Global Information Assurance Certification Paper

Designing a Responsive Intrusion Detection System on HP-UX 11i

GCUX - Version 1.9
Option 1 - Securing Unix Step by Step

Hazem Mahmoud
November 2003

Table of Contents

Abstract:

The purpose of this paper is to develop and implement the concept of a Responsive Intrusion Detection System (RIDS). The purpose of RIDS is to add intelligence to an already existing Intrusion Detection System. RIDS empowers an IDS system to respond to attacks in an intelligent manner, in order to reduce the damage or eliminate them completely. This allows a System Administrator to focus on the more advanced situations that may arise in an attack, and will provide a quicker response time to handle the intrusion. In this practical I will be implementing the RIDS concept from Hewlett-Packard's implementation of IDS, "IDS/9000".

1.0 Introduction

An Intrusion Detection System (IDS) is a system designed to detect unauthorized or malicious activities on a system. A system here could mean a standalone host or a network of hosts. For this report, I will implement an IDS server whose function is to detect and respond to certain unauthorized or malicious activity on a network of hosts. IDS is not meant to be a standalone security feature, in the sense that you cannot rely entirely on IDS to secure your network. Another security feature, such as a firewall, must be installed to secure the network from external attacks. While a firewall secures a network from outsiders, the IDS is responsible for detecting any intrusions that may occur from external or internal attacks.

IDS/9000 is Hewlett-Packard's implementation of an intrusion detection system. For my research I will initially be installing, configuring, and testing an IDS/9000 Administration Server, whose purpose is to search for suspicious activity on a network. One of the major advantages of IDS/9000 over other IDS products is the fact that the detection of intrusions is done at the operating system or kernel level. This allows the system to stop core types of attacks, such as race condition attacks and does not concern itself with the many flavors and forms that a race condition attack can take. Therefore, it is basically focusing on the core attack concept as opposed to the many different methods that a hack can be executed. The major focus of my research however will be on developing a Responsive IDS (RIDS) system using IDS/9000.

What do I mean by a "Responsive" IDS system? It is basically an IDS system with intelligence. The purpose of IDS is to *monitor* and *detect* certain unauthorized behavior. It is not designed to *react* to situations as they occur. This is where I am heading: to create an IDS system that responds to certain scenarios appropriately. The method I will use to accomplish this is to create a Unix script that will allow the IDS system to act accordingly to certain intrusions. The reasons for this development from a real-world perspective are many. Hackers do not necessarily (if ever) work the eight to five workday. Therefore, more than likely, a system will be compromised when there is no one around to

detect it and stop it. So assuming that the system administrator has installed and configured the IDS/9000 system properly, delays in alerts reaching the administrator's pager or the time it takes for the administrator to wake up and log into the system, might be all the time an intruder needs to wreck havoc. Wouldn't it be nice to empower the IDS system to perform some simple security housekeeping procedures whenever an intrusion is detected, in order to prevent further damage to the host or the network? This would buy the system administrator some time and allow him/her to focus on the problem and assessing the damage caused instead of spending time on performing certain tasks that can be automated by the IDS system. Please note, this is not intended to replace a system (or security) administrator's role in responding to intrusions, but merely buy them time and allow them to focus on the other aspects of resolving a system intrusion. RIDS is also not meant to be an all-inclusive security function. It does not cover all the types of intrusions that can possibly occur, unless of course it is constantly updated for new types of attacks. This paper is merely designed to introduce the concept of RIDS, and therefore only addresses a sample set of intrusions.

There are many responses that the Unix response script can take, depending on the nature of the attack. However, for the majority of the detected attacks, an email will be generated and sent to the Administrator, notifying him/her of the attempt. Many times an attacker will attempt to cover their tracks by modifying or removing system log files. The response in that case will be to attempt to preserve the logs by emailing them to the Administrator or copying them as hidden files to another location. The response for brute-force type attacks, where the attacker attempts to break into a user's account by guessing the password, will be to lock that user's account so that the hacker cannot enter. For user's already on the system and attempting to hack into another user's account through the "su" command, we will want to lock the account of the user executing the "su" command. For race condition attacks, we will deactivate the user's account from which the attack originated, but we will also remove the attacking program from the system to prevent future attacks. Before we remove the infected program however, we will preserve it for analysis. Regarding the dangerous buffer overflow attacks, the response will be to kill the malicious program and again preserve it for analysis. All these are examples of how RIDS will work through the Unix response script, and I will go into more detail on each one later in the paper. The one important note here is that the types of attacks and responses I have mentioned are only a sample. The purpose of this paper is not to address every type of attack, but to present a concept that can be built on and enhanced. A key benefit to this system is that the Unix response script can be easily modified and enhanced as new attacks emerge.

The concept of an Intrusion Prevention System (IPS) is similar in theory to what I am implementing, but there exists a fundamental difference between the two. An Intrusion Prevention System can take it one step further and actually prevent an attack from occurring. To accomplish that, an Intrusion Prevention System,

creates near real-time responses to attacks, and therefore, *prevents* unauthorized activities from occurring. In order for the “near real-time” responses to become practical, the code for the response programs must be embedded in the core of the system. In Unix, this would probably be the kernel. Other Intrusion Prevention Systems are actual hardware devices that plug into your network and can communicate with the firewall to block packets received from suspicious intruders. We can safely say that Intrusion Prevention Systems are much more advanced and require a great deal of time and money to develop. If I were to call this paper an Intrusion Prevention System, I would be oversimplifying the concept of an IPS. Hence the phrase “Responsive IDS” is more appropriate, because it responds to certain intrusions intelligently to minimize or contain damage, but it does not necessarily prevent them like an Intrusion Prevention System would.

In order to test my research, I will be using an HP-UX Visualize workstation as the IDS/9000 Server, referred to in this document as the IDS/9000 Administration Server. I will also be using another HP-UX Visualize workstation as the IDS/9000 Client host machine, referred to in this document as an “Agent”. The Agent will serve as a host on the network to which the IDS/9000 Administration Server will attempt to protect. In implementing this in the real world, there would most likely exist many Agents that the Administration Server would monitor. For testing purposes, I will only be implementing one Agent. The same procedures, with slight modifications, can be used for multiple Agents.

The first step in developing RIDS is pre-configuring the system for the installation. This includes determining vulnerabilities and threats, hardening the system to protect from those threats, and installing any necessary patches. The second step is to install the IDS/9000 software on both the Server and Agent. The third step is to configure the software so that it can detect intrusion attempts. The fourth step is to develop and deploy the custom script(s) to turn the system into a Responsive IDS. Finally the system will be tested, and we will discuss ways of properly monitoring the system and performing regular system maintenance checks to ensure the highest level of security while maintaining the efficiency of the systems.

1.1 Role of Host

After the setup and configuration is complete, the final state of the Server on the network is to serve as a Responsive Intrusion Detection System (IDS). The goal of the Responsive IDS (RIDS) is to detect and respond to unauthorized or malicious activity on the network. This will be accomplished using IDS/9000 and custom Unix script(s) developed to turn the system into a RIDS system.
The role of the Agent Host is to allow for testing RIDS over the network. The Agent will detect intrusions in its local workstation and notify the Administration Server of these intrusions. It will further go on to respond accordingly to certain types of intrusions by executing the appropriate portions of the script.

1.2 System Description

For this setup, I will be using an HP C3000 Visualize workstation (A4986A) as the RIDS Server. The Agent will be an HP B132L workstation (A4190A) to test the functionalities of RIDS on its local machine.

The IDS/9000 administration software is GUI-based and can be accessed either through the network or the console. If accessed through the console, you will not have as much of a security concern as if accessed over the network. If accessed over the network, proper security measures can be taken with X-Windows to ensure that the data is secure. It can be tunneled through an SSH connection to ensure that the data is encrypted in transit over the network. To perform this, you need to establish an SSH connection with the Administration server, perform the necessary commands in order for the GUI display to appear on your computer (ie: “export DISPLAY=<ip_address>:0.0”), then start up the administration software. Regarding the Agent systems, there are no GUI applications that will run on them. They will require the IDS/9000 agent software as well as SSH to be installed on them. We will go into more detail regarding the software installation later in the paper.

The table below displays the specifications for the Administration and Agent systems, which are running HP-UX 11i. Please note that the systems required a CD-ROM for the installation of the HP-UX 11i Operating System.

Table 1 - System Description

Hostname / Purpose	IP Address	Processor	Memory	Disk Space	Operating System
orion / Admin System	xxx.xxx.0.7	400 MHz PA-RISC	512 MB	2 x 9 GB	HP-UX 11i
gemini / Agent System	xxx.xxx.0.5	132 MHz PA-RISC	384 MB	1 x 9 GB	HP-UX 11i

1.3 Software Required

HP-UX 11i Core OS
HP Secure Shell A.03.61.001
IDS/9000 Release 2.1
JAVA SDK/RTE 1.3.1
Unix Shell Script to implement RIDS (Appendix A)

1.4 Additional Patches

These are the patches required for an HP-UX 11i OS to be able to run the IDS/9000 Software:
PHKL_26074 (fix for system panic for Agent systems)
JAVA Patches: PHCO_26061, PHCO_27632, PHCO_27740, PHCO_27958, PHKL_24751, PHKL_25233, PHKL_25993, PHKL_25994, PHKL_27091, PHKL_27094, PHKL_27096, PHKL_27316, PHKL_27317, PHKL_27686, PHKL_28122, PHKL_28267, PHNE_27703, PHNE_28089, PHNE_28103, PHSS_26971, PHSS_26973, PHSS_26975, PHSS_28370, PHSS_28470

2.0 Risk Analysis

The major security concern of this system is for the RIDS Server or one of the Agents to be the subject of an attack. The RIDS Server is the backbone of this system. It contains the IDS/9000 System Manager, which manages all the hosts on the network. Unauthorized access to this Server, as user *root* or *ids*, can cause the IDS/9000 software to be disabled or rendered non-functional, leaving all the hosts on the entire network open for malicious attacks. Therefore it is imperative to properly secure the RIDS Server. An attack on the RIDS Agents is just as critical. The Agent systems house the Unix response script and an attack on them can cause the response script to also be nonfunctional, therefore not allowing them to carry out the proper response in the event of an attack or intrusion. All threats to these systems from any vulnerabilities must be examined and corrected.

2.1 Vulnerabilities & Threats

In order to secure the IDS Server, we need to first define the vulnerabilities in the system and the threats that exist. After defining the vulnerabilities and threats, we can go ahead and determine how to secure the server. According to Pfleeger's book "Security in Computing", a vulnerability is defined as a "weakness in the security system", while a threat is a "set of circumstances that has the potential to cause loss or harm". Meaning that a threat is present when the system is vulnerable. For example, not having an intrusion detection system in place is a vulnerability in the system. The threat caused by this vulnerability is that someone can attack a host on the network without being detected. Let us now examine the vulnerabilities and threats to this specific system, and then we can discuss how we can protect the systems from these threats by eliminating the vulnerabilities.

2.1.1. Vulnerabilities

In order to ensure proper implementation of the RIDS system, we must eliminate the vulnerabilities. Out of the box, the HP-UX system has many vulnerabilities. There are a lot of unneeded services and daemons that are by default left on. These include network services such as telnet,ftp,rlogin, time, talk, daytime, etc. These are all nice features that come activated when HP-UX is installed, but have many security holes in them that attackers can exploit. There may also be unnecessary suid/sgid programs. These are programs that run as the owner or group of the file. Other vulnerabilities include weak password policies, expired user accounts that are still active, unencrypted data over the network, and much more. These must also be identified and dealt with. The rule when it comes to eliminating vulnerabilities is: if you do not need something then get rid of it. Another important vulnerability that is often overlooked is the physical security of the system. It serves no good to have the latest and greatest security applications and devices if an attacker can just waltz in and pull the plug. Physical security can be a huge vulnerability in many corporations. We just

touched on some of the vulnerabilities but will discuss in detail how to eliminate them in the section entitled “Hardening HP-UX System”.

2.1.2. Threats

The threats to the system arise from vulnerabilities that exist. The first vulnerability we mentioned above was unneeded network services. Network services such as rlogin are weak in nature when it comes to security. First off, the transmission of data between the two systems using rlogin is not encrypted. Second, rlogin (and all the other “r” commands) do not require a password to log in from one system to the next. When rlogin is used, the /etc/hosts.equiv file on the target system is consulted to see if the user from the source system can log in without a password. If not, then the .rhosts file in the home directory of the user for the target system is checked to see if permission should be granted. This basically allows for the same user to access different servers, if the same username exists, without having to provide a password each time. The threat to this functionality is obvious, and must be addressed. Examples of other services that need to be secured are Telnet and FTP. While there functionality is necessary for usage of the system, they are vulnerable in nature. Their vulnerability lies in the fact that they transmit their data in plain clear text over the network. Anyone with a simple network sniffer can listen in on packets on the network and steal passwords of users as they enter their username/password to log onto their systems. This is a serious vulnerability, and fortunately there are substitutes that can be used for both. We can use SSH to provide a secure shell which encrypts data before sending it off over the network. For FTP we can use SCP (secure copy) or SFTP (secure FTP). HP provides an SSH (A.03.61.001) product free of charge. The HP Secure Shell package has commands to perform remote login similar to Telnet called “ssh”, and remote file transferring similar to FTP called “scp” or “sftp”. This solution will definitely resolve the vulnerabilities inherent in Telnet and FTP.

The second vulnerability mentioned was the suid/sgid programs. These programs run as the owner of the file. For example, if root is the owner of a program that has the suid bit sit, then a smart hacker might find a way to compromise the program and take control of the shell that the program is running in. Therefore they would have a root shell available to them. To resolve this vulnerability, you would need to keep track of all the suid/sgid scripts and programs on the system. This can be done through a simple “find” command and is discussed below. If a new suid/sgid script appears, you would need to determine why it is there and if it contains any vulnerabilities. These, and many others, are the types of threats that exist on out-of-the-box systems. To eliminate these threats, we do what is called “hardening” the operating system. This involves securing the operating system and the services it provides to ensure that the vulnerabilities are not exploited by hackers. Hardening the system will be covered in the step-by-step guide in the next section under “RIDS Installation & Configuration”.

3.0 RIDS Installation & Configuration

In this section, I will go over the step-by-step procedures involved in installing the HP-UX operating system. We will then discuss the methods and techniques of hardening and securing the HP-UX operating system. After the system is in a secure state, we can then proceed to install the IDS/9000 application and then configuring it to our specific needs. After the installation and configuration of the IDS/9000 product is complete, we can then proceed to implement the focus of this paper, which is to convert the IDS system into a Responsive IDS system. This approach will be taken for the RIDS server as well as all the RIDS clients. However, the implementation will be different for both.

3.1 Installing HP-UX Operating System

The installation of the HP-UX operating system is the first step in the process of creating our RIDS system. Before proceeding to install HP-UX, you must first collect some networking information to help configure the networking portion of the installation. The information you will need to gather includes the hostname and IP address of the system, subnet mask, gateway, and DNS server. You will also need to obtain the HP-UX 11i core operating system, which usually resides on a CD. For this installation process we will be using the CD to install the HP-UX operating system.

Boot up your system and wait for the message:

Processor is starting autoboot
To discontinue press a key within 10 seconds.

At this point hit any key to interrupt the boot process. Insert the HP-UX 11i Core OS CD into the CD-ROM drive. You will be given a menu of available options, and then the following prompt will display:

Main Menu: Enter command or menu >

You will most likely not know the path number that the CD-ROM is associated with. Therefore type in the following:

Main Menu: Enter command or menu > search

You will be given a list of the devices on the system and their associated path number. Find the path number of the CD-ROM (let's assume the path is P0) and enter the following:

Main Menu: Enter command or menu > boot P0
Interact with IPL (Y or N)? > N

The system will now boot from the CD-ROM and begin the installation process. Throughout the boot process, you will be asked to make decisions and to select certain items. This will not be GUI-based (Graphical User Interface) and so you will not be able to use the mouse at this point. To navigate the screen you can press the “Tab” key on the keyboard. This will allow you to jump from one option to the next. To select a specific option, just hit the “Return” or “Enter” key on the keyboard. The first screen you will encounter will give more detailed instructions on the navigation methods in the installation process.

After the system has finished booting off the CD, you will then see a welcome display and will be requested to choose between installing HP-UX, running a recover shell or other advanced options available to you. Select “Install HP-UX” by hitting “Return”. The next screen will prompt for the installation source, which you will want to choose “Media only installation”. The option for an Ignite server is generally used if you have a large number of systems that you want to roll out the operating system to, along with any other additional software. For the purposes of not overcomplicating this section, we will not use this option. Our installation will be through a CD media. On the same screen you will be prompted for the interface of the installation, and you will want to choose “Advanced Installation”. This will allow us more options in case you should need to perform any more advanced settings specific to your environment and setup. After those two are selected (with the asterisk character, “*”) tab over to “OK” and hit “Return”. The next screen will prompt for a general configuration of the system. Select the default option of “HP-UX B.11.11 Default”. The next screen will include five (5) tabs labeled “Basic”, “Software”, “System”, “File System”, and “Advanced”. You will most likely want to keep the default settings for the majority of these options, but depending on your environment, you will probably want to modify some of the settings. For the purposes of this paper, you will want to make sure to perform at least the following setting. IDS/9000 stores its log files in /var/opt/ids/, and they can get considerably large. For that reason we will want to create a separate logical volume for /var/opt/ids. This will ensure that the log files will not fill up the /var logical volume, causing the system to crash. To perform this, tab over to the “File System” tab. Inside this tab you can select to add a new logical volume. Specify the following values to add logical volume /var/opt/ids, and then select “Add”:
Usage: HFS
VG Name: vg00
Mount Dir: /var/opt/ids
Size: Fixed MB (specify the size depending on how much disk space you are willing to allocate)

After you have added the new logical volume, you may want to reconfigure some of the existing default logical volumes by selecting the volume and then select “Modify”. For the purposes of this paper, we will use the default values specified by the installation process. You can now select “Go!”, and at this point the configuration process for the installation begins.

You will receive a welcome screen and will be asked:

Are you ready to link this system to a network?

Press [y] for yes or [n] for no, then press [Return]

Hit “y”, then return and you will then be asked to continue if you have the hostname IP address and time zone. Press “y” again and then return. You will then be prompted with the following:

Enter the system name, then press [Return] or simply press [Return] to retain the current host name (host name): orion

Enter the hostname at this point, “orion” in my case, and hit return. You will then be prompted for the location and time zone. Enter the appropriate corresponding number to each entry and hit return. It will then confirm that the current system is correct. If it is, type “y” and hit return. If not, type “n” and it will walk you through setting up the correct time. Next, you will be asked about choosing a password at this time. It is recommended to set it now. Make sure to use good password selection policies when selecting a password. Do not use words from the dictionary, variants of the hostname, or any other password that can be cracked with a few iterations. A good source of information on proper password policies can be found at http://www.sans.org/resources/policies/Password_Policy.pdf.

