Global Information Assurance Certification Paper

Using OSSEC with NETinVM

GIAC (GCIH) Gold Certification

Author: Jon Mark Allen, jm@allensonthe.net
Advisor: Antonios Atlasis

Accepted: 17 September 2010

Abstract

We've long heard OSSEC was an excellent option for HIDS, but there is a scarcity of detailed documentation on how to set up an OSSEC system. This paper will step through the installation, configuration, and use of OSSEC in a NETinVM environment. As a side effect, we will also see the benefits of having a NETinVM environment available for testing network attacks and defenses. We will very quickly review an OSSEC install (with some special considerations for the NETinVM environment), and jump right into developing an OSSEC policy. Once complete, we will develop a customized OSSEC configuration, based on our policy. To test our setup, we will launch an attack and see what OSSEC alerts are generated. Next, we will configure OSSEC to automatically respond to detected attacks and test the results.

1. Introduction

The days of installing a firewall at the "edge" of the network and monitoring traffic from a single point have long vanished into the history books. Today's security "edge" has collapsed all the way to the desktop and traffic from practically every system in the network must be monitored, analyzed, and acted on to maintain a secure posture (Cummings, 2004). This type of intense monitoring requires a combination of intrusion detection systems (IDS), event correlation, and analysis.

For many organizations, OSSEC provides a low-cost, flexible Host IDS, including the ability to respond to attacks without requiring administrator intervention. ("About OSSEC," n.d.) The architecture is simple to understand and the configuration is easy to setup and maintain. But care must be taken when implementing changes to monitoring rules and responses, as any misstep could potentially significantly disrupt normal business operations (and/or your plans for the weekend).

What is needed is a simple yet flexible lab network, where changes can be made and reverted quickly and easily. VMWare snapshots provide the ability to revert a system to a given state and host networking allows virtual machines to communicate via an "internal" network isolated from the "real" production network. But this would still require enough resources to run a full installation of each guest operating system (OS) and does not easily allow for configurable firewalls between guests. NETinVM allows an administrator that flexibility without requiring a lot of expensive hardware. NETinVM guest machines can share configurations quickly and easily, and firewall rules are managed by a full standard iptables firewall.

2. Overview of Tools

OSSEC

According to OSSEC's website, OSSEC is "a scalable, multi-platform, open source Host-based Intrusion Detection System (HIDS). It has a powerful correlation and analysis engine, integrating log analysis, file integrity checking, Windows registry monitoring, centralized policy enforcement, rootkit detection, real-time alerting and active response."

("About OSSEC," n.d.) OSSEC was originally developed by Daniel Cid and has since been purchased by Third Brigade and Trend Micro in sequence. Currently, the project averages more than 5,000 downloads per month and "is being used by ISPs, universities, governments and neven large corporate data centers as their main HIDS solution. In addition to being deployed as an HIDS, it is commonly used strictly as a log analysis tool, monitoring and analyzing firewalls, IDSs, web servers and authentication logs." (ibid.).

OSSEC utilizes a simple client-server architecture. The configuration files are standard XML files, allowing for easy comprehension and maintenance. The project is under active development, and is frequently supported via an active community mailing list, though Trend now offers some official support channels. ("OSSEC Commercial Support," n.d.)

The latest version as of this writing is 2.4, released April 1, 2010 (no, really. It's not an April Fool's joke....). The single best resource for OSSEC documentation is the online manual at http://www.ossec.net/main/manual.

NETinVM

NETinVM was announced in June of 2008 by Carlos and David Perez. They describe it on their website as "a single VMware virtual machine image that contains, ready to run, a series of User-mode Linux (UML) virtual machines which, when started, conform a whole computer network inside the VMware virtual machine. … NETinVM has been conceived mainly as an educational tool for teaching and learning about operating systems, computer networks and system and network security..." (D. Perez & C. Perez, n.d.)

The UML model allows an administrator to run multiple guests, while requiring minimal storage space and modest memory resources. The NETinVM network consists of two protected network segments and one "external" segment, all of which must route through a guest firewall (also a standard UML virtual machine) to communicate across segments. The VMWare machine has network interfaces on each UML segment, but does not route traffic between them. This arrangement makes traffic monitoring and troubleshooting simple, while still enforcing network security best practices within the lab.

The latest version of NETinVM was released in April of 2009. We will certainly cover all aspects required to understand NETinVM for this project, but a comprehensive explanation of NETinVM, including the architecture and design, are available on the NETinVM website. (ibid.)

3. Understanding the NETinVM Environment

In order to fully take advantage of NETinVM, it is necessary to spend a little bit of time more fully discussing its network architecture, file system design, and key directories.

The NETinVM Network

As stated above, the NETinVM network consists of three networks, who are interconnected via a UML firewall, `fw` (itself a UML guest). Traffic from the internal and "secure server" network (aka DMZ) are routed through the firewall out through to the SuSE linux VMWare instance, hereafter referred to as `base.example.net` or simply `base`. Each guest is named after the network segment on which it resides, i.e. `inta`, `dmza`, `dmzb`, `exta`, or simply `fw`. This is perhaps better understood by referring to the diagram in Appendix A, graciously provided by the NETinVM authors.

While NETinVM is capable of running up to 19 UML instances, our lab will only require the use of five: a firewall (`fw`), two DMZ guests (`dmza` and `dmzb`), one internal guest (`inta`), and one external guest (`exta`) used to launch attacks. We will use the default domain, example.net, as provided by NETinVM. All UML guests are running linux kernel 2.6.18 on Debian, while `base` is running SuSE 10.3, which will be our OSSEC server.

A Brief Explanation of File Storage in NETinVM

You can imagine that if NETinVM used separate files for each UML guest's file system, our VMWare machine would very quickly run out of disk space. Instead, NETinVM uses what are known as "sparse files". The concept is similar to a "growing" VMDK VMWare disk file, in that a sparse file is a "disk file that saves space by storing only actual data in the sectors and not storing consecutive runs of non-data (nulls). When

a file system supports sparse files, it saves meta-data about the file that indicates where the runs of non-data are located. The reported file size is always the size of the entire file." (PCMag, 2010)

We can see the real space used by including the `-s` option to `ls`:

```
user1@base:~/uml/data> ls -lsh uml_root_fs
755M -r--r--r-- 1 user1 users 1.0G 2009-04-01 10:25 uml_root_fs
```

The far left field shows the actual size of the data stored on disk, while the "virtual" size of the file is reported in the normal size column.

Each UML guest shares the same root file system (or "Reference File System" per the NETinVM authors), but changes to the file system are actually written to a "copy-on-write" (COW) disk instead of the original file. NETinVM stores the COW files in `/home/user1/uml/machines/<machine>/cow`. The COW files are also sparse files:

```
user1@base:~/uml/machines/inta> ls -lsh cow
46M -rw-r--r-- 1 user1 users 1.1G 2010-08-07 11:17 cow
```

This provides the ability to have different programs and configurations on each UML guest, but still maintain a low disk usage profile.

NETinVM Directory Notes

The core of the NETinVM files reside in several directories under `/home/user1/uml`. We will review items pertinent to our lab here, but if you want to dive deeper, there is a full description on the official NETinVM site.

The UML control scripts are stored in the `bin` directory. While there are several scripts there, we currently only need to make note of a few. `uml.sh` and `uml_halt.sh` are used to start and stop individual UML guests, while `uml_run_all.sh` is used to start `fw`, `exta`, `inta`, `dmza`, and `dmzb` simultaneously. The latter script is great for when all the guests have already been configured and you want to pick up on a lab where you left off. Of course, all of these are simple bash scripts and can easily be modified if desired.

The `data` directory is where the UML reference file system is stored, `uml_root_fs`. When we create a new file system for the OSSEC install, that will be stored here as well.

The `machines` directory contains subdirectories for each UML configured guest:

```
user1@base:~/uml> ls machines
dmza  dmzb  exta  fw  inta
```

Inside each of these directories is a COW and swap memory file for each guest. To reset a guest and have the initial configuration scripts re-run, simply delete the machine's subdirectory from here, like so:

```
user1@base:~/uml/machines> rm -rf inta
```

The `mntdirs` directory contains three subdirectories – `config`, `data`, and `tmp` – which are made available to each UML guest as `/mnt/config`, `/mnt/data`, and `/mnt/tmp` respectively. Changes made in these directories are shared between all UML guests.

The `/mnt/config` directory is where NETinVM stores guest and network specific configuration instructions, found in the `commands_to_run.sh` file for each location. As an example, configuration commands intended for every system in the DMZ should be placed in `network_dmz/commands_to_run.sh`, while commands specific to the firewall are in `fw/commands_to_run.sh`. This script is run the first time a "new" UML guest is started. (The guest is considered "new" if it does not have a subdirectory in the `uml/machines` directory.)

We will not utilize the `/mnt/data` directory here, and the `/mnt/tmp` directory is self-explanatory.

With these details laid out, we can begin developing an OSSEC Policy, and then we will be ready to install the OSSEC server.

4. Creating an OSSEC Policy

One of the keys to a good OSSEC install is a good policy, and a critical aspect of a good policy is knowing what you need to protect (Danchev, 2003). While in an ideal world your client (or company) would already know all the services running on the network or particular host, it has been my experience that is rarely the case. Therefore, I would like to briefly highlight the process of service discovery since I believe it is so critical to a good HIDS implementation.

There are many methods to gather a list of running services, and I highly recommend the use of more than one, as any single source may not present a complete picture of the environment. A good first step is to review the network documentation.

From the diagram in Appendix A, we see that `dmza` runs an HTTP server while `dmzb` runs an FTP service. There are no known services running on the internal network, and no version information is included in the available documentation. This could certainly represent the level of detail available in some environments and therefore highlights the need for multiple sources of information.

