Global Information Assurance Certification Paper

Implementing a Secure IRC server with Fedora Core 1 and grsecurity

GCUX Practical version 2.1
Option 1: Securing Unix Step by Step
November 1, 2004

George D. (Danny) Walker II
GCIA, GCIH, GCFW, ISSAP, CISSP, SSCP, CISA

Table of Contents

Introduction

GIAC Enterprises has become a huge success and is a multi-national conglomerate with offices in 30 countries. Since Jack, John and Adam Chan took over the fortune cookie operation it has grown by leaps and bounds. Adam's heavy use of open source solutions has enabled the company to leverage their state-of-the-art network without heavy infrastructure costs. With such a large diverse network the company now has information technology staff all over the world. There is a need for a simple secure method to provide real-time communication especially when coordinating network issues, security investigations or virus/worm outbreaks. For an overview of GIAC Enterprises initial network it can be found at http://www.giac.com/practical/GCFW/Danny_Walker_GCFW.pdf.

Server Role

This server will be used to enable secure real-time communication between the company's IT staff. This will provide the ability for users to communicate via electronic chat as well as transfer files in real-time on a hardened platform resistant to penetration and other interruptions. Since this server will have its own accounts and authentication requirements it could be considered safer than using email especially if the primary network is suspected of being compromised. This machine will sit in the DMZ of GIAC Enterprises server farm located in Boulder, Colorado.

Hardware

Adam decided to upgrade the mail server and decided to use the following hardware for this server:

- Dell PowerEdge 650
  - P4 2.6 GHz processor
  - 1GB DDR RAM
  - RAID-1 card
  - 2-7.2K RPM 20GB IDE HD
  - 100bT NIC
  - Onsite-4hr parts service

Operating System

- Fedora CORE 1 http://fedora.redhat.com
- Linux 2.6.7 Kernel http://www.kernel.org

Third Party Software

- Unreal IRCd v3.2.1 http://www.unrealircd.org
- OpenSSL 0.9.7d http://www.openssl.org
- grsecurity 2.01 patch http://www.grsecurity.net
- gradm-2.01 http://www.grsecurity.net

- chpax-0.7 http://pax.grsecurity.net
- paxctl-0.2 http://pax.grsecurity.net

Services/Processes Required

- IRCd daemon IRC services
- SSHd daemon SSH services
- Syslogd daemon Syslog services
- Yum client software Updates the OS with latest RPM packages
- Rdate client software Updates the OS with the latest time

User Access

- Users – The primary network service offered by this server will be IRC. It will be accessible both inside on the internal network as well as from the Internet.
- Administrator – Administrators will only be able to access this server from the internal network.

Risk Mitigation Plan

Definitions

The following definitions will be used throughout this document.

Information System (IS) - Set of information resources organized for the collection, storage, processing, maintenance, use, sharing, dissemination, disposition, display, or transmission of information. [1]

Risk - Possibility that a particular threat will adversely impact an IS by exploiting a particular vulnerability. [1]

Vulnerability - Weakness in an IS, system security procedures, internal controls, or implementation that could be exploited. [1]

Attack - Attempt to gain unauthorized access to an IS's services, resources, or information, or the attempt to compromise an IS's integrity, availability, or confidentiality. [1]

Threat - Any circumstance or event with the potential to adversely impact an IS through unauthorized access, destruction, disclosure, modification of data, and/or denial of service. [1]

Countermeasure - Action, device, procedure, technique, or other measure that reduces the vulnerability of an IS. [1]

Threat Modeling

Threat Modeling is a structured approach that allows the systematic identification and rating of threats, which can impact a system or application. Below is the process used to analyze the threats that could affect our server [2][3].

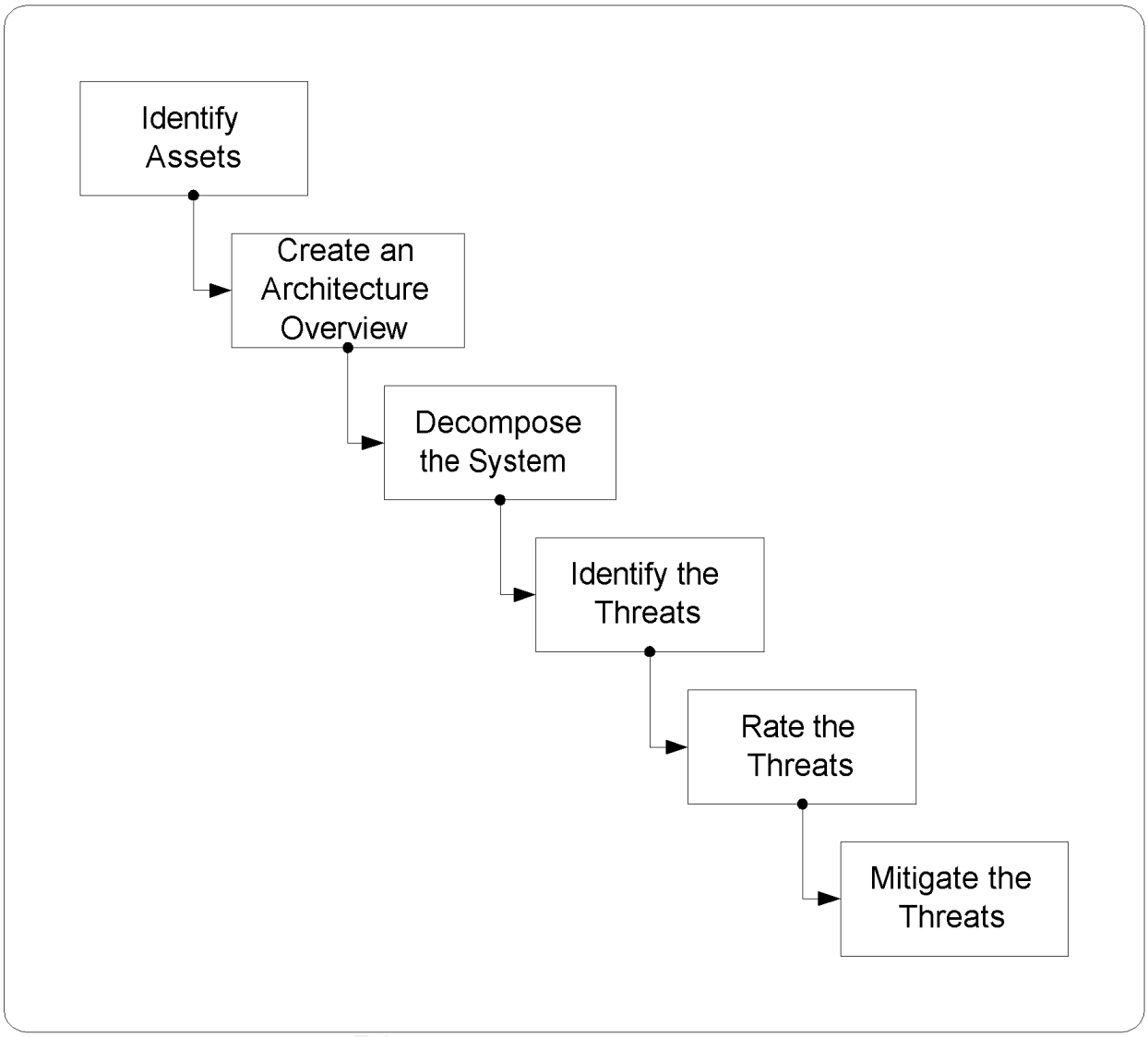


Figure 1: Threat Modeling Process

1. Identify Assets – Identify the assets that you need to consider in your analysis.
2. Create an Architecture Overview – Draw an overview of the system.
3. Decompose the System – Breakdown the system into its components including users, interfaces and system dependencies. The goal is to uncover vulnerabilities in the design, implementation or deployment of the system.
4. Identify the Threats – While thinking like attackers, identify the threats that could affect your system.
5. Rate the Threats – Document and prioritize the threats.
6. Mitigate the Threats- Develop countermeasures for each threat.

Identify Assets

Only the IRCd server is being evaluated for this part of the paper. For threat modeling to be complete the entire network would need to be analyzed.

Create an Architecture Overview

For the overview, the server has been laid out in relation to the network, users and other servers.

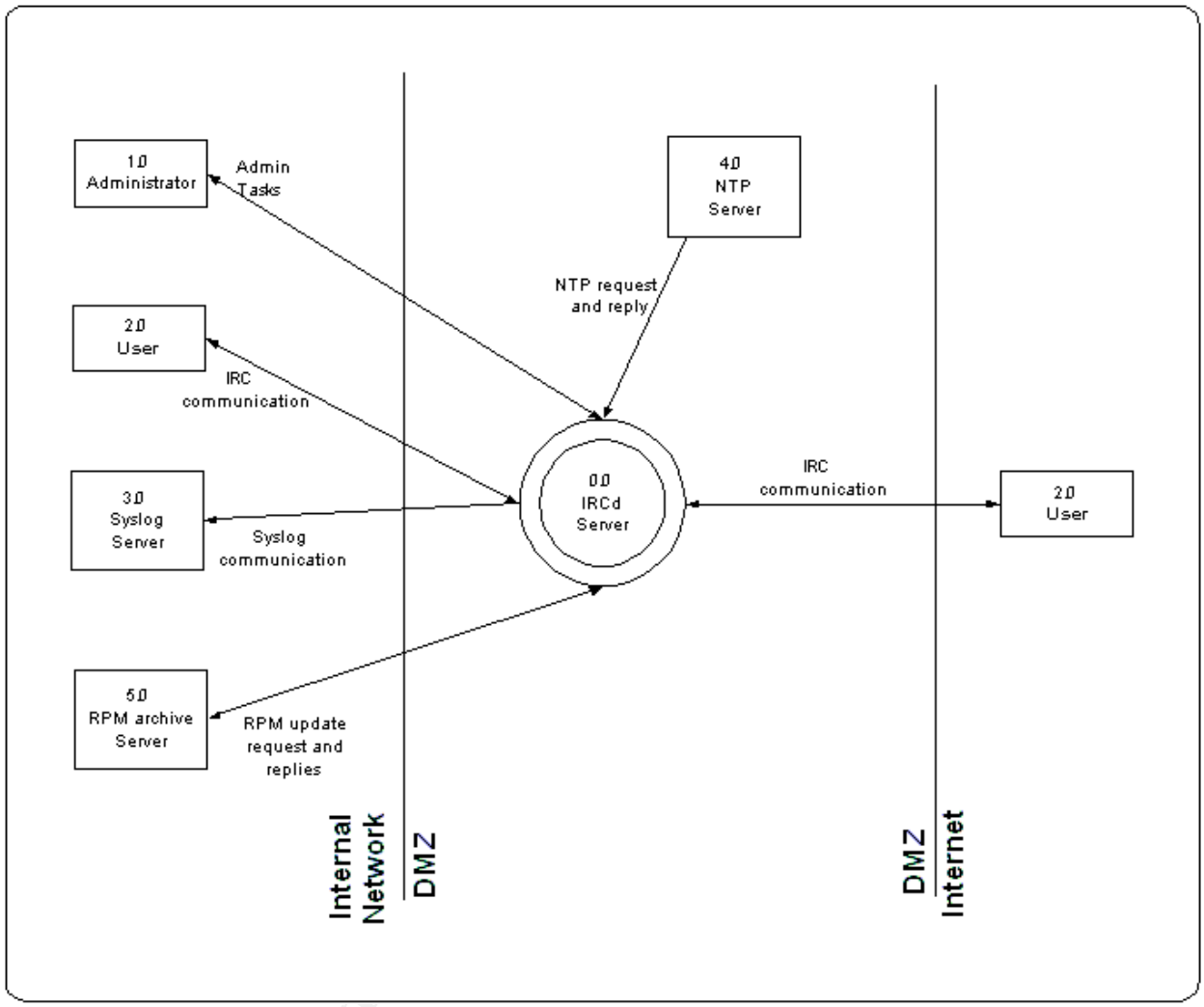


Figure 2: Context DFD

Also for this section the technologies used will be identified.

Technology/Platform	Implementation details
Fedora Core Linux	Operating system
Unreal IRCd	Provides user authentication and IRC services.
Secure Sockets Layer	Provides encryption of the IRC traffic.
Secure Shell (SSHd)	Provides administrator access to server
Syslogd	Provides remote logging
NTPd/Rdate	Provides accurate time data
Yum	Provides updates for patched components

Decompose the System

Next the system will be broken down to create a security profile based on the traditional areas of vulnerability. Potential weaknesses in the system would be reviewed to identify the areas of vulnerability.

Trust boundaries – Consider the inbound and outbound data flows, user input, and server trust relationships.