At this point in the configuration you are done with the system specific settings, and next comes the network specific settings. You will be prompted to enter your IP address first:

Enter your IP address, then press [Return] or press [Return] to select the current address (192.168.0.7): 192.168.0.7

Enter your IP address, and then you will be asked whether you would like to proceed with configuring additional network settings such as the subnet mask, gateway, DNS, and NIS service. These are specific to how your network environment is structured. For the purposes of this paper, I have not set up these parameters. When prompted with the following I entered “n”:

Do you want to configure these additional network parameters?
Press [y] for yes or [n] for no, then press [Return]

Finally, the basic installation is complete. If you decide to perform any modifications or add additional settings to your basic configuration, you can execute ***set_parms*** at the HP-UX command prompt and it will walk you through any modifications you would like to make. At this point, the system is in its most basic form and is therefore in its most vulnerable state. All network services are turned on, security patches are not installed, and various other configurations are weakining the system. We will now discuss the many security measures that need to be taken to properly secure all our hosts.

3.2 Hardening HP-UX System

To secure the vulnerabilities in the IDS/9000 Server and the Agents, we are basically hardening or securing the Operating System. To accomplish this, the least we need to do is the following:

1. Disable network services not needed
2. Turn System into a Trusted System
3. Install HP-UX Secure Shell
4. Install Security Patches
5. Implement Additional Security Precautions
6. Examine Physical Security

I have created a script that can be used to harden the system. It will convert a system to a Trusted System, restrict root access only to the console, carry out additional security precautions and make recommendations for changes that need to be made on the system, but are optional. The script can be found in Appendix A and is called "secure.sh". Copy this script into the Administration system and each Agent system and execute it as root. You need to be root in order for it to perform the security conversion on the system. Kevin Steve's *Building a Bastion Host* was the source in developing the concepts discussed in this section.

3.2.1 Disable Unused Network Services

There is a huge risk involved in keeping network services that are not used enabled. The more features you empower your computer to have, the more vulnerable it is. By keeping network services enabled, you allow an intruder more passages and openings to try and hack in. A hacker can exploit known vulnerabilities in network services, and it is therefore highly recommended to disable any service that we do not need.

Services that are running on the system are listed in a file called /etc/inetd.conf. To disable a service, you just need to comment out the line that lists the service by placing a "#" sign at the beginning of the line. The services that are not needed will vary from system to system. If your server functions as an IDS/9000 Administration server, and is also an FTP server, than you will want to keep the entry for the ftp service. However, if the system is solely used as an IDS/9000 Administration server, then the list below displays the network services I recommend that we disable. The same list also applies to the Agent systems. The script "secure.sh" disables these services for you automatically. If for any reason you need to reactivate any of these services, you can go into /etc/inetd.conf and remove the "#" sign at the beginning of the line that contains the service.

Network services not needed:
login, telnet, ftp, shell, exec, ntalk, daytime, time, echo, discard, chargen

3.2.2 Convert into Trusted System

One of the biggest security risks is someone hacking the password file (/etc/passwd). On a normal HP-UX system, the password file is readable to all. Even though you may restrict access to the IDS/9000 Server, you still run the risk of a user finding a vulnerability and gaining access to the password file. The passwords are encrypted, however, someone can run a crack program offline and determine the passwords. Once they have the root password, they have root access, and can basically take control of your entire second line of defense. They can manipulate log data, or corrupt a configuration file to make the IDS/9000 seem like it is doing its job, while the whole time it is being manipulated to not inform the administrator of any attacks. To prevent such a threat from someone gaining access to the password file, we would need to convert the system into a Trusted System.

A trusted system is a secure system, which among other features, removes the passwords from the /etc/passwd file and places them in files readable only by root. The passwords are still encrypted, however now only the root user can read the encrypted file, as opposed to a world-readable password file. The basic command for converting a system into a Trusted System is /etc/tsconvert. While having a Trusted System is an added security feature, it is not required in order to install the IDS/9000 application. The script “secure.sh” will automatically convert the system into a Trusted System.

3.2.3 Install HP-UX Secure Shell

HP-UX Secure Shell, is the HP implementation of the popular SSH technology. This technology allows the encryption of data over the network to prevent eavesdropping on the network by attackers. A common hacking tool is something called a network sniffer. A network sniffer listens to packets transmitted over the network. Therefore if you are logging on to a server using Telnet/FTP and you input your username/password, the hacker with the network sniffer can read the username/password as well as any other data that you send over the network. SSH encrypts data before being transmitted over the network, therefore protecting you from the hacker with the network sniffer. This is required for properly implementing the RIDS system in a secure manner. Later on we will see that during the installation/configurarion portion of IDS/9000, we will need to copy over the certificates from the Administration system to each Agent system. Certificates are the means by which a host authenticates itself to another host. SSH is the most secure method of copying these certificates across the network. If SSH is not used, you can risk someone intercepting the certificate.
To install, perform the following steps:

1. Go to: https://payment.ecommerce.hp.com/portal/swdepot/try.do?productNumber=T1471AA fill out the required information and download the depot

T1471AA.depot to your depot directory (ie:/depot) on the Administration server.
2. Execute: “swinstall -s /depot/T1471AA.depot”
3. Verify that the SSH daemon (sshd) is running by executing: “ps -ef | grep sshd”
4. Depending on your operating system version, you may need to install additional patches.
5. Attempt to establish a connection with another server using the “ssh” command.

3.2.4 Install Security Patches

There are constantly new vulnerabilities that are discovered in the operating system and the services it offers. Therefore it is important for us to ensure that we keep the latest security patches up to date on the Administration system as well as all the Agents. The links listed below will aide you in determining which patches are relevant to your systems.
ftp://ftp.itrc.hp.com/hp-ux_patches/ - This location contains the patches
ftp://ftp.itrc.hp.com/export/patches/hp-ux_patch_matrix - Matrix of latest patches
ftp://ftp.itrc.hp.com/export/patches/security_catalog - Patch catalog

3.2.5 Additional Security Precautions

There are many other additional security measures that we can take to help solidify our systems. These are not necessary, but do add extra features to help keep our systems in a more secure state. I will mention just a few measures that we can take to help secure our systems.

First measure we can take is to restrict root login to the console. The benefit of this is that it allows limited remote access. Only users set up on the system can log on remotely. Since we have disabled Telnet, they will need to use SSH to log in. After logging on, they can then execute “su –“ to log in to the root account after providing a password. So it basically allows for more control over the root account. The script “secure.sh” will perform this. To accomplish this, we can also do the following:

```
# echo console > /etc/securetty      #As root user
# chmod 400 > /etc/securetty
```

The second measure we can take is to check the /etc/passwd file for any users that are no longer using the system. Also check the /etc/group file for any inactive groups. If there are any users/groups no longer using the system, you can remove them with the following:

```
# userdel <username>      #As root user
# groupdel <groupname
```

This will remove the files from /tcb/files/auth, which is the location (in a Trusted System) of the user properties. They are actually located under /tcb/files/auth/<first_letter_of_username>/. The script "secure.sh" will list the users and groups in the system so you can examine them.

The third measure we can take is to locate any suid/sgid programs on the system. These are program that when executed, run in a shell logged on as the user owner (or group if sgid is found) of the program. To determine if a program has the suid bit set, execute "ll <program_name>" and if the execute bit, "x", is set in the owner permissions to "s", then the program will run as the owner of the program. To check if the sgid bit is set, check the execute bit for the group permissions. "secure.sh" will locate all the suid and sgid programs on the system. To do it manually, execute the following as root:

```
# find / \( -perm –4000 –o –perm –2000 \) –type f –exec ls –ld {} \;
```

The last, but not least, measure we can take is to make sure that command history is enabled. Command history stores the commands executed by a user, in a file called .sh_history. This serves as a good auditing tool to see what the user executed. For root, it is saved under "/", and for other users, the file is saved under "/home/<userid>/". To enable, just make sure that .sh_history exists in the users home directory. The environment variable HISTFILE defines the location of the history file and HISTSIZE defines the number of previous commands to save in .sh_history. Both these variables can be set in the users .profile file. "secure.sh" will notify the user if which users has .sh_history activated.

3.2.6 Physical Security Considerations

Physical security is one of the most critical, yet often overlooked, issue when it comes to security. When all is said and done, if an unauthorized or disgruntled individual gains physical access to the Administration server, all security measures can be considered worthless. Booting the server into single-user mode, will allow the intruder to gain root access to the system. After that there is nothing to hold them back from causing all the damage they want. If they are really disgruntled, they can even take it one step further and physically damage all the hardware. Therefore it is imperative that proper physical security measures are taken. Physical security involves allowing access to authorized individuals. If an unauthorized individual has access to the physical hardware, your physical security is considered weak. This is a risk that should not be taken lightly.

3.3 IDS/9000 Installation

We will be installing both the IDS/9000 software as well as the Java software on the Administration system required for the GUI screens in IDS/9000. Before continuing however, you must have root access. I will use depots to install all the software together at one time, so that we only need to reboot once. For more information on the installation, and any specific patches you may need for your system, please refer to *HP Intrusion Detection System/9000 Release 2.1 Release Notes*. This document proved to be very helpful in installing IDS/9000 properly on my systems.

To install IDS/9000, you will need to download it from the Internet at http://software.hp.com, click on “security and manageability”, then click on the link for “hp intrusion detection system 9000”. The software is free, and below is the product information:

IDS/9000 Information:
Name: J5083AA
Version: 2.1
Information: HP Intrusion Detection System 9000 (IDS/9000)
Size: 30 MB
Architecture: HP-UX_B.11.11_32/64

3.3.1 Pre-Installation Configuration

IDS/9000 generates logs, such as /var/opt/ids/alert.log, which logs all the alerts generated. These logs can often get quite large, and therefore it is recommended that a separate logical volume be created for /var/opt/ids. Otherwise, as the logs from IDS/9000 get larger, they will fill up the entire /var file system. So it is better to keep the logs on their own file system (/var/opt/ids) so as to not cause any system crashes. Instructions for creating this logical volume can be found under the section “Installing HP-UX Operating System”. As far as the remaining file system structure, I am keeping the defaults that come with the installation, because there is no need to modify any of them. You may choose to increase or decrease some of the sizes, but as far as the structure is concerned, we are only adding one more file system (/var/opt/ids) to the default structure.

Installing IDS/9000 requires a system reboot. In the case where something might go wrong, it might be wise to preserve the old kernel. To preserve the old kernel, make a copy of /stand/vmunix and rename it to /stand/vmunix.old. In the case where the system might not come up after installing IDS/9000, you can log back in to single-user mode and restore the old kernel. Another recommendation is to take a full-system backup before the installation and after the installation. All this will ensure that the system can be restored if anything wrong should happen.

3.3.2 Installing IDS/9000 Administration System

We will now proceed to install IDS/9000 on the Administration system. The approach will be to create a depot to place the IDS/9000 software, the Java software and the required patches. After all has been copied to the depot through "sftp" - the secure FTP solution, we can proceed with the installation. This installation includes both the IDS/9000 Administration and Agent software. The IDS/9000 Administration server must also have the Agent software installed on it. Below are the instructions for installing the IDS/9000 Administration system. You must be logged in as *root*.

```
# mkdir /var/depot/                    #create depot directory

ftp HIDS-J5083AA_11.11.depot to /tmp
ftp PHKL_26074 to /tmp
ftp sdk13_13109_1100.depot to /tmp
ftp JAVA patches (listed above)

# swcopy –s /tmp/HIDS-J5083AA_11.11.depot \* @ /var/depot/ids_11i_admin+agent
# sh –c 'for i in /tmp/PH*; do sh $i; done'
# sh –c 'for i in /tmp/PH*.depot; do swcopy –s $i \* @ /var/depot/ids_11i_admin+agent; done'
# swinstall –x autoreboot=true –s \ orion:/var/depot/ids_11i_admin+agent \*
```

3.3.3 Installing IDS/9000 Agent System

The next stop involves installing the IDS/9000 Agent system. The IDS/9000 Agent needs to be installed on each Agent you wish to monitor on the network. The same approach to installing the IDS/9000 Administration system will be taken here as well. We will create a depot and copy, through "sftp", the IDS/9000 software and all necessary patches and perform the installation in one step. The Java software does not need to be installed on the Agent systems. If you have a large number of hosts, you might want to consider automating the installation through an Ignite server. Below are the instructions for manually installing the Agent software on each host on the network. You must also be logged in as *root* to perform these installations.

```
# mkdir /var/depots/                   #create depot directory

ftp HIDS-J5083AA_11.11.depot to /tmp
ftp PHKL_26074 to /tmp

# swcopy –s /tmp/HIDS-J5083AA_11.11.depot IDS.IDS-Agent @ /var/depot/ids_11i_agent
# sh PHKL_26074
# sh –c 'for i in /tmp/PHKL_26074.depot; do swcopy –s $i \* @ /var/depot/ids_11i_agent; done'
# swinstall –x autoreboot=true –s \ gemini:/var/depot/ids_11i_agent \*
```

The installation creates many important directories. It is worth learning about these directories and understanding what they contain. This will in help better securing the Administration server and Agent hosts and will also help in

troubleshooting if a problem arises. It is also recommended to backup these directories so you can restore them in the case of any corruption. The following are the directories and their significance:

Administration System:
/opt/ids/bin – Programs (idsagent, idsadmin, idsgui, etc.)
/etc/opt/ids – Configuration file
/etc/opt/ids/certs – Certificates for Administration and Agent systems
/var/opt/ids/gui/logs – Agent logs (for all Agents)
/var/opt/ids/gui/SurveillanceSchedules – Configured schedules
/var/opt/ids/gui/SurveillanceGroups – Configured groups
/var/opt/ids/gui/Templates – Configured templates
/opt/ids/share/man – MAN Pages

Agent System:
/var/opt/ids – Agent logs (alert.log and error.log)
/etc/opt/ids – Agent configuration file (ids.cf)
/etc/opt/ids/certs/agent – Certificates (agent.pem and cacert.pem)
/opt/ids/bin – Programs (idsagent, etc.)
/opt/ids/response – Response scripts

3.4 IDS/9000 Configuration

After installation, we need to go through the following steps in order to properly configure the Administration and Agent Systems:

1. Generate certificates for all systems
2. Create hosts to monitor through System Manager
3. Create surveillance schedules through System Manager

For more information on configuring IDS/9000, please refer to *HP Intrusion Detection System/9000 Administrator's Guide*. This guide was extremely valuable in helping me to configure IDS/9000 on the system.

3.4.1 Certificate Generation

Communication between the Administration System and the Agent Systems must be secure. The importance of this security is to prevent the threat of someone interrupting the communication link, such as a man-in-the-middle attack. The threat of disruption of the communication link can cause several problems such as feeding the Administration System the wrong information or preventing accurate information to the Administration System. For example, an intruder can capture a message and alter it and send it to the Administration server, therefore hiding his/her presence. For the sake of simplicity, they can simply capture the messages and drop them so they never reach the Administration server. All these attacks are possible, but are eliminated with the use of secure communications through certificates between the Administration server and the Agents.

To properly secure the communication link, IDS/9000 provides security between the Administration and Agent Systems using the Secure Sockets Layer (SSL) protocol. Certificates are the components that verify the identity of the system through the SSL protocol, and allow for reliable communication. It allows the Administration System to identify and verify the Agents. This serves as an authentication tool so the Administration System can verify that the Agent is a legitimate system. Certificates are generated by the Root Certification Authority (Root CA), which uses the ITU-T X.509 authentication standard.

We will first generate the certificate for the Administration System by logging on to the Administration Server and executing the following sequence of commands:

# su – ids	***#Must be user ids***
$ cd /opt/ids/bin	***#Location of scripts***
$ IDS_genAdminKeys install	***#Creates Root CA #(Certification #Authority) and #Admin certificate***

The agent certificates are generated on the Administration System and then moved over to each individual Agent. To generate the agent certificates, execute the following command on the Administration Server:

```
$ IDS_genAgentCerts                    #Create Agent certificate(s). #Enter hostname/IP
                                       of each #Agent #then CTRL-D when done.

==> Be sure to run this script on the IDS Administration host.

Generate keys for which host? gemini
Generating key pair and certificate request for IDS Agent on gemini…
Signing certificate for IDS Agent on gemini…
Certificate package for IDS Agent on gemini is /var/opt/ids/tmp/gemini.tar.Z
Next hostname (^D to quit)? CTRL-D
```

The agent certificates are stored on the Administration System in the file: /var/opt/ids/tmp/hostname.tar.Z. Each certificate must be transferred to its respective Agent system through some secure medium. This is where SSH comes into play, allowing us to securely transfer the files to each Agent. It is critical that it is transferred securely so that it is not intercepted and copied. Transfer them to the following /var/opt/ids/tmp directory on each Agent System and execute the following sequence of commands on the Agent Systems to install the certificate:

```
# su – ids                             #Must be user ids

$ cd /opt/ids/bin                      #Location of scripts

$ IDS_importAgentKeys /var/opt/ids/tmp/hostname.tar.Z & <AdminHostName>
        #Import certificate into IDS Agent System
```

One issue that might arise is before executing the last command (IDS_importAgentKeys), make sure that the certificate file (hostname.tar.Z) has the owner and group set to “ids”. Otherwise the script will exit with an error:

```
$ IDS_importAgentKeys /var/opt/ids/tmp/hostname.tar.Z & <AdminHostName>

Usage: IDS_importAgentKeys key_bundle.tar.Z admin_hostname
Could not find the bundled key file /var/opt/ids/tmp/hostname.tar.Z
```

The reason it exits with that error is because of this portion of the IDS_importAgentKeys script:

```
if [ ! -r $1 ]; then
    echo "\nUsage: $0 key_bundle.tar.Z admin_hostname\n"
    echo "Could not find the bundled key file $1\n"
    exit 1
fi
```

In this portion, it is checking to see if the file, /var/opt/ids/tmp/hostname.tar.Z, is readable. So if you are user "ids", which you should be, and you try to execute the script, which is also user *ids,* but the "hostname.tar.Z" file is owned by user "root" or "hmahmoud", then the script will not be able to read the file. The error message is a little confusing however, because it leads you to believe that the script cannot even find the file.
In order to resolve this problem, you would need to execute the following commands, which will change the owner and group of "hostname.tar.Z" to *ids*:

```
# chown ids /var/opt/ids/tmp/hostname.tar.Z   #as user "root"
# chgrp ids /var/opt/ids/tmp/hostname.tar.Z   #as user "root"
```

3.4.2 Create Hosts in the System Manager

After the certificates have been installed on the Administration Server and the Agents, the systems are now ready to start communicating. However, we first need to notify the System Manager of the Agents to monitor. The System Manager is the administration console utility shown below that allows for system administrative tasks of the IDS/9000 product. It allows you to add hosts (Agents) to monitor, create and deploy schedules to Agents, and monitor alerts. The GUI for the System Manager will require X-Windows if running through the network. In that case, you will want to ensure that X-Windows is properly secured. Tunneling the connection through an SSH connection would be the best approach to ensure that data going back and forth is encrypted.