Once documentation has been reviewed, it would be wise to verify the documentation accurately reflects the services running in production. One method of verification is the use of netstat. A list of listening network services and their ports and processes can be obtained from each individual linux system by running netstat as root (output trimmed):

```
dmzb:~# netstat -atunp
tcp   0  0 0.0.0.0:2049    0.0.0.0:*    LISTEN   -
tcp   0  0 0.0.0.0:871     0.0.0.0:*    LISTEN   1116/rpc.mountd
tcp   0  0 0.0.0.0:3019    0.0.0.0:*    LISTEN   -
tcp   0  0 0.0.0.0:111     0.0.0.0:*    LISTEN   946/portmap
tcp   0  0 0.0.0.0:80      0.0.0.0:*    LISTEN   1259/apache
tcp   0  0 0.0.0.0:21      0.0.0.0:*    LISTEN   1288/vsftpd
tcp   0  0 0.0.0.0:2935    0.0.0.0:*    LISTEN   1212/rpc.statd
tcp   0  0 127.0.0.1:25    0.0.0.0:*    LISTEN   1176/master
tcp6  0  0 :::22           :::*         LISTEN   1193/sshd
udp   0  0 0.0.0.0:2048    0.0.0.0:*             -
udp   0  0 0.0.0.0:2049    0.0.0.0:*             -
udp   0  0 0.0.0.0:2050    0.0.0.0:*             1212/rpc.statd
udp   0  0 0.0.0.0:964     0.0.0.0:*             1212/rpc.statd
udp   0  0 0.0.0.0:868     0.0.0.0:*             1116/rpc.mountd
udp   0  0 0.0.0.0:111     0.0.0.0:*             946/portmap
```

The first portion of the output shows the TCP services, where we verify the FTP service, but also see RPC, portmap, SSH, and Apache services running. The RPC and portmap services aren't surprising for a default linux install, but in the real world should be verified with the server administrators and added to the OSSEC policy if appropriate. There is also a postfix process running (shown under the `master` name), listening only on the loopback interface and is only accessible from the local machine. In our lab environment, we will choose not to monitor this process. Under the UDP listings we see more RPC services and a portmap service, which should also be verified with the administrators and added to the policy if necessary. The same process (or equivalent) should be carried out on all monitored hosts.

Lastly, it may be a good idea to run a network scan to verify network activity and look for services listening that did not show up from the local copy of netstat. An excellent option for this is an nmap scan with version detection. Of course, if you wanted to run this in your production environment, be sure to get permission first. This again highlights one of the benefits of a NETinVM lab, as you are the only person you need permission from!

This scan targeted the DMZ network from `base.example.net`. I ran the scan as:

```
sudo nmap -sV -T5 --reason -oA netinvm-service-scan -F -O \
--script=default,discovery 10.5.1.0/24
```

the full output of which can be found in Appendix B.

In this case, the scan report (with the abbreviated results shown below) is consistent with our results from the netstat command and verifies our previous findings:

```
Nmap scan report for dmza.example.net (10.5.1.10)
22/tcp   open  ssh     syn-ack OpenSSH 4.3p2 Debian 9etch3
(protocol 2.0)
80/tcp   open  http    syn-ack Apache httpd 1.3.34 ((Debian))
111/tcp  open  rpcbind syn-ack 2 (rpc #100000)
| rpcinfo:
| 100000  2        111/udp  rpcbind
| 100005  1,2,3    868/udp  mountd
| 100021  1,3,4   2048/udp  nlockmgr
| 100003  2,3     2049/udp  nfs
| 100024  1       2050/udp  status
| 100000  2        111/tcp  rpcbind
| 100005  1,2,3    871/tcp  mountd
| 100003  2,3     2049/tcp  nfs
| 100024  1       2254/tcp  status
|_100021  1,3,4   2568/tcp  nlockmgr
2049/tcp open  nfs     syn-ack 2-3 (rpc #100003)

Nmap scan report for dmzb.example.net (10.5.1.11)
22/tcp   open  ssh     syn-ack OpenSSH 4.3p2 Debian 9etch3
(protocol 2.0)
111/tcp  open  rpcbind syn-ack
| rpcinfo:
| 100000  2        111/udp  rpcbind
| 100005  1,2,3    868/udp  mountd
| 100021  1,3,4   2048/udp  nlockmgr
| 100003  2,3     2049/udp  nfs
| 100024  1       2050/udp  status
| 100000  2        111/tcp  rpcbind
| 100005  1,2,3    871/tcp  mountd
| 100003  2,3     2049/tcp  nfs
| 100024  1       2182/tcp  status
|_100021  1,3,4   2226/tcp  nlockmgr
2049/tcp open  nfs     syn-ack 2-3 (rpc #100003)
```

Now that the running services have been determined, specific goals should be

documented to list which services or other host logs should be monitored. SSH, Apache, ftp, rpcbind, and nfs services should be monitored for all guests running these services on the example.net network. Lastly, OSSEC will monitor critical system files on all systems in an attempt to detect rootkit activity. (Normally, firewall logs are a natural and logical area to monitor and report on, but the default iptables rules on `fw` do not log any packets. You will need to modify the iptables configuration if you wish to include this area in your lab.) The NETinVM sample OSSEC policy is listed in Appendix C.

Of course, one of the advantages to running VMWare is the ability to create and restore snapshots. I recommend creating a snapshot at the start of installation, and at various points during the process as you are comfortable.

5. OSSEC Server Installation

Configuring the Firewall

The first step is to configure the iptables firewall (on `base`) to permit the OSSEC agents to communicate with the server. This is most easily done through the standard configuration utility for SuSE linux: YaST. Based on the network diagram in Appendix A, we know the agent traffic will be traversing the tap1 and tap2 interfaces on `base`. The stated purpose of this firewall, per the NETinVM authors, is to maintain the routing integrity of the lab environment, while also permitting ease of network monitoring on each segment. For this reason, `base` is configured to filter certain types of traffic from all three three tap interfaces. This is accomplished by the use of "zones" in the YaST configuration. By default, each interface is added to the external, or "`EXT`", zone on the `base` firewall.

Adding a rule to permit OSSEC traffic from the `EXT` zone would permit any guest in the NETinVM lab network to contact the OSSEC server service, including guests on the "external" NETinVM segment (10.5.0.0/24). Since this is a lab "external" segment, all devices are still under the full control of the lab administrator and this rule is seen as permissible in order to significantly ease the administrative setup of the lab. This also has the added benefit of permitting an OSSEC agent on any of the external guest virtual machines if desired.

First, create a rule to permit traffic to base on UDP port 1514 (the default OSSEC server listening port), and then restart the firewall:

```
base:~ # yast firewall services add udpport=1514 zone=EXT
base:~ # service SuSEfirewall2_setup restart
Starting Firewall Initialization (phase 2 of 2) SuSEfirewall2:
Warning: no default firewall zone defined, assuming 'ext'  done
base:~ #
```

Firewall service settings can be verified with:

```
base:/var/ossec # yast firewall services show

Allowed Services in Zones:
--------------------------

    Zone  Service ID      Service Name
    ----------------------------------------------
    INT   *All services*  *Entire zone unprotected*
    EXT   service:sshd    Secure Shell Server

Additional Allowed Ports:
-------------------------

    Zone  Protocol     Port
    ----------------------------------------
    INT   *All ports*  *Entire zone unprotected*
    EXT   UDP          1514


Allowed Additional IP Protocols in Zones:
-----------------------------------------

    Zone  IP Protocol
    ----------------------------------------------------
    INT   *All IP protocols*  *Entire zone unprotected*
```

OSSEC Installation and Configuration

Next, download the desired version of OSSEC and verify the MD5 sums. OSSEC installation is a simple "wizard-type" interface, but requires the GNU C compiler, `gcc`, which is already installed on both `base` and all UML guests. Extract the download file, change to the new directory, and enter (still as root):

```
base:~# tar xzf ossec-hids-2.4.tar.gz
base:~# cd ossec-hids-2.4
base:~/ossec-hids-2.4 # ./install.sh
```

There is no need for email notifications in this environment and we will start with active response disabled, so these are the only two options that need modified during installation. See Appendix D for a full sample installation output. The default installation directory is `/var/ossec`, and all primary OSSEC utilities will be installed in

the `bin` subdirectory during the install script. Do not start the OSSEC server when complete, as we have a number of steps to complete to customize the lab, first.

OSSEC encrypts all communication between the server and agents through the use of agent keys. ("OSSEC Manual manage_agents tool," n.d.) For simplicity, we will configure one key for the `fw` guest and one for each of the two protected network segments in the lab: `int` and `dmz`. These keys are stored on the server in the `client.keys` file, located in the `etc` subdirectory of the OSSEC install. Agent keys are created with the `manage_agents` utility. The process of adding an agent will look something like this (with user keystrokes in bold)[1]:

```
base:/var/ossec/bin # ./manage_agents

****************************************
* OSSEC HIDS v2.3 Agent manager.     *
* The following options are available: *
****************************************
   (A)dd an agent (A).
   (E)xtract key for an agent (E).
   (L)ist already added agents (L).
   (R)emove an agent (R).
   (Q)uit.
Choose your action: A,E,L,R or Q: A

- Adding a new agent (use '\q' to return to the main menu).
  Please provide the following:
   * A name for the new agent: int
   * The IP Address of the new agent: 10.5.2.0/24
   * An ID for the new agent[004]:
Agent information:
   ID:004
   Name:int
   IP Address:10.5.2.0/24

Confirm adding it?(y/n): y
Agent added.
```

Simply repeat the steps above for each agent. Once configured, the `client.keys` file will look something like this (keys shortened here for display):

```
001 fw 10.5.2.254   6f91727341699b9d24fb505087a0919bba0c357861...
002 dmz 10.5.1.0/24 876480835ff75bd6efada7bf6b0a66206785acdb0c...
003 int 10.5.2.0/24 bdb11209067fd2b2b1e0bbb8743a9f8b37d66f6cc4...
```

Create a directory under `/home/user1/uml/mntdirs/config` called `ossec-config` and copy the `client.keys` file there. Be sure to change the owner of the copied file back to `user1` and a group ownership of `users`, or the UML guests will

1 The version number shown here for the example manage_agents script is exactly as found in OSSEC release v2.4 and is not a typo. I imagine this will be resolved in a later release.

not be able read it. We will use this file when copying the keys to the OSSEC agents during the installation script.