- Inbound data flows
  - SSH client connecting to SSHd
  - IRC client connecting to IRCd
- Outbound data flows
  - Rdate client connecting to NTP server
  - Syslog connecting to Syslog server
  - Yum client connecting to RPM archive server
- User input
  - IRCd – Users access with IRC clients
  - SSHd – For the few users with access to this service
- Server trust relationships – This server intentionally would not have any server trust relationships to limit the possibility of a compromise taking this server down.

Data flows – Data flow can be looked at between applications as well as inside the system.

- Inbound data flows
  - SSH client connecting to SSHd
  - IRC client connecting to IRCd
- Outbound data flows
  - Rdate client connecting to NTP server
  - Syslog connecting to Syslog server
  - Yum client connecting to RPM archive server
- Rdate updates the operating system once the correct time has been received from the NTP server.
- Syslog receives log data from several applications and the kernel.
- After the Yum client pulls over the new packages it uses RPM to install them.
- After SSHd authenticates a user it allows user to access a bash shell.

Entry points – Consider anything that could serve as an entry point. Review the authentication type at each point.

- Network (TCP/IP)
  - SSHd access on TCP port 22
  - IRCd access on TCP port 6697
- Application
  - Yum pulls over a bad RPM package and installs it
  - Rdate gets bad NTP data and attempts to update the system

Privileged code – Consider any code that accesses specific types of secure resources (e.g. environment variables, event logs, file systems, registry, etc.) or performs other privileged operations.

- Syslog receiving log data from applications and kernel
- SSH clients access to bash shell
- Rdate updating kernel with new time and date
- Yum installing new packages

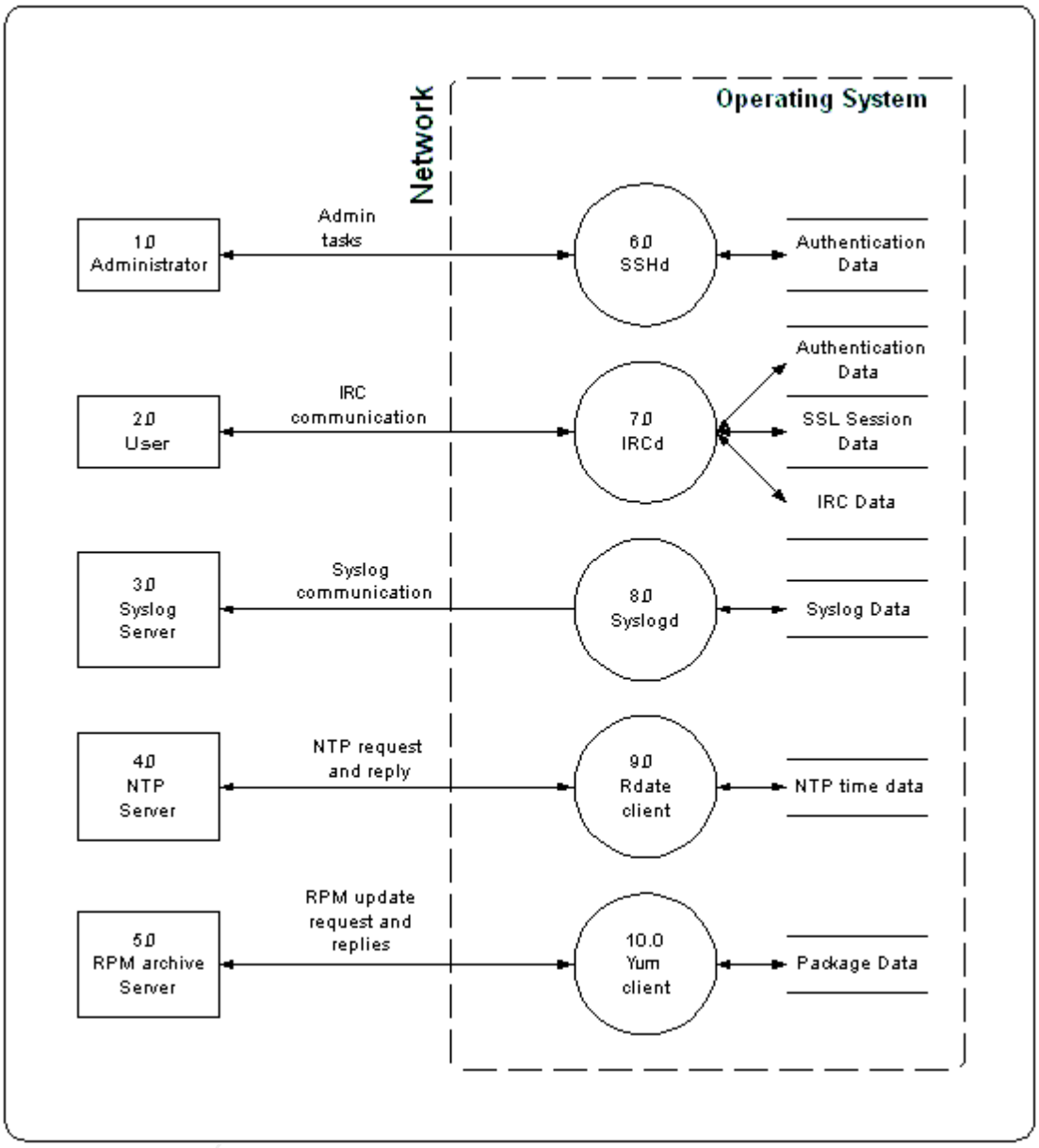


Figure 3: Level 0 DFD

Identify the Threats

Compile the data collected to date and attempt to identify potential threats. One basic approach uses categorized threat lists, which breaks down a list of common threats grouped by network, host and application categories. [2] Another is the use of the STRIDE model that is an acronym for **S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service and **E**levation of privilege. [3]

Component	Question Asked	Examples
Spoofing	Can an attacker pretend to be a valid user, resource, client or server?	DNS Poisoning Forged email Traffic Replay Man-in-the-Middle Session Hijacking Password Compromise
Tampering	Can an attacker modify data without authorization?	Unauthorized file modification Data modification via malware Network packet injection
Repudiation	Can the system prove what a user or process does?	Anonymous network access File deletion No unauthorized login attempt detection
Information Disclosure	Can the system protect data from unauthorized users or processes?	Network sniffing Data access via malware Applications that allow access of data Physical access to servers Information gathering/Footprinting
Denial of Service	Can an attacker deny service to valid users?	Service DoS Network DoS Coding errors Authentication DoS
Elevation of Privilege	Can an attacker gain additional unauthorized system privileges?	Buffer/Heap overflows Race conditions Keystroke monitoring Network sniffing

As can be seen from the examples, there are attacks that may fall under two or more components. This persists when defining threats where multiple threats may have the same attack.

A threat tree is a method of collecting and documenting the potential threats on the system in a structured way. This will be used to reflect the threats associated with this server. In addition, attack patterns, which are generic representations of commonly occurring attacks that can enable many different threats to occur, will be addressed.

1. Gather Information about network and servers

1.1. Open source investigation and google (or another search engine)
- 1.1.1. Newsgroups
- 1.1.2. Listserve archives
- 1.1.3. Personal and company websites
- 1.1.4. Personal employee resumes

1.2. Enumerate network structure
- 1.2.1. Traceroute/TCPtraceroute
- 1.2.2. DNS

1.3. Enumerate Router Access Controls
1.4. Enumerate Firewall Access Controls
1.5. Enumerate Hosts
    1.5.1. Port scan host
        1.5.1.1. Identify services based on banners
    1.5.2. OS Fingerprinting

2. Denying service to users

2.1. SSH
    2.1.1. Service DoS
        2.1.1.1. SYN floods
        2.1.1.2. Flood authentication requests
        2.1.1.3. Crash service
    2.1.2. Intercept traffic from client to the server without delivering it
    2.1.3. Network DoS
        2.1.3.1. Distributed DoS
        2.1.3.2. Take down Router
        2.1.3.3. Take down Firewall
    2.1.4. Turn off power to server
        2.1.4.1. Requires physical access AND
        2.1.4.2. Account to access server
    2.1.5. Crash server
        2.1.5.1. Filling up disk space
        2.1.5.2. Server DoS exploit
2.2. IRC
    2.2.1. Service DoS
        2.2.1.1. SYN floods
        2.2.1.2. Flood authentication requests
        2.2.1.3. Crash service
    2.2.2. Intercept traffic from client to the server without delivering it
    2.2.3. Network DoS
        2.2.3.1. Distributed DoS
        2.2.3.2. Take down Router
        2.2.3.3. Take down Firewall
    2.2.4. Turn off power to server
        2.2.4.1. Requires physical access AND
        2.2.4.2. Account to access server
    2.2.5. Crash server
        2.2.5.1. Filling up disk space
        2.2.5.2. Server DoS exploit

3. Intercept a network connection for a user

3.1. SSH
    3.1.1. Break RSA encryption
    3.1.2. Obtain a key
    3.1.3. Man-in-the-middle attack
    3.1.4. Poor Installation/Configuration
    3.1.5. Software vulnerability

3.1.6. Modify software distribution
3.2. IRC
3.2.1. Break AES encryption
3.2.2. Obtain a key
3.2.3. Man-in-the-middle attack
3.2.4. Poor Installation/Configuration
3.2.5. Software vulnerability
3.2.6. Modify software distribution

4. Gaining remote access to server

4.1. SSH Service vulnerability
4.1.1. Poor Installation/Configuration
4.1.2. Software vulnerability
4.2. SSH protocol
4.2.1. Library vulnerability
4.3. IRC Service vulnerability
4.3.1. Poor Installation/Configuration
4.3.2. Bypass authentication
4.3.2.1. Clear text credentials sent over network AND
4.3.2.2. Attacker users sniffing tool
4.3.2.2.1. Attacker recognizes credential data
4.3.3. Software vulnerability
4.4. IRC protocol
4.4.1. Library vulnerability
4.4.2. Eavesdrop session
4.4.3. Hijack session

5. Gain user credentials

5.1. Brute force password
5.2. Guess using user intelligence (e.g. Name of kids, spouse, pets, etc.)
5.3. Observe password being entered
5.4. Social engineer password
5.5. Authorized user AND
5.6. Attempt to crack password
5.6.1. Obtain password file
5.6.1.1. Steal backup tape OR
5.6.1.2. Break filesystem security OR
5.6.1.3. Buffer overflow AND
5.6.1.4. Pull hash with Pwdump
5.7. Malicious software
5.7.1. User acquires malicious code (virus/worm/spyware) AND
5.7.2. Malicious code installs keystroke monitor AND
5.7.3. Communicates back to attacker

Several attack patterns emerged from the lists above. These are usually not goals in and of themselves but more of a means to achieving the threat.

Rate the Threats

Rating the threats is important. The cost of mitigating a threat can be more costly than the damage potential from the threat. A formula commonly used to calculate risk is *Risk = Event Probability * Damage Potential*. This formula is not sufficient, as it does not look at all of the necessary dimensions to provide the ability to rate threats.