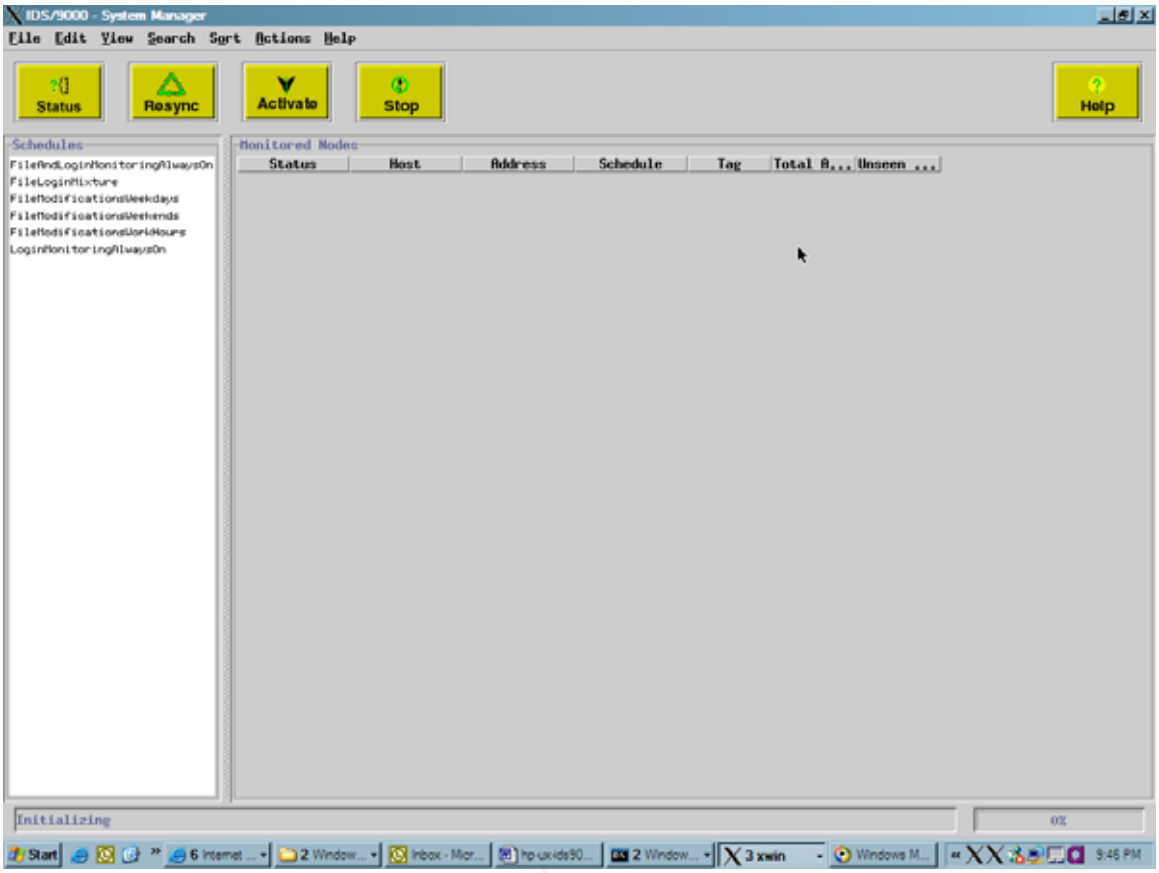

Figure 1: System Manager Screen

The System Manager needs to be notified that there exists on the network an Agent who has the IDS/9000 Agent software and certificate installed. To add an Agent to the IDS/9000 System Manager, follow the following steps on the Administration Server:

 # su – ids #Must be user ids
$ /opt/ids/bin/idsgui #Launches System Manager
1. In the "System Manager" Window:
Edit → Host Manager… (Figure 2)
2. In the "IDS/9000 Host Manager" Window:
Click on the "Add" button (Figure 3)
3. In the "Add Host" Window:
Enter the Hostname, IP Address, and an
Optional Tag (nickname) for the Agent (Figure 4)
4. In the "IDS/9000 Host Manager" Window:
Enable the Agent (host) for monitoring by
Checking the box under "Monitored"
5. In the "IDS/9000 Host Manager" Window:
File → Save (Figure 5)
(Repeat Steps 2-5 for additional Agents)

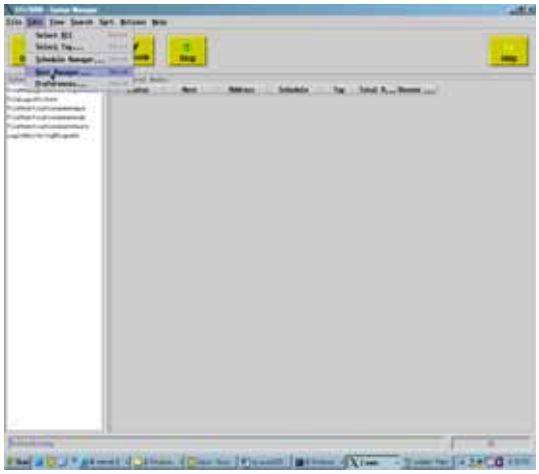

Figure 2: System Manager

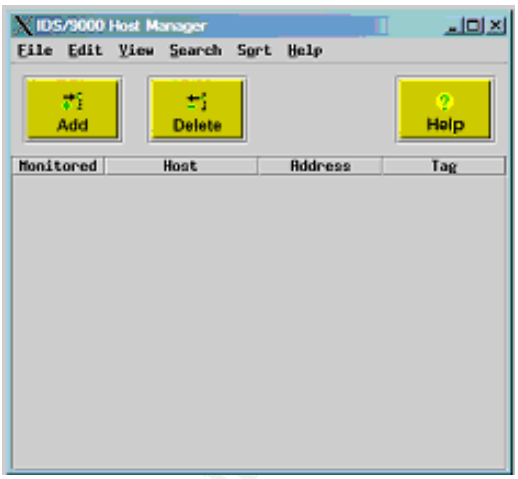

Figure 3: Host Manager

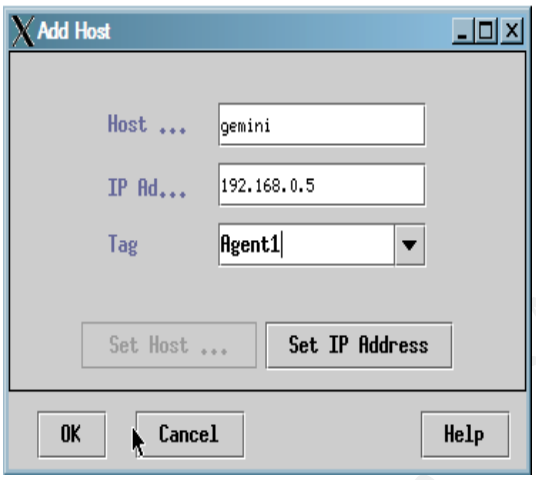

Figure 4: Add Host

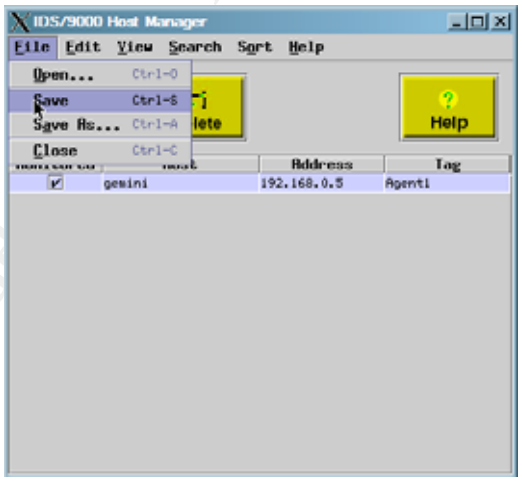

Figure 5: Save Host Changes

You can also add all the hosts listed in the /etc/hosts file on the Administration Server all at once by performing the following:

In the "IDS/9000 Host Manager" Window:
Edit → Add Host → Load /etc/hosts File

This can save a considerable amount of time if your network is large. It is also quicker to delete Agent entries in the "Host Manager" window than it is to create them. So you can add all the hosts listed in /etc/hosts, then delete the select few that you do not want to monitor.

3.4.3 Create Surveillance Schedules

A surveillance schedule is a function inside the IDS/9000 software that allows you to create schedules to monitor the Agents for different types of attacks. A surveillance schedule is at the heart of the IDS/9000 System. Its purpose is to

monitor activities on the Agents according to certain rules and criteria's. These rules are defined in what are called Templates. However, let's back up a little to explain the structure.

Inside a surveillance schedule is one or more surveillance groups. A surveillance group is a grouping of templates. Templates are the actual intrusion possibilities that the IDS/9000 looks out for. Each template consists of properties that may have values. These properties are what define the specific criteria that the IDS/9000 system goes by to determine if it should send an alert. Below is a table that defines the templates provided by IDS/9000 and their corresponding properties and default values:

Table 2: Templates, Properties, & Default Values

Templates	Properties	Default Values
Monitor Start of Interactive Sessions	Notify when these users begin a session	root, ids, www, news, daemon, bin, sys, adm, uucp, lp, nuucp, hpdb
Monitor Logins/Logouts	Ignore these users	-
Changes to Log Files	Files which should only be appended to	/var/adm/btmp, /var/adm/wtmp, /etc/btmp, /etc/wtmp, /var/adm/messages, /var/adm/syslog/mail.log, /var/adm/syslog/syslog.log, /var/adm/pacct, /var/adm/sulog
Modification of Files/Directories	Watch these files for modifications/creation	/stand/vmunix, /stand/kernrel, /stand/bootconf, /etc/passwd, /etc/group, /.rhosts, /.shosts, /etc/hosts.equiv, /etc/hosts.allow, /etc/hosts.deny, /etc/inetd.conf
	Ignore these files	/etc/ptmp, /etc/.pwd.lock, /etc/utmp, /etc/utmpx, /etc/rc.log, /etc/lvconf/lvm_lock
	Watch these directories for modification	/etc, /bin, /sbin, /stand, /lib, /usr/bin, /opt
	Ignore these directories	-
Creation of setUID Files	List of critical user IDs to be monitored	0, 1, 2, 3, 4, 5, 9, 11
Creation of World-Writable Files	List of critical user IDs to be monitored	0, 1, 2, 3, 4, 5, 9, 11
	Ignore these values	-
	Ignore these directories	-
Repeated Failed Logins	Number of failures to exceed	2
	Time span to detect failures over	10
	Suppression period for reporting	30
Repeated Failed su Command	Number of failures to trigger on	2
	Time span to detect failures over	24

Modification of Another User's Files	Ignore changes to these files	/dev/null, /etc/rc.log, /etc/lvmconf/lvm_lock
	Ignore changes to these directories	/var/opt/OV/tmp/OpC
	List of user IDs to be ignored	-
Race Condition Attacks	What user IDs to monitor for being attacked	0, 1, 2, 3, 4, 5, 9, 11
	How many paths to keep track of per process (0 is all)	10
Buffer Overflow Attacks	What user IDs to monitor for being attacked	0, 1, 2, 3, 4, 5, 9, 11

Below you can see a screen of the Schedule Manager. To open the Schedule Manager, in the "System Manager" window, go to "Edit" → "Schedule Manager…"

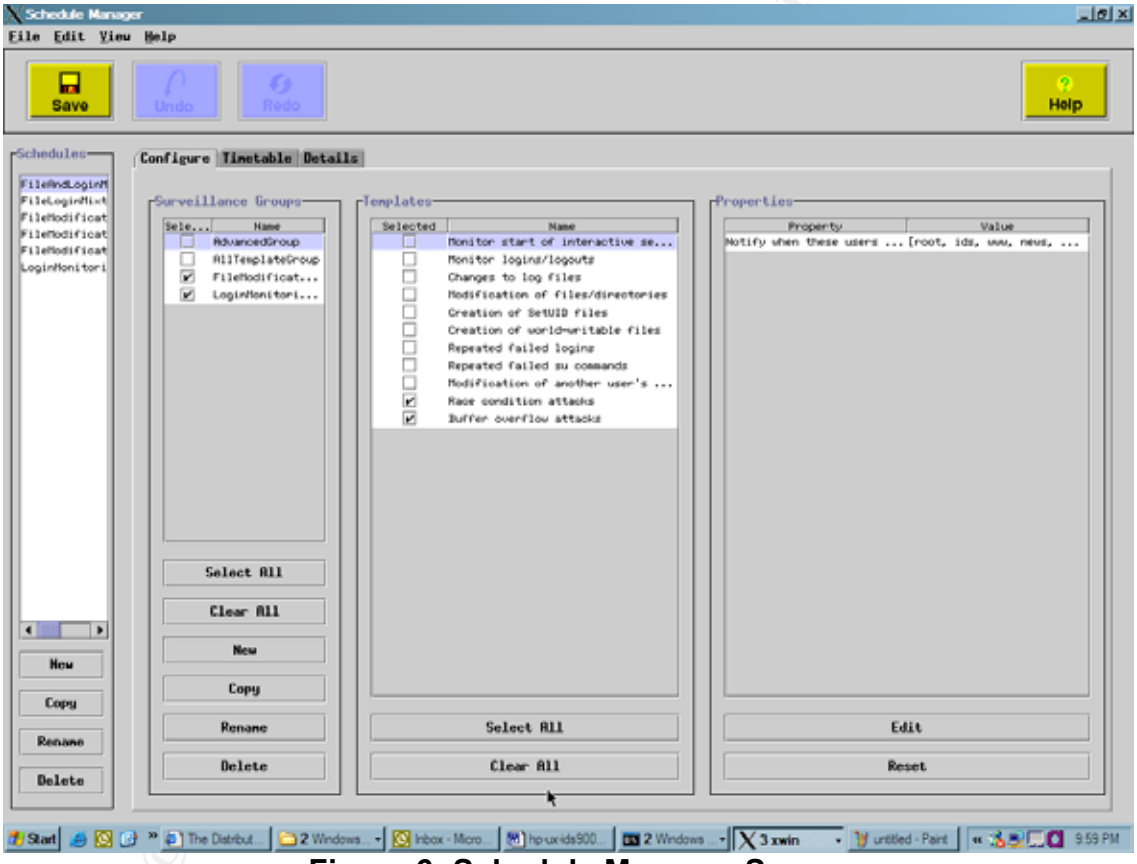

Figure 6: Schedule Manager Screen

As you can see on the left side of this screen shot, IDS/9000 comes with default schedules that you can use. In any case, all the schedules that are created will be listed in the "Schedules" box. From the Schedules box, you can create a new schedule, copy an existing schedule, rename an old schedule, or completely delete a schedule.

To the right of the schedule names are the settings associated with each schedule. There are three tabs that are used to configure the highlighted schedule. The first tab, configure, is used to configure the surveillance groups in that schedule. The second tab, timetable, is used to configure the day and time the schedule runs. The third tab, details, is used to view the details of the schedule. Each schedule can have one or more surveillance groups that it uses. You can think of a surveillance group as a collection of procedures that define what intrusions to detect. These procedures in IDS/9000 are called templates, and they define what types of intrusions to look out for. Each template has one or more properties associated with it, and these properties carry some type of value. The administrator can choose which templates they want included in the surveillance group, in order to detect certain types of suspicious behavior. For example, a server that allows Telnet access to it for a certain purpose can have a surveillance group associated with it, which monitors logins and logouts, and for failed repeated logins. Monitoring the logins is implemented through the template. You can then set the specific characteristic for this template by defining how many failed logins should occur before an alert is sent, or which users you do not need to monitor for logins, etc. These are what are called the "properties" of the template.

To create a schedule, there must be a certain strategy or purpose in mind. It would not be wise to create a schedule that includes all templates and checks for all possible values for each property; this would consume all the resources of all the Agents and the entire network. Each schedule has to serve a specific purpose. Depending on the Agent we are monitoring, the templates and the values for the properties will be different. However for the sake of not spending too much time on this extensive subject of how to properly develop a schedule for each type of Agent, since it is beyond the focus of this paper, I will create a general-purpose schedule that can serve the purpose for many types of Agents. From this schedule we will develop the Responsive Unix script(s). However, the Responsive Unix script(s) can be used for any schedules you decide to develop, since they are dependent on the templates and not the schedules.

Let us now create a new general-purpose schedule and configure it:

1. In the "Schedule Manager" Window:
Click on the "New" button under "Schedules"
2. In the "New Surveillance Schedule" Window:
Enter the name of the Schedule (GeneralPurpose) and click OK (Figure 7)

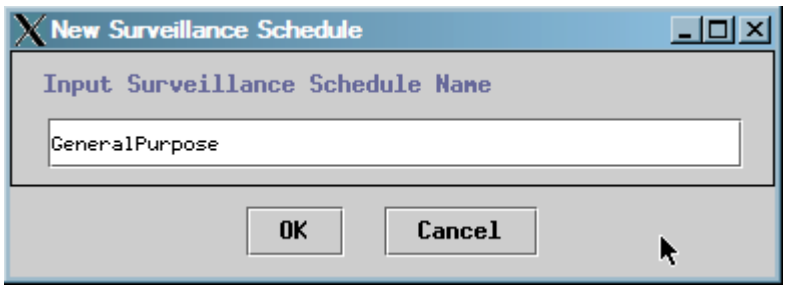

Figure 7: New Surveillance Schedule

After creating the GeneralPurpose schedule, it will appear under the list of "Schedules" in the "Schedule Manager" Window. We will then want to associate surveillance groups with this schedule. By default, there are four groups we can choose from or we can add our own groups customized to what we are trying to do. So let us add a new group:

3. In the "Schedule Manager" Window:
Click on the "New" button under "Surveillance Groups"
4. In the "New Surveillance Group" Window:
Enter the name of the Group (BasicGroup) and click OK (Figure 8)

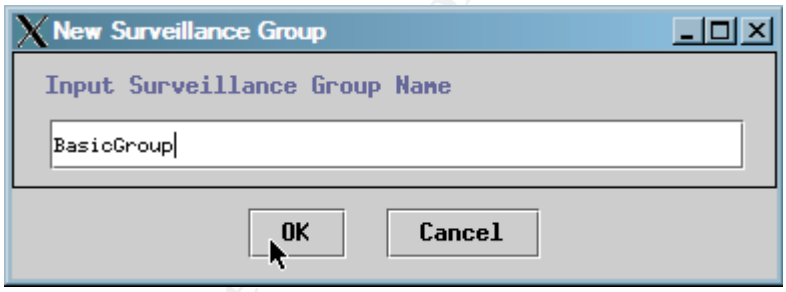

Figure 8: New Surveillance Group

At this point, we are ready to define the characteristics of our intrusion detection system. We need to select which templates we want the system to use to protect our Agents. We can use Table 2 to view the options we have. For the purpose of our example, I have analyzed the options and chosen only certain templates and modified some of the default values to look for only certain types of patterns. Table 3 is the table of the templates we want to implement with their modified property values. To select these templates, check the box in the "Schedule Manager" window next to the desired template. To see the reasons why I chose the specific property values defined in Table 3, please refer to Appendix B. Below is a sample of how to modify a value for a property. The same procedure can be used to change the values of the other properties.