We still need to configure the `ossec.conf` file before starting the server for the first time. This is a standard XML file, so to disable a line or feature, simply enclose the line or lines with `<!--` and `-->`. Based on our policy, we disable all rules except the those required, making the active `<rules>` portion of `ossec.conf` look like:

```
  <rules>
    <include>rules_config.xml</include>
    <include>pam_rules.xml</include>
    <include>sshd_rules.xml</include>
    <include>syslog_rules.xml</include>
    <include>named_rules.xml</include>
    <include>vsftpd_rules.xml</include>
    <include>web_rules.xml</include>
    <include>apache_rules.xml</include>
    <include>firewall_rules.xml</include>
    <include>ossec_rules.xml</include>
    <include>attack_rules.xml</include>
    <include>local_rules.xml</include>
</rules>
```

Since all the linux devices in our lab our debian based, we can also disable the non-debian portions of the `<rootcheck>` element, as follows[2]:

```
<rootcheck>
<rootkit_files>/var/ossec/etc/shared/rootkit_files.txt</rootkit_fi
les>
<rootkit_trojans>/var/ossec/etc/shared/rootkit_trojans.txt</rootki
t_trojans>
<system_audit>/var/ossec/etc/shared/system_audit_rcl.txt</system_a
udit>
<system_audit>/var/ossec/etc/shared/cis_debian_linux_rcl.txt</syst
em_audit>
<!-- All linux boxes are debian in this environment
<system_audit>/var/ossec/etc/shared/cis_rhel_linux_rcl.txt</system
_audit>
<system_audit>/var/ossec/etc/shared/cis_rhel5_linux_rcl.txt</syste
m_audit>
-->
</rootcheck>
```

The last modification required for this file is to add an `<allowed-ips>` element to the `<remote>` section:

```
  <remote>
    <connection>secure</connection>
    <allowed-ips>10.5.0.0/16</allowed-ips>
    <port>1514</port>
  </remote>
```

Because we disabled the anti-virus rules in the above configuration, we need to make

2 The base device is SuSE, but there is no CIS syscheck file included by default.

one modification to prevent the server choking on a now invalid OSSEC alert group. Comment out rule 40113 in `attack_rules.xml`, located in the `rules` subdirectory of the OSSEC install path:

```
<!--   <rule id="40113" level="12" frequency="6" timeframe="360">
    <if_matched_group>virus</if_matched_group>
    <description>Multiple viruses detected - Possible
outbreak.</description>
    <group>virus,</group>
  </rule> -->
```

Lastly, because our UML guests have a potential to be reset on a regular basis (that *is* the point of a lab...), we need to make a change in the `internal_options.conf` file on the server. Disable the option to have the server verify message IDs from the clients:

```
# Verify msg id (set to 0 to disable it)
remoted.verify_msg_id=0
```

Now we can start the OSSEC server (as root) and verify it is listening on the network:

```
base:/var/ossec/bin # ./ossec-control start
Starting OSSEC HIDS v2.4 (by Trend Micro Inc.)...
2010/04/10 05:22:02 ossec-maild: INFO: E-Mail notification
disabled. Clean Exit.
Started ossec-maild...
Started ossec-execd...
Started ossec-analysisd...
Started ossec-logcollector...
Started ossec-remoted...
Started ossec-syscheckd...
Started ossec-monitord...
Completed.
base:/var/ossec/bin # netstat -anp | grep -i ossec
udp   0  0 0.0.0.0:1514  0.0.0.0:*  7834/ossec-remoted
```

The entire `ossec.conf` file is included in Appendix E for reference.

Modifying the NETinVM Startup Scripts

As we've already seen, NETinVM is designed for a flexible, dynamic configuration. To take advantage of this flexibility, we need to take a few extra steps for the initial agent installation, which helps automate subsequent agent installs. While these tasks may seem numerous, once complete, they will make future labs much easier to configure and/or reset.

Each UML guest only has around 1 gigabyte of space on the / partition which, after the Debian install, leaves only around 260 megabytes free. To allow enough space for the OSSEC installation and subsequent logs, we need to create another sparse file system to be mounted at `/var/ossec`. This is actually a fairly simple process.

First, create a new 1 gigabyte sparse file in the `/home/user1/uml/data` directory, which we discussed when reviewing the NETinVM file structure. `dd` performs this trick nicely:

```
user1@base:~/uml/data> dd if=/dev/zero of=uml_ossec_fs \
seek=1047552 count=1 bs=1024
```

Next, we need to create a file system inside this newly created file:

```
user1@base:~/uml/data> /sbin/mkfs.ext2 uml_ossec_fs
mke2fs 1.40.2 (12-Jul-2007)
uml_ossec_fs is not a block special device.
Proceed anyway? (y,n) y
file system label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
131072 inodes, 261888 blocks
13094 blocks (5.00%) reserved for the super user
First data block=0
Maximum file system blocks=268435456
8 block groups
32768 blocks per group, 32768 fragments per group
16384 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376

Writing inode tables: done
Writing superblocks and file system accounting information: done

This file system will be automatically checked every 33 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to
override.
```

Now that the file system is created, we need to tell the UML startup scripts to mount the new file system in the guests. Make a copy of the `uml.sh` script[3] in the `/home/user1/uml/bin` directory and call it `uml_ossec.sh`. We will add both the file system and the COW file parameters to the script, using two new variables, `VM_OSSEC_COW` and `UBD2`, as highlighted below:

```
        VM_PID=$VM_DIR/pid
        VM_COW=$VM_DIR/cow
        VM_OSSEC_COW=$VM_DIR/ossec_cow
        VM_SWAP=$VM_DIR/swap
        UBD0="$VM_COW,$UML_HOME/data/uml_root_fs"
        UBD1="$VM_SWAP"
        UBD2="$VM_OSSEC_COW,$UML_HOME/data/uml_ossec_fs"
```

Next, find the line further down the script, in the `run_vm()` function, that reads (all on one line):

```
VM="$LINUX xterm=$XTERM umid=$UMID mem=$MEM $ETH ubd0=$UBD0
```

3 A patch in unified diff format is available for the uml.sh script in Appendix G

```
ubd1=$UBD1"
```

Append the newly created `ubd2` variable before the closing quotes:

```
VM="$LINUX xterm=$XTERM umid=$UMID mem=$MEM $ETH ubd0=$UBD0
ubd1=$UBD1 ubd2=$UBD2"
```

Now, our new ext2 file system will be presented to the UML guest as the partition `/dev/ubdc` and is ready to be used for storing the OSSEC install.

6. OSSEC Agent Installation

Preparation

We need to create a compiled installation of an OSSEC agent, which can then be copied to the other agents. This can be done on any UML guest, as we will create a tar file of the install and then reset the configuration for that system. Copy the OSSEC install tarball (the same one used to install the server) to the `uml/mntdirs/tmp` directory, where it can be accessed by the UML guests. Start the UML guest:

```
user1@base:~> uml_ossec.sh -d 5 a int
```

This will start UML guest `inta` on virtual desktop #5 in the KDE window manager. Switch to the console window that opens in that desktop and login as `root` with the password "`You should change this passphrase`" (minus the quotes, obviously).

As already stated, there isn't enough space in the root UML file system, so we need to mount our new file system at the `/var/ossec` directory:

```
inta:~# mkdir /var/ossec
inta:~# mount -t ext2 /dev/ubdc /var/ossec
```

Then install OSSEC in agent mode:

```
inta:~# tar xzf /mnt/tmp/ossec-hids-2.4.tar.gz -C ~/
inta:~# cd ossec-hids-2.4
inta:~/ossec-hids-2.4# ./install.sh
```

Follow the sample agent install script, as found in Appendix F. If the process detects that the `/var/ossec` directory already exists, do not delete it when prompted.

Once the compile is finished, there are a few edits we need to make before creating the tar file for distribution. Open the `ossec.conf` file and replace the OSSEC server IP inside the `<server-ip>` element (located at the top of the file) with "`OSSEC_SERVER`" (minus the quotes). Then move down the file to the `<active-response>` block and

replace it with:

```
  <command>
    <name>firewall-drop</name>
    <executable>firewall-drop.sh</executable>
    <expect>srcip</expect>
    <timeout_allowed>yes</timeout_allowed>
  </command>

  <active-response>
    <disabled>yes</disabled>
    <command>firewall-drop</command>
    <location>local</location>
    <level>6</level>
    <timeout>120</timeout>
  </active-response>
```

This will greatly simplify enabling active response when the time comes.

Now we can proceed with creating a portable archive, to be installed on future UML guests. One way to accomplish this is found on page 56 of the OSSEC book, which I have modified slightly below:

```
inta:~/ossec-hids-2.4# cd /; tar --exclude client.keys \
--exclude lost+found -cvzf /mnt/tmp/ossec-agent.tar.gz \
var/ossec etc/init.d/ossec etc/ossec-init.conf
```

Back on `base`, move the agent archive, ossec-agent.tar.gz, to the `uml/mntdirs/config/ossec-config` directory where it will be accessed during the startup sequence.

Creating the Startup Scripts

Now we will add a new script to automate the installation of the OSSEC agent on the UML guests. This script, `configure-ossec.sh`, should be placed in the `uml/mntdirs/config/ossec-config` directory. Be sure to change permissions on the script to make it executable:

```
user1@base:~/uml/mntdirs/config/ossec-config> chmod 755 \
configure-ossec.sh
```

This script requires two arguments, the network segment name (`INT`, `DMZ`, or `FW`) and the OSSEC server IP, and performs the following tasks:

1. Adds the `/dev/ubdc` partition (which is the newly-created `uml_ossec_fs` file system) to `/etc/fstab`, to be automounted on future startups.
2. Creates the `/var/ossec` directory and mounts `/dev/ubdc`, since the `/etc/fstab` has already been processed for this boot sequence.