The DREAD model, which is an acronym for **D**amage potential, **R**eproducibility, **E**xploitability, **A**ffected users and **D**iscoverability [2][3], will be used. Here is a breakdown on DREAD:

Rating	**Question**	**High (3)**	**Medium (2)**	**Low (1)**
Damage potential	How much damage is done if it is exploited?	Gain admin or system access; bypass all security; deny access	Gain user access; gaining sensitive information	Gain trivial information
Reproducibility	How easy is it to reproduce the attack?	Easy to reproduce; works each time	Reproduced only under certain conditions	Difficult to reproduce results
Exploitability	How easy is it to launch the attack?	Script kiddies can launch it; exploit found on net	Requires skilled programmer to create; Requires skilled attacker	Requires skilled attacker with inside information
Affected users	How many users would be affected and at what level?	All users, standard configuration, key customers	Some users; non-standard configuration	Few users; Unique conditions required
Discoverability	How easy is it to locate the vulnerability?	Vulnerability published; easy to locate	Vulnerability in software not typically used	Obscure bug; difficult to locate

The above attacks should be categorized into the following rates:

- Low (<8)
- Medium (8-11)
- High (>11)

Attack (Threats)	D	R	E	A	D	Total	Rating
Open source investigation (1)	2	2	3	3	3	13	HIGH
Enumerate network structure (1)	2	2	3	3	3	13	HIGH
Enumerate Router Access Controls (1)	2	2	3	2	3	12	HIGH
Enumerate Firewall Access Controls (1)	2	2	3	2	3	12	HIGH
Enumerate Hosts (1)	2	2	3	3	3	13	HIGH
SYN Foods (2.1 & 2.2)	3	2	3	3	3	14	HIGH
Flood authentication requests (2.1 & 2.2)	3	2	3	3	3	14	HIGH
Crash service (2.1 & 2.2)	3	2	3	3	2	13	HIGH
Intercept traffic from client to server without delivering it (2.1 & 2.2)	3	1	2	3	2	11	MEDIUM
Distributed DoS (2.1 & 2.2)	3	2	3	3	3	14	HIGH
Take down Router (2.1 & 2.2)	3	2	2	3	2	12	HIGH
Take down Firewall (2.1 & 2.2)	3	2	2	3	2	12	HIGH
Turn off power to server (2.1 & 2.2)	3	2	2	3	3	13	HIGH
Filling up disk space (2.1 & 2.2)	3	1	2	2	3	11	MEDIUM
Server DoS exploit (2.1 & 2.2)	3	3	2	3	1	12	HIGH
Break RSA encryption (3.1)	2	2	1	1	1	7	LOW
Break AES encryption (3.2)	2	2	1	1	1	7	LOW
Obtain a private key (3.1 & 3.2)	3	2	2	1	1	9	MEDIUM
Man-in-the-middle attack (3.1 & 3.2)	3	1	1	1	1	7	LOW
Poor Installation/Configuration (3.1, 3.2, 4.1, 4.3)	3	2	2	1	1	9	MEDIUM
Software vulnerability (3.1, 3.2, 4.1, 4.3)	3	3	3	3	2	14	HIGH
Modify software distribution (3.1 & 3.2)	3	1	1	1	1	7	LOW
Library vulnerability (4.2 & 4.4)	3	2	2	2	2	11	MEDIUM
Bypass authentication (4.3)	3	2	2	2	2	11	MEDIUM
Eavesdrop session (4.4)	3	1	2	1	1	8	MEDIUM
Hijack session (4.4)	3	1	1	1	1	7	LOW
Brute Force password (5)	3	2	2	2	2	11	MEDIUM
Guess password (5)	3	1	2	2	1	9	MEDIUM
Observe password being entered (5)	3	1	1	1	1	7	LOW
Social Engineer a password (5)	3	1	1	1	1	7	LOW
Crack password file (5)	3	2	2	3	3	13	HIGH
Introduce malicious software (5)	3	2	3	2	2	12	HIGH

Two areas that impact the rating system above would be Motivation and Capability. These areas would be useful when attempting to model an adversary. When modeling adversaries different DREAD tables would have to be used that included these areas.

Mitigating the Threats

The final step is to determine what to do about these issues. There are four options typically taken.

1. Do nothing and accept the risk of this attack happening. In some situations this is actually the only possible option. For instance, breaking AES encryption is highly unlikely although it is considered an attack.
2. Simply warn the user that the threat exists. For instance, possible social engineering attacks can be partially mitigated by educating the user about such attack methods.

3. Remove the problem. For instance, if a particular service has a security issue then perhaps a patch would remove it or if no patch exists simply remove the service from use or replace it with a secure version.
4. Fix the problem using technology or an additional process.

The next section gives an overview of threat mitigation techniques and technologies.

High DREAD Rating

Attack (Threats)	**Mitigation**
Open source investigation (1)	User education on the operational risks of posting questions to listserves, newsgroups, and question boards; operating personal websites and employee resumes.
Enumerate network structure (1)	Minimize data stored in external DNS servers. By using split DNS servers only information required to be external would be accessible to attackers. Both a perimeter router and outer firewall will assist with this type of enumeration.
Enumerate Router Access Controls (1)	Primarily uses ICMP Destination Unreachable and Time Exceeded messages to provide attackers with access control information but these message types are required to maintain internetwork health.
Enumerate Firewall Access Controls (1)	Outer and Host firewalls can assist to protect against this.
Enumerate Hosts (1)	Denying ICMP Echo Requests, Destination Unreachable and Time Exceeded. Also grsecurity provides protection against OS fingerprinting.
SYN Floods (2.1 & 2.2)	Provide protection against common attacks at perimeter with routers and firewalls. Enable tcp_syncookies on server.
Crash service (2.1 & 2.2)	Patch maintenance and maintain knowledge on security issues regarding this product. Provide protection against common attacks at perimeter with routers and firewalls. By minimizing the number of public services available to the network there are fewer targets.
Distributed DoS (2.1 & 2.2)	Provide protection against common attacks at perimeter with routers and firewalls.
Take down Router (2.1 & 2.2)	Patch maintenance and maintain knowledge on security issues regarding this product.
Take down Firewall (2.1 & 2.2)	Patch maintenance and maintain knowledge on security issues regarding this product.
Turn off power to server (2.1 & 2.2)	Physical security only allowing authorized personnel into the server room. Use of a UPS in case of a power failure.
Server DoS exploit (2.1 & 2.2)	Patch maintenance and maintain knowledge on security issues regarding this product. Provide protection against common attacks at perimeter with routers and firewalls.
Software vulnerability (3.1, 3.2, 4.1, 4.3)	Patch maintenance and maintain knowledge on security issues regarding this product. PaX (non-executable memory pages and full address space layout randomization) provides protection against stack overflows and other common attacks. Using RBAC on server to ensure a compromised users account follows the principal of least privilege. Provide additional auditing of services to look for problem areas and error messages.
Crack password file (5)	User education on how to create strong passwords and protect the password file. Using RBAC on server to ensure a compromised users account follows the principal of least privilege.

Introduce malicious software (5)	User education on how to avoid malware. Anti-virus software deployed on all service gateways and kept up to date. Using RBAC on server to ensure a compromised users account follows the principal of least privilege. Provide additional auditing of services to look for problem areas and error messages.

Medium DREAD Rating

Attack (Threats)	**Mitigation**
Intercept traffic from client to server with no delivery (2.1 & 2.2)	Use encryption to protect the network traffic.
Filling up disk space (2.1 & 2.2)	Proper system administration and maintain knowledge on security issues regarding the operating system and software used.
Obtain a private key (3.1 & 3.2)	Educate or force users to protect private keys with passwords. Using RBAC on server to ensure a compromised users account follows the principal of least privilege.
Poor Installation/Configuration (3.1, 3.2, 4.1, 4.3)	Proper system administration and maintain knowledge on security issues regarding the operating system and software used. Security audits need to be performed on a weekly basis to ensure no new vulnerabilities have been introduced to the network as packages are updated and changes to the network are made. Using RBAC on server to ensure a compromised users account follows the principal of least privilege.
Library vulnerability (4.2 & 4.4)	Patch maintenance and maintain knowledge on security issues regarding this product or software. PaX (non-executable memory pages and full address space layout randomization) provides protection against stack overflows and other common attacks. Using RBAC on server to ensure a compromised users account follows the principal of least privilege. Provide additional auditing of services to look for problem areas and error messages.
Bypass authentication (4.3)	Use two-factor authentication when possible. Using RBAC on server to ensure a compromised users account follows the principal of least privilege.
Eavesdrop session (4.4)	Use encryption to protect the network traffic. Using RBAC on server to ensure a compromised users account follows the principal of least privilege.
Brute Force password (5)	User education on how to create strong passwords and protect the password file. Auditing of applications for failed logins. Using RBAC on server to ensure a compromised users account follows the principal of least privilege.
Guess password (5)	User education on how to create strong passwords and protect the password file. Using RBAC on server to ensure a compromised users account follows the principal of least privilege.

Low DREAD Rating

Attack (Threats)	**Mitigation**
Break RSA encryption (3.1)	Building in a Defense-in-Depth network design strategy will minimize damage if successful. Not much else to do.
Break AES encryption (3.2)	Building in a Defense-in-Depth network design strategy will minimize damage if successful. Not much else to do.

Man-in-the-middle attack (3.1 & 3.2)	Use encryption to protect the network traffic. Use firewalls/TCP wrappers to determine who can use services if it can be locked down. Using RBAC on server to ensure a compromised users account follows the principal of least privilege.
Modify software distribution (3.1 & 3.2)	All software should have its integrity verified preferably with a PGP signature of the software package along with independent verification of the signers PGP key. Md5 checksums can be easily forged the same time an attacker trojans the original distribution on the hacked site. Testing software packages before deployment to production systems can help with this issue. Security audits need to be performed on a weekly basis on test and production networks to ensure no new vulnerabilities have been introduced to the network as packages are updated.
Hijack session (4.4)	Use encryption to protect the network traffic. Use firewalls/TCP wrappers to determine who can use services if it can be locked down. Using RBAC on server to ensure a compromised users account follows the principal of least privilege.
Observe password being entered (5)	User education on operational risks. Using RBAC on server to ensure a compromised users account follows the principal of least privilege.
Social Engineer a password (5)	User education on operational risks and social engineering. Using RBAC on server to ensure a compromised users account follows the principal of least privilege, which can minimize damage.

Grsecurity

Grsecurity is a suite of kernel patches for Linux that started as a port of the Openwall kernel patch to version 2.4. The author, Bradley Spengler, then added other functionality in an attempt to meet four goals [4]:

- Configuration-free operation
- Complete protection against all forms of address space modification bugs
- Feature-rich ACL and auditing systems
- Operation on multiple processor architectures and operating systems

All of this is implemented with low overhead to the processor.

Role Based Access Control (RBAC)

The basic concept of RBAC is that access control is determined by assigned roles that users take on as part of an organization. When a user is then associated with this role they are given only the privileges required to do their job. This supports the concept of least privilege which requires identifying the user's job functions, determining the minimum set of privileges required to perform that function, and restricting the user to a domain with those privileges and nothing more. [5]

In this implementation, roles are split into users, groups and special roles. Users and groups are tied to specific user or group IDs. Special roles traverse the traditional user or group ID system and could be used to create several administrative roles that would cause root privileges to be split up. [6]

This is an extension to the original Access Control Lists (ACL) by adding role support. ACL can be provided for resources, sockets, files and processes.

This feature assists with mitigating the following threats:

- Software vulnerability
- Introduce malicious software
- Obtain a private key
- Library vulnerability
- Bypass authentication
- Eavesdrop session
- Brute Force password
- Guess password
- Man-in-the-middle attack
- Hijack session
- Observe password being entered
- Social Engineer a password

PaX Address Space Protection

Provides non-executable memory pages and full address space layout randomization (ASLR) for a wide variety of architectures. [7] Its purpose is to provide exploit prevention and containment that focuses on three attack methods: [8]

1. Introduction and execution of arbitrary code (e.g. stack/heap-based buffer overflows and shellcode injection)
2. Execution of existing code out of original program order (e.g. return-to-libc method)
3. Execution of existing code in original program order with arbitrary data (e.g. user credentials, access rights, etc.)

PaX does this by implementing the following:

1. NOEXEC – its goal is to prevent the injection and execution of code into a task's address space. [8] Includes both the non-executable page feature and mmap/mprotect restrictions. More in-depth information can be found at http://pax.grsecurity.net/docs/noexec.txt.
2. Address Space Layout Randomization (ASLR) – its goal is to introduce randomness into addresses used by a given task. [8] More in-depth information can be found at http://pax.grsecurity.net/docs/aslr.txt.

Please review http://pax.grsecurity.net to see the latest in information regarding the PaX project.

This feature assists with mitigating the following threats:

- Software vulnerability
- Library vulnerability

Limitations: Fedora Core distribution does not natively support the PT_PAX_FLAGS header marking and instead uses EI_PAX as documented later. It also does not have ET_DYN ELF executables and instead uses the ET_EXEC type. To implement this it would require essentially a recompilation of the entire distribution. Those interested in working with distributions that support these are encouraged to take a look at the Adamantix (http://www.adamantix.org) and Hardened Gentoo projects (http://www.gentoo.org/proj/en/hardened/).

Filesystem Protection

Provides the ability to restrict the /proc filesystem, a source of information, to only provide process information to its owner. Restrictions can be placed on symlinks (symbolic links) and FIFOs(data structure for buffering a data stream). Chroots is making the root directories become something other than its default, If chroots are used they can be strengthened by making syscalls unrelated to the filesystem chroot-aware (e.g. denying double-chroots, pivot_root, fchdir outside of chroot and mounting), enforcing chdir ("/") upon chroot and fixing other ways chroots can be broken.

This feature assists with mitigating the following threats:

- Software vulnerability
- Library vulnerability

Kernel Auditing

Provides the ability to increase the amount of auditing provided by the kernel. Some of the auditing events include exec, signals, failed forks, IPC creation and removal, chdirs and mount/unmounts.

This feature assists with mitigating the following threats:

- Crash service
- Software vulnerability
- Introduce malicious software
- Library vulnerability

Executable Protection

Provides features to assist with the protection of executables and keeps non-users from being able to view dmesg and the log buffer. Enables randomized Process IDs and /proc restrictions so attackers would have to guess these.