5. In the "Schedule Manager" Window:
Highlight the property of interest under "Properties" and click "Edit"
6. In the "Edit List" Window:
Add, Edit or Delete the values you want (Figure 9)

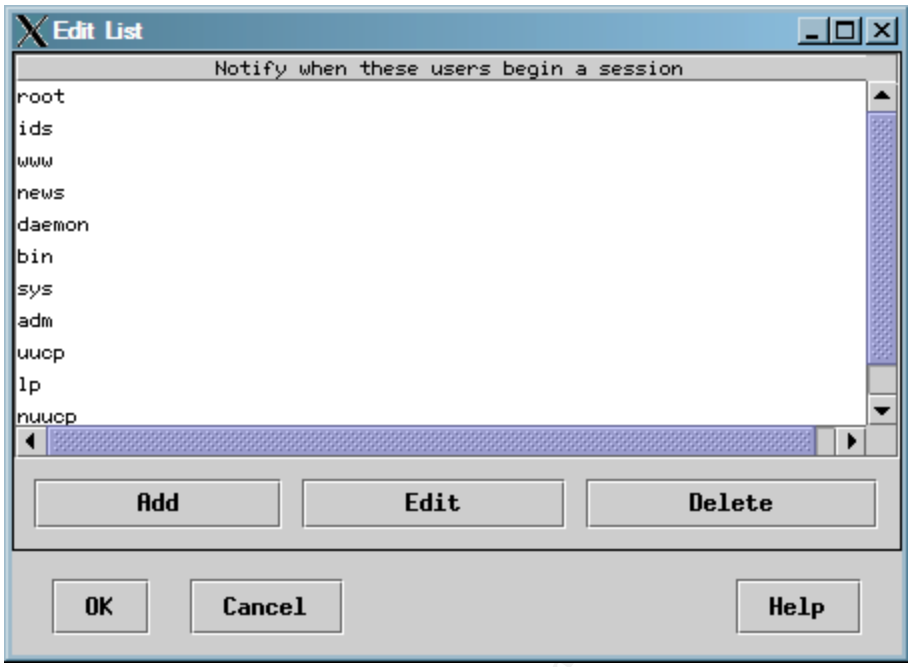

Figure 9: Edit Property Values

Table 3: Customized Templates for BasicGroup

Templates	Properties	Modified Values
Monitor Start of Interactive Sessions	Notify when these users begin a session	root, ids
Monitor Logins/Logouts	Ignore these users	-
Changes to Log Files	Files which should only be appended to	/var/adm/syslog/syslog.log, /var/opt/ids/alert.log
Modification of Files/Directories	Watch these files for modifications/creation	/stand/vmunix, /stand/kernrel, /stand/bootconf, /etc/passwd, /etc/group, /.rhosts, /.shosts, /etc/hosts.equiv, /etc/hosts.allow, /etc/hosts.deny, /etc/inetd.conf
	Ignore these files	/etc/ptmp, /etc/.pwd.lock, /etc/utmp, /etc/utmpx, /etc/rc.log, /etc/lvconf/lvm_lock
	Watch these directories for modification	/stand
	Ignore these directories	-
Creation of setUID Files	List of critical user IDs to be monitored	0, 102
Repeated Failed Logins	Number of failures to exceed	2
	Time span to detect failures over	10
	Suppression period for reporting	30
Repeated Failed su Command	Number of failures to trigger on	2
	Time span to detect failures over	10
Race Condition Attacks	What user IDs to monitor for	0, 102

	being attacked	
	How many paths to keep track of per process (0 is all)	10
Buffer Overflow Attacks	What user IDs to monitor for being attacked	0, 102

The two templates I chose not to implement are “Creation of world-writable files” and “Modifications of another user’s files”. These two templates are not necessary for our group, BasicGroup. While they are dangerous, they do not pose as significant of a threat as some of the other templates, and I have therefore chosen not to include them at this point. After the surveillance schedules, groups, and their associated templates are configured, we can now proceed to configure the timetables that these surveillance schedules will run on. Timetables define the times and days that the groups will activate on the Agents. The following are the step-by-step procedures for configuring the timetables:

7. In the “Schedule Manager” Window:
Click on the “Timetable” tab
8. Under the “Schedules” Window:
Select the schedule that you wish to configure its timetable
9. Under the “Selected Groups” Window:
Select the group “BasicGroup”
10. Under “Criteria” select “Always On”
11. Finally, click the “Save” button to save configuration (Figure 10)

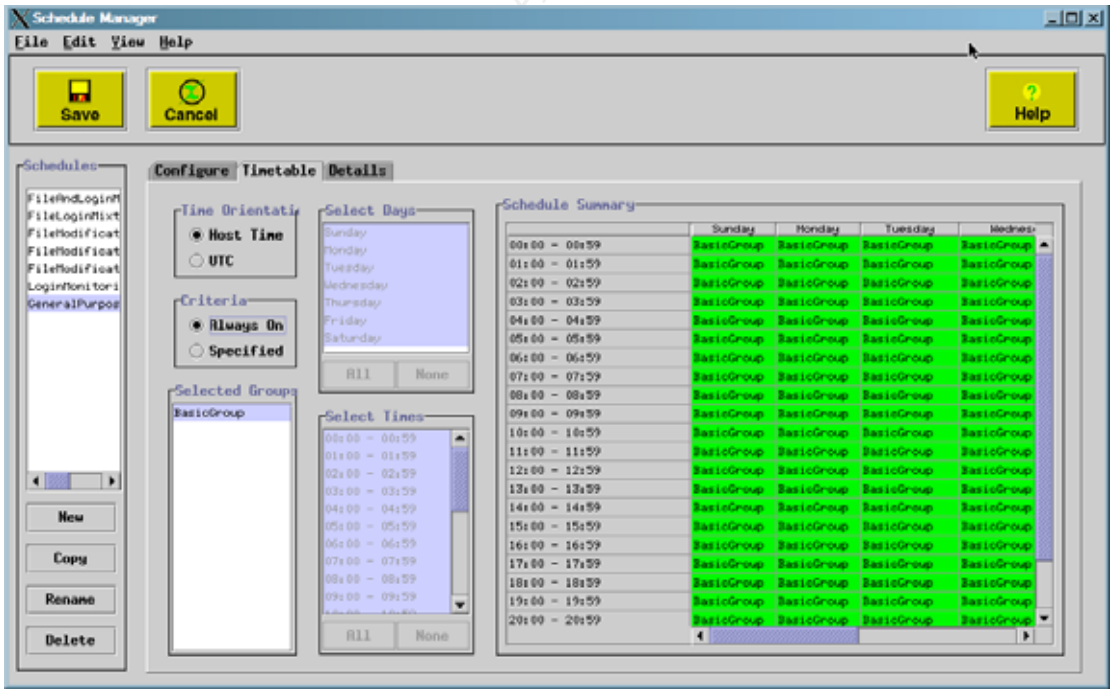

Figure 10: Configure Timetables

Since we are developing a general-purpose schedule that includes a basic group, I decided to keep the group “Always on”, which means that it is running

constantly. If the surveillance group was more comprehensive, this would not be a wise selection. Keeping the surveillance running constantly hogs the system and the network, and not to mention the amount of alerts that you would be receiving. In our case, since the surveillance is basic, we can afford to have it running constantly. If a surveillance group was developed that was more comprehensive, you can specify certain times in the day and certain days in the week that the group can become active. You would just select “Specified” under the “Criteria” window and then under the “Select Days” and “Select Times” windows you can specify the days and times that the group is activated. Finally, once the timings of when the groups will run is completed, a visual representation of the timetable is shown to the right under “Schedule Summary”. This allows you to quickly and easily determine when a certain group will be active.

The locations for the configuration files for the schedules, groups, and templates are located under /var/opt/ids/gui. Inside this directory is a directory for each of the three. The schedules are in the directory called “SurveillanceSchedules”. The groups are in the directory called “SurveillanceGroups”. And the templates are in a directory called “Templates”. The configuration files are not in a readable format, however, it is good to know their locations so that you can make sure that they are backed up regularly.

The hosts are now configured and the certificates have been generated and distributed. The schedules, groups, templates, and timetables were customized to the needs of our system. However, we still have not defined which schedule will run for which host. Therefore, our final task to fully complete the configuration of the IDS/9000 system is to activate the schedules to their appropriate hosts. To do this, we do the following:

12. In the “System Manager” Window:
Highlight the desired schedule you wish to activate
13. In the “System Manager” Window:
Highlight the desired host under “Monitored Nodes”
14. Click on the “Actions” pull-down menu:
Select “Activate Schedule” from the menu
15. You may need to click on the “Status” button to update (Figure 11)

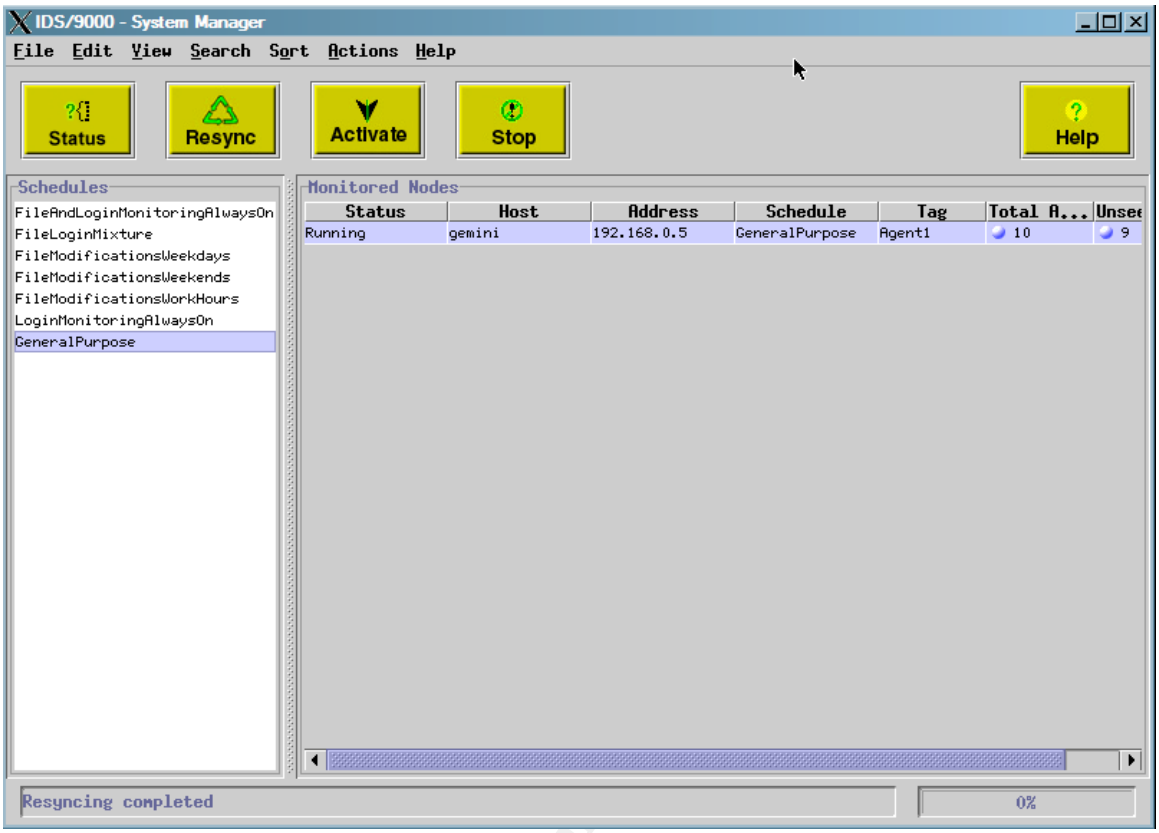

Figure 11: Final Configuration

You can see the final configuration of the Agent, gemini, and its appropriate schedule above. It is important to make sure that the “Status” is always “Running”. You can also see which “Schedule” is used to monitor that Agent (GeneralPurpose). It also displays the “Total Alerts” as well as the “Unseen Alerts”. To obtain more information on the alerts, double-click on the listed Agent, which will bring up the “Network Node” window. This will be discussed in more detail in section 5.0, Monitoring & Maintaining Systems. The IDS/9000 system is now fully functioning and properly running. We will now proceed to implement the “responsive” functionality in IDS.

3.5 Responsive Functionality

We are now in a position to investigate and implement the focus of this paper. As I had mentioned previously, the focus of my research is on a concept that I call Responsive Intrusion Detection System (RIDS). The concept of RIDS is to configure IDS to respond intelligently to certain types of attacks. This will further strengthen the functionality of IDS. It will allow for a faster response time for handling intrusions, considering that the System Administrator may not always be available to respond. It will also allow the System/Security Administrator to focus on some other angles of the intrusion, such as determining the source and protecting from future attacks. The responsive features of this research are housekeeping tasks that any administrator would have to carry out if an intrusion is detected, but it can also serve to prevent further attacks as well as providing audit information. **An important note to mention: this is a simplistic model of the RIDS concept.** This only addresses the needs of a sample of all the possible attacks. My research is meant to develop a concept and not an all-inclusive product. The research does allow for further growth and can be built on in the future.

The strength of RIDS is in developing custom Unix script(s) to respond to certain scenarios. These scripts must be kept in the directory /opt/ids/response in order for IDS/9000 to recognize them. The scripts respond according to alerts activated by the IDS/9000 system. Depending on the alert message fields/variables, the script can respond appropriately. Each alert generated sends the information listed in Table 4 in its message, and this information can be captured to decide on what the scripts need to do.

Table 4: Alert Message Content

Variable Name	Description
Code	Code for template. Three digits, with leading zeros
Version	Version # of template
Severity	Severity level of alert (1 – most serious, 3 – least serious)
UTC Time	Time of alert
Attacker	Source of attack
Target ID	System attacked
Attack Type	Type of attack
Details	Full alert information

Typically, a System Administrator has many methods of staying in touch with the system. The System Administrator is not constantly sitting and watching the IDS/9000 System Manager while at work. Therefore if an alert is generated, and it has a medium criticality level, the best way to notify the System Administrator is through email. The email can also be forwarded to his/her pager device to notify him/her when they are not in front of their email. I have devised a numbering scheme that will define the seriousness of the event based on the severity level produced by IDS/9000 and the nature of the attack. Critical level 1 is the highest

level of an alert, while 3 is the lowest level. Table 5 lists the level of attack and the response needed for such an attack. Again, this is not to be confused with IDS/9000's Severity Level numbering system. This new system determines its level of severity based on IDS/9000's Severity Level as well as the type of attack, and then decides what the best response should be.

Table 5: Critical Levels and Responses

Level	Response
1	Alert/Email/Respond
2	Alert/Email
3	Alert

If an alert is generated by IDS/9000 that is not in any way critical and is simply a notification of someone logging on to the system, then this will carry a level 3 response. Level 3 responses merely send an alert to the IDS/9000 System Manager. If the event is more serious, such as *root* or *ids* logging on to the system, then this will raise the alert to level 2. Level 2 involves sending an alert to the IDS/9000 System Manager as well as emailing (and maybe paging) the System Administrator. At that point, the System Administrator can decide if further action needs to take place. Finally, if the event is severe, and has the potential of causing serious damage, this will take the alert to level 1. Level 1 involves everything that level 2 involves, but goes further to respond in a manner that will reduce the damage or completely eliminate it, as well as providing audit-related information.

Table 6 lists the responses that the RIDS system will take in each sample attack listed in our group, BasicGroup. Some do not require any responses, and an email to the System Administrator is sufficient.

Table 6: Responses to Detected Intrusions

Intrusion	Templates	Code	RIDS Level	Response
	Monitor Start of Interactive Sessions	030	2	
	Monitor Logins/Logouts	031	3	
	Changes to Log Files	028	2	Preserve copy of log file
	Modification of Files/Directories	027	1	Preserve copy of files
	Creation of setUID Files	009	2	
	Repeated Failed Logins	016	1	Lock account
	Repeated Failed su Command	015	1	Lock account
	Race Condition Attacks	006	1	Kill program
	Buffer Overflow Attacks	005	1	Kill program

I have chosen to develop the RIDS script in the Unix shell programming language. I have packaged all the responses into one script, and based on the arguments it receives from the alert (Table 5), it will determine the nature of the attack and the appropriate response. You can also create different scripts for each attack type, but I decided to package it all in one to prevent overcomplicating the solution. The script is called “response.sh”, and you will find it in Appendix A. Download (or copy and paste) the script into each Agent system under /opt/ids/response. The response scripts run off the Agent systems, not off the Administration system.

I chose the responses listed in Table 6 based on the nature of the attack. Let us now look into why RIDS should respond in the following manner for each type of attack. Please review the code in Appendix A to see the implementation of the below descriptions. Most of the responses taken involve manipulating the “Details” field sent by the alert. What I do is grab the information I need from that field, with some Unix commands, then I perform the response on the data I grab.

Interactive sessions are important events to monitor because many attacks originate from interactive sessions. Examples of interactive sessions are ftp, remote logins (such as rlogin or telnet), and the su command. Interactive sessions are a little more serious than monitoring for logins/logouts because interactive sessions can be the source of many attacks. For interactive sessions, we will only look for *root* or *ids* users, and send an email to the System Administrator. An email is sufficient enough considering that it is common practice for people to telnet or su into an account. Again, this would probably depend on the type of server. If the server does not allow any interactive sessions to take place, then you might want to modify “response.sh” to respond more harshly by disabling networking and locking the account.

When a log file is changed (appending to it is ok) it means that someone is trying to cover his or her trail. The attacker is trying to erase the audit trail that can lead to them. This is a serious attack because it prevents us from knowing the source of the attack. The two log files that we set in IDS/9000 to monitor for, are the /var/adm/syslog/syslog.log and the /var/opt/ids/alert.log log files. The first log is the general system log that monitors for system activity. This is critical to preserve since it will provide a lot of valuable audit information. The second is more critical because it contains the log files for the IDS/9000 alerts. If an attacker is able to modify these files, then the IDS/9000 system may never generate alerts or may send us incorrect information. The response we want to take here is to preserve these log files in the event that they are edited. We preserve them by emailing them to the System Administrator. The problem with this however is that the response is not real-time, in the sense that the response will occur only after that change is done. This does not help us much, because we then receive a file through email after it has been altered. This response will only serve a useful purpose if the attacker saves the file in the middle of editing it.

After it is saved, it will then get emailed to the System Administrator. So as you can see, this may not be the best approach, but it is the best type of response we can get with the resources we have. Another thought is to preserve these files whenever *root* or *ids* log in, and not necessarily after the log file is opened and edited. This will work since it is only those two users that can edit their respective log files. This approach would work, but would carry some extra overhead on the system, considering that it is more common for a normal *root* or *ids* user to log in than it is for an attacker to log in. So most of the time the system will be executing commands that do not serve an immediate purpose.