3. Creates the required OSSEC users and group.
4. Extracts the `ossec-agent.tar.gz` binary install.
5. Inserts the proper OSSEC server IP into `ossec.conf`.
6. Extracts the correct OSSEC agent key (based on the network segment name) and places it into the agent's `client.keys` file.
7. Configures OSSEC for the proper runlevels (start and stop sequences).
8. Starts the OSSEC agent.

The full `configure-ossec.sh` script is found in Appendix H.

Configuring the guest to actually run the OSSEC install script is simple. Edit the `commands-to-run.sh` script for each guest or network where an agent should be installed, appending this line:

```
/mnt/config/ossec-config/configure-ossec.sh <NETWORK> <OSSEC-IP>
```

where `<NETWORK>` can be `INT`, `DMZ`, or `FW`; and `OSSEC-IP` is 10.5.2.1 for the internal network or firewall, and 10.5.1.1 for the DMZ.

Now start a new agent (we will use `dmza` as an example):

```
user1@base:~> uml_ossec.sh -d 3 a dmz
Creating directory: /home/user1/uml/machines/dmza
Creating swap file of 256M: /home/user1/uml/machines/dmza/swap
```

This starts `dmza` on KDE desktop #3. Switch to the console there and you should see OSSEC installing as shown here:

```
[other startup messages]
--- Applying network-specific UML configuration ---
Running commands...
Adding the ubdc disk to fstab
Making /var/ossec directory
Mounting /dev/ubdc on /var/ossec
Adding the OSSEC users and group
var/ossec/
var/ossec/logs/
var/ossec/logs/ossec.log
var/ossec/bin/
[tar extraction output trimmed]
Editing ossec.conf to use the proper OSSEC server
Copying client.keys file
Adding OSSEC to startup process
 System startup links for /etc/init.d/ossec already exist.
Starting OSSEC HIDS v2.4 (by Trend Micro Inc.)...
Started ossec-execd...
```

```
Started ossec-agentd...
Started ossec-logcollector...
Started ossec-syscheckd...
Completed.
[more startup messages]
```

Login to the guest as root with the same default password as before: "`You should probably change this passphrase`" (again, minus the quotes).

Verifying Communication

Communication between the OSSEC server and agents can be verified at both endpoints. Progress can be followed on the agent with:

```
dmza:/var/ossec/logs# tail -f ossec.log
[output trimmed]
2010/08/07 19:21:55 ossec-agentd(4102): INFO: Connected to the
server (10.5.1.1:1514).
```

On the server, run (as root):

```
base:/var/ossec/bin # ./agent_control -l

OSSEC HIDS agent_control. List of available agents:
   ID: 000, Name: base (server), IP: 127.0.0.1, Active/Local
   ID: 001, Name: int, IP: 10.5.2.0/24, Inactive
   ID: 002, Name: dmz, IP: 10.5.1.0/24, Active
   ID: 003, Name: fw, IP: 10.5.2.254, Inactive
```

Follow this process for each guest you want to run.

At this point, you probably want to modify the `uml_run_all.sh` script to use the modified `uml_ossec.sh` launch script instead of `uml.sh`. Or simply rename `uml_ossec.sh` to `uml.sh`.

7. Testing OSSEC

Now it's time to have some fun! To generate some alerts, we will start `exta` and then install some security scanners and to run against the servers in the "DMZ".

Generating Alerts via (Malicious) Web Traffic

Nikto is a web server vulnerability scanner which "performs comprehensive tests against web servers for multiple items, including over 6400 potentially dangerous files/CGIs, checks for outdated versions of over 1000 servers, and version specific problems on over 270 servers". ("Nikto2 | CIRT.net," n.d.) It is also a great choice for testing basic functionality, as it is small and makes no attempt to be stealthy by default.

(ibid.)

Our attacking host, `exta`, is plagued with the same storage limitation as the other UML guests. But here we can "cheat" and use the same file system we created for OSSEC to house any required additional tools needed:

```
exta:/# mkdir /tools
exta:/# echo "/dev/ubdc /tools  ext2  defaults 0 1" >> /etc/fstab
exta:/# mount -t ext2 /dev/ubdc /tools
```

This works because changes to the `/dev/ubdc` file system are written to the COW file, so OSSEC is not in the way on `exta` and neither do the attack tools interfere with the OSSEC installation on the other UML guests.

The NETinVM UML guests do not have access to the Internet by default. First, I chose to download Nikto[4] onto `base` and saved it to `uml/mntdirs/tmp` so it can be accessed from `exta`. Installation is a simple tar command:

```
exta:/# tar xzf /mnt/tmp/nikto-2.1.2.tar.gz -C /tools/
```

Once finished, launch a basic Nikto scan against the web server (`dmza`):

```
exta:/tools/nikto-2.1.2# ./nikto.pl -Format html \
-output /mnt/tmp/nikto-scan.html -host 10.5.1.10 \
-vhost www.example.net -port 80
```

This will also save the results of the scan on `/mnt/tmp/nikto-scan.html`, which can be viewed with the browser on `base`.

Once the scan has completed, review the `alerts.log` file in `/var/ossec/logs` on `base`. While the default Apache installation on `dmza` doesn't generate a lot of terribly "interesting" entries in Nikto, the scan generated roughly 5900 alerts in OSSEC!

A visual inspection of the alerts first reveals a handful of alerts generated by rule 31101:

```
** Alert 1281208821.136962: - web,accesslog,
2010 Aug 07 21:20:21 (dmz) 10.5.1.0->/var/log/apache/access.log
Rule: 31101 (level 5) -> 'Web server 400 error code.'
Src IP: 10.5.0.10
User: (none)
10.5.0.10 - - [07/Aug/2010:21:20:20 +0200] "GET /2BPt9ZCF.nn
HTTP/1.1" 404 289 "-" "Mozilla/4.75 (Nikto/2.1.2) (Evasions:None)
(Test:map_codes)" "-"
```

This single entry is a level 5 alert, which by itself is probably normally just noise. This rule simply shows that the web server issued a 400 class error, in this case the

4 Nikto is available at http://cirt.net/nikto2

famous "404 File Not Found" error. A busy web server would certainly generate these codes all the time, but further down the list we see an alert from rule 31153:

```
** Alert 1281209115.429789: mail  - web,accesslog,attack,
2010 Aug 07 21:25:15 (dmz) 10.5.1.0->/var/log/apache/access.log
Rule: 31153 (level 10) -> 'Multiple common web attacks from same
souce ip.'
Src IP: 10.5.0.10
User: (none)
10.5.0.10 - - [07/Aug/2010:21:25:14 +0200] "GET /index.jsp%00x
HTTP/1.1" 404 287 "-" "Mozilla/4.75 (Nikto/2.1.2) (Evasions:None)
(Test:000469)" "-"
10.5.0.10 - - [07/Aug/2010:21:25:14 +0200] "GET /%00/ HTTP/1.1"
404 278 "-" "Mozilla/4.75 (Nikto/2.1.2) (Evasions:None)
(Test:000459)" "-"
10.5.0.10 - - [07/Aug/2010:21:25:14 +0200] "GET /%00/ HTTP/1.1"
404 278 "-" "Mozilla/4.75 (Nikto/2.1.2) (Evasions:None)
(Test:000458)" "-"
10.5.0.10 - - [07/Aug/2010:21:25:14 +0200] "GET /%00/ HTTP/1.1"
404 278 "-" "Mozilla/4.75 (Nikto/2.1.2) (Evasions:None)
(Test:000457)" "-"
10.5.0.10 - - [07/Aug/2010:21:25:09 +0200] "GET
/_mem_bin/remind.asp HTTP/1.1" 404 297 "-" "Mozilla/4.75
(Nikto/2.1.2) (Evasions:None) (Test:000317)" "-"
10.5.0.10 - - [07/Aug/2010:21:25:09 +0200] "GET
/_mem_bin/auoconfig.asp HTTP/1.1" 404 300 "-" "Mozilla/4.75
(Nikto/2.1.2) (Evasions:None) (Test:000316)" "-"
10.5.0.10 - - [07/Aug/2010:21:25:09 +0200] "GET
/_mem_bin/auoconfig.asp HTTP/1.1" 404 300 "-" "Mozilla/4.75
(Nikto/2.1.2) (Evasions:None) (Test:000315)" "-"
10.5.0.10 - - [07/Aug/2010:21:25:06 +0200] "GET /newuser?
Image=../../database/rbsserv.mdb HTTP/1.1" 404 285 "-"
"Mozilla/4.75 (Nikto/2.1.2) (Evasions:None) (Test:000246)" "-"
10.5.0.10 - - [07/Aug/2010:21:25:04 +0200] "GET /web-
console/ServerInfo.jsp%00 HTTP/1.1" 404 304 "-" "Mozilla/4.75
(Nikto/2.1.2) (Evasions:None) (Test:000165)" "-"
```

Now we see that there are several 400 class errors being generated by the same web client. More importantly, that client is sending requests associated with common web attacks. This alert is rated as more severe (level 10) and might deserve some attention.

A basic review of the logs reveal OSSEC fired 506 level 10 alerts, from three different rules:

```
base:/var/ossec/logs/alerts # grep -c level\ 10 alerts.log
506

base:/var/ossec/logs/alerts # grep level\ 10 alerts.log | sort -u
Rule: 31151 (level 10) -> 'Mutiple web server 400 error codes from
same source ip.'
Rule: 31153 (level 10) -> 'Multiple common web attacks from same
souce ip.'
Rule: 31154 (level 10) -> 'Multiple XSS (Cross Site Scripting)
attempts from same souce ip.'
```

Generating Alerts via Brute Force Password Attacks

While Nikto is a good, small(er) footprint web scanner, no security administrator's tool set is complete without Nmap. With the addition of the Nmap Scripting Engine (NSE), Nmap has grown into a powerful vulnerability and pentest scanner. (Lyon, 2009) We already used Nmap to verify services running in the NETinVM network, but that was from `base`. And since we will be launching attacks this time, and eventually telling OSSEC to automatically block traffic from the attacking machine, it is preferable to install Nmap locally on `exta`, so OSSEC's Active Response doesn't disconnect the agents from the server!