This feature assists with mitigating the following threats:

- Software vulnerability
- Library vulnerability

Network Protection

Provides randomness features from OpenBSD to minimize prediction-based network attacks including random IP IDs (Identification), RPC XIDs, TCP Initial Sequence

Numbers, and altered Ping IDs. These features also hinder OS fingerprinting and keep the machine from being a bounce point in a blind port scan by causing its IP ID scheme to be unpredictable.

This feature assists with mitigating the following threats:

- Enumerate network structure
- Enumerate Hosts

Perimeter Protections

This server will be in the DMZ of GIAC Enterprises and will sit behind a border router as well as an external firewall. Each one of these devices provides some protection against attacks on the new server.

Border Router

The border router is the first line of defense for the network. It uses access control lists (ACL) to filter traffic that crosses it, which can provide a basic method for eliminating unwanted traffic. It provides the following protections for this server:

- Prevents IP source routing
- Restricts the use of subnet zero addresses
- Disables 'Host Unreachable' ICMP messages (type 3)
- Disables 'Redirect' ICMP messages (type 5)
- Disables Ad-hoc routing
- Denies spoofed and non-routable addresses
- Denies outgoing time-exceeded and unreachable ICMP messages

This device assists with mitigating the following threats:

- Enumeration of network
- Enumerate Router access controls
- Enumerate Firewall access controls
- Enumerate Hosts
- Distributed DoS

External Firewall

The external firewall is the second line of defense for the network. It uses iptables for its firewall and provides the following protections for this server:

- Disables response to Broadcasts to keep it from being SMURF amplifier.
- Enables anti-spoofing support
- Disables source routing
- Disables ICMP redirect support (type 5)
- Provides ability to blacklist IP addresses of known threats.

This device assists with mitigating the following threats:

- Enumeration of network
- Enumerate Router access controls

- Enumerate Firewall access controls
- Enumerate Hosts
- SYN Floods
- Crash service
- Distributed DoS
- Server DoS exploit

User Education

Unfortunately security cannot be solved by technical solutions alone. A comprehensive user education program that keeps users up-to-date on the latest threats to the network can largely protect against the threats listed below. This is especially important for those users that work from their home network. This would assist with mitigating the following threats:

- Open source investigation
- Crack password file
- Introduce malicious software
- Obtain a private key
- Brute force password
- Guess password
- Observe password being entered
- Social Engineer a password

System Administration Function

This function includes all ongoing functions performed by the system administrators. This includes software patching and updating, router firmware updating and ACLs modifications, firewall ruleset modification, virus updates, periodic vulnerability scans, data backups and keeping up on the latest threats. This would assist with mitigating the following threats:

- Crash service
- Take down Router
- Take down Firewall
- Server DoS exploit
- Software vulnerability
- Introduce malicious software
- Filling up disk space
- Poor Installation/Configuration
- Library vulnerability
- Modify software distribution

Network Communication Encryption

Encryption will be used to protect network communication. This will assist with mitigating the following threats:

- Intercept traffic from client to server with no delivery
- Eavesdrop session

- Man-in-the-middle attack
- Hijack session

Physical Server Room

GIAC Enterprises has upgraded to a 20x30 server room in the building of Boulder, Colorado's largest ISP. The server room includes raised flooring, environmental controls, well ventilated locking server racks, fire control system, Uninterruptible Power Supplies, Voltage spike filters and a backup generator. He has also upgraded to a T3 connection and will connect directly to the ISP router via Ethernet. The ISP is a tier one Internet provider offering a service level agreement with guaranteed 99.5% uptime. In addition, the office space has its own physical security system, card access control system, fire detection and control systems, and security guards. This would assist with mitigating the following threats:

- Take down Router
- Take down Firewall
- Turn off power to server

Server Installation and Hardening Process

Fedora CORE 1 OS Installation

1. Boot up machine and install Fedora Core 1 disk 1
2. When startup screen comes up, type linux text to install.
3. Skip the CD media test unless needed
4. Welcome to Fedora Core screen - <next>
5. Language Selection - English - <next>
6. Keyboard Configuration - U.S.English - <next>
7. Monitor Configuration - Generic LCD Panel 1024x768 - <next>
8. Installation Type - Server
9. Disk Partitioning Setup – Manually partition with Disk Druid - <next>
    a. Select New button to add partition
        i. Mount point: /boot
        ii. File System Type: ext3
        iii. Allowable Drives: sda
        iv. Size (MB): 120
        v. Additional Size Options: Fixed size
        vi. Check Force to be a primary partition
        vii. <ok>
    b. Select New button to add another partition
        i. Mount point: / (this will become blank when file system is selected)
        ii. File System Type: swap
        iii. Allowable Drives: sda
        iv. Size (MB): 1024
        v. Additional Size Options: Fixed size
        vi. Check Force to be a primary partition
        vii. <ok>
    c. Select New button to add another partition
        i. Mount point: /
        ii. File System Type: ext3
        iii. Allowable Drives: sda
        iv. Size (MB): Leave default. Will change.
        v. Additional Size Options: Fill to maximum allowable size
        vi. Check Force to be a primary partition
        vii. <ok>
    d. <next>
10. Boot Loader Configuration
    a. Use GRUB Boot Loader
    b. Check Use a GRUB Password and enter password
    c. <next>
11. Network Configuration
    a. Edit Network Devices, Unclick DHCP and add IP address

b. Hostname, add manually
c. Add Miscellaneous Settings (Gateway, DNS servers) - <next>

12. Firewall Configuration - No firewall (one will be added later) - <next>
    a. (click Proceed at Warning - No Firewall message)
13. Additional Language Support - <next>
14. Time Zone Selection - America/Denver - <next>
15. Set Root Password - <next>
16. Package Group Selection
    a. Select minimal - <next>
17. Reboot

Remove unnecessary packages

Use “rpm –e <package>” to remove the following packages in this order.

Package	Reason for removal
ash-0.3.8-15	Smaller version of Bourne shell. (Not needed)
system-config-mouse-1.1.2-1	Mouse configuration. (Not needed)
up2date-4.1.16-1	Using Yum for package management. (Not needed)
system-config-network-tui-1.3.10-1	Network configuration. (Not needed)
rhpl-0.121-1	Library of python code used by programs.
pyxf86config-0.3.12-1	Python wrappers for the Xserver config file library. Not running Xfree86. (Not needed)
anacron-2.3-29	Runs timed jobs lost during downtime. (Not needed)
aspell-0.50.3-16	Spell checker. This is a server. Will have to use –nodeps to remove due to circular dependencies. (Not needed)
aspell-en-0.51-6	Spell Checker. This is a server. (Not needed)
wireless-tools-26-1	No wireless hardware. (Not needed)
autofs-3.1.7-42	Automatically mounts and unmounts filesystems. (Not needed)
ypbind-1.12-3	NIS daemon. Will have to use –nodeps to remove due to circular dependencies. (Not needed)
yp-tools-2.8.2	NIS client. (Not needed)
wvdial-1.53-12	Dialup software. No modem. (Not needed)
rp-pppoe-3.5-8	PPP over Ethernet client. (Not needed)
ppp-2.4.1-15	Point-to-point protocol daemon. (Not needed)
irda-utils-0.9.15-1.1	IRDA utilities. (Not needed)
nano-1.2.1-3	Tiny console text editor. (Not needed)
ftp-0.17-18	FTP client. Only use SSH. (Not needed)
stunnel-4.04-6	SSL-encrypting socket wrapper. Only use SSH. (Not needed)
isdn4k-utils-3.2-5.p1	ISDN utilities. (Not needed)

at-3.1.8-46.1	Time oriented job control program. Using cron. (Not needed)
rsh-0.17-19	Rsh client. Only use SSH. (Not needed)
talk-0.17-21	Talk client. (Not needed)
telnet-0.17-26.2	Telnet client. Only use SSH. (Not needed)
sendmail-8.12.10-1.1.1	Sendmail MTA mail service. (Not needed)
procmail-3.22-11	Mail processing program used by Sendmail. (Not needed)
rdist-6.1.5-30.1	Remote file distribution. (Not needed)
rsync-2.5.6-19	Client for synchronizing files over a network. (Not needed)
minicom-2.00.0-17	Terminal program. (Not needed)
bc-1.06-15.1	A numeric processing language. (Not needed)
lftp-2.6.5-4	Commandline ftp client. Only use SSH. (Not needed)
tcsh-6.12-5	Enhanced version of the C shell. (Not needed)
apmd-3.0.2-20	Advanced Power Management daemon utilities. (Not needed)
kernel-pcmcia-cs-3.1.31-13	PCMCIA modules. (Not needed)
system-config-securitylevel-tui-1.2.11-1	Security level configuration during install. (Not needed)
ed-0.2-34	Old text editor. (Not needed)
finger-0.17-18.1	Finger client. (Not needed)
reiserfs-utils-3.6.8-1.1	Utilities for reiser FS. (Not needed)
nfs-utils-1.0.6-1	Utilities for NFS. (Not needed)
jfsutils-1.1.3-1	Utilities for JFS. (Not needed)
portmap-4.0-57	Manages RPC connections. (Not needed)
mkbootdisk-1.5.1-1	Utility for making a boot disk. (Not needed)
dosfstools-2.8-11	Utilities for making and checking MS-DOS FAT file systems. (Not needed)
authconfig-4.3.801	Sets NIS and shadow passwords at install. (Not needed)
ntsysv-1.3.9-1	Sets symbolic links in /etc/rc.d directory at install. (Not needed)
setuptool-1.13-2	Sets text mode menu utility for install. (Not needed)
pam_krb5-2.0.5-1	Tie between PAM and kerberos. (Not needed)
krbafs-1.2.2-1	Tie between AFS and kerberos. (Not needed)
eject-2.0.13-3	Software used to eject removable media. (Not needed)
hotplug-2003_08_05-1	Helper application for loading modules for USB devices. (Not needed)
mailcap-2.1.14-1.1	Associates helper applications with file types. (Not needed)
fedora-logos-1.1.20-1	Contains various logo graphics. (Not needed)
setserial-2.17-13	Utility for setting serial port. (Not needed)
time-1.7-22	Utility that collects resource information on programs while executing. (Not needed)
nscd-2.3.2-101.4	Caches name service lookups for NIS+. (Not needed)
nss_ldap-207-6	LDAP access clients. (Not needed)

pam_smb-1.1.7-2	PAM module for use with SMB servers. (Not needed)
fbset-2.1-14	Tools for managing a frame buffer's video mode. (Not needed)
statserial-1.1-33	Helps debug serial lines. (Not needed)
gettext-0.12.1-1	Utilities for producing multi-lingual messages. (Not needed)
libgcj-3.3.2-1	Java Class library. (Not needed)
dhclient-3.0pl2-6.16	DHCP client. (Not needed)
vconfig-1.8-1	802.1q VLAN configuration utility. (Not needed)
lha-1.14i-12.2	Archiving and compression utility for LHarc format. (Not needed)
slocate-2.7-4	Finds files on a system via central database. (Not needed)
setarch-1.0-1	Architecture personality setter. (Not needed)
rhnlib-1.4-1	Collection of python modules used by Red Hat Network. (Not needed)
pyOpenSSL-0.5.1-11	Python wrapper module around OpenSSL library. (Not needed)
mt-st-0.7-12.1	Programs to control tape device operations. (Not needed)
logwatch-4.3.2-2.1	Summarizes the logs for admin. Function is done on central log server.
dump-0.4b34-1	Programs for backing up and restoring filesystems. (Not needed)

Remove unneeded services and verify services left

1. Stop the following services using "chkconfig --level 0123456 <service> off".
    a. kudzu - Recognizes hardware at startup. Is not needed after initial install unless additional hardware is added and this service can be run manually if needed.
    b. smartd - Script that starts the Self Monitoring and Reporting Technology (SMART) Daemon.
2. Ensure the following services are on using "chkconfig --level 2345 <service> on".
    a. syslog - Needed for system logging.
    b. sshd - Provides for secure shell services.
    c. acpid - Starts the acpi daemon which listens and dispatches ACPI events from the kernel.
    d. crond - Handles background/timed job scheduling.
    e. network - Starts networking up.
    f. random - Seeds /dev/random. Deals with random number generation.
    g. yum - Automates yum, a program updater.
3. Remove the following unnecessary scripts using "chkconfig --del <service>.
    a. irqbalance - Distributes interrupts across the CPUs on a multiprocessor system with purpose of spreading the load.
    b. netfs - Mount network filesystems.
    c. saslauthd - Server process which handles plaintext authentication.
    d. microcode_ctl - Script to apply CPU microcode.
    e. rawdevices - Script assigns raw devices to block devices.
    f. gpm - Mouse support to text-based linux applications.

g. iptables – Default firewall that will be replaced below.