Modification or corruption of certain files or directories can cause a serious system failure, especially when it involves the kernel or bootconf files. In this case, our response will be to once again attempt to preserve these files. The approach for that will be to relocate the files to another directory under a different name and convert them to hidden files (or "." files). This will attempt to hide the files under a different name and at a different location so that the Administrator can go back later and retrieve them, in case the originals are corrupted from the hacker. However, with a little more patience, the hacker can still find the copied files, but it will hopefully buy the Administrator enough time to preserve the copies and disable the hacker. Another option is to perform an "rcp", and copy it off the Agent to the Administration server for example, but in that case, you would have to enable "login" in /etc/inetd.conf to allow for "r" command execution. For purposes of simplicity, we will just copy it to another location on the Agent. The new location the hidden file will be saved to is /opt/ids/lib.

Repeated failed logins and repeated failed su attempts indicates to us that someone is trying a brute-force attack to attempt to crack the password for a particular user. In a Trusted System, an account will automatically lock after a certain number of failed attempts. However, if it is a Trusted System, let us still empower RIDS, through "response.sh", to lock the user account. In the case that the system is not a Trusted System, a user account can still be locked. There is a difference between the response I chose between failed login attempts and failed su attempts. If an attack is a "failed login attempt", then the response will be to lock the account the person is trying to log into. This makes sense since most likely the hacker is logging in from an external server, and the only user we can control is the user on the local machine, and therefore we can lock that user so that the hacker cannot try to attack it. If an attack is a "failed su attempt", I chose to lock the account that the "su" command originated from. This is possible, since "su" is a command run on the local machine, by a local user. So we have control over local users and can therefore lock the account of this particular malicious user. This also includes emailing the System Administrator to notify him/her that this has occurred so they can investigate and proceed to unlock any accounts if all is secured.

Race condition attacks and buffer overflow attacks are quite dangerous. At the end, they can leave the hacker with a *root* shell to do with it what they please.

- A race condition attack is when an attacker uses the idle time between two commands or operations in a program to input his/her set of commands. If the program is running as *root*, well then, the attacker now has a *root* shell.
- A buffer overflow attack is when an attacker floods a program buffer with too much data, causing the program to execute commands at a different location in memory. This new location is where the attacker places his/her code to exploit the system. The way buffer overflow attacks actually work is that initially there is a non-malicious program running. Malicious code is inserted so a new malicious program is executed when the buffer overflow occurs, and gives the user privileged access.

These are both described in great detail in Hal Pomeranz's *Common Issues and Vulnerabilities in Unix Security*. Now that we understand race condition attacks and buffer overflow attacks, we can talk about how to prevent them through the RIDS response program.

Let us first discuss the response for each of the two attacks.

- For the buffer overflow attack what we ultimately want to do is to kill the malicious program that is currently running, since that is the primary source of damage at this point. We also want to remove the program that the buffer overflow originated from, to prevent further attacks. "response.sh" grabs the currently running malicious process id from the "Details" field from the alert and kills that process. "response.sh" then grabs the program name from the "Details" field for the initial program that contains the buffer overflow problem and emails it (to preserve it so the Administrator can see where the error originated) and then removes it to prevent further attacks.
- For the race condition attack, what we would like to do is to remove the attacking program and deactivate the user who generated this attack. "response.sh" will capture the user ID of the attack, and then based on whether it is a Trusted System or not, we will perform the proper command to lock the account. Next we will email a copy of the attacking program to the Administrator and then remove it to prevent further attacks.

4.0 Testing Configuration

Testing the configuration is one of the most important steps in ensuring that the RIDS system is up and running securely. Testing however is sometimes not easy, because you need to simulate the different types of possible intrusions. We can divide the testing into two main categories: system security testing and RIDS testing. System security testing involves testing the Unix system for general security concerns. It basically ensures that what we discussed in the section entitled “Hardening HP-UX System” is working properly. RIDS testing involves testing the functionality of RIDS to ensure that it is functioning as intended.

4.1 Unix System Security Testing

Testing the security of the Unix system is just as important as testing the RIDS functionality. What good is it if you have a well-toned RIDS system, but your Unix system has fundamental security flaws that can be exploited by the most trivial tools used by a “script kiddie hacker”? The first check we want to perform is to run the secure.sh script located in Appendix A against the Administration system and all the Agents. This will perform a general security check on the system. The output will give us a lot of valuable information that we can use to analyze our systems. I would recommend that you run this at least once a month to give you an idea of changes that may have occurred on the systems. For example, the script checks for suid/sgid scripts. If any new suid/sgid scripts have been added since the last time it ran, then it is something you will need to look into. Below is the output from running “secure.sh” on the Administration system. As you can see the system is Trusted, and if it is not, it will prompt you to convert the system into a Trusted System. For more advanced security features, ensure that the system is always Trusted. Root access is also restricted to the console, which is another important security feature. It then notifies you of the users and groups setup on the system. This allows you to keep track of any expired or suspicious users/groups. After that, you will see a listing of all the suid/sgid programs. It is imperative that you keep track of these programs and inspect any new ones. A quick method for keeping track of all these is to store this list in a file every month, then execute the “diff” command on the previous month’s file and the current month’s file. Any new suid/sgid programs will be displayed on the terminal. This will help protect your systems from a hacker who, for example, has placed a malicious program that uses the suid/sgid functionality to gain access to a root shell.
“secure_results” file (output of “secure.sh”):

This file contains the results of hardening the system:

Checking to see if system is trusted...
System is already trusted!

Root access is now restricted to the console

Below is a list of users configured on the system:

root
daemon
bin
sys
adm
uucp
lp
nuucp
hpdb
www
webadmin
smbnull
ids
hmahmoud

Below is a list of groups configured on the system:

root
other
bin
sys
adm
daemon
mail
lp
tty
nuucp
users
nogroup
smbnull
ids

Below is a list of suid/sgid programs on the system:

```
-r-sr-xr-x   1 root     bin        39448 Nov 14  2000 /opt/dce/bin/ep_scavenger
-r-sr-xr-x   1 root     bin        16384 Oct  2  2000 /opt/webadmin/parmgr/startParMgr.cgi
-r-sr-xr-x   1 root     bin        53248 Jun 27  2000 /opt/graphics/common/lbin/gwind
-r-sr-xr-t   5 root     bin      4407296 Jun 27  2000 /opt/graphics/common/lbin/sb_daemon_11.0
-r-sr-xr-t   5 root     bin      4407296 Jun 27  2000 /opt/graphics/common/lbin/sb_daemon_8.02
-r-sr-xr-t   5 root     bin      4407296 Jun 27  2000 /opt/graphics/common/lbin/sb_daemon_8.05
-r-sr-xr-t   5 root     bin      4407296 Jun 27  2000 /opt/graphics/common/lbin/sb_daemon_8.07
-r-sr-xr-t   5 root     bin      4407296 Jun 27  2000 /opt/graphics/common/lbin/sb_daemon_9.0
-r-sr-xr-x   1 daemon   bin        45056 Jun 27  2000 /opt/graphics/common/lbin/timd
-r-sr-xr-t   1 root     bin        40960 Jun 27  2000 /opt/graphics/phigs/bin/cgmiui
-r-sr-xr-t   1 root     bin        65536 Jun 27  2000 /opt/graphics/phigs/lbin/phg_daemon
-rwsr-xr-x   1 root     users      57536 Sep 13  2000 /opt/cifsclient/bin/cifslist
-rwsr-xr-x   1 root     users      57529 Sep 13  2000 /opt/cifsclient/bin/cifslogin
-rwsr-xr-x   1 root     users      53434 Sep 13  2000 /opt/cifsclient/bin/cifslogout
-rws--x--x   1 root     bin        77824 Sep 18 04:57 /opt/ssh/PA-RISC1.1/libexec/ssh-chauthtok-helper
```

```
-rws--x--x  1 root     bin      1093632 Sep 18 04:57 /opt/ssh/PA-RISC1.1/libexec/ssh-keysign
-r-sr-x---  1 root     ids       266240 Mar 22  2002 /opt/ids/lbin/idssysdsp
-r-sr-x---  1 root     ids       274432 Mar 22  2002 /opt/ids/lbin/updaterc
-r-sr-xr-x  1 root     bin        32768 Nov  9  2000 /usr/bin/mediainit
-r-sr-xr-x  1 root     bin        20480 Nov 14  2000 /usr/bin/bdf
-r-sr-xr-x  1 root     bin        45056 Nov 14  2000 /usr/bin/rcp
-r-sr-xr-x  1 root     bin        28672 Nov 14  2000 /usr/bin/remsh
-r-sr-xr-x  1 root     bin        45056 Nov 14  2000 /usr/bin/at
-r-sr-xr-x  1 root     bin        24576 Nov 14  2000 /usr/bin/crontab
-r-sr-sr-x  2 root     mail       45056 Nov 14  2000 /usr/bin/mail
-r-sr-sr-x  2 root     mail       45056 Nov 14  2000 /usr/bin/rmail
-r-xr-sr-x  1 bin      sys        24576 Nov 14  2000 /usr/bin/ipcs
-r-sr-xr-x  1 root     bin        16384 Nov 14  2000 /usr/bin/newgrp
-r-xr-sr-x  1 bin      sys        28672 Nov 14  2000 /usr/bin/top
-r-xr-sr-x  2 bin      sys        16384 Nov 14  2000 /usr/bin/uptime
-r-xr-sr-x  2 bin      sys        16384 Nov 14  2000 /usr/bin/w
-r-sr-xr-x  1 root     bin        24576 Nov 14  2000 /usr/bin/nfsstat
-r-xr-sr-x  1 bin      sys       118784 Nov 14  2000 /usr/bin/strdb
-r-xr-sr-x  1 bin      sys        16384 Nov  9  2000 /usr/bin/iostat
-r-xr-sr-x  1 bin      sys        98304 Oct 23  2002 /usr/bin/netstat
-r-xr-sr-x  1 bin      sys        24576 Nov  9  2000 /usr/bin/vmstat
-r-sr-xr-x  5 root     bin        45056 Nov 14  2000 /usr/bin/chfn
-r-sr-xr-x  1 root     bin        73728 Nov 14  2000 /usr/bin/df
-r-sr-xr-x  1 root     bin        53248 Nov 14  2000 /usr/bin/login
-r-sr-xr-x  1 root     bin        24576 Nov 14  2000 /usr/bin/su
-r-sr-xr-x  5 root     bin        45056 Nov 14  2000 /usr/bin/chsh
-r-sr-xr-x  5 root     bin        45056 Nov 14  2000 /usr/bin/nispasswd
-r-sr-xr-x  5 root     bin        45056 Nov 14  2000 /usr/bin/passwd
-r-sr-xr-x  5 root     bin        45056 Nov 14  2000 /usr/bin/yppasswd
-r-sr-xr-x  1 root     bin        36864 Nov 14  2000 /usr/bin/chkey
-r-sr-xr-x  1 root     bin       344064 Nov 14  2000 /usr/bin/pppd
-r-sr-xr-x  1 root     bin        69632 Nov 14  2000 /usr/bin/rdist
-r-sr-xr-x  1 root     bin        20480 Nov 14  2000 /usr/bin/rexec
-r-sr-xr-x  1 root     bin        36864 Nov 14  2000 /usr/bin/rlogin
-r-xr-sr-x  1 bin      bin       163840 Dec 23  2002 /usr/bin/X11/xfs
-r-sr-sr-x  1 root     sys       299008 Nov 14  2000 /usr/bin/X11/hpterm
-r-sr-xr-x  1 root     bin       294912 Nov 14  2000 /usr/bin/X11/xterm
-r-xr-sr-x  1 bin      mail      507904 Nov 14  2000 /usr/bin/elm
-r-sr-xr-x  1 root     bin        40960 Nov 14  2000 /usr/bin/lp
-r-sr-xr-x  1 lp       bin        36864 Nov 14  2000 /usr/bin/cancel
-r-sr-xr-x  1 lp       bin        24576 Nov 14  2000 /usr/bin/disable
-r-sr-xr-x  1 lp       bin        20480 Nov 14  2000 /usr/bin/enable
-r-sr-xr-x  1 root     bin        36864 Nov 14  2000 /usr/bin/lpalt
-r-sr-xr-x  1 lp       bin        45056 Nov 14  2000 /usr/bin/lpstat
-r-sr-xr-x  1 lp       bin        16384 Nov 14  2000 /usr/bin/slp
-r-sr-xr-x  1 root     bin        45056 Nov 14  2000 /usr/bin/ct
-r-sr-xr-x  1 root     bin        36864 Nov 14  2000 /usr/bin/cu
-r-sr-sr-x  1 bin      daemon    1699840 Nov 14  2000 /usr/bin/kermit
-r-sr-xr-x  1 uucp     bin         57344 Nov 14  2000 /usr/bin/uucp
-r-sr-xr-x  1 uucp     bin         20480 Nov 14  2000 /usr/bin/uuls
-r-sr-xr-x  1 uucp     bin         16384 Nov 14  2000 /usr/bin/uuname
-r-sr-xr-x  1 uucp     bin         16384 Nov 14  2000 /usr/bin/uusnap
-r-sr-xr-x  1 uucp     bin         36864 Nov 14  2000 /usr/bin/uustat
-r-sr-xr-x  1 uucp     bin         53248 Nov 14  2000 /usr/bin/uux
```

```
-r-xr-sr-x  1 bin      bin      245760 Nov 14  2000 /usr/bin/stmkfont
-r-sr-xr-x 11 root     bin     1888256 Nov 14  2000 /usr/sbin/swinstall
-r-sr-xr-x  2 root     bin     1011712 Nov 14  2000 /usr/sbin/swpackage
-r-sr-xr-x 11 root     bin     1888256 Nov 14  2000 /usr/sbin/swacl
-r-sr-xr-x 11 root     bin     1888256 Nov 14  2000 /usr/sbin/swconfig
-r-sr-xr-x 11 root     bin     1888256 Nov 14  2000 /usr/sbin/swcopy
-r-sr-xr-x 11 root     bin     1888256 Nov 14  2000 /usr/sbin/swlist
-r-sr-xr-x 11 root     bin     1888256 Nov 14  2000 /usr/sbin/swremove
-r-sr-xr-x 11 root     bin     1888256 Nov 14  2000 /usr/sbin/swverify
-r-sr-xr-x 11 root     bin     1888256 Nov 14  2000 /usr/sbin/swreg
-r-sr-xr-x  2 root     bin     1011712 Nov 14  2000 /usr/sbin/swmodify
-r-sr-xr-x  1 root     bin       61440 Oct 23  2002 /usr/sbin/arp
-r-sr-xr-x  1 root     bin       32768 Oct 23  2002 /usr/sbin/ping
-r-xr-sr-x  1 root     sys       40960 Nov 14  2000 /usr/sbin/lanscan
-r-sr-xr-x 26 root     sys      536576 Nov 14  2000 /usr/sbin/lvchange
-r-sr-xr-x 26 root     sys      536576 Nov 14  2000 /usr/sbin/lvcreate
-r-sr-xr-x 26 root     sys      536576 Nov 14  2000 /usr/sbin/lvdisplay
-r-sr-xr-x 26 root     sys      536576 Nov 14  2000 /usr/sbin/lvextend
-r-sr-xr-x 26 root     sys      536576 Nov 14  2000 /usr/sbin/lvlnboot
-r-sr-xr-x 26 root     sys      536576 Nov 14  2000 /usr/sbin/lvreduce
-r-sr-xr-x 26 root     sys      536576 Nov 14  2000 /usr/sbin/lvremove
-r-sr-xr-x 26 root     sys      536576 Nov 14  2000 /usr/sbin/lvrmboot
-r-sr-xr-x 26 root     sys      536576 Nov 14  2000 /usr/sbin/pvchange
-r-sr-xr-x 26 root     sys      536576 Nov 14  2000 /usr/sbin/pvck
-r-sr-xr-x 26 root     sys      536576 Nov 14  2000 /usr/sbin/pvcreate
-r-sr-xr-x 26 root     sys      536576 Nov 14  2000 /usr/sbin/pvdisplay
-r-sr-xr-x 26 root     sys      536576 Nov 14  2000 /usr/sbin/pvmove
-r-sr-xr-x 26 root     sys      536576 Nov 14  2000 /usr/sbin/pvremove
-r-sr-xr-x 26 root     sys      536576 Nov 14  2000 /usr/sbin/vgcfgbackup
-r-sr-xr-x 26 root     sys      536576 Nov 14  2000 /usr/sbin/vgcfgrestore
-r-sr-xr-x 26 root     sys      536576 Nov 14  2000 /usr/sbin/vgchange
-r-sr-xr-x 26 root     sys      536576 Nov 14  2000 /usr/sbin/vgchgid
-r-sr-xr-x 26 root     sys      536576 Nov 14  2000 /usr/sbin/vgcreate
-r-sr-xr-x 26 root     sys      536576 Nov 14  2000 /usr/sbin/vgdisplay
-r-sr-xr-x 26 root     sys      536576 Nov 14  2000 /usr/sbin/vgexport
-r-sr-xr-x 26 root     sys      536576 Nov 14  2000 /usr/sbin/vgextend
-r-sr-xr-x 26 root     sys      536576 Nov 14  2000 /usr/sbin/vgimport
-r-sr-xr-x 26 root     sys      536576 Nov 14  2000 /usr/sbin/vgreduce
-r-sr-xr-x 26 root     sys      536576 Nov 14  2000 /usr/sbin/vgremove
-r-sr-xr-x 26 root     sys      536576 Nov 14  2000 /usr/sbin/vgscan
-r-sr-xr-x  1 root     bin       12288 Nov 14  2000 /usr/sbin/acct/accton
-r-xr-sr-x  1 bin      tty       16384 Nov 14  2000 /usr/sbin/wall
-r-sr-xr-x  1 root     bin       12288 Nov 14  2000 /usr/sbin/keyenvoy
-r-sr-xr-x 11 root     bin     1888256 Nov 14  2000 /usr/sbin/sd
-r-sr-sr-t  1 root     mail     856064 Nov 14  2000 /usr/sbin/sendmail
-r-sr-xr-x 11 root     bin     1888256 Nov 14  2000 /usr/sbin/swask
-r-sr-xr-x 11 root     bin     1888256 Nov 14  2000 /usr/sbin/swjob
-r-sr-xr-x  1 lp       bin       16384 Nov 14  2000 /usr/sbin/accept
-r-sr-xr-x  1 root     bin       40960 Nov 14  2000 /usr/sbin/lpadmin
-r-sr-xr-x  1 lp       bin       20480 Nov 14  2000 /usr/sbin/lpfence
-r-sr-xr-x  1 lp       bin       28672 Nov 14  2000 /usr/sbin/lpmove
-r-sr-xr-x  1 root     bin       53248 Nov 14  2000 /usr/sbin/lpsched
-r-sr-xr-x  1 lp       bin       16384 Nov 14  2000 /usr/sbin/lpshut
-r-sr-xr-x  1 root     bin       32768 Nov 14  2000 /usr/sbin/rcancel
```