I chose to download the latest stable version of Nmap (as of this writing) from the official site, http://nmap.org/download/, (again, using `base`) and placed it in `uml/mntdir/tmp`, so `exta` could extract from there:

```
exta:/mnt/tmp# tar xjvf nmap-5.21.tar.bz2 -C /tools
exta:/mnt/tmp# cd /tools/nmap-5.21
exta:/tools/nmap-5.21# ./configure --prefix=/tools/nmap \
--without-zenmap && make && make install
```

To avoid a litany of dependencies and keep things lite, I disabled the graphical interface for Nmap, by using the `--without-zenmap` flag during the `configure` process. Once the install was complete, I launched an FTP brute force password attack, this time against `dmzb`:

```
exta:/tools/nmap/bin# ./nmap -sS -PN -p 21 --script=ftp-brute \
--script-args=passlimit=100 --reason 10.5.1.11
```

This uses the default `usernames.lst` and `passwords.lst` files distributed with nmap and located in the `nmap/share/nmap/nselib/data` directory. This script can take awhile, so for this demo I limited the password guesses to 100 per user account. The results on my system were as follows:

```
Starting Nmap 5.21 ( http://nmap.org ) at 2010-08-18 04:26 CEST
NSE: Script Scanning completed.
Nmap scan report for 10.5.1.11
Host is up, received user-set (0.051s latency).
PORT   STATE SERVICE REASON
21/tcp open  ftp     syn-ack
| ftp-brute:
|_  anonymous: IEUser@

Nmap done: 1 IP address (1 host up) scanned in 51.16 seconds
```

A new review of the the `alerts.log` showed a plethora of alerts from rules 11401

and 11402:

```
** Alert 1282097694.2787: - syslog,vsftpd,connection_attempt
2010 Aug 18 04:14:54 (dmz) 10.5.1.0->/var/log/vsftpd.log
Rule: 11401 (level 3) -> 'FTP session opened.'
Src IP: 10.5.0.10
User: (none)
Wed Aug 18 04:14:53 2010 [pid 1429] CONNECT: Client "10.5.0.10"

** Alert 1282097694.3048: - syslog,vsftpd,authentication_success,
2010 Aug 18 04:14:54 (dmz) 10.5.1.0->/var/log/vsftpd.log
Rule: 11402 (level 3) -> 'FTP Authentication success.'
Src IP: (none)
User: (none)
Wed Aug 18 04:14:53 2010 [pid 1428] [ftp] OK LOGIN: Client
"10.5.0.10", anon password "IEUser@"
```

Once again, these are level 3 alerts, simply logging an FTP session has started and even a successful authentication to the FTP service. These are certainly normal (and frequent) events during the course of a regular business day, and by themselves are very uninteresting. But very soon thereafter I found a level 10 alert from rule 11452:

```
** Alert 1282097694.5961: mail  - syslog,vsftpd,recon,
2010 Aug 18 04:14:54 (dmz) 10.5.1.0->/var/log/vsftpd.log
Rule: 11452 (level 10) -> 'Multiple FTP connection attempts from
same source IP.'
Src IP: 10.5.0.10
User: (none)
Wed Aug 18 04:14:54 2010 [pid 1452] CONNECT: Client "10.5.0.10"
Wed Aug 18 04:14:54 2010 [pid 1450] CONNECT: Client "10.5.0.10"
Wed Aug 18 04:14:54 2010 [pid 1448] CONNECT: Client "10.5.0.10"
Wed Aug 18 04:14:54 2010 [pid 1446] CONNECT: Client "10.5.0.10"
Wed Aug 18 04:14:54 2010 [pid 1444] CONNECT: Client "10.5.0.10"
Wed Aug 18 04:14:54 2010 [pid 1442] CONNECT: Client "10.5.0.10"
Wed Aug 18 04:14:54 2010 [pid 1440] CONNECT: Client "10.5.0.10"
Wed Aug 18 04:14:54 2010 [pid 1438] CONNECT: Client "10.5.0.10"
Wed Aug 18 04:14:54 2010 [pid 1436] CONNECT: Client "10.5.0.10"
Wed Aug 18 04:14:54 2010 [pid 1434] CONNECT: Client "10.5.0.10"
Wed Aug 18 04:14:54 2010 [pid 1432] CONNECT: Client "10.5.0.10"
```

OSSEC has detected several connections from the same client – a behavior that is not normal for a regular FTP session. I verified the range and number of alerts in similar fashion as before:

```
base:/var/ossec/logs/alerts # grep FTP alerts.log | sort -u
Rule: 11401 (level 3) -> 'FTP session opened.'
Rule: 11402 (level 3) -> 'FTP Authentication success.'
Rule: 11452 (level 10) -> 'Multiple FTP connection attempts from
same source IP.'

base:/var/ossec/logs/alerts # grep -c level\ 3.*FTP alerts.log
2194

base:/var/ossec/logs/alerts # grep -c level\ 10.*FTP alerts.log
197
```

At this point, we literally have thousands of alerts to sift through, and a lot of noise that makes it more difficult to find the important information and respond appropriately. Instead of trying to respond to these attacks by hand, we can enable OSSEC's Active Response and automatically block the offending client.

8. Active Response

Overview and Options

As described in the OSSEC book, "Active response allows a scripted action to be performed whenever a rule is matched in your OSSEC HIDS rule set." (Hay, Cid, & Bray, 2008) In fact, if you desired, it is possible to fire off more than one response for any given alert or set of alerts.

There are a number of pre-configured response scripts that ship with OSSEC, one of which is the `firewall-drop.sh` script. This script will accept from the rule the offending IP address and append a local host firewall rule to drop any traffic from that address, optionally for a given amount of time. Shown here again for review is the initial active response configuration:

```
<command>
  <name>firewall-drop</name>
  <executable>firewall-drop.sh</executable>
  <expect>srcip</expect>
  <timeout_allowed>yes</timeout_allowed>
</command>

<active-response>
  <disabled>yes</disabled>
  <command>firewall-drop</command>
    <location>local</location>
    <level>6</level>
    <timeout>120</timeout>
</active-response>
```

Once enabled, this configuration will fire the response, firewall-drop, when an alert of level 6 or greater is received against a given address. The response script is also configured to remove the firewall drop rule after the given timeout of 120 seconds. When a response rule fires, it is logged to an agent-local, dedicated file for active response, `/var/ossec/logs/active-response.log`.

Other response scripts include `host-deny.sh`, which requires a given service

support `tcpwrappers`. Alternatively, the `route-null.sh` script will simply modify the agent's local routing table, preventing reply traffic from being delivered. This is an excellent option when the `host-deny.sh` or `firewall-drop.sh` scripts are not an option. If the attack is utilizing a specific account, the `disable-account.sh` script may be appropriate. But my favorite "toy" response is the recently added `ossec-tweeter.sh` script, which as you probably guessed, sends a "tweet" to a specified account, as a notification of the alert. While probably not terribly practical, it is fun to play with and demonstrates the possibilities of active response scripts.

With almost all of the responses (the possible exception being a twitter message), you should take care in how an active response command is configured, as you can easily create a self-inflicted denial of service (DoS). To safeguard against just such a possibility, it is recommended to add any hosts that should never be blocked to the `<white_list>` element (one address per element) in `ossec.conf` ("OSSEC Manual Active Response," n.d.). A good example of an IP that could be placed here is the OSSEC server itself. It should be noted that alerts will still be generated for these hosts, but no active responses will fire.

Enabling Active Response

We already included the required configuration for active response when we setup the agent install process, but left it disabled. To enable it we just need to replace the "yes" with a "no" in the `<disabled>` element:

```
    <disabled>no</disabled>
```

This should be done on the agents and the server, and then restart OSSEC on each system:

```
base:/var/ossec/bin # ./ossec-control restart
```

Now we can return to `exta` and re-run the Nikto attack, then follow the logs on `dmza`:

```
dmza:/var/ossec/logs# tail -f active-responses.log
Mon Aug  9 03:26:39 CEST 2010 /var/ossec/active-
response/bin/firewall-drop.sh add - 10.5.0.10 1281317198.9984758
31151
Mon Aug  9 03:29:40 CEST 2010 /var/ossec/active-
response/bin/firewall-drop.sh delete - 10.5.0.10
1281317198.9984758 31151
```

```
Mon Aug  9 03:29:45 CEST 2010 /var/ossec/active-
response/bin/firewall-drop.sh add - 10.5.0.10 1281317384.10014556
31151
Mon Aug  9 03:32:51 CEST 2010 /var/ossec/active-
response/bin/firewall-drop.sh delete - 10.5.0.10
1281317384.10014556 31151
Mon Aug  9 03:33:58 CEST 2010 /var/ossec/active-
response/bin/firewall-drop.sh add - 10.5.0.10 1281317638.10061832
31151
Mon Aug  9 03:36:58 CEST 2010 /var/ossec/active-
response/bin/firewall-drop.sh delete - 10.5.0.10
1281317638.10061832 31151
[trimmed]
```

Here we see the date and time of the action, as well as the script that fired and the actions it performed. The last number of the line ("31151" in this case) shows the rule that triggered the response:

```
<rule id="31151" level="10" frequency="10" timeframe="120">
  <if_matched_sid>31101</if_matched_sid>
  <same_source_ip />
  <description>Mutiple web server 400 error codes </description>
  <description>from same source ip.</description>
  <group>web_scan,recon,</group>
</rule>
```

Likewise, re-run the Nmap scan from `exta` and follow the `active-responses.log` file again, this time showing the trigger rule of 11452:

```
dmzb:/var/ossec/logs# tail -f active-responses.log
Wed Aug 18 07:15:07 CEST 2010 /var/ossec/active-
response/bin/firewall-drop.sh add - 10.5.0.10 1282108507.778773
11452
Wed Aug 18 07:18:20 CEST 2010 /var/ossec/active-
response/bin/firewall-drop.sh delete - 10.5.0.10 1282108507.778773
11452
```

And notice the difference in the Nmap scan results after Active Response was enabled:

```
Starting Nmap 5.21 ( http://nmap.org ) at 2010-08-18 07:14 CEST
NSE: Script Scanning completed.
Nmap scan report for 10.5.1.11
Host is up, received user-set (0.051s latency).
PORT   STATE SERVICE REASON
21/tcp open  ftp     syn-ack

Nmap done: 1 IP address (1 host up) scanned in 64.64 seconds
```

This time, Nmap did not report any valid users for the FTP service, even though anonymous access is enabled.