Cleanup unnecessary files

When the packages above were removed, rpmsave extensions are used to save the old configuration information. Do the following to search for and remove all of these files:

```
[root@hermes]# find / -iname "*.rpmsave" | xargs rm –f
```

Below are files and directories left over from packages that have been removed or unnecessary services that have been removed. These files need to be removed manually using either "rm –f" for a single file or "rm –rf" for an entire directory.

Files	Reason for removal
/etc/exports	Defines remote mount points for NFS.
/etc/printcap	Printer capability database.
/etc/lilo.conf	Not using lilo for bootloader.
/etc/krb5.conf	Not using kerberos.
/etc/updfstab.conf /etc/updfstab.conf.default	Used with hotplug.
/etc/csh.cshrc /etc/csh.login /root/.tcshrc /root/.cshrc	Configuration files for C shell that has been removed.
/root/.Xresources /usr/bin/X11 /usr/lib/X11	Removed X11 package
/usr/bin/chfn	Changes finger information. Removed finger service

Directories	Reason for removal
/etc/xinetd.d/	Not using xinetd.
/etc/hotplug/	Removed hotplug package.
/etc/ppp/	Removed ppp package.
/etc/pcmcia/	Removed pcmcia package.
/etc/X11/ /usr/X11R6/	Removed X11 package
/var/nis/ /var/yp/	Not using NIS/NIS+.
/var/spool/anacron/	Removed anacron.
/var/spool/clientmqueue/	Removed sendmail.
/var/spool/lpd/	Removed printing capability.
/var/lib/dhcp/	Removed dhcp client.
/var/lib/games/	Removed games.
/var/lib/nfs/	Removed NFS support.

Package Repository

Due to the amount of servers running Fedora Core, GIAC Enterprises has installed a central RPM package repository that updates twice daily from http://download.fedora.redhat.com. The repository also performs a gpg signature check on all packages pulled from the source to minimize integrity issues. The test network built inside of VMware then applies the updates and the network machines are reviewed for signs of any problems. In addition, the administrators stay current on the latest security issues relating to the OS. Such issues could cause this process to need special attention or patch timeline to be hastened.

Remove Unnecessary Users and Groups

A default installation has many users and groups that are created but not needed. The fewer users in /etc/passwd and /etc/groups the greater chance of noticing a new entry.

Before these users and groups are deleted the following files need to be removed. These files are currently owned by the soon-to-be removed users or groups and should be deleted to alleviate un-owned files.

Remove files associated with group lp

```
[root@hermes]# find / -group lp | xargs rm –f
```

Remove files associated with group uucp

```
[root@hermes]# find / -group uucp | xargs rm –f
```

Remove files associated with group floppy

```
[root@hermes]# find / -group floppy | xargs rm –f
```

Now the unneeded users and groups can be removed.

```
[root@hermes]# for user in lp sync shutdown halt news uucp operator;
    do /usr/sbin/userdel $user;
done

[root@hermes]# for user in ftp gopher games nscd rpc mailnull smmsp;
    do /usr/sbin/userdel $user;
done

[root@hermes]# for group in lp news uucp games dip floppy;
    do /usr/sbin/groupdel $group;
done
```

Auditing for SUID and GUID programs

Programs that use SUID (Set UID) and SGID (Set GID) can be dangerous since they give the user the privilege of the owner or group of that program upon execution. This only modifies privileges within that process but this could be a problem if that program

was exploited and a shell was accessed. For this server the administrator will be the only authorized user for most utilities so these can have the SUID or SGID removed.

To locate SUID programs:

```
[root@hermes]# find / -type f –perm +4000 -ls
```

Which resulted in the following programs that need the SUID removed.

```
[root@hermes]# chmod a-s <file>
```

/usr/sbin/usernetctl – allows a user to manipulate a network interface
/usr/sbin/userhelper – A helper interface to pam
/usr/bin/chage – change user password expiry information
/usr/bin/gpasswd – administers the /etc/group file
/usr/bin/chsh – changes your login shell
/usr/bin/crontab – maintains crontab files for users
/bin/ping6 - send ICMP ECHO_REQUEST to IPv6 network hosts
/bin/traceroute – print the route packets take to network host
/bin/traceroute6 – print the route packets take to IPv6 network host
/bin/mount – mount a file system
/bin/umount – unmount a file system
/sbin/pam_timestamp_check – authenticate using cached successful authentication attempts
/sbin/pwdb_chkpwd – check the password of the invoking user
/sbin/unix_chkpwd – check the password of the invoking user

The remainder files are necessary for users and should be left SUID.

/bin/su – run a shell with substitute user and group IDs
/usr/bin/sudo – execute a command as another user
/usr/bin/newgrp – login to a new group
/usr/bin/passwd – updates user password
/usr/libexec/openssh/ssh-keysign – creates keys for ssh
/bin/ping – send ICMP ECHO_REQUEST to network hosts

To locate SGID programs:

```
[root@hermes]# find / -type f –perm +2000 -ls
```

Which resulted in the following programs that need the SGID removed.

```
[root@hermes]# chmod a-s <file>
```

/usr/sbin/wall – send a message to everyone’s terminal
/usr/sbin/write – send a message to another user
/sbin/netreport – request notification of network interface change

The remainder files are necessary for operations and should be left SGID.

/usr/sbin/lockdev – manage device lockfiles
/usr/sbin/utempter – utility which allows non-privileged applications to modify utmp database without having to be SUID

Locate files with no User or Group Owner

This searches the filesystem for files with no user or group owner. There should be no files found so investigate further should you find any.

```
[root@hermes]# find / -nouser –o -nogroup
```

Use of TCPwrappers

Although this server doesn't use TCPwrappers, this version of SSH was compiled with the libwrap library. This causes the SSH daemon to make access checks with /etc/hosts.allow and /etc/hosts.deny. Since libwrap checks the hosts.allow file first and then hosts.deny the following setup will be used.

/etc/hosts.allow

```
# hosts.allow
# network access control for programs invoked by tcpd or
# using libwrap. See hosts_access(5) and hosts_options(5).

# Only allow access to SSHd from security subnet.

SSHD : 192.168.24.0/255.255.255.0
```

/etc/hosts.deny

```
# hosts.deny
# network access control for programs invoked by tcpd or
# using libwrap. See hosts_access(5) and hosts_options(5).

# Deny everyone not accepted by hosts.allow
ALL : ALL
```

NOTE: This is in addition to the Host Firewall found below. This would add another layer of protection should the firewall fail or becomes misconfigured in the future.

Host Firewall

A custom firewall script will be used as a drop in replacement for the iptables init script found in /etc/init.d. It has the filename of "netfilter" to differentiate it from the original iptables name and can be found in Appendix A. This will start up the host firewall and

will require no maintenance unless service configuration or host configuration changes. To add the firewall script to the initialization directories do the following:

```
[root@hermes]# chkconfig –-add netfilter
```

BIOS Security

As another layer of security against unauthorized users with physical access, use a password to protect your BIOS. By doing this you can ensure that the hardware will only allow access the chosen internal storage device (usually hard drive) to boot from. This will keep an attacker from changing the boot sequence to a CDROM, USB device or floppy.

Single user mode

Single user mode should be password protected to keep users from accessing it at the command line (init 1). Add the following to /etc/inittab to password protect single-user mode.

```
#password protect single-user mode
~~:S:wait:/sbin/sulogin
```

GRUB configuration

Grub should be configured to only boot the kernel supporting grsecurity. If another kernel was to be booted that did not support it the Role Based Access Controls could be easily bypassed. Grub should also be password protected to keep users from adding in parameters.

Rebooting the system

Make rebooting more difficult by disabling ctrl-alt-del. Remove the following from the /etc/inittab file:

```
ca::ctrlaltdel:/sbin/shutdown –t3 –r now
```

Banners

Login banners will be used on all services on this server. The company has a standard banner that addresses the following five points [9]:

- Access to the system is limited to authorized activity.
- Unauthorized access and modification is prohibited.
- Unauthorized user may face criminal or civil penalties.
- Use of the system may be monitored and recorded.
- Law enforcement may be notified if monitoring reveals criminal activity.

“WARNING! Unauthorized use of this system is strictly prohibited and may be subject to criminal prosecution or employee discipline. Authorized personnel may monitor any

activity or communication on the system and may retrieve any information stored within the system. By accessing and using this computer, you are consenting to such monitoring and information retrieval for law enforcement and other purposes. Users should have no expectation of privacy as to any communication on or information stored within the system."

This banner is put in every banner file including:
/etc/issue – displayed at the system login prompt.
/etc/issue.net – used for our SSH server.
/usr/local/etc/ircd.motd – this file is displayed on connection to the IRC server.

Recompile kernel with grsecurity patch

The kernel will need to be recompiled to include the grsecurity patch in order to provide the additional security components listed above. In order to do this, the compiler and the development environment will need to be installed.

Compiler and development environment

In order to install the compiler and development environment the following packages need to be installed in this order due to dependencies:

1. glibc-kernheaders-2.4-8.36.i386.rpm (Disk 2)
2. glibc-headers-2.3.2-101.i386.rpm (Disk 2)
3. glibc-devel-2.3.2-101.i386.rpm (Disk 2)
4. binutils-2.14.90.0.6-3.i386.rpm (Disk 2)
5. cpp-3.3.2-1.i386.rpm (Disk 1)
6. gcc-3.3.2-1.i386.rpm (Disk 2)
7. patch-2.5.4-18.i386.rpm (Disk 1)
8. flex-2.5.4a-30.i386.rpm (Disk 1)
9. bison-1.875-5.i386.rpm (Disk 1)
10. m4-1.4.1-14.i386.rpm (Disk 1)

To remove these files just reverse the order. This environment should be removed after the kernel and software is compiled, configured and tested.

Get new kernel

Download linux-2.6.7.tar.bz2 and linux-2.6.7.tar.bz2.sign from http://www.kernel.org.

Obtain the PGP key from http://www.kernel.org/signature.html. This will be needed to import into your gpg.

```
[root@hermes]# gpg --import kernel_pgpkey
gpg: /root/.gnupg/trustdb.gpg: trustdb created
gpg: key 517D0F0E: public key "Linux Kernel Archives Verification Key
<ftpadmin@kernel.org>" imported
gpg: Total number processed: 1
gpg:               imported: 1

[root@hermes]# gpg –fingerprint 517D0F0E
pub  1024D/517D0F0E 2000/10/10 Linux Kernel Archives Verification Key
<ftpadmin@kernel.org>
```

```
     Key fingerprint = C75D C40A 11D7 AF88 9981  ED5B C86B A06A 517D 0F0E
                               Linux Kernel Archives Verification Key
<ftpadmin@kernel.org>
sub  4096g/E50A8F2A 2000-10-10
```

Remember, the issue with using MD5 checksums is that unless you have obtained the checksum from an independent source, there is no way of knowing if an attacker forged a new checksum at the same time they trojaned the software package at the hacked website. As the above public key was pulled from the same website that the software package was downloaded from, independent verification of the signers key needs to be obtained. Most developers publish their public key on key servers so independent verification can be done.

By going to http://pgp.mit.edu (or any other PGP key server) you can do a search for the key ID of 0x517D0F0E (remember the “0x”) and you would produce these results.

Public Key Server -- Index ``0x517D0F0E ''

```
Type bits /keyID    Date       User ID
pub  1024D/517D0F0E 2000/10/10 Linux Kernel Archives Verification Key
<ftpadmin@kernel.org>
     Key fingerprint = C75D C40A 11D7 AF88 9981  ED5B C86B A06A 517D 0F0E
                               Linux Kernel Archives Verification Key
<ftpadmin@kernel.org>
```

This search provides the same user ID and fingerprint of the imported key above so independent verification is successful.

Now verify the kernel source.

```
[root@hermes]# gpg --verify linux-2.6.7.tar.bz2.sign linux-2.6.7.tar.bz2
gpg: Signature made Wed 16 Jun 2004 02:22:25 AM EDT using DSA key ID
517D0F0E
gpg: Good signature from "Linux Kernel Archives Verification Key
<ftpadmin@kernel.org>"
gpg: checking the trustdb
gpg: no ultimately trusted keys found
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the
owner.
Primary key fingerprint: C75D C40A 11D7 AF88 9981  ED5B C86B A06A 517D 0F0E
```

This ouput from GnuPG shows that:

- The software package was signed on June 16, 2004 by a key with the ID 517D0F0E and the fingerprint “C75D C40A 11D7 AF88 9981 ED5B C86B A06A 517D 0F0E” (which matches our independent verified key above)
- The owner of this key claims to be “Linux Kernel Archives Verification Key <ftpadmin@kernel.org>”
- GnuPG does not know whether this key really belongs to this owner. This is a common message unless a trusted path to this owner has been built.