-r-sr-xr-x 1 lp bin 16384 Nov 14 2000 /usr/sbin/reject
-r-sr-xr-- 1 root lp 24576 Nov 14 2000 /usr/sbin/rlp
-r-sr-xr-x 1 root bin 65536 Nov 14 2000 /usr/sbin/rlpdaemon
-r-sr-xr-x 1 root bin 32768 Nov 14 2000 /usr/sbin/rlpstat
-rwxr-sr-x 1 bin sys 28296 Nov 9 2000 /usr/sbin/sysdef
-r-sr-xr-x 1 root bin 20480 Nov 14 2000 /usr/lib/lanadmin/libdsbtlan.1
-r-sr-xr-x 1 root bin 16384 Oct 4 2000 /usr/lib/lanadmin/libdsfddi4.1
-r-sr-xr-x 1 root bin 139264 Oct 2 2000 /usr/lib/lanadmin/libdsgelan.1
-r-xr-sr-x 1 bin sys 69632 Nov 14 2000 /usr/lbin/fs/vxfs/diskusg
-r-xr-sr-x 1 bin sys 24576 Nov 9 2000 /usr/lbin/fs/hfs/diskusg
-r-sr-sr-x 1 root bin 212992 Nov 14 2000 /usr/lbin/chgpt
-r-sr-xr-x 1 root bin 12288 Nov 14 2000 /usr/lbin/protect_pty
-r-xr-sr-x 1 bin mail 16384 Nov 14 2000 /usr/lbin/rmmail
-r-sr-xr-x 1 root bin 16384 Nov 14 2000 /usr/lbin/rwrite
-r-sr-xr-x 1 root bin 20480 Nov 14 2000 /usr/lbin/exrecover
-r-sr-xr-x 1 uucp bin 118784 Nov 14 2000 /usr/lbin/uucp/uucico
-r-sr-xr-x 1 uucp bin 32768 Nov 14 2000 /usr/lbin/uucp/uuclean
-r-sr-xr-x 1 uucp bin 28672 Nov 14 2000 /usr/lbin/uucp/uusched
-r-sr-xr-x 1 uucp bin 32768 Nov 14 2000 /usr/lbin/uucp/uusub
-r-sr-xr-x 1 uucp bin 57344 Nov 14 2000 /usr/lbin/uucp/uuxqt
-r-sr-xr-x 1 daemon bin 303104 Nov 14 2000 /usr/lbin/grmd
-r-sr-xr-x 1 root bin 49152 Oct 23 2002 /usr/contrib/bin/traceroute
-r-sr-xr-x 1 root bin 212992 Nov 14 2000 /usr/contrib/bin/X11/xconsole
-r-xr-sr-x 1 bin sys 229376 Nov 14 2000 /usr/contrib/bin/X11/xload
-r-sr-xr-x 1 root bin 385024 Nov 14 2000 /usr/contrib/bin/X11/xterm
-r-sr-xr-x 1 root bin 16384 Nov 14 2000 /usr/sam/lbin/rsam
-r-sr-xr-x 1 root bin 65536 Nov 14 2000 /usr/dt/bin/dtterm
-r-sr-sr-x 1 root sys 32768 Nov 14 2000 /usr/dt/bin/dtaction
-r-sr-xr-x 1 root bin 77824 Nov 14 2000 /usr/dt/bin/dtappgather
-r-xr-sr-x 1 bin mail 921600 Nov 14 2000 /usr/dt/bin/dtmail
-r-xr-sr-x 1 bin mail 258048 Nov 14 2000 /usr/dt/bin/dtmailpr
-r-sr-xr-x 1 root bin 442368 Nov 14 2000 /usr/dt/bin/dtprintinfo
-r-sr-xr-x 1 root bin 237568 Nov 14 2000 /usr/dt/bin/dtsession
-r-sr-xr-x 1 root bin 12288 Nov 14 2000 /usr/tsm/sys/tsm.root
-r-sr-xr-x 1 root bin 16384 Nov 14 2000 /usr/tsm/sys/tsm.utmp
-r-sr-xr-x 1 root bin 32768 Nov 14 2000 /var/adm/sw/save/PHNE_28089/NET-RUN/usr/contrib/bin/traceroute
-r-sr-xr-x 1 root bin 61440 Nov 14 2000 /var/adm/sw/save/PHNE_28089/NET-RUN/usr/sbin/arp
-r-sr-xr-x 1 root bin 24576 Nov 14 2000 /var/adm/sw/save/PHNE_28089/NET-RUN/usr/sbin/ping
-r-xr-sr-x 1 bin sys 98304 Nov 9 2000 /var/adm/sw/save/PHNE_28089/SYS-ADMIN/usr/bin/netstat
-r-xr-sr-x 1 bin bin 126976 Nov 14 2000 /var/adm/sw/save/PHSS_28470/X11-FONTSRV/usr/bin/X11/xfs
-r-sr-xr-x 1 root bin 49152 Oct 23 2002 /var/depot/java/PHNE_28089/NET-RUN/usr/contrib/bin/traceroute
-r-sr-xr-x 1 root bin 61440 Oct 23 2002 /var/depot/java/PHNE_28089/NET-RUN/usr/sbin/arp
-r-sr-xr-x 1 root bin 32768 Oct 23 2002 /var/depot/java/PHNE_28089/NET-RUN/usr/sbin/ping
-r-xr-sr-x 1 bin sys 98304 Oct 23 2002 /var/depot/java/PHNE_28089/SYS-ADMIN/usr/bin/netstat

-r-xr-sr-x 1 bin sys 98304 Oct 23 2002 /var/depot/java/PHNE_28089/SYS-ADMIN.2/usr/bin/netstat
-r-xr-sr-x 1 bin bin 163840 Dec 23 2002 /var/depot/java/PHSS_28470/X11-FONTSRV/usr/bin/X11/xfs
-r-sr-x--- 1 root root 138203 Mar 22 2002 /var/depot/ids_11i_admin+agent/IDS/IDS-AGT-RUN/opt/ids/lbin/idssysdsp
-r-sr-xr-x 26 root sys 811008 Nov 14 2000 /sbin/lvchange
-r-sr-xr-x 26 root sys 811008 Nov 14 2000 /sbin/lvcreate
-r-sr-xr-x 26 root sys 811008 Nov 14 2000 /sbin/lvdisplay
-r-sr-xr-x 26 root sys 811008 Nov 14 2000 /sbin/lvextend
-r-sr-xr-x 26 root sys 811008 Nov 14 2000 /sbin/lvlnboot
-r-sr-xr-x 26 root sys 811008 Nov 14 2000 /sbin/lvreduce
-r-sr-xr-x 26 root sys 811008 Nov 14 2000 /sbin/lvremove
-r-sr-xr-x 26 root sys 811008 Nov 14 2000 /sbin/lvrmboot
-r-sr-xr-x 26 root sys 811008 Nov 14 2000 /sbin/pvchange
-r-sr-xr-x 26 root sys 811008 Nov 14 2000 /sbin/pvcreate
-r-sr-xr-x 26 root sys 811008 Nov 14 2000 /sbin/pvdisplay
-r-sr-xr-x 26 root sys 811008 Nov 14 2000 /sbin/pvmove
-r-sr-xr-x 1 root sys 811008 Nov 14 2000 /sbin/sdstolvm
-r-sr-xr-x 26 root sys 811008 Nov 14 2000 /sbin/vgcfgbackup
-r-sr-xr-x 26 root sys 811008 Nov 14 2000 /sbin/vgcfgrestore
-r-sr-xr-x 26 root sys 811008 Nov 14 2000 /sbin/vgchange
-r-sr-xr-x 26 root sys 811008 Nov 14 2000 /sbin/vgcreate
-r-sr-xr-x 26 root sys 811008 Nov 14 2000 /sbin/vgdisplay
-r-sr-xr-x 26 root sys 811008 Nov 14 2000 /sbin/vgexport
-r-sr-xr-x 26 root sys 811008 Nov 14 2000 /sbin/vgextend
-r-sr-xr-x 26 root sys 811008 Nov 14 2000 /sbin/vgimport
-r-sr-xr-x 26 root sys 811008 Nov 14 2000 /sbin/vgreduce
-r-sr-xr-x 26 root sys 811008 Nov 14 2000 /sbin/vgremove
-r-sr-xr-x 26 root sys 811008 Nov 14 2000 /sbin/vgscan
-r-sr-xr-x 26 root sys 811008 Nov 14 2000 /sbin/pvck
-r-sr-xr-x 26 root sys 811008 Nov 14 2000 /sbin/pvremove
-r-sr-xr-x 26 root sys 811008 Nov 14 2000 /sbin/vgchgid
-r-s------ 1 root bin 286720 Nov 14 2000 /sbin/passwd
-r-sr-xr-x 1 root bin 270336 Nov 14 2000 /sbin/shutdown
The following users have command history activated:

/home/ids/.sh_history
/.sh_history

The other test we will want to perform is a network penetration test. We can test several items in this portion. First, we will want to ensure that Telnet and FTP are truly disabled. To perform this, from any host on the network, execute the command “telnet <hostname>” or “ftp <hostname>”, where <hostname> is the name of the host that has telnet/ftp disabled from the /etc/services file. The response you should expect is seen in Figure 12. It will state, “Connection refused” and will leave you at a connectionless telnet/ftp prompt. I also displayed a screenshot of the /etc/services file with FTP commented out – hence disabled.

Second, we will want to test that “root” cannot log in over the network, a feature which is enabled through the “secure.sh” script. I removed the /etc/securetty file and Figure 13 shows that the file /etc/securetty does not exist. For the purposes

of this test, I enabled Telnet and we can see in that same figure that a connection was established for root over the network. In Figure 14, I created and configured the /etc/securetty file again – which is performed by the "secure.sh" script. After attempting to connect to "orion" again, as root, it stated "Login incorrect" and took me to the login prompt again. This proves that root cannot log in over the network with the existence and proper configuration of the /etc/securetty file.

Third, we will want to test the "Trusted System" functionality of the systems. One way to test this is to attempt to log onto an account using the brute-force attack methodology. Our goal in this test is to ensure that the account locks up after three failed login attempts, as it is supposed to under a Trusted System. Figure 15 displays a proper login of user "hmahmoud" to ensure that the account is setup and configured properly. Figure 16 shows the response of the Trusted System on a brute-force type attack on a user's account. After three failed attempts, the user account "hmahmoud" locked up. In Figure 16 you can see the response that states, "Account is disabled – see Account Administrator", and proves that this test was successful and the system security held its ground in this attack.

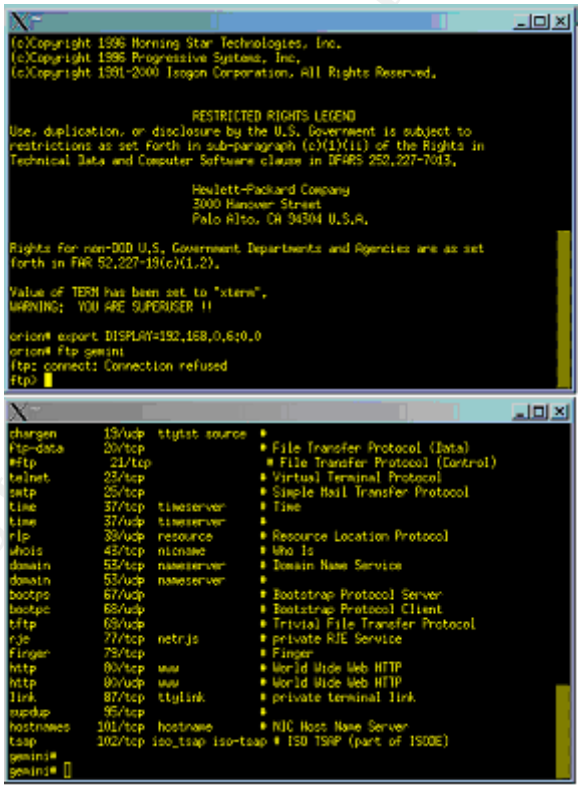

Figure 12: Disable FTP Connections

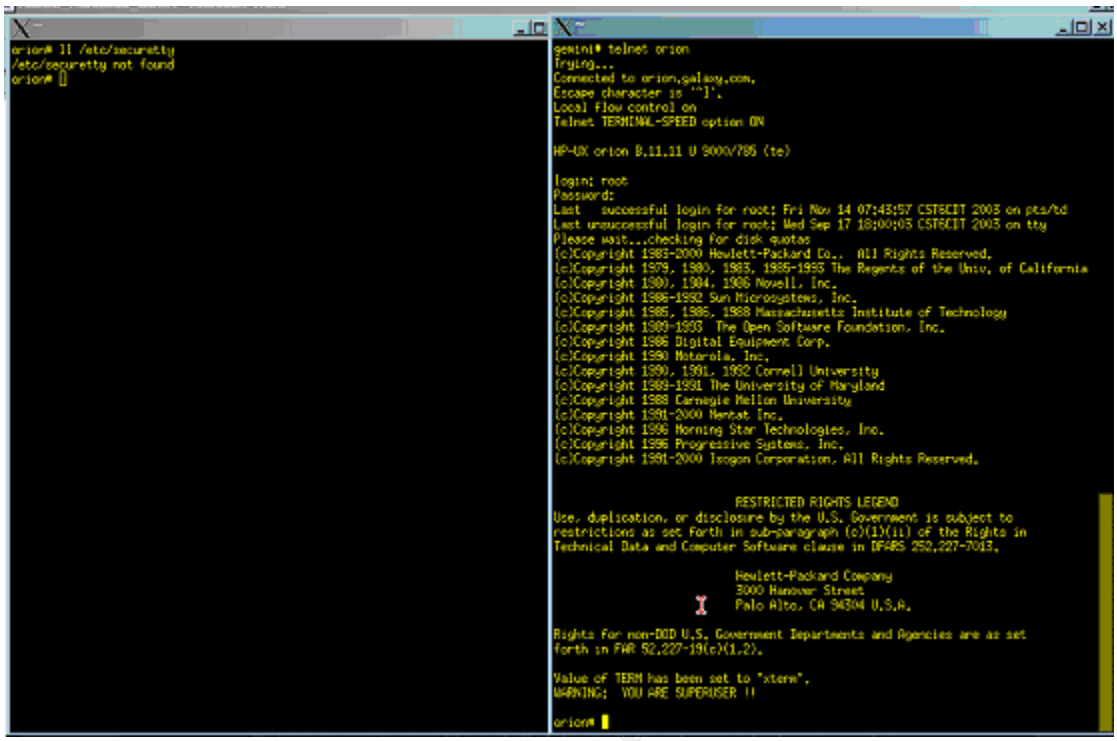

Figure 13: Root Logging in Over the Network

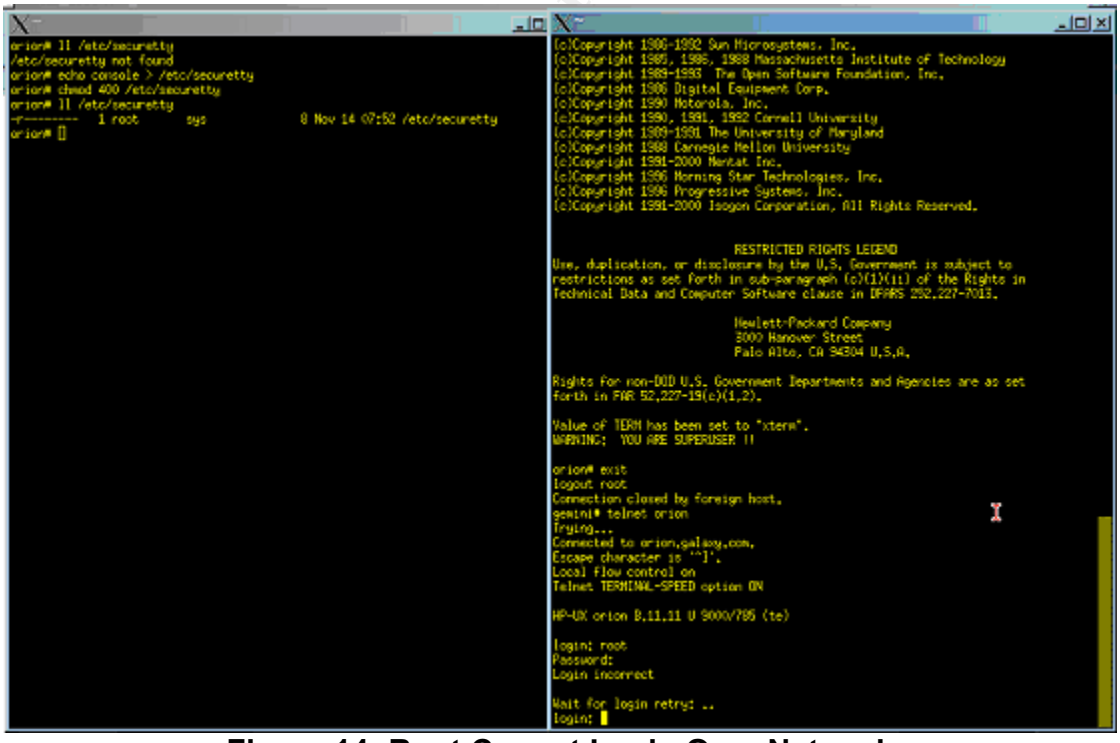

Figure 14: Root Cannot Login Over Network

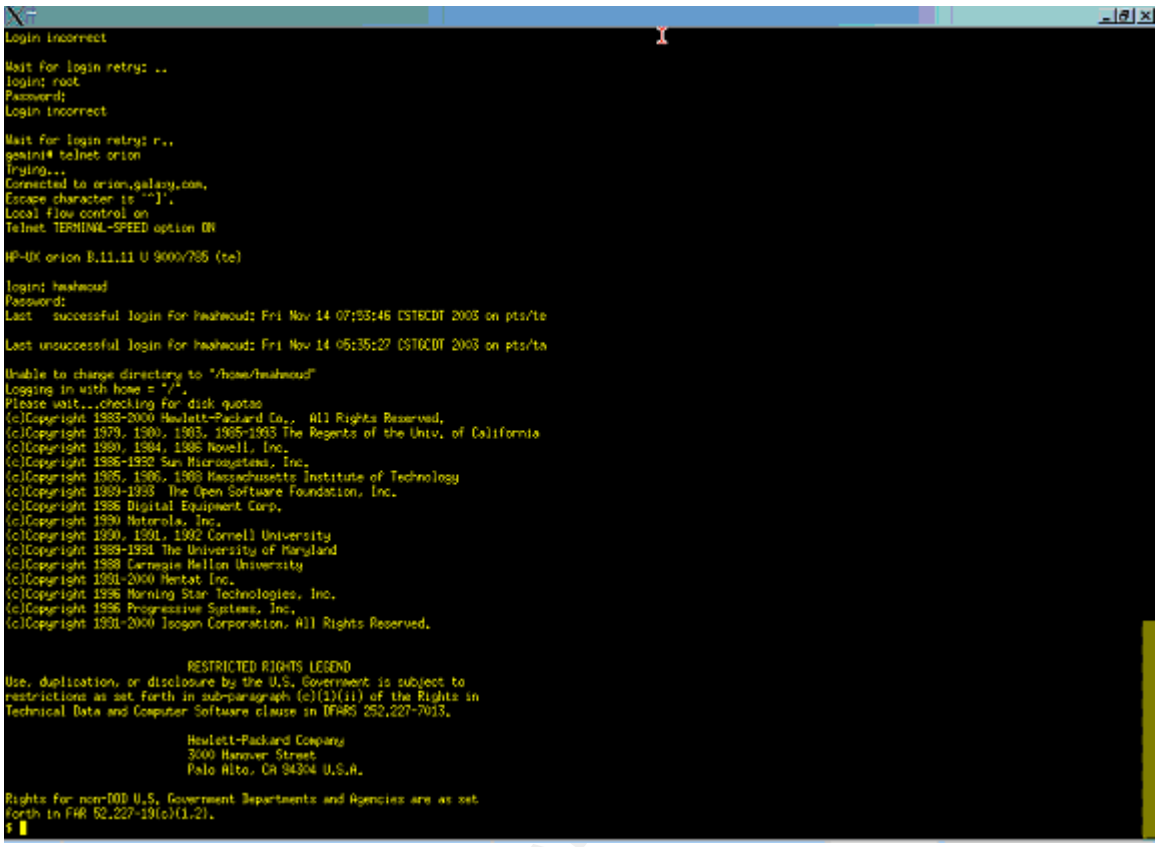

Figure 15: Successful Login of “hmahmoud”