9. Conclusion

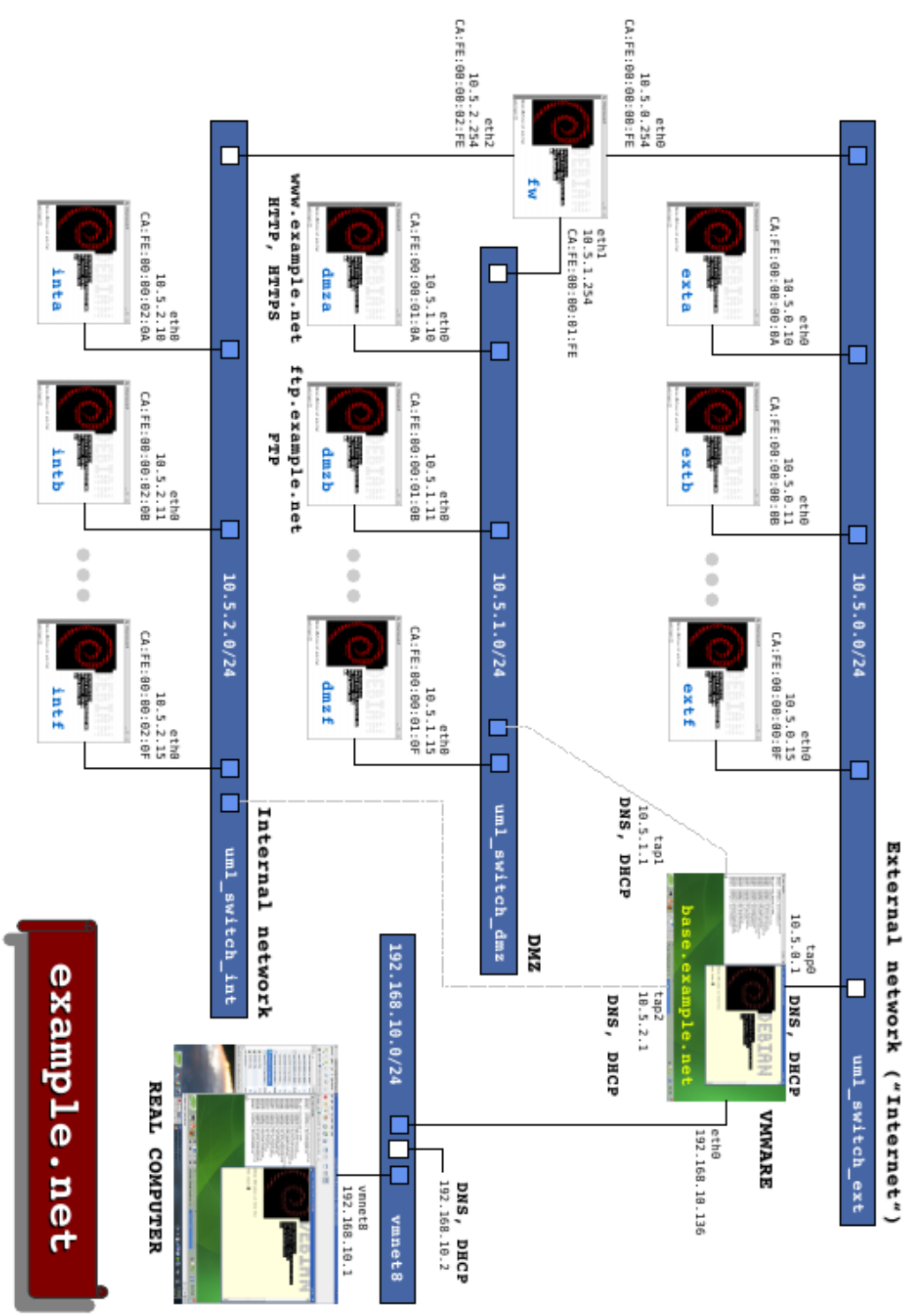
OSSEC is a useful tool to detect and automatically act on attacks against hosts on the

network. It also provides a powerful, flexible framework for automating responses to an attack or aggregated event. Paired with a solid understanding of the network and a good policy, OSSEC's scriptable nature and simple architecture make it an excellent option for protecting the network on a shoe-string budget.

Because OSSEC's Active Response can create undesirable outages if not configured properly, it is preferable to perform initial tests in a lab environment. NETinVM provides that lab in a neatly wrapped, flexible package for testing various network settings, attacks, and defenses, without requiring a great deal of hardware. With a little additional effort, the combination of these two solutions provides a security administrator an excellent, dynamic platform to test configurations and learn from mistakes, without costly repercussions.

Appendix A

NETinVM Network Diagram (provided by Carlos and David Perez)



Appendix B

Full listing of nmap service discovery scan against the DMZ.

```
# Nmap 5.21 scan initiated Fri Apr  9 19:06:34 2010 as: nmap -sV
-T5 --reason -oA netinvm-service-scan -F -O
--script=default,discovery 10.5.1.0/24

Nmap scan report for base.example.net (10.5.1.1)
Host is up, received localhost-response (0.00013s latency).
Not shown: 97 closed ports
Reason: 97 resets
PORT    STATE SERVICE REASON  VERSION
22/tcp  open  ssh     syn-ack OpenSSH 4.6 (protocol 2.0)
|_banner: SSH-2.0-OpenSSH_4.6
| ssh-hostkey: 1024
ce:0b:4d:f2:b6:ca:5d:15:24:bb:c7:02:29:9d:72:4f (DSA)
|_1024 e7:85:aa:50:e2:99:f3:05:e6:5b:8e:76:1e:79:b2:77 (RSA)
53/tcp  open  domain  syn-ack ISC BIND 9.4.1-P1
111/tcp open  rpcbind syn-ack 2 (rpc #100000)
| rpcinfo:
| 100000  2    111/udp  rpcbind
|_100000  2    111/tcp  rpcbind
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.17 - 2.6.30
Network Distance: 0 hops

Nmap scan report for dmza.example.net (10.5.1.10)
Host is up, received arp-response (0.00072s latency).
Not shown: 96 closed ports
Reason: 96 resets
PORT     STATE SERVICE REASON  VERSION
22/tcp   open  ssh     syn-ack OpenSSH 4.3p2 Debian 9etch3
(protocol 2.0)
|_banner: SSH-2.0-OpenSSH_4.3p2 Debian-9etch3
| ssh-hostkey: 1024
1e:76:e1:c7:83:2d:2f:ec:11:5f:f4:85:6b:52:84:fd (DSA)
|_2048 6d:1a:3e:02:72:4d:52:fd:91:6e:f7:e2:0f:a9:f6:bb (RSA)
80/tcp   open  http    syn-ack Apache httpd 1.3.34 ((Debian))
|_citrix-enum-apps-xml:
| http-headers:
|   Date: Fri, 09 Apr 2010 17:07:09 GMT
|   Server: Apache/1.3.34 (Debian)
|   Last-Modified: Thu, 29 Nov 2007 15:14:45 GMT
|   ETag: "13065-148a-474ed765"
|   Accept-Ranges: bytes
|   Content-Length: 5258
|   Connection: close
|   Content-Type: text/html; charset=iso-8859-1
|
|_  (Request type: HEAD)
|_citrix-enum-servers-xml:
|_http-date: Fri, 09 Apr 2010 17:07:09 GMT; 0s from local time.
|_html-title: Placeholder page
| http-enum:
|_  /icons/: Icons and images
111/tcp  open  rpcbind syn-ack 2 (rpc #100000)
| rpcinfo:
```

```
| 100000  2        111/udp  rpcbind
| 100005  1,2,3    868/udp  mountd
| 100021  1,3,4   2048/udp  nlockmgr
| 100003  2,3     2049/udp  nfs
| 100024  1       2050/udp  status
| 100000  2        111/tcp  rpcbind
| 100005  1,2,3    871/tcp  mountd
| 100003  2,3     2049/tcp  nfs
| 100024  1       2254/tcp  status
|_100021  1,3,4   2568/tcp  nlockmgr
2049/tcp open  nfs     syn-ack 2-3 (rpc #100003)
MAC Address: CA:FE:00:00:01:0A (Unknown)
Aggressive OS guesses: Chumby Internet radio (99%), DD-WRT v24
(Linux 2.6.22) (99%), Linux 2.6.9 - 2.6.30 (98%), Linux 2.6.13 -
2.6.28 (97%), Linux 2.6.9 - 2.6.19 (97%), RGB Spectrum MediaWall
1500 video processor (96%), AXIS 207W Network Camera (96%), AXIS
207 Network Camera (Linux 2.6.16) or 241Q Video Server (96%),
Linux 2.4.18 - 2.4.35 (likely embedded) (96%), Linux 2.6.20 (96%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop
Service Info: OS: Linux

Host script results:
|_sniffer-detect: Likely in promiscuous mode (tests: "11111111")

Nmap scan report for dmzb.example.net (10.5.1.11)
Host is up, received arp-response (0.00081s latency).
Not shown: 95 closed ports
Reason: 95 resets
PORT     STATE SERVICE REASON  VERSION
21/tcp   open  ftp     syn-ack vsftpd 2.0.5
|_banner: 220 (vsFTPd 2.0.5)
22/tcp   open  ssh     syn-ack OpenSSH 4.3p2 Debian 9etch3
(protocol 2.0)
|_banner: SSH-2.0-OpenSSH_4.3p2 Debian-9etch3
| ssh-hostkey: 1024
1e:76:e1:c7:83:2d:2f:ec:11:5f:f4:85:6b:52:84:fd (DSA)
|_2048 6d:1a:3e:02:72:4d:52:fd:91:6e:f7:e2:0f:a9:f6:bb (RSA)
80/tcp   open  http    syn-ack Apache httpd 1.3.34 ((Debian))
|_citrix-enum-apps-xml:
|_html-title: Placeholder page
|_citrix-enum-servers-xml:
|_http-date: Fri, 09 Apr 2010 17:07:09 GMT; 0s from local time.
| http-headers:
|   Date: Fri, 09 Apr 2010 17:07:10 GMT
|   Server: Apache/1.3.34 (Debian)
|   Last-Modified: Thu, 29 Nov 2007 15:14:45 GMT
|   ETag: "13065-148a-474ed765"
|   Accept-Ranges: bytes
|   Content-Length: 5258
|   Connection: close
|   Content-Type: text/html; charset=iso-8859-1
|
|_  (Request type: HEAD)
| http-enum:
|_  /icons/: Icons and images
111/tcp  open  rpcbind syn-ack
| rpcinfo:
| 100000  2        111/udp  rpcbind
| 100005  1,2,3    868/udp  mountd
```

```
| 100021  1,3,4   2048/udp  nlockmgr
| 100003  2,3     2049/udp  nfs
| 100024  1       2050/udp  status
| 100000  2        111/tcp  rpcbind
| 100005  1,2,3    871/tcp  mountd
| 100003  2,3     2049/tcp  nfs
| 100024  1       2182/tcp  status
|_100021  1,3,4   2226/tcp  nlockmgr
2049/tcp open  nfs     syn-ack 2-3 (rpc #100003)
MAC Address: CA:FE:00:00:01:0B (Unknown)
Device type: media device
Running: Chumby embedded
OS details: Chumby Internet radio
Network Distance: 1 hop
Service Info: OSs: Unix, Linux

Host script results:
|_nfs-showmount:
|_sniffer-detect: Likely in promiscuous mode (tests: "11111111")

OS and Service detection performed. Please report any incorrect
results at http://nmap.org/submit/ .
# Nmap done at Fri Apr  9 19:07:39 2010 -- 256 IP addresses (3
hosts up) scanned in 65.84 seconds
```