To build a trusted path to this owner you will have to do one of the following:

- Physically verify the identification of the key owner and accept their key.
- Find someone you trust that will verify the identification and sign the key with his/her own key.

As building a trusted path is not simple, most administrators accept this risk if independent verification of public key is successful.

Unpack the new kernel using gzip and tar under /usr/src/linux-2.6.7.

Kernel information requirements

This information is needed during the configuration process while it is decided which hardware will be supported under our new kernel. The information gathered includes: Processor, Drive type and Controller (SCSI, IDE), Ethernet devices, Graphics and Sound Cards, USB HUB.

Start by running the following utility to print information about the hardware:

```
[root@hermes]# /sbin/lspci
```

The following command will give out CPU information:

```
[root@hermes]# cat /proc/cpuinfo
```

Patch with grsecurity

Obtain the grsecurity patch from http://grsecurity.net/download.php. Be sure to include the file's GPG signature and the GPG key used to sign the packages. The following files will be needed:

- grsecurity-2.0.1-2.6.7.patch & grsecurity-2.0.1-2.6.7.patch.sign
- gradm-2.0.1.tar.gz & gradm-2.0.1.tar.gz.sign
- spender-gpg-key.asc

First import the gpg key

```
[root@hermes]# gpg --import spender-gpg-key.asc
gpg: key 4245D46A: public key "Bradley Spengler (spender)
<spender@grsecurity.net>" imported
gpg: Total number processed: 1
gpg:               imported: 1

[root@hermes]# gpg –-fingerprint 4245D46A
pub  1024D/4245D46A 2002/12/02 Bradley Spengler (spender)
<spender@grsecurity.net>
     Key fingerprint = 9F74 393D 7E7F FF3C 6500  E778 9879 B649 4245 D46A
                                 Bradley Spengler (spender)
<spender@grsecurity.net>
sub  2048g/271E4404 2002-12-02
```

For independent key verification, http://pgp.mit.edu yielded:

```
Public Key Server -- Index "0x4245D46A "
```

```
Type bits /keyID    Date       User ID
pub  1024D/4245D46A 2002/12/02 Bradley Spengler (spender)
<spender@grsecurity.net>
     Key fingerprint = 9F74 393D 7E7F FF3C 6500  E778 9879 B649 4245 D46A
                               Bradley Spengler (spender)
<spender@grsecurity.net>
```

Then verify each file as follows:

```
[root@hermes]# gpg --verify grsecurity-2.0.1-2.6.7.patch.sign grsecury-
2.0.1-2.6.7.patch
gpg: Signature made Sun 08 Aug 2004 06:33:14 AM EDT using DSA key ID
4245D46A
gpg: Good signature from "Bradley Spengler (spender)
<spender@grsecurity.net>"
gpg: checking the trustdb
gpg: no ultimately trusted keys found
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the
owner.
Primary key fingerprint: 9F74 393D 7E7F FF3C 6500  E778 9879 B649 4245 D46A

[root@hermes]# gpg --verify gradm-2.0.1.tar.gz.sign gradm-2.0.1.tar.gz
gpg: Signature made Tue 10 Aug 2004 02:02:55 PM EDT using DSA key ID
4245D46A
gpg: Good signature from "Bradley Spengler (spender)
<spender@grsecurity.net>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the
owner.
Primary key fingerprint: 9F74 393D 7E7F FF3C 6500  E778 9879 B649 4245 D46A
```

Again an explicit trusted path was not found but the independent key verification above was successful.

Move the grsecurity-2.0.1-2.6.7.patch into the /usr/src/ directory and patch.

```
[root@hermes]# patch –p0 < grsecurity-2.0.1-2.6.7.patch
```

Verify there were no problems and the patch was applied correctly by looking for .rej files:

```
[root@hermes]# find ./ -name \*.rej
```

Make the directory /usr/src/linux-2.6.7-grsec now that the patch has been added.

```
[root@hermes]# mv /usr/src/linux-2.6.7 /usr/src/linux-2.6.7-grsec
```

Configure kernel

Goto /usr/src/linux-2.6.7-grsec and configure the kernel.

The configuration process is the most strenuous part of the kernel compilation process. In this step, you are deciding which features will be included in the final kernel and it can

require lots of hardware knowledge. For this paper, the focus will be placed on where the grsecurity patch fits in. The following resources would be helpful if help is needed with the overall kernel configuration.

- http://www.digitalhermit.com/linux/Kernel-Build-HOWTO.html
- http://www.siliconvalleyccie.com/linux-adv/kernel.htm
- http://www.bitbenderforums.com/vb22/showthread.php?postid=296104

Under Security Options you will want to proceed to the Security Options section and do the following:

- Enable different security models – Deselect
  - This option allows you to use custom security models over the default model. This includes various Socket and Networking Security Hooks and NSA SELinux Support.

Since this patch directly modifies the kernel it does not require any LSM (Linux Security Modules) component to plug-in security. The author of the patch makes some good points against LSM, which can be found at http://grsecurity.net/lsm.php.

The kernel configuration settings for grsecurity have been broken down along with comments and whether the option was selected. This can be found in Appendix B at the end of the paper. The description was pulled from the kernel configuration information and can also be found at http://grsecurity.net/confighelp.php.

Compile and install kernel

Now the kernel is ready to be compiled.

1. Backup the kernel .config information to another file in case a problem arises.
2. Create necessary include files and generate dependency information.

```
[root@hermes]# make dep
```

3. Clean up unneeded object files.

```
[root@hermes]# make clean
```

4. Start actual kernel build.

```
[root@hermes]# make bzImage
```

5. Build and install the modules.

```
[root@hermes]# make modules
[root@hermes]# make modules_install
```

6. Make an initrd file.

```
[root@hermes]# mkinitrd /boot/initrd-2.6.7-grsec.img 2.6.7-grsec
```

7. Copy kernel and system.map to the /boot directory

```
[root@hermes]# cp arch/i386/boot/bzImage /boot/bzImage-2.6.7-grsec
[root@hermes]# cp System.map /boot/System.map-2.6.7-grsec
[root@hermes]# ln –s /boot/System.map-2.6.7-grsec /boot/System.map
```

8. Configure GrUB located at /etc/grub.conf with following:

```
Title Fedora Core with grsec (2.6.7-grsec)
    root (hd0,0)
    kernel /boot/bzImage-2.6.7-grsec ro root=LABEL=/
    initrd /initrd-2.6.7-grsec.img
```

9. Reboot. This will cause two kernels to show so pick the kernel labeled “Fedora Core with grsec (2.6.7-grsec).

Install gradm-2.0.1

This is the RBAC administration program that interfaces with grsecurity. Since this file was verified above gradm-2.0.1.tar.gz needs to be unpacked and compiled.

```
[root@hermes]# make
[root@hermes]# make install
mkdir -p /sbin
/usr/bin/install -c -m 0755 gradm /sbin
/usr/bin/strip /sbin/gradm
/usr/bin/install -c -m 0700 grlearn /sbin
/usr/bin/strip /sbin/grlearn
mkdir -p -m 700 /etc/grsec
mkdir -p /usr/share/man/man8
/usr/bin/install -c -m 0644 gradm.8 /usr/share/man/man8/gradm.8
Setting up grsecurity RBAC password
Password:
Re-enter Password:
Password written to /etc/grsec/pw.
```

Usage information for this program can be found later in this document.

chpax-0.7 and paxctl-0.2

PaX uses a header marking for ELF binaries to enable control of settings on a per-binary basis. Our kernel has support for both the legacy EI_PAX and newer PT_PAX_FLAGS. Both chpax and paxctl are tools that can modify the PaX flags on a per-binary basis. The tool chpax modifies the legacy EI_PAX flags and paxctl modifies the newer PT_PAX_FLAGS. In order to determine which header markings a default Fedora Core OS supports, it was necessary to compile and run each tool to see if support was added to the OS binaries.

For chpax-0.7 the file was located at http://pax.grsecurity.net/chpax-0.7.tar.gz. Since neither md5 checksums or gpg signatures were provided there is no protection from source tampering. Unpack and compile the file.

```
[root@hermes]# make
[root@hermes]# make install
```

Now to see if the EI_PAX header marking is included in a binary let us look at /bin/ls.

```
[root@hermes]# chpax –v /bin/ls

----[ chpax 0.7 : Current flags for /bin/ls (PeMRxS) ]---

* Paging based PAGE_EXEC          : enabled (overridden)
* Trampolines                     : not emulated
* mprotect()                      : restricted
* mmap() base                     : randomized
* ET_EXEC base                    : not randomized
* Segmentation based PAGE_EXEC    : enabled
```

So the EI_PAX header marking is being used. This command should show similar results for all binaries checked.

For paxctl-0.2 the file was located at http://pax.grsecurity.net/paxctl-0.2.tar.gz. Since neither md5 checksums or gpg signatures were provided there is no protection from source tampering. Unpack and compile the file.

```
[root@hermes]# make
[root@hermes]# make install
```

Now to see if PT_PAX_FLAGS header marking is included let us again look at /bin/ls.

```
[root@hermes]# paxctl –v /bin/ls
PaX control v0.2
Copyright 2004 PaX Team <pageexec@freemail.hu>
```

This would indicate that the PT_PAX_FLAGS header marking is not being used. This can be verified by executing the above command for a few other key binaries including /usr/bin/ld and /usr/bin/gcc.

UnrealIRCd

The Unreal IRCd will require the use of the OpenSSL source installed since the RPM package only contains actual libraries and executables. This is necessary to compile SSL into the IRCd.

Compiling OpenSSL

The OpenSSL package was downloaded from http://www.openssl.org/source/openssl-0.9.7d.tar.gz and the authenticity of the package needs to be checked. The OpenSSL FAQ stated that the OpenSSL team public key should be used which can be pulled from http://pgp.mit.edu/.

```
Public Key Server -- Index "OpenSSL team "
pub  1024D/89A36572 1999/12/12 OpenSSL Team Security Key (WARNING: SHARED
KEY) <openssl-security@openssl.org>
     Key fingerprint = 7369 E60D 2FA5 ECF3 DA38  66FD 901E 804A 89A3 6572
```

When attempting to verify the signature the following error was received when attempting to use gpg.

```
[root@hermes]# gpg --verify openssl-0.9.7d.tar.gz.asc openssl-0.9.7d.tar.gz
gpg: error while loading shared libraries: cannot make segment writable for
relocation: Permission denied
```

This error is due to PaX and that ELF text relocation was disallowed. To allow gpg to run, security must be sacrificed and runtime code generation must be allowed for that program. This means mprotect() has to be unrestricted.

```
[root@hermes]# chpax –v /usr/bin/gpg

----[ chpax 0.7 : Current flags for /usr/bin/gpg (PeMRxS) ]---

* Paging based PAGE_EXEC          : enabled (overridden)
* Trampolines                     : not emulated
* mprotect()                      : restricted
* mmap() base                     : randomized
* ET_EXEC base                    : not randomized
* Segmentation based PAGE_EXEC    : enabled
```

Unrestrict mprotect() and verify it was unrestricted.

```
[root@hermes]# chpax –m /usr/bin/gpg
[root@hermes]# chpax –v /usr/bin/gpg

----[ chpax 0.7 : Current flags for /usr/bin/gpg (PeMRxS) ]---

* Paging based PAGE_EXEC          : enabled (overridden)
* Trampolines                     : not emulated
* mprotect()                      : not restricted
* mmap() base                     : randomized
* ET_EXEC base                    : not randomized
* Segmentation based PAGE_EXEC    : enabled
```

And now gpg should work fine. In addition this and any other errors would be entered into the change control log as defined in the next section.

While verifying the file http://www.openssl.org/source/openssl-0.9.7d.tar.gz.asc the following error message was received.

```
[root@hermes]# gpg --verify openssl-0.9.7d.tar.gz.asc openssl-0.9.7d.tar.gz
gpg: Signature made Wed 17 Mar 2004 07:09:54 AM EST using RSA key ID
49A563D9
gpg: BAD signature from "Mark Cox <mjc@redhat.com>"
```

The error message stated the package signature file was signed by key 49A563D9 owned by Mark Cox, one of the current OpenSSL core team members. The decision was made to pull the RSA key ID 49A563D9 (0x49A563D9) from http://pgp.mit.edu/. Using this key for verification proved successful.