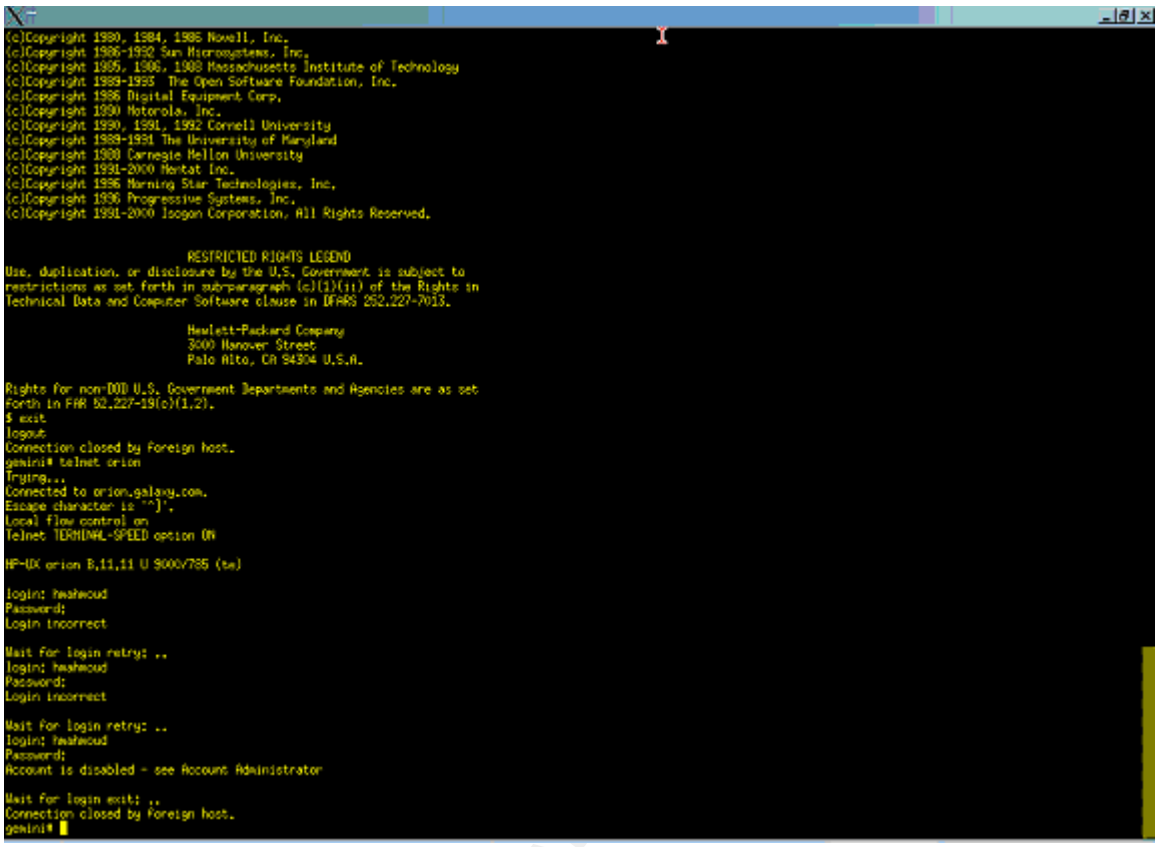

Figure 16: Trusted System Functionality – User Locked Out

4.2 RIDS Functionality Testing

We will now look into testing the RIDS functionality to verify its setup. To test the configuration, I have created a new schedule called “Testing”, and a new group called “TestGroup”. My approach is to initially test each template in BasicGroup individually to make sure that the response scripts are executed when an intrusion occurs and that they respond correctly. However I do not want to alter the already created production schedule, “GeneralPurpose”, or the already existing group, “BasicGroup”. That is why I created a new schedule and group for testing. The assumption is that if each template is tested individually and passes, then the response scripts are functioning correctly. For TestGroup, I will activate each template one-by-one and test them. If the response script works properly for the first template, then I will unselect it from TestGroup and select the next template and test it. In this process I get to test each template individually, by running it under TestGroup.

After creating the test schedule and group and selecting only one template, I stopped the schedule, “GeneralPurpose”, running on gemini (the Agent), and started the new schedule, “Testing”, to run instead. This was done from the System Manager window. Every time we complete the testing of a template, we need to stop the schedule, update TestGroup to include the next template to test, save it then restart the schedule again on gemini.

The first event tested was “monitor start of interactive sessions”. This carries a RIDS level of 2, meaning that an email will be generated when this occurs and sent to orion (the Administration system). It was also configured to run only if *root* or *ids* are the users targeted in this attack. The next step was to log into gemini and see if we receive the email that we had logged into gemini (therefore starting an “interactive session”). So from orion, I logged into gemini through “ssh”, and Figure 17 shows the email sent to *root* on orion. This proves that RIDS detected the logon and responded appropriately by sending an email to the administrator. It also shows that the user logged on to gemini from orion.

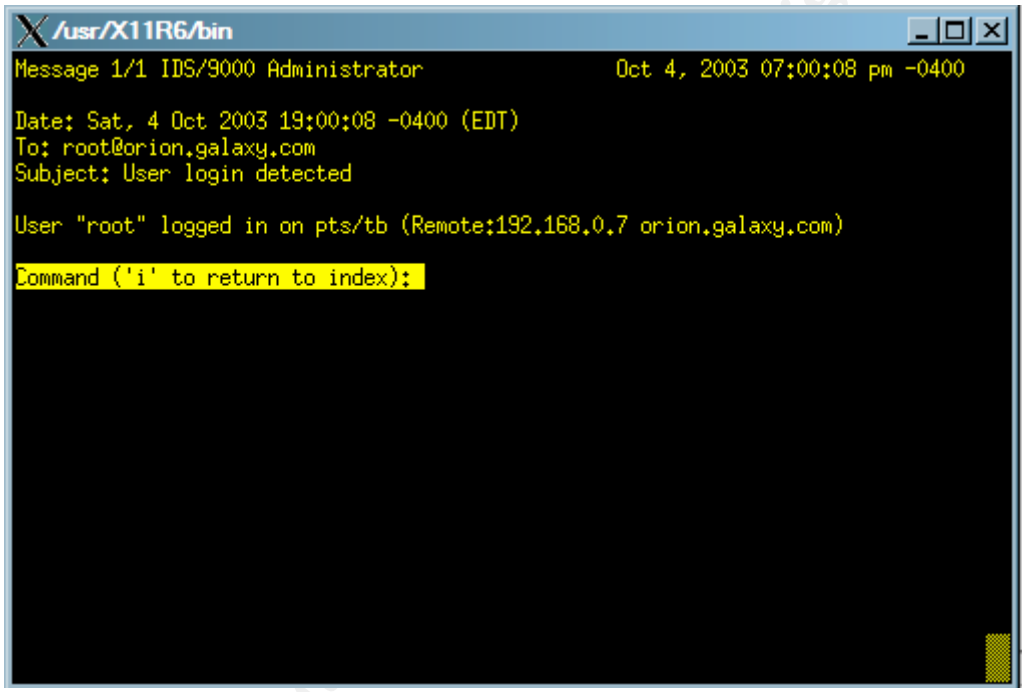

Figure 17: Monitor start of interactive session response

The next step involves testing for “changes to log files”. This has a RIDS level of 1, which means that “orion” will receive an email but a response will also take place. The response in this case is preserving the log files by emailing them to the System Administrator. This will allow the System Administrator to view the log files and obtain information about the attacks. Figure 18,19, and 20 display the emails “orion” received from this event. RIDS was able to detect changes to the log files, and took the appropriate response by emailing the System Administrator the log files, in an attempt to preserve them.

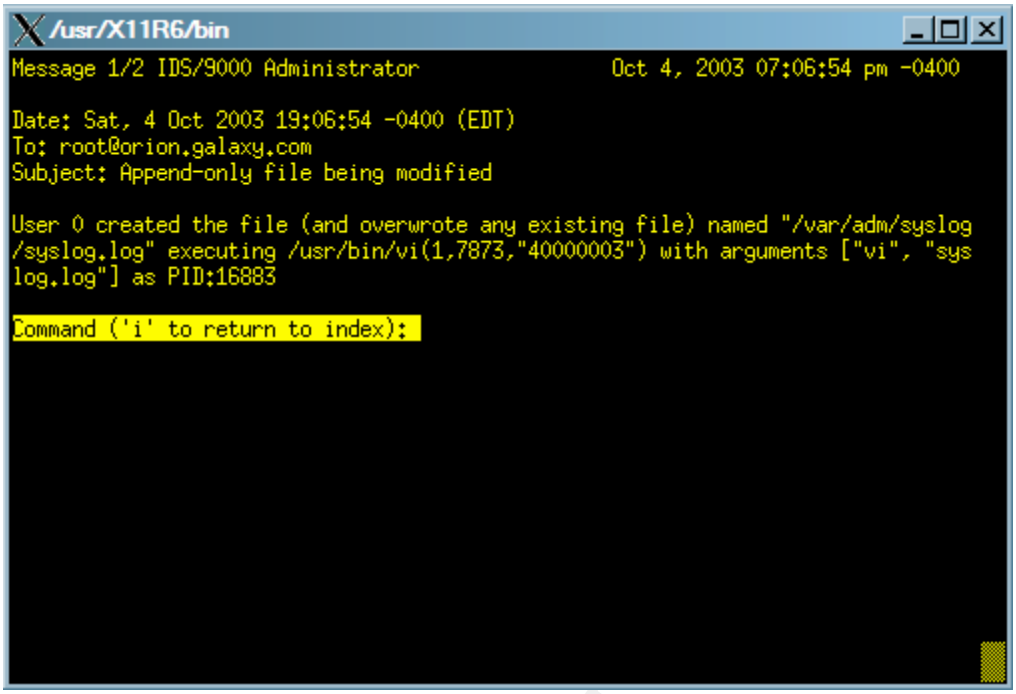

Figure 18: Changes to log files

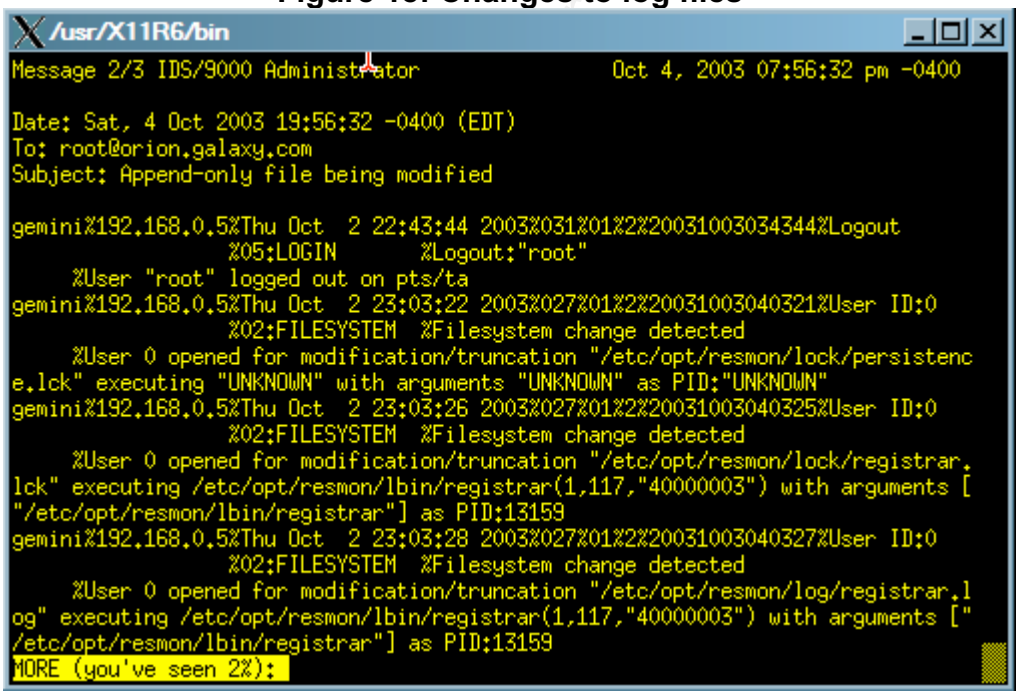

Figure 19: Preserved alert.log

Figure 20: Preserved syslog.log

The next test we want to perform is to check for the appropriate response when someone attempts to log on to a user's account and fails three times. The response we are looking for if someone is attempting to log on externally through "ssh", is that we want to lock the user account for the user under attack. I created

a new user, “hmahmoud”, performed three failed logons, and the account was locked. The response script takes into consideration whether the system is a Trusted System or not, and in either case it locks the account. The next step was to check that three failed “su” attempts will produce the appropriate response. In this case, the appropriate response is to lock the account of the user executing the “su” command. Therefore after unlocking the account “hmahmoud”, I performed three failed “su” commands to try and log on to “hmahmoud”, and my user, “hazem”, who initiated the attack got locked out.

A race condition attack is unfortunately (or fortunately) difficult to recreate, and therefore it makes it difficult to test for this kind of attack. The difficulty is not in the implementation of a race condition attack; the concept is easy and straightforward. The problem is in the amount of time it would take to recreate it. The same is true for a buffer overflow attack. However, I did test that the response script responds appropriately by simulating the attack and forcing the script to run that portion of the code. The code kills the malicious program and locks out the user who executed that program.

Testing is an important component in ensuring that you have a properly functioning system that performs as intended. It should not be overlooked, nor trivialized. In testing, I made sure to test each response feature to ensure it is functioning properly.

5.0 Monitoring & Maintaining Systems

5.1 Monitor RIDS

The RIDS system is only as good as the System Administrator who is monitoring it. If the system is not monitored properly and regularly, then having RIDS installed is pointless. Monitoring the alerts that take place for each Agent is done through the “Network Node” window in IDS/9000. To open this window, in the System Manager, click “View” then “Network Node”, or you can just double-click on the Agent listed. A snapshot of the “gemini” Network Node window is displayed in Figure 21.

The Network Node is a color-coded, GUI-based display of the alerts originated from IDS/9000. Each severity level is a different color, which allows you to quickly filter out and identify the serious alerts. Blue represents the least serious alert, and carries a severity number of 3. Yellow represents a medium-risk alert and carries a severity number of 2. Red is the most serious risk alert and it carries a severity number of 1.

Another important monitoring task is to check that the “idsagent” program is running on all the Agents. “idsagent” is the program that communicates with the Administration system to notify the Administration system that a possible intrusion has been detected and to generate an alert. If the “idsagent” program is running, then the Administration system assumes that the Agent is not generating any alerts, which it considers to be good. When in actuality, it is very bad, because now the Agents are completely vulnerable. Unfortunately IDS/9000 does not do a thorough job of testing whether “idsagent” is running on the Agents. A simple “ps -ef | grep idsagent” on each Agent will do the job of checking that the program is running. If you have a large number of Agents, this can be automated (through cron, for example) to run on each Agent machine and when it does not detect that “idsagent” is running it can email the System Administrator.

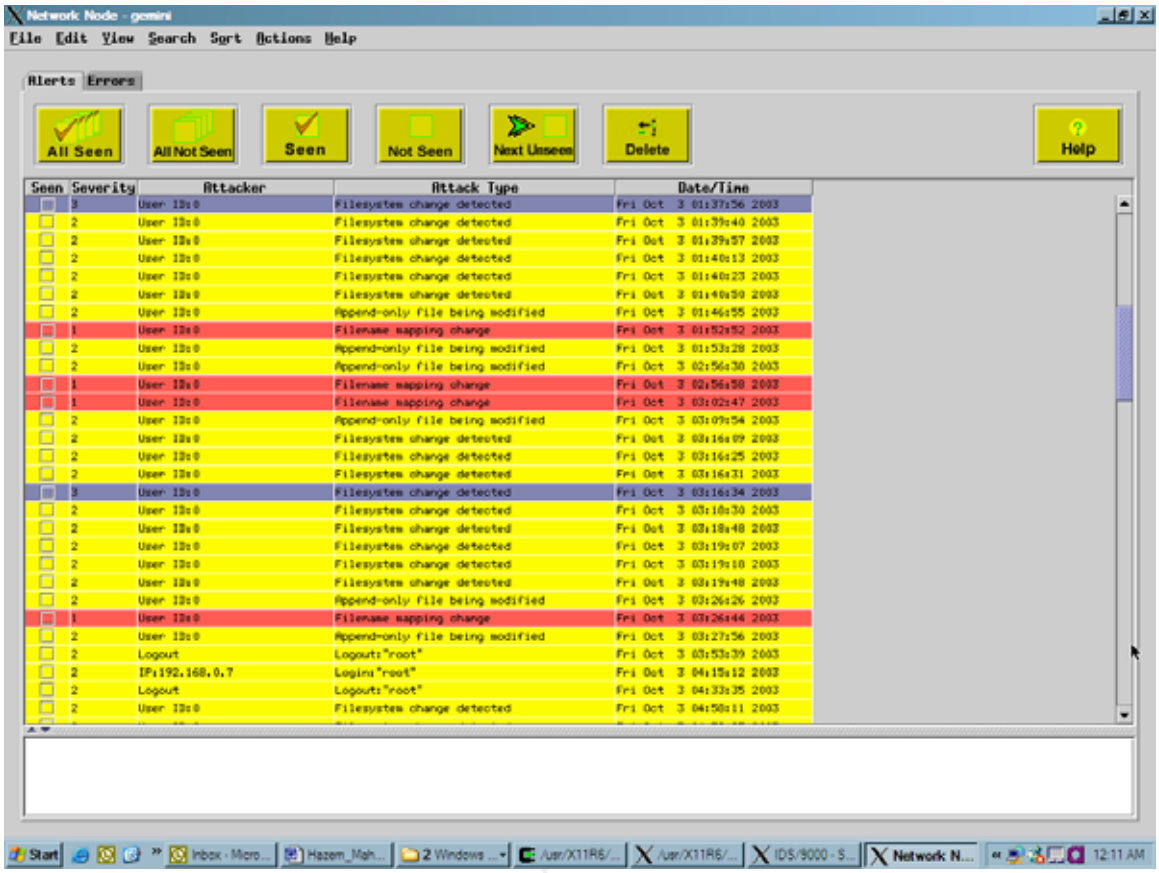

Figure 21: Network Node for "gemini"

5.2 RIDS Maintenance

It does not take much to maintain RIDS, but maintenance is required. The single biggest problem would occur if the log files filled up under /var/opt/ids. If /var/opt/ids is in its own filesystem, then the problem is not as severe since the /var filesystem is not affected. This can still cause the IDS/9000 system to stop generating alerts. Therefore regular log file cleanups are necessary. If a separate filesystem is not created for /var/opt/ids, then the problem is worse because it can cause a system crash. A simple script can be written and automated to check the size of the log files on the Administration and Agent systems and notify the Administrator. The action you would need to take is to remove the oldest logs first and then check if any more needs to be removed. A more advisable approach is to backup the logs, as opposed to just removing them completely. Creating a backup rotation to ensure that you keep a year's worth of logs is recommended. If you find yourself committing too much time into maintaining the logs in /var/opt/ids, you would probably want to consider expanding the /var/opt/ids logical volume.