Appendix C

Example.Net OSSEC Policy

Hosts and Services

Host: dmza
Services: SSH, Apache, rpcbind, nfs

Host: dmzb
Services: SSH, ftp, rpcbind, nfs

Host: inta
Services: SSH, Apache, rpcbind, nfs

Goals

Monitor critical system files on all hosts for signs of rootkit activity
Monitor SSH logs on all hosts for signs of attempted account compromise
Monitor Apache logs on host dmza for signs of intrusion attempts/successes
Monitor FTP logs on host dmzb for signs of intrusion attempts/successes

Appendix D

Sample OSSEC Server Installation

```
base:~/ossec-hids-2.4# ./install.sh

  ** Para instalação em português, escolha [br].
  ** 要使用中文进行安装，请选择 [cn].
  ** Fur eine deutsche Installation wohlen Sie [de].
  ** Για εγκατάσταση στα Ελληνικά, επιλέξτε [el].
  ** For installation in English, choose [en].
  ** Para instalar en Español , eliga [es].
  ** Pour une installation en français, choisissez [fr]
  ** Per l'installazione in Italiano, scegli [it].
  ** 日本語でインストールします.選択して下さい.[jp].
  ** Voor installatie in het Nederlands, kies [nl].
  ** Aby instalować w języku Polskim, wybierz [pl].
  ** Для инструкций по установке на русском ,введите [ru].
  ** Za instalaciju na srpskom, izaberi [sr].
  ** Türkçe kurulum için seçin [tr].
  (en/br/cn/de/el/es/fr/it/jp/nl/pl/ru/sr/tr) [en]: <enter>


 OSSEC HIDS v2.4 Installation Script - http://www.ossec.net

 You are about to start the installation process of the OSSEC
HIDS.
 You must have a C compiler pre-installed in your system.
 If you have any questions or comments, please send an e-mail
 to dcid@ossec.net (or daniel.cid@gmail.com).

  - System: Linux e65 2.6.28-19-generic
  - User: root
  - Host: e65


  -- Press ENTER to continue or Ctrl-C to abort. --
<enter>

1- What kind of installation do you want (server, agent, local or
help)? server

  - Server installation chosen.

2- Setting up the installation environment.

 - Choose where to install the OSSEC HIDS [/var/ossec]: <enter>

    - Installation will be made at  /var/ossec .

3- Configuring the OSSEC HIDS.

  3.1- Do you want e-mail notification? (y/n) [y]: n

   --- Email notification disabled.

  3.2- Do you want to run the integrity check daemon? (y/n) [y]: n
```

```
   - Not running syscheck (integrity check daemon).

  3.3- Do you want to run the rootkit detection engine? (y/n) [y]:

<enter>

   - Running rootcheck (rootkit detection).

  3.4- Active response allows you to execute a specific
       command based on the events received. For example,
       you can block an IP address or disable access for
       a specific user.
       More information at:
       http://www.ossec.net/en/manual.html#active-response

   - Do you want to enable active response? (y/n) [y]: n

     - Active response disabled.

  3.5- Do you want to enable remote syslog (port 514 udp)? (y/n)
[y]: <enter>

   - Remote syslog enabled.

  3.6- Setting the configuration to analyze the following logs:
    -- /var/log/messages
    -- /var/log/auth.log
    -- /var/log/syslog
    -- /var/log/mail.info
    -- /var/log/dpkg.log

 - If you want to monitor any other file, just change
   the ossec.conf and add a new localfile entry.
   Any questions about the configuration can be answered
   by visiting us online at http://www.ossec.net .


   --- Press ENTER to continue ---

<enter>

5- Installing the system
 - Running the Makefile

[compiler output trimmed]

- System is Debian (Ubuntu or derivative).
 - Init script modified to start OSSEC HIDS during boot.

 - Configuration finished properly.

 - To start OSSEC HIDS:
          /var/ossec/bin/ossec-control start

 - To stop OSSEC HIDS:
          /var/ossec/bin/ossec-control stop

 - The configuration can be viewed or modified at
/var/ossec/etc/ossec.conf
```

```
    Thanks for using the OSSEC HIDS.
    If you have any question, suggestion or if you find any bug,
    contact us at contact@ossec.net or using our public maillist
at
    ossec-list@ossec.net
    ( http://www.ossec.net/main/support/ ).

    More information can be found at http://www.ossec.net

    ---  Press ENTER to finish (maybe more information below). ---

<enter>

 - In order to connect agent and server, you need to add each
agent to the server.
   Run the 'manage_agents' to add or remove them:

   /var/ossec/bin/manage_agents

   More information at:
   http://www.ossec.net/en/manual.html#ma
```