Then the package was unpacked and the directory was changed to /opt/openssl-0.9.7d.

```
[root@hermes]#./config
[root@hermes]# make
[root@hermes]# make test
[root@hermes]# make install
```

The openssl directory will be in /usr/local/ssl and will be used to compile support for SSL into Unreal IRCd. Once the packages have been installed the source files from /opt should be removed.

Compiling UnrealIRCd

Next the IRCd server needs to be compiled. It was not possible to locate any type of PGP signature to verify the integrity of the software package. An md5 checksum was pulled from http://64.84.10.70:81/?page=md5s. This provides limited protection from source tampering as the md5 checksum was found on the same site as the source meaning that an attacker simply has to modify the md5 html page with the trojaned version's new md5 checksum. This is better than doing nothing but the risk needs to be understood.

The md5 checksum found on the above site was "ebe56fd42fc229681f527932eaa173cc" and it matched the results of checking the package with md5sum. Move into the /opt/Unreal3.2.1 directory and do the following:

```
[root@hermes]#./Config

(does auto detection of software until user configuration section below)

Many older operating systems have an insecure TCP/IP stack
which may be vulnerable to IP spoofing attacks, if you run
an operating system that is vulnerable to such attacks
enable this option. This option can also be useful to prevent
blind proxies from connecting (eg: HTTP POST proxies).

Do you want to enable the server anti-spoof protection?
[No] -> <enter>

What directory are all the server configuration files in?
[/opt/Unreal3.2] -> /usr/local/etc/

What is the path to the ircd binary including the name of the binary?
[/opt/Unreal3.2/src/ircd] -> /usr/local/sbin/ircd

Would you like to compile as a hub or as a leaf?
Type Hub to select hub and Leaf to select leaf.
[Hub] -> leaf

What is the hostname of the server running your IRCd?
[IRCd] -> hermes
```

```
What should the default permissions for your configuration files be? (Set
this to 0 to disable)
It is strongly recommended that you use 0600 to prevent unwanted reading of
the file
[0600] -> <enter>

Do you want to support SSL (Secure Sockets Layer) connections?
[No] -> Yes

If you know the path to OpenSSL on your system, enter it here. If not leave
this blank
[] -> /usr/local/ssl

Do you want to enable IPv6 support?
[No] -> <enter>

Do you want to enable ziplinks support?
[No] -> <enter>

Do you want to enable remote includes?
[No] -> <enter>

Do you want to enable prefixes for chanadmin and chanowner?
This will give +a the & prefix and ~ for +q (just like +o is @)
Supported by the major clients (mIRC, xchat, epic, eggdrop, Klient, PJIRC,
etc.)
with the notable exceptions of irssi, KVIrc and CGI:IRC.
This feature should be enabled/disabled network-wide.
[No] -> <enter>

What listen() backlog value do you wish to use?  Some older servers
have problems with more than 5, others work fine with many more.
[5] -> <enter>

How far back do you want to keep the nickname history?
[2000] -> <enter>

What is the maximum sendq length you wish to have?
[3000000] -> <enter>

How many buffer pools would you like?
This number will be multiplied by MAXSENDQLENGTH.
[18] -> <enter>

How many file descriptors (or sockets) can the IRCd use?
[1024] -> <enter>

Would you like any more parameters to configure?
Write them here:
[]-> <enter>

(Compilation messages)

Generating certificate request ..
/usr/bin/openssl req -new \
              -config src/ssl.cnf -out server.req.pem \
              -keyout server.key.pem -nodes
```

```
Generating a 1024 bit RSA private key
...++++++
.....................................++++++
writing new private key to 'server.key.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name [US]: <enter>
State/Province [New York]:Colorado
Locality Name (eg, city) []:Boulder
Organization Name (eg, company) [IRC geeks]:GIAC Enterprises
Organizational Unit Name (eg, section) [IRCd]:Information Systems
Common Name (Full domain of your server) []:hermes.giacenterprises.com
Generating self-signed certificate ..
/usr/bin/openssl req -x509 -days 365 -in server.req.pem \
               -key server.key.pem -out server.cert.pem
Generating fingerprint ..
/usr/bin/openssl x509 -subject -dates -fingerprint -noout \
    -in server.cert.pem
subject= /C=US/ST=Colorado/L=Boulder/O=GIAC Enterprises/OU=Information
Systems/CN=hermes.giacenterprises.com
notBefore=Sep 14 19:03:21 2004 GMT
notAfter=Sep 14 19:03:21 2005 GMT
MD5 Fingerprint=DA:99:31:CE:46:08:99:FF:97:98:B6:AA:83:55:5B:DE
Setting o-rwx & g-rwx for files...
chmod o-rwx server.req.pem server.key.pem server.cert.pem
chmod g-rwx server.req.pem server.key.pem server.cert.pem
Done!. If you want to encrypt the private key, run
make encpem

[root@hermes]# make
```

Once the compilation is done the following needs to be done.

```
[root@hermes]# cp /opt/Unreal3.2/src/ircd to /usr/local/sbin
[root@hermes]# cp /opt/Unreal3.2/help.conf to /usr/local/etc
[root@hermes]# cp /opt/Unreal3.2/server.cert.pem /usr/local/etc
[root@hermes]# cp /opt/Unreal3.2/server.key.pem /usr/local/etc
```

Check to ensure that /usr/local/lib is in /etc/ld.so.conf and if it is added rerun “ldconfig.”

```
[root@hermes]# cp /opt/Unreal3.2/src/modules/commands.so to /usr/local/lib
[root@hermes]# cp /opt/Unreal3.2/src/modules/cloak.so to /usr/local/lib
```

Finally since this server should not be run as root, a new user is created. This user account will have the path to the ircd executable set as the users shell, so when it is su’d into the user account it will only be able to start the ircd daemon. Then by changing ownership of the executable and key files this daemon will now run under the user ircd.

```
[root@hermes]# groupadd –g 80 ircd
```

```
[root@hermes]# useradd –u 80 –g 80 –c UnrealIRCd –d /usr/local/sbin –s
/usr/loca/sbin/ircd ircd
[root@hermes]# chown ircd:ircd /usr/local/sbin/ircd
[root@hermes]# chown –R ircd:ircd /usr/local/etc/*
[root@hermes]# chown ircd:ircd /var/log/ircd.log
```

Now test this by running:

```
[root@hermes]# su ircd
[root@hermes]# ps –aux
```

Look for the line that shows command ircd and you should see the user as ircd.

Configuration Files

Next create the file as seen in Appendix C as unrealircd.conf in /usr/local/etc/. Configure IRCd to start up on boot using the init file found in Appendix D which should be located in /etc/init.d. Configure it to start by doing the following:

```
[root@hermes]# chkconfig –-add ircd
```

User Management

The users.conf file will maintain the users and passwords required for all authorized employees. This file will be modified as new users are added and removed. The syntax is as follows if username is made up of first initial plus last name. This information will reside in the /usr/local/etc/users.conf file.

```
allow {
    ip          <username>*@*;
    hostname    NOHOSTNAME;
    class       clients;
    password    "password";
    options {
                noident;
    };
};
```

Note that this file stores the user passwords in cleartext. As this is a limitation of the software all that can be done is to lock the permissions of this file down to the ircd user and use the protection provided by our RBAC configuration.

```
[root@hermes]# chmod 0600 /usr/local/etc/users.conf
```

Of course, since this file is being created after the ownership changes above it will need to be modified also.

```
[root@hermes]# chown ircd:ircd /usr/local/etc/users.conf
```

SSHd

SSHd is setup as an RPM and comes with the OS. Use yum to update to the latest package. The decision to stay with a packaged version of SSH was made for several reasons:

- As new versions come out they will be automatically updated.
- Convenience of dependencies being met, including new updated dependency versions.
- Configuration is applied to new versions without having to reconfigure.

Of course, a possible downside is that attackers like to develop exploits that might focus on such packages since they are deployed to a larger group. The default configuration is not used for this server.

GIAC Enterprises uses SSH throughout the network so this server will just be added to the existing design. The company uses SSH to authenticate to the other servers and they all use public key authentication. Our standard is 2048bit RSA key pairs for reasons that were decided upon when the network was stood up. Only administrators have accounts on this server and would use their existing public/private key pair but should a new administrator require it they would do the following:

```
[achan@client achan]$ ssh-keygen –t rsa –b 2048
Generating public/private rsa key pair.
Enter file in which to save the key (/home/achan/.ssh/id_rsa): <enter for
default>
Created directory ‘/home/achan/.ssh’.
Enter passphrase (empty for no passphrase): *******
Enter same passphrase again: *******
Your identification has been saved in /home/achan/.ssh/id_rsa.
Your public key has been saved in /home/achan/.ssh/id_rsa.pub.
The key fingerprint is
27:4f:06:be:c4:bf:98:26:b7:32:99:b9:68:f1:62:ef  achan@client
```

The user should verify that their private key (/home/achan/.ssh/id_rsa) is mode 0600 while everything else should be 0740 using chmod. The users public key needs to be put on the server where they want to connect to by doing the following:

```
[achan@client achan]$ eval `ssh-agent`
Agent pid 3651
[achan@client achan]$ ssh-add
Enter passphrase for /home/achan/.ssh/id_rsa:
Identity added: /home/achan/.ssh/id_rsa (/home/achan/.ssh/id_rsa)
[achan@client achan]$ ssh achan@hermes

“WARNING! Unauthorized use of this system is strictly prohibited and may be
subject to criminal prosecution or employee discipline. Authorized
personnel may monitor any activity or communication on the system and may
retrieve any information stored within the system. By accessing and using
this computer, you are consenting to such monitoring and information
retrieval for law enforcement and other purposes. Users should have no
expectation of privacy as to any communication on or information stored
within the system.”
```

```
[achan@hermes achan]$ exit
```

To minimize unauthorized access the users public key should be added to the /home/<userid>/.ssh/authorized keys file located on the server they are to have access to. By disabling the ability to log in to any server by password authentication it eliminates brute forcing of accounts.

Here is the commented sshd.conf file from our server. It is identical to our other servers on GIAC Enterprises network.

/etc/ssh/sshd.conf

```
# GIAC Enterprises sshd.conf file for Server Farm
# Developed: Adam Chan 030623
# Using defaults for any configuration variable not listed.

# Port to bind to
Port 22

# SSH Protocol to use. Only use version 2.
Protocol 2

# Listen on all interfaces. On machines with more than one the actual IP
address you want the server to listen on should be entered.
ListenAddress 0.0.0.0

# HostKeys for protocol version 2. Only use RSA key.
HostKey /etc/ssh/ssh_host_rsa_key

# Logging Information
SyslogFacility AUTHPRIV

# Do not allow direct root account login
PermitRootLogin no

# Disables pure RSA1 authentication since protocol version 1 is not used.
RSAAuthentication no

# Enable Public Key authentication
PubkeyAuthentication yes

# Define Authorized Key file
AuthorizedKeysFile      .ssh/authorized_keys

# Disable Rhosts authentication
RhostsAuthentication no

# Disable password authentication
PasswordAuthentication no

# Disable Challenge Response authentication
ChallengeResponseAuthentication no
```

```
# Disable Kerberos authentication
KerberosAuthentication no

# Disable PAM keyboard-interactive authentication
PAMAuthenticationViaKbdInt no

# Disable X11 Forwarding since X it is not installed
X11Forwarding no

# Enables privilege separation
UsePrivilegeSeparation yes

# Enable user environment
PermitUserEnvironment yes

# Use /etc/issue.net for the banner
Banner /etc/issue.net

#Verify the remote host name
VerifyReverseMapping yes

# Add in sftp subsystem
Subsystem   sftp  /usr/libexec/openssh/sftp-server
```

Since only the RSA host keys for SSH protocol version 2 (/etc/ssh/ssh_host_rsa_key) is being used the others can be removed. To remove the other keys the auto generation check found in the /etc/init.d/sshd initialization file needs to be removed. Remove do_rsa1_keygen and do_dsa_keygen from the start() function. Then remove ssh_host_dsa_key, ssh_host_dsa_key.pub, ssh_host_key and ssh_host_key.pub from /etc/ssh. Again, this minimizes the number of unneeded files found in key directories, which makes it easier for administrators to spot out of place files.

Privilege separation functionality has been enabled in SSH. This uses a privileged parent process that monitors the progress of an unprivileged child process. This results in complete privilege separation where a bug in the unprivileged child process does not result in system compromise. More information on this can be found at http://www.citi.umich.edu/u/provos/ssh/privsep.html.

Yum

GIAC Enterprises now runs a local archive server. The server was configured with the following yum.conf that will run nightly as per the yum service. The local archive server will put up files ready to be downloaded after they are verified on the VMware test network.

