - this target allows for packets to be
#                               logged
#
#                     ipt_REJECT - this target DROPs the packet and returns
#                                  a configurable ICMP packet back to the
#                                  sender.
#

echo -e "   Done loading modules.\n"

#CRITICAL:  Enable IP forwarding since it is disabled by default since
#
#           Redhat Users:  you may try changing the options in
#                          /etc/sysconfig/network from:
#
#                        FORWARD_IPV4=false
#                              to
```

```
#                       FORWARD_IPV4=true
#
echo "   Enabling forwarding.."
echo "1" > /proc/sys/net/ipv4/ip_forward


# Dynamic IP users:
#
#   If you get your IP address dynamically from SLIP, PPP, or DHCP,
#   enable this following option.  This enables dynamic-address hacking
#   which makes the life with Diald and similar programs much easier.
#
echo "   Enabling DynamicAddr.."
echo "1" > /proc/sys/net/ipv4/ip_dynaddr


# Enable simple IP forwarding and Masquerading
#
#  NOTE:  In IPTABLES speak, IP Masquerading is a form of SourceNAT or SNAT.
#
#  NOTE #2:  The following is an example for an internal LAN address in the
#            192.168.0.x network with a 255.255.255.0 or a "24" bit subnet mask
#            connecting to the Internet on external interface "eth0".  This
#            example will MASQ internal traffic out to the Internet but not
#            allow non-initiated traffic into your internal network.
#
#
#         ** Please change the above network numbers, subnet mask, and your
#         *** Internet connection interface name to match your setup
#


#Clearing any previous configuration
#
#  Unless specified, the defaults for INPUT and OUTPUT is ACCEPT
#    The default for FORWARD is DROP (REJECT is not a valid policy)
#
#   Isn't ACCEPT insecure?  To some degree, YES, but this is our testing
#   phase.  Once we know that IPMASQ is working well, I recommend you run
#   the rc.firewall-*-stronger rulesets which set the defaults to DROP but
#   also include the critical additional rulesets to still let you connect to
#   the IPMASQ server, etc.
#
echo "   Clearing any existing rules and setting default policy.."
$IPTABLES -P INPUT ACCEPT
$IPTABLES -F INPUT
$IPTABLES -P OUTPUT ACCEPT
$IPTABLES -F OUTPUT
```

```
$IPTABLES -P FORWARD DROP
$IPTABLES -F FORWARD
$IPTABLES -t nat -F

echo "   FWD: Allow all connections OUT and only existing and related ones IN"
$IPTABLES -A FORWARD -i $EXTIF -o $INTIF -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A FORWARD -i $INTIF -o $EXTIF -j ACCEPT
$IPTABLES -A FORWARD -j LOG

echo "   Enabling SNAT (MASQUERADE) functionality on $EXTIF"
$IPTABLES -t nat -A POSTROUTING -o $EXTIF -j MASQUERADE

echo -e "\nrc.firewall-iptables v$FWVER done.\n"
```

Appendix B

SSL server setip instructions

SSL_server was tested on Debian and Redhat Linux 9. As SSL_server needs to connect to a mysql database that maintains the keys of registered users, you may need to change the connection information for the database in db_config.h (DB_SERVER_IP, DB_SERVER_PORT, DB_DATABASE_NAME, DB_USER, DB_PASSWORD). To install the SSL_server you need:

- **Libpcap**. Available at `http://www.tcpdump.org`
- **Libnet 1.1**. Available at `http://www.packetfactory.net/libnet/`. Note for Debian users: You can install libdnet through "apt-get install libnet1 libnet1-dev"
- OpenSSL libraries. OpenSSL is available at `http://www.openssl.org/source/`. Current version of SSL_server was tested with openssl-0.9.8a. Follow the instructions of the openssl distribution for configuring and building it. Note for Debian users: For debian systems, an "apt-get install libssl-dev" will install the necessary files
- **MySQL 5.0 client libraries**. Available at `http://dev.mysql.com/downloads/mysql/5.0.html#downloads`. Note for Debian users: For debian systems, an "apt-get install libmysqlclient15-dev" will install the necessary files.

Compile the ssl_server component by simply typing make. You can run the client using the following command: ./ssl_server ¡port¿ ¡interface¿ port is the TCP port that server will listen on and ¡interface¿ is the network interface through which it will send the received packets to honeyd. If honeyd runs on the same physical machine with ssl_server, this interface will be the loopback.

Appendix C

Connection Tracker: Networking Setup Scripts

C.1 Qemu-ifup

Listing C.1: Script for bridge configuration, place it in /etc

```
#!/bin/bash
#place it in the folder /etc/

###################################
# The bridge
###################################

clear

echo //////////////////////////////////
echo // Configuring the bridge...
echo //////////////////////////////////

if ifconfig br0
then
        echo Bridge br0 exists already.
        sudo ifconfig br0 192.168.3.55 # The bridge gets this IP address
else
        echo Bridge br0 does not exist yet, I am creating it now.
        # adds a bridge br0
        sudo brctl addbr br0
        # bridge br0 with 192.168.3.55 is the gateway for the guest OS...
        sudo ifconfig br0 192.168.3.55
fi
echo Bridge-IP-Address: 192.168.3.55

###################################
# The virtual interface
###################################

echo ////////////////////////////////////////////
echo // Configuring the virtual interface $1...
echo ////////////////////////////////////////////

# $1 is the ifname-argument from argos (for us tun1)
# The bridge connects to the guest OS via this IP
# The guest OS itself doesn't know this and just sees eth0...
sudo ifconfig $1 192.168.3.50
sudo brctl addif br0 $1 # adds the interface $1 to the bridge br0
echo The interface $1 gets the IP 192.168.3.50

echo ////////////////////////////////////////////////////////////
echo // Configuration at the Guest OS:
echo // Network-Controller IP: 192.168.3.51 - Host IP
echo // Default Gateway IP: 192.168.3.55
echo // Note: The virtual tap interface is not addressed here
echo ////////////////////////////////////////////////////////////

exit 0
```

C.2 Qemu-ifdown

Listing C.2: Script for removing virtual interfaces, place it in /etc

```
#!/bin/bash
sudo /sbin/ifconfig br0 down
sudo /sbin/ifconfig $1 down
sudo /usr/sbin/brctl delif br0 $1
sudo /usr/sbin/brctl delbr br0
```

C.3 Start-argos

Listing C.3: Script that starts Argos with network capability

```
#!/bin/sh

# without the next line, the mouse won't work
export SDL_VIDEO_X11_DGAMOUSE=0

#without the next line, there would be a warning because of the frequency
sudo sh -c 'echo 1024 > /proc/sys/dev/rtc/max-user-freq'

#set access permissions
sudo chmod 666 /dev/net/tun

#without the next line, you have to start argos as root (only on kernel >2.6.??)
sudo tunctl -u user -t tap0

#start argos, snapshot mode
argos -m 256 -hda /home/user/DiplomaThesis/images/win2kEN.img -net nic,vlan=0
-net tap,vlan=0,ifname=tap0,script=/etc/qemu-ifup -win2k -csaddr 192.168.2.55

#shutdown interface
/etc/qemu-ifdown
```