5.3 HP-UX System Maintenance

HP-UX system maintenance is a general maintenance routine for all types of servers and workstations. Performing regular incremental and full backups are required. Check the system log files (ie: /var/adm/syslog/syslog.log) to make sure they do not fill up. Check the mounted logical volumes on a regular basis ("bdf") to make sure that none of the file systems is nearing its maximum limit. Perform regular cleaning of old user accounts and groups. Commands such as "pwck" and "grpck" check for inconsistencies in the password and group files. Leaving a supposedly deactivated account active is one of the worst security practices. Perform regular checks for suid/sgid programs on the systems. Be aware of all the programs that have this functionality and make sure they do what they are supposed to do. Ensure there is always proper physical security for critical servers and workstations. The list is vast and books are written on how to maintain HP-UX systems. This is just a short list to get you thinking of the types of things you need to look out for. The script "secure.sh", located in Appendix A, can be run on a regular basis and it will address most of the items mentioned above. It contains the command that will notify you of all the suid/sgid programs on the system. A simple comparison (using the "diff" command) from the last time it ran can easily filter out the new suid/sgid programs. It will also give you a listing of all the users/groups setup on the system, just in case any of them need to be removed. System maintenance is as critical, if not more critical, as maintaining the RIDS systems. You need to know your systems inside out. Most of the system maintenance tasks can be automated and can therefore save you a lot of time.

On a regular basis, there are new vulnerabilities discovered and new attacks created, and that is why it is important to constantly check for new updates and patches released by HP. This is a critical measure that needs to be addressed in regular system maintenance tasks. To search for the latest patches, you can visit ftp://ftp.itrc.hp.com/export/patches/hp-ux_patch_matrix where you will find a matrix listing all the new patches released. Once you determine the appropriate patches needed on your system, you can download the patches from ftp://ftp.itrc.hp.com/hp-ux_patches/ and install them on your system. Another good approach is to download a benchmark utility that you run on your system that notifies you of the appropriate patches needed to keep your system safe and secure. One good benchmarking tool is developed by CIS (Center for Internet Security) and can be downloaded at http://www.cisecurity.org/bench_hpux.html.

6.0 Summary

The concept of Intrusion Detection Systems are great, but nowadays, they are just not powerful/capable enough. We need to do more to protect our network and our systems. We need intelligent, more sophisticated systems to care for our systems. The days of manually sifting through logs and logs of alerts is just not practical any more. We need a more intelligent system that can detect if an alert is serious or not, and if it is, to take appropriate measures to reduce or eliminate the attack. We also need this intelligent system to provide us with auditing information. RIDS (Responsive Intrusion Detection System) is the implementation behind this concept. I attempted to demonstrate the concept of a simplified RIDS system in this paper. It is the concept of building your system to respond more intelligently to certain scenarios. It is no longer just notifying the System Administrator of an attack, but it takes it one step further and attempts to deal with the situation. Empowering your system to make decisions is what RIDS is all about.

As simplified as the RIDS implementation may have been in this paper, it addressed some of the most critical types of attacks and how to respond. For buffer overflow attacks and race condition attacks, we would like to respond by eliminating the malicious program, if it is still running, and still provide the program for the Administrator for auditing purposes. If it is known, it will also deactivate the user's account who originated the attack. For repeated failed login attempts and repeated failed su commands, we would like to deactivate the account to protect from further attacks. Regarding malicious manipulation of log files, our response will be to attempt to preserve the original file for auditing. Most likely, the attacker is trying to cover their trail. The same approach is considered when important files, such as the kernel, come under attack.

RIDS has a lot of room for growth and the many different implementation methods or tools are endless. However, it all originates from the same concept: empowering our IDS systems to respond intelligently to intrusion attacks, to reduce the damage or completely eliminate it, to allow the System Administrator more time to investigate the attack more thoroughly.

Appendix A – Unix Shell Scripts

secure.sh:

```
#!/bin/sh
##############################################
# Script to harden system                    #
# Developed by Hazem Mahmoud on 5/5/03.      #
##############################################

rm secure_results    #results file
echo "This file contains the results of hardening the system:\n" >> secure_results

#To convert to Trusted System
echo "\nChecking to see if system is trusted...\c" >> secure_results
/usr/lbin/getprdef -r
# grep for "System is not trusted." then if statement if want to convert.
if [ $? -eq 4 ]
then
echo "\nWould you like to convert to a trusted system (y/n)? \c"
read answer;
if [ $answer = "y" ]
then
echo "Will now convert to a Trusted System...\n" >> secure_results
/etc/tsconvert
echo "\nWARNING:";
echo "Open new telnet session and make sure you can log on as root!";
echo "If not, use this session to recreate password!\n";
fi
else
echo "\nSystem is already trusted!\n" >> secure_results
fi

echo "\nNow restricting root access to console...\n"
echo console > /etc/securetty
chmod 400 /etc/securetty
echo "\nRoot access is now restricted to the console\n" >> secure_results

echo "Now checking for users and groups configured on the system...\n"
echo "\nBelow is a list of users configured on the system:\n" >> secure_results
cat /etc/passwd | cut -f 1 -d : >> secure_results

echo "\nBelow is a list of groups configured on the system:\n" >> secure_results

cat /etc/group | cut -f 1 -d : >> secure_results
```

```
echo "\nNow checking for suid/sgid programs on system...\n"
echo "\nBelow is a list of suid/sgid programs on the system:\n" >> secure_results
find / \( -perm -4000 -o -perm -2000 \) -type f -exec ls -ld {} \; >> secure_results

echo "Now checking for command history...\n"
echo "The following users have command history activated:\n" >> secure_results
find / -name .sh_history >> secure_results
```

response.sh:

```
#!/usr/bin/sh

###########################################
# Author: Hazem Mahmoud
#
# This script is a response program which
# works with the IDS/9000 software. It
# carries out a specific response, based
# on the severity level generated by
# IDS/9000 and the nature of the attack.
###########################################

IDS_BASE="/opt/ids"
IDS_ETC="/etc/opt/ids"
IDS_VAR="/var/opt/ids"
RESPONSE_BASE=$IDS_BASE/response

# Response for "Monitor Start of Interactive Sessions":
# Send an email alert
if [[ $1 -eq 30 && $3 -eq 2 ]]
then
        echo "$8" | /usr/bin/mailx -s "$7" root@orion
fi

# Response for "Changes to Log Files":
# Send an email alert and email log file under attack to Administration server
if [[ $1 -eq 28 && $3 -eq 2 ]]
then
        logfile=`echo $8 | cut -f 12 -d " " | sed -e s/\"/""/g`
        cat $logfile | /usr/bin/mailx -s "$7" root@orion    #email log file
        echo "$8" | /usr/bin/mailx -s "$7" root@orion
fi

# Response for "Modification of Files/Directories":
# Copy file under attack to another new location as hidden file
if [[ $1 -eq 27 ]]
then
        file=`echo $8 | cut -f 12 -d " " | sed -e s/\"/""/g`
        cp $file /opt/ids/lib.$file                                  #copy file
        echo "$8\nNew File Location: /opt/ids/lib$file" | /usr/bin/mailx -s "$7"
root@orion
fi

# Response for "Creation of setUID files":
```

```
# Send an email alert
if [[ $1 -eq 9 && $3 -eq 1 ]]
then
        echo "$8" | /usr/bin/mailx -s "$7" root@orion
fi

# Response for "Repeated Failed Logins":
# Send an email and lock user account that hacker is attempting to hack into
if [[ $1 -eq 16 ]]
then
        victim=`echo $8 | cut -f 8 -d " "`
        /usr/lbin/getprdef -r
        if [ $? -eq 4 ]
        then
                /usr/sbin/usermod -s /usr/bin/false $victim
        else
                /usr/lbin/modprpw -l -m alock=YES $victim
        fi
        echo "$8" | /usr/bin/mailx -s "$7" root@orion
fi

# Response for "Repeated Failed su command":
# Send an email and lock user account of person executing su command
if [[ $1 -eq 15 ]]
then
        attacker=`echo $8 | cut -f 2 -d " " | sed -e s/\"/""/g`
        /usr/lbin/getprdef -r
        if [ $? -eq 4 ]
        then
                /usr/sbin/usermod -s /usr/bin/false $attacker
        else
                /usr/lbin/modprpw -l -m alock=YES $attacker
        fi
        echo "$8" | /usr/bin/mailx -s "$7" root@orion
fi

# Response for "Race Condition Attacks":
# Deactivate malicious user's account, preserve copy of attacking program
# but then remove it, send and email
if [[ $1 -eq 6 ]]
then
        attacker=`echo $8 | sed -e s/^.*ATTACKER was \(.*\)/\1/`
        /usr/lbin/getprdef -r
        if [ $? -eq 4 ]
        then
                /usr/sbin/usermod -s /usr/bin/false $attacker
```

```
        else
                /usr/lbin/modprpw -l -m alock=YES $attacker
        fi
        badprog=`echo $8 | sed -e s/^.*running \(.*\)/\1/`
        cat $badprog | /usr/bin/mailx -s "$7" root@orion
        echo "$8" | /usr/bin/mailx -s "$7" root@orion
fi

# Response for "Buffer Overflow Attacks":
# Kill malicious program, preserve copy of attacked program but then
# remove it, send and email
if [[ $1 -eq 5 ]]
then
        pid=`echo $8 | sed -e s/^.*PID:\(.*\)/\1/`
        kill $pid
        badprog=`echo $8 | sed -e s/^.*executing \(.*\)/\1/`
        cat $badprog | /usr/bin/mailx -s "$7" root@orion
        echo "$8" | /usr/bin/mailx -s "$7" root@orion
fi
exit 0
```

Appendix B – Reasons for Template Configurations

Monitor Start of Interactive Sessions:

- **Notify when these users begin a session:**
  Default: root, ids, www, news, daemon, bin, sys, adm, uucp, lp, nuucp, hpdb
  Revised Values: root, ids
  Reason: For the purposes of implementing the IDS/9000 system, the two most critical users are *root* and *ids*. If an intruder gains access to root, they have access to the entire system. "ids" must also be secured because if they have access to this user they can manipulate the IDS/9000 system to not report alerts for example, therefore defeating the whole purpose of the system. While the other users, such as bin and daemon, are important, they are not as critical when it comes to the overall general security of the system. They are still limited in their scope and power.

Monitor Logins/Logouts:

- **Ignore these users:**
  Default: -
  Revised Values: N/A
  Reason: We will monitor all logins/logouts into the system because that is valuable auditing information to keep. However, we will run a response script when user *root* or *ids* log in to notify the administrator of this, since they are critical users and their presence is worth noting.

Changes to Log Files:

- **Files which should only be appended to:**
  Default: /var/adm/btmp, /var/adm/wtmp, /etc/btmp, /etc/wtmp, /var/adm/messages, /var/adm/syslog/mail.log, /var/adm/syslog/syslog.log, /var/adm/pacct, /var/adm/sulog
  Revised Values: /var/adm/syslog/syslog.log, /var/opt/ids/alert.log
  Reason: The reason I took out the btmp and wtmp files is because btmp is a file that contains bad login attempts and wtmp contains a record of all logins/logouts and both of these features are already monitored in IDS/9000 under the templates "Repeated Failed Logins" and "Monitor Logins/Logouts", respectively. /var/adm/syslog/syslog.log is the system log file and therefore it needs to be properly monitored for any malicious formatting. The alert log for IDS/9000 is kept in /var/opt/ids/gui/logs and if it is deleted or edited, the alerts may never reach the Administration system.

Modification of Files/Directories

- **Watch these files for modifications/creation:**
  Default: /stand/vmunix, /stand/kernrel, /stand/bootconf, /etc/passwd, /etc/group, /.rhosts, /.shosts, /etc/hosts.equiv, /etc/hosts.allow, /etc/hosts.deny, /etc/inetd.conf
  Revised Values: /stand/vmunix, /stand/kernrel, /stand/bootconf, /etc/passwd, /etc/group, /.rhosts, /.shosts, /etc/hosts.equiv, /etc/hosts.allow, /etc/hosts.deny, /etc/inetd.conf
  Reason: I have decided to keep all the default values. Each file listed here is a critical file and impacts the security of the system.
- **Ignore these files:**
  Default: /etc/ptmp, /etc/.pwd.lock, /etc/utmp, /etc/utmpx, /etc/rc.log, /etc/lvconf/lvm_lock
  Revised Values: N/A
  Reason: I see no reason to check any of these files. “utmp” contains record of all users logged onto the system, which is already tracked through IDS/9000. “utmpx” is similar to “utmp” but contains more information about the user logged in. Therefore, ignoring them is fine.
- **Watch these directories for modification:**
  Default: /etc, /bin, /sbin, /stand, /lib, /usr/bin, /opt
  Revised Values: /stand
  Reason: To monitor all these default directories would drain the system resources. Performing any type of function will certainly modify at least one of these directories. I kept it on the default values over night to get a feel for how many alerts would be generated. Considering that there was no major activity on the system (no users were logged in and performing normal functions), there were over 100 alerts that were generated. However it is not common for the /stand directory to be modified, and it is probably the most important directory since it contains the actual kernel and boot configuration file. Therefore, it would be wise to be notified when this directory is modified.
- **Ignore these directories:**
  Default: -
  Revised Values: N/A
  Reason: The above property sufficiently incorporates this property.

Creation of setUID Files:

- **List of critical user IDs to be monitored (for suid):**
  Default: 0, 1, 2, 3, 4, 5, 9, 11
  Revised Values: 0, 102
  Reason: setUID files have the suid bit set and can allow an attacker to gain access to a shell as that user. The two users we are concerned with their security are *root* and *ids*.

Repeated Failed Logins:

- **Number of failures to exceed:**
  Default: 2
  Revised Values: N/A
  Reason: 2 is a good value. It allows a user to retype their password if they accidentally typed it, and yet generates an alert if a brute-force login attack is attempted.
- **Time span to detect failures over:**
  Default: 10
  Revised Values: N/A
  Reason: Again, this is a good number between detecting failures that occur to close together or not too far apart.
- **Suppression period for reporting:**
  Default: 30
  Revised Values: N/A
  Reason: Once again a good average number.

Repeated Failed "su" Command:

- **Number of failures to trigger on:**
  Default: 2
  Revised Values: N/A
  Reason: This should carry the same ruling as that for the Repeated Failed Logins
- **Time span to detect failures over:**
  Default: 24
  Revised Values: 10
  Reason: This should carry the same value as that for the Repeated Failed Logins. 10 is a good average value.

Race Condition Attacks:

- **What user IDs to monitor for being attacked:**
  Default: 0, 1, 2, 3, 4, 5, 9, 11
  Revised Values: 0, 102
  Reason: Our main focus of users getting compromised through Race Condition Attacks are the *root* and *ids* users.
- **How many paths to keep track of per process (0 is all):**
  Default: 10
  Revised Values: N/A
  Reason: The lower the number, the more paths we keep track of, and therefore the more process time we are consuming.

Buffer Overflow Attacks:

- **What user IDs to monitor for being attacked:**
  Default: 0, 1, 2, 3, 4, 5, 9, 11
  Revised Values: 0, 102
  Reason: The major concerns for buffer overflow attacks are for *root* or *ids*. A buffer overflow attack can allow a user to gain control of the shell that the program is running in. If the program is a setuid program, then the attacker gains control of the shell that the program is running in.

Appendix C – Tables

Appendix D – Figures

References:

1. Hewlett-Packard, *HP Intrusion Detection System/9000 Administrator's Guide (J5083-90007)*
2. Hewlett-Packard, *HP Intrusion Detection System/9000 Release 2.1 Release Notes (J5083-90008)*
3. Hewlett-Packard, *HP-UX Secure Shell A.03.61.001 Release Notes (T1471-90008)*
4. Steves, Kevin, *Building a Bastion Host*
   URL:http://downloads.securityfocus.com/library/bastion11.html
5. Wong, Chris (2002). *HP-UX lli Security*. New Jersey: Prentice Hall PTR
6. Pfleeger, Charles and Shari Lawrence. *Security in Computing*, *3rd ed.* Prentice Hall, 2003
7. Arthur, Lowell Jay and Burns, Ted. *Unix Shell Programming, 4th ed.* Wiley Computer Publishing, 1997
8. Rehman, Rafeeq Ur. *HP Certified*. Prentice Hall, 2000
9. HP-UX "man" pages (/opt/ids/share/man/man1m and /opt/ids/share/man/man5)
10. HP (Hewlett-Packard). IT Resource Center (ITRC), URL: http://itrc.hp.com
11. Pomeranz, Hal. *Common Issues and Vulnerabilities in Unix Security*. SANS Institute, 2002

Last Updated: May 28th, 2017

Upcoming Training

SANS London July 2017	**London, United Kingdom**	**Jul 03, 2017 - Jul 08, 2017**	**Live Event**
SANSFIRE 2017	**Washington, DC**	**Jul 22, 2017 - Jul 29, 2017**	**Live Event**
SANS Network Security 2017	**Las Vegas, NV**	**Sep 10, 2017 - Sep 17, 2017**	**Live Event**
SANS OnDemand	**Online**	**Anytime**	**Self Paced**
SANS SelfStudy	**Books & MP3s Only**	**Anytime**	**Self Paced**