Appendix E

Sample `ossec.conf` for base.example.net

```
<ossec_config>
  <global>
    <email_notification>no</email_notification>
  </global>

  <rules>
    <include>rules_config.xml</include>
    <include>pam_rules.xml</include>
    <include>sshd_rules.xml</include>
    <include>syslog_rules.xml</include>
    <include>named_rules.xml</include>
    <include>vsftpd_rules.xml</include>
    <include>web_rules.xml</include>
    <include>apache_rules.xml</include>
    <include>firewall_rules.xml</include>
    <include>ossec_rules.xml</include>
    <include>attack_rules.xml</include>
    <include>local_rules.xml</include>

<!-- Disabled rules below
    <include>telnetd_rules.xml</include>
    <include>arpwatch_rules.xml</include>
    <include>symantec-av_rules.xml</include>
    <include>symantec-ws_rules.xml</include>
    <include>pix_rules.xml</include>
    <include>smbd_rules.xml</include>
    <include>pure-ftpd_rules.xml</include>
    <include>proftpd_rules.xml</include>
    <include>ms_ftpd_rules.xml</include>
    <include>ftpd_rules.xml</include>
    <include>hordeimp_rules.xml</include>
    <include>roundcube_rules.xml</include>
    <include>wordpress_rules.xml</include>
    <include>cimserver_rules.xml</include>
    <include>vpopmail_rules.xml</include>
    <include>vmpop3d_rules.xml</include>
    <include>courier_rules.xml</include>
    <include>nginx_rules.xml</include>
    <include>php_rules.xml</include>
    <include>mysql_rules.xml</include>
    <include>postgresql_rules.xml</include>
    <include>ids_rules.xml</include>
    <include>squid_rules.xml</include>
    <include>cisco-ios_rules.xml</include>
    <include>netscreenfw_rules.xml</include>
    <include>sonicwall_rules.xml</include>
    <include>postfix_rules.xml</include>
    <include>sendmail_rules.xml</include>
    <include>imapd_rules.xml</include>
    <include>mailscanner_rules.xml</include>
    <include>dovecot_rules.xml</include>
    <include>ms-exchange_rules.xml</include>
```

```
    <include>racoon_rules.xml</include>
    <include>vpn_concentrator_rules.xml</include>
    <include>spamd_rules.xml</include>
    <include>msauth_rules.xml</include>
    <include>mcafee_av_rules.xml</include>
    <include>trend-osce_rules.xml</include>
    <include>ms-se_rules.xml</include>
    <include>zeus_rules.xml</include>
    <include>solaris_bsm_rules.xml</include>
    <include>vmware_rules.xml</include>
    <include>ms_dhcp_rules.xml</include>
    <include>asterisk_rules.xml</include>
    <include>policy_rules.xml</include>
-->
  </rules>

  <syscheck>
    <!-- Frequency that syscheck is executed - default to every 22
hours -->
    <frequency>79200</frequency>

    <!-- Directories to check  (perform all possible
verifications) -->
    <directories
check_all="yes">/etc,/usr/bin,/usr/sbin</directories>
    <directories check_all="yes">/bin,/sbin</directories>

    <!-- Files/directories to ignore -->
    <ignore>/etc/mtab</ignore>
    <ignore>/etc/mnttab</ignore>
    <ignore>/etc/hosts.deny</ignore>
    <ignore>/etc/mail/statistics</ignore>
    <ignore>/etc/random-seed</ignore>
    <ignore>/etc/adjtime</ignore>
    <ignore>/etc/httpd/logs</ignore>
    <ignore>/etc/utmpx</ignore>
    <ignore>/etc/wtmpx</ignore>
    <ignore>/etc/cups/certs</ignore>
    <ignore>/etc/dumpdates</ignore>
    <ignore>/etc/svc/volatile</ignore>

    <!-- Windows files to ignore -->
<!-- We don't have any Windows devices in this environment
    <ignore>C:\WINDOWS/System32/LogFiles</ignore>
    <ignore>C:\WINDOWS/Debug</ignore>
    <ignore>C:\WINDOWS/WindowsUpdate.log</ignore>
    <ignore>C:\WINDOWS/iis6.log</ignore>
    <ignore>C:\WINDOWS/system32/wbem/Logs</ignore>
    <ignore>C:\WINDOWS/system32/wbem/Repository</ignore>
    <ignore>C:\WINDOWS/Prefetch</ignore>
    <ignore>C:\WINDOWS/PCHEALTH/HELPCTR/DataColl</ignore>
    <ignore>C:\WINDOWS/SoftwareDistribution</ignore>
    <ignore>C:\WINDOWS/Temp</ignore>
    <ignore>C:\WINDOWS/system32/config</ignore>
    <ignore>C:\WINDOWS/system32/spool</ignore>
    <ignore>C:\WINDOWS/system32/CatRoot</ignore>
-->
  </syscheck>
```

```
  <rootcheck>
    <rootkit_files>/var/ossec/etc/shared/rootkit_files.txt</rootki
t_files>
    <rootkit_trojans>/var/ossec/etc/shared/rootkit_trojans.txt</ro
otkit_trojans>
    <system_audit>/var/ossec/etc/shared/system_audit_rcl.txt</syst
em_audit>
    <system_audit>/var/ossec/etc/shared/cis_debian_linux_rcl.txt</
system_audit>
<!-- All linux boxes are debian in this environment
    <system_audit>/var/ossec/etc/shared/cis_rhel_linux_rcl.txt</sy
stem_audit>
    <system_audit>/var/ossec/etc/shared/cis_rhel5_linux_rcl.txt</s
ystem_audit>
-->
  </rootcheck>

  <command>
     <name>firewall-drop</name>
     <executable>firewall-drop.sh</executable>
     <expect>srcip</expect>
     <timeout_allowed>yes</timeout_allowed>
  </command>

  <active-response>
    <disabled>no</disabled>
     <command>firewall-drop</command>
     <location>local</location>
     <level>6</level>
     <timeout>120</timeout>
  </active-response>

  <remote>
    <connection>secure</connection>
    <allowed-ips>10.5.1.0/24</allowed-ips>
    <allowed-ips>10.5.2.0/24</allowed-ips>
    <port>1514</port>
  </remote>

  <alerts>
    <log_alert_level>1</log_alert_level>
  </alerts>
  <!-- Files to monitor (localfiles) -->

  <localfile>
    <log_format>syslog</log_format>
    <location>/var/log/messages</location>
  </localfile>

  <localfile>
    <log_format>syslog</log_format>
    <location>/var/log/mail.info</location>
  </localfile>
</ossec_config>
```

Appendix F

Sample OSSEC agent install script output

```
inta:~/ossec-hids-2.4# ./install.sh

 OSSEC HIDS v2.4 Installation Script - http://www.ossec.net

 You are about to start the installation process of the OSSEC
HIDS.
 You must have a C compiler pre-installed in your system.
 If you have any questions or comments, please send an e-mail
 to dcid@ossec.net (or daniel.cid@gmail.com).

  - System: Linux inta 2.6.18
  - User: root
  - Host: inta


  -- Press ENTER to continue or Ctrl-C to abort. --
<ENTER>

1- What kind of installation do you want (server, agent, local or
help)? agent

  - Agent(client) installation chosen.

2- Setting up the installation environment.

 - Choose where to install the OSSEC HIDS [/var/ossec]: <ENTER>

    - Installation will be made at  /var/ossec .

3- Configuring the OSSEC HIDS.

 3.1- What's the IP Address of the OSSEC HIDS server?: 10.5.2.1

   - Adding Server IP 10.5.2.1

 3.2- Do you want to run the integrity check daemon? (y/n) [y]: y

   - Running syscheck (integrity check daemon).

 3.3-Do you want to run the rootkit detection engine? (y/n) [y]: y

   - Running rootcheck (rootkit detection).

  3.4 - Do you want to enable active response? (y/n) [y]: n

   - Active response disabled.

  3.5- Setting the configuration to analyze the following logs:
    -- /var/log/messages
    -- /var/log/auth.log
    -- /var/log/syslog
    -- /var/log/vsftpd.log
    -- /var/log/mail.info
```

```
    -- /var/log/dpkg.log
    -- /var/log/snort/alert (snort-fast file)
    -- /var/log/apache/error.log (apache log)
    -- /var/log/apache/access.log (apache log)

 - If you want to monitor any other file, just change
   the ossec.conf and add a new localfile entry.
   Any questions about the configuration can be answered
   by visiting us online at http://www.ossec.net .


   --- Press ENTER to continue ---
<enter>
[remainder of interaction identical to server install]
```

Appendix G

UML Startup Script Patch

```
--- uml.sh 2010-07-27 20:52:25.000000000 -0500
+++ uml_ossec.sh      2010-07-12 19:40:20.000000000 -0500
@@ -126,2 +126,3 @@
     VM_COW=$VM_DIR/cow
+        VM_OSSEC_COW=$VM_DIR/ossec_cow
     VM_SWAP=$VM_DIR/swap
@@ -129,2 +130,3 @@
     UBD1="$VM_SWAP"
+        UBD2="$VM_OSSEC_COW,$UML_HOME/data/uml_ossec_fs"

@@ -170,3 +172,3 @@
     export UMID DESKTOP
-    VM="$LINUX xterm=$XTERM umid=$UMID mem=$MEM $ETH ubd0=$UBD0
ubd1=$UBD1"
+        VM="$LINUX xterm=$XTERM umid=$UMID mem=$MEM $ETH
ubd0=$UBD0 ubd1=$UBD1 ubd2=$UBD2"
     cd $HOME || exit 1
```

Appendix H

configure-ossec.sh, to be placed in the `/home/user1/uml/mntdirs/config/ossec-configure` directory, along with the `ossec-agent.tar.gz` file created in the *OSSEC Agent Install Preparation* steps.

```
#!/bin/bash
# This setup assumes the client keys are managed on a network
# basis

#*****************************************************************
# Add the UBDC drive to /etc/fstab
#*****************************************************************
echo "Adding the ubdc disk to fstab"
echo "/dev/ubdc /var/ossec  ext2  defaults    0 1" >> /etc/fstab

#*****************************************************************
# Make OSSEC Dir
#*****************************************************************
echo "Making /var/ossec directory"
mkdir /var/ossec

#*****************************************************************
# Mount the new drive
#*****************************************************************
echo "Mounting /dev/ubdc on /var/ossec"
mount -t ext2 /dev/ubdc /var/ossec

#*****************************************************************
# Accept the network segment and # OSSEC Server IP from the
# calling routine
#*****************************************************************
NETWORK=$1
OSSEC_SERVER=$2

#*****************************************************************
# Add the OSSEC user and group
#*****************************************************************
echo "Adding the OSSEC users and group"
groupadd ossec
useradd -g ossec -d /var/ossec ossec
useradd -g ossec -d /var/ossec ossecm
useradd -g ossec -d /var/ossec ossecr

#*****************************************************************
# Extract the OSSEC install
#*****************************************************************
tar xzvf /mnt/config/ossec-config/ossec-agent.tar.gz -C /
chmod 755 /etc/init.d/ossec
```

```
#*****************************************************************
# Copy the appropriate client.keys entry to
# /var/ossec/etc/client.keys
#*****************************************************************
echo "Copying client.keys file"
grep -i ${NETWORK} /mnt/config/ossec-config/etc/client.keys \
| head -n 1 > /var/ossec/etc/client.keys

#*****************************************************************
# Point the agent to the correct OSSEC server IP
#*****************************************************************
sed -i s/OSSEC_SERVER/${OSSEC_SERVER}/ /var/ossec/etc/ossec.conf

#*****************************************************************
# Add ossec to startup process
#*****************************************************************
echo "Adding OSSEC to startup process"
update-rc.d ossec start 98 2 3 4 5 . stop 20 0 1 6 .

#*****************************************************************
# Start OSSEC
#*****************************************************************
/var/ossec/bin/ossec-control start
```

Bibliography

About OSSEC. (n.d.). Retrieved March 26, 2010, from http://www.ossec.net/main/about/

Cummings, J. (2004, September 27). "Security in a world without borders". *Network World*, Retrieved August 12, 2010 from http://www.networkworld.com/buzz/2004/092704perimeter.html

Hay, Andrew, Cid, Daniel, & Bray, Rory. (2008). *Ossec host-based intrusion detection guide*. Syngress.

Lyon, G. (2009). *Nmap network scanning* (Online Edition), Retrieved August 17, 2010, from http://nmap.org/book/nse.html#nse-intro

Nikto2 | CIRT.net. (n.d.). Retrieved August 11, 2010, from http://cirt.net/nikto2

OSSEC Commercial Support. (n.d.). Retrieved August 12, 2010, from http://www.ossec.net/main/get-commercial-support/

OSSEC Manual Active Response. (n.d.). Retrieved August 12, 2010, from http://www.ossec.net/main/manual/manual-active-responses/

OSSEC Manual manage_agents tool. (n.d.). Retrieved August 10, 2010, from http://www.ossec.net/main/manual/manual-manage_agents-tool/

Perez, D., & Perez, C. (n.d.). NETinVM. Retrieved March 14, 2010, from http://informatica.uv.es/~carlos/docencia/netinvm/