/etc/yum.conf

```
# GIAC Enterprises yum.conf file for Server Farm
# Developed: Adam Chan 031123

# main options
```

```
[main]

# Directory where yum should store its cache and db files
cachedir=/var/cache/yum

# Debug level. Range is 0-10.
debuglevel=1

# Full directory and file name where logs are written
logfile=/var/log/yum.log

# Determines the "version" of the distribution
distroverpkg=redhat-release

# List of packages to exclude from updates or installs
exclude=*kernel*

# Define Local Archive Server
[local]
name=Local Archive $releasever - $basearch
baseurl=http://<local archive server>/updates/$releasever/$basearch/
```

When attempting to check for updated packages using yum the following error was received.

```
[root@hermes]# yum update
Traceback (most recent call last):
  File "/usr/bin/yum", line 22, in ?
     Import yummain
<snip>
ImportError: libbeecrypt.so.6: cannot enable executable stack as shared
object requires: Permission denied
```

After checking the forums at http://forums.grsecurity.net/ it was determined that this error is again due to PaX and that ELF text relocation was disallowed. Again mprotect() has to be unrestricted on the binary. Since yum is a python script, the binary that needs modifying would actually be python itself.

```
[root@hermes]# chpax -m /usr/bin/python
[root@hermes]# chpax -v /usr/bin/python

----[ chpax 0.7 : Current flags for /usr/bin/python (PeMRxS) ]---

* Paging based PAGE_EXEC         : enabled (overridden)
* Trampolines                    : not emulated
* mprotect()                     : not restricted
* mmap() base                    : randomized
* ET_EXEC base                   : not randomized
* Segmentation based PAGE_EXEC   : enabled
```

Another error was generated when rpm was used to query the package database.

```
[root@hermes]# rpm -qa
```

```
error while loading shared libraries: libbeecrypt.so.6: cannot enable
executable stack as shared object requires: Permission denied
```

As this is the same exact error as above the following binaries were modified to ensure rpm functions correctly.

```
[root@hermes]# chpax –m /bin/rpm
[root@hermes]# chpax –m /usr/lib/rpm/rpmd
[root@hermes]# chpax –m /usr/lib/rpm/rpmi
[root@hermes]# chpax –m /usr/lib/rpm/rpmk
[root@hermes]# chpax –m /usr/lib/rpm/rpmq
```

Rdate

This client connects the to the company's time server and updates the system time on a daily basis. Using cron the server will schedule a daily connection as follows:

```
[root@hermes]# rdate –s –l <time server IP address>
```

This requests a time to set the system clock and uses syslog to ouput errors to cron.warning and output to cron.info.

Cron

Cron is used to run a specific program at a specific time. It is configured as follows:

- cron.hourly - nothing
- cron.daily
  - logrotate – starts /usr/sbin/logrotate using /etc/logrotate.conf
  - prelink – prelinks ELF shared libraries and binaries to speed up startup time.
  - tmpwatch – cleans up directories that are used for temporary files
  - yum.cron – checks the repository server using yum for updated packages.
  - rdate.cron – runs rdate to connect to company's time servers.
- cron.weekly
  - makewhatis.cron – updates whatis database
  - yum.cleanup.cron – cleans up old packages and old headers
- cron.monthly - nothing

Log rotation

Even though a central log server is being used, it is good to maintain a certain amount of logs on the server for troubleshooting or investigation purposes. Logrotate is run daily to manage them.

Syslog Logging

GIAC Enterprises maintains a central logging server on the secure subnet. More in depth information on this infrastructure can be found at http://www.giac.com/practical/GCFW/Danny_Walker_GCFW.pdf.
The logging server uses a non-IP-addressed interface plugged into the DMZ switch. Since a switch is being used in the DMZ the bogus host will represent the syslog server

and has to be located using its MAC address. For that a static ARP entry of “arp –s 39.2.1.62 00:00:00:00:00:00” (where 39.2.1.62 is the bogus host (syslog server) and 00:00:00:00:00:00 is the actual MAC address of our non-IP-addressed interface of the Syslog server) will be added. This will ensure that the data gets sent to the proper switch port. On the Syslog server a stealth-logger using Snort will be used. This logger will implement the frag2 preprocessor in case the packets come across fragmented. Then the logger will simply pass the logged data into the msyslog database. As syslogd is being used on this server, our configuration file is as follows:

/etc/syslog.conf

```
# GIAC Enterprises syslog.conf file for Server Farm
# Developed: Adam Chan 040920

# Log auth facility with warning priority or higher to log and logserver
auth.warning                                /var/log/auth
auth.warning                                @39.12.1.62

# Log mail facility with err priority or higher to log
mail.err                                    /var/log/maillog

# Log kern facility with any priority to log and logserver
kern.*                                      /var/log/kernel
kern.*                                      @39.12.1.62

# Log cron facility with warning priority or higher to log and logserver
cron.warn                                   /var/log/cron
cron.warn                                   @39.12.1.62

# Log all except authpriv, mail, cron & kernel of info priority or higher
to log.
*info;authpriv.none;mail.none;cron.none;kernel.none   /var/log/messages

# Log authpriv facility with any priority to log and logserver
authpriv.*                                  /var/log/secure
authpriv.*                                  @39.12.1.62

# Log all facilities (except authpriv) with any priority to log
# This will only be used until the log configuration file has been tuned.
*.*;authpriv.none                           /var/log/all

# Log local3 facility with info priority to logserver (firewall notices)
local3.info                                 @39.12.1.62
```

As with any other new server added to the network, part of the log review process is to review both the central log server and local logs and tune this configuration file. This enables us to limit the amount of unnecessary information sent to the central log server.

Grsecurity Kernel auditing

Grsecurity adds additional kernel auditing that monitors exec, signals, failed forks, IPC creation and removal, chdirs and mount/unmounts. This creates a lot of log data and its

use for a production system would require the use of some type of log filtering tool like Swatch (http://swatch.sourceforge.net/), SEC (http://www.estpak.ee/~risto/sec/) or logwatch (http://www2.logwatch.org:81/). By using such software events that are due to normal server operation can be filtered out leaving only the abnormal errors. This may assist when defending against zero-day exploits or other unknown attacks. This area is being investigated by GIAC Enterprises.

Role Based Access Control (RBAC) Configuration

Now that all of the software and services have been installed you will need to configure the RBAC system that grsecurity provided.

Since this is role based access control, the roles to be used for this server must be determined. The goal is to split up the privileges so that no one user (e.g. root) would have unlimited access to the server.

admin – manages RBAC policy and system administration duties.
root – manages operating system basics and service startup.
ircd – manages IRC server
achan – system administrator

Gradm command

This tool interfaces with and controls the functionality provided by the grsecurity patch.

Argument	Function
-E	Enable the RBAC system
-R	Reload the RBAC system
-D	Disable the RBAC system
-S	Check the status of RBAC system
-P [rolename]	Create password for RBAC administration or a special role
-a <rolename>	Change to a special role that requires authentication
-u	Change back to original role
-n <rolename>	Change to a special role that does not require authentication
-F	Enable full system learning
-L <filename>	Specify the pathname for learning logs
-O <filename>	Specify where to place policies generated from learning mode

Learning Mode

Grsecurity offers a very nice full-system learning mode to assist with creating a least privilege policy. When activated this program will track the executables and resources required for that executable (libraries, files, binding to the network, system resources, etc.).

The following gradm initialization file will serve two purposes. First gradm needs to be set in learning mode to generate a least privilege policy for the system. This will enable the system to be rebooted since the learning.log will simply be appended. The second

purpose will be to use the script to start gradm for production after the policy has been developed. Under start() either option can be selected while commenting out the other.

/etc/init.d/gradm

```
#!/bin/bash
#
# chkconfig: 2345 90 02
# (run in init levels 2345 & start priority of 90 and stop priority of 02)
# processname: gradm
# description: Init file for gradm
# Used both for learning mode and production
# config: /etc/grsec/policy

# source function library
. /etc/rc.d/init.d/functions

lockfile=/var/lock/subsys/gradm

RETVAL=0
GRADM=/sbin/gradm

start()
{
     echo -n $"Starting RBAC:"

     # Use this for initial learning mode
     $GRADM -F -L /root/learning.log

     # Use this for production
     # $GRADM -E
     touch "$lockfile" && success || failure
     RETVAL=$?
     echo
}

stop()
{
      echo -n $"Stopping RBAC: "
      $GRADM -D
      rm -f "$lockfile" && success || failure
      RETVAL=$?
      echo
}

case "$1" in
    start)
          start
          ;;
      stop)
            stop
            ;;
      status)
            $GRADM -S
            ;;
```

```
    *)
          echo $"Usage: $0 {start|stop|status}"
            RETVAL=1
esac
exit $RETVAL
```

All communication needs to be blocked with the following script until after the RBAC system is in place. Once RBAC is enabled the normal firewall defined earlier> can be implement. In fact, the initialization order is as follows:

1. killcomm – startup firewall that kills all communication.
2. Startup all other services.
3. gradm – start up RBAC.
4. netfilter – replace killcomm with normal production firewall.

And upon reboot, halt or shutdown the order is as follows:

1. killcomm – replace netfilter firewall with one that kills all communication.
2. gradm – shutdown RBAC.
3. Shuts down all other services.

Note that the netfilter firewall never gets shut down, as that would reconfigure the firewall to allow everything.

/etc/init.d/killcomm

```
#!/bin/bash
#
# chkconfig: 2345 01 01
# (run in init levels 2345 & start priority of 01 and stop priority of 01)
# description: Init script to kill all possible communication upon both  #
# starting or stopping script
# processname: iptables

# source function library
. /etc/rc.d/init.d/functions

lockfile=/var/lock/subsys/killcomm

RETVAL=0
IPT=/sbin/iptables

start()
{
    echo -n $"Killing all communication:"

    $IPT -t filter -P INPUT DROP
    $IPT -t filter -P FORWARD DROP
    $IPT -t filter -P OUTPUT DROP
    $IPT -t nat -P PREROUTING DROP
    $IPT -t nat -P POSTROUTING DROP
    $IPT -t nat -P OUTPUT DROP
    $IPT -t mangle -P PREROUTING DROP
    $IPT -t mangle -P OUTPUT DROP
```

```
      $IPT -t mangle -P POSTROUTING DROP
      $IPT -t mangle -P INPUT DROP
      $IPT -t mangle -P FORWARD DROP

      $IPT -t filter -F
      $IPT -t nat -F
      $IPT -t mangle -F
      $IPT -X

      echo 0 > /proc/sys/net/ipv4/ip_forward

      touch "$lockfile" && success || failure
      RETVAL=$?
  }

  stop()
  {
      echo –n $" Killing all communication: "

      $IPT -t filter -P INPUT DROP
      $IPT -t filter -P FORWARD DROP
      $IPT -t filter -P OUTPUT DROP
      $IPT -t nat -P PREROUTING DROP
      $IPT -t nat -P POSTROUTING DROP
      $IPT -t nat -P OUTPUT DROP
      $IPT -t mangle -P PREROUTING DROP
      $IPT -t mangle -P OUTPUT DROP
      $IPT -t mangle -P POSTROUTING DROP
      $IPT -t mangle -P INPUT DROP
      $IPT -t mangle -P FORWARD DROP

      $IPT -t filter -F
      $IPT -t nat -F
      $IPT -t mangle -F
      $IPT -X

      echo 0 > /proc/sys/net/ipv4/ip_forward
      rm –f "$lockfile" && success || failure
      RETVAL=$?
   }

  case "$1" in
     start)
           start
           ;;
       stop)
           stop
           ;;
     *)
           echo $"Usage: $0 {start|stop}"
           RETVAL=1
  esac
```

The command chkconfig will be used to manage the new scripts. Since each has a chkconfig setting defined in the beginning of the script the following commands can be used to add them:

```
[root@hermes]# chkconfig –-add gradm
[root@hermes]# chkconfig –-add killcomm
```

Since cron is considered a normal operation, it would be time consuming to wait on the hourly, daily, weekly and monthly jobs to run in order to complete the policy. By saving the existing crontab and creating a new temporary crontab that runs all cron jobs each hour (as below), the process can be sped up.

/etc/crontab

```
# Temporary crontab to speed up the execution of the various cron jobs.
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:usr/bin
MAILTO=root
HOME=/

#run-parts
01 * * * *      root  run-parts   /etc/cron.hourly
10 * * * *      root  run-parts   /etc/cron.daily
20 * * * *      root  run-parts   /etc/cron.weekly
42 * * * *      root  run-parts   /etc/cron.monthly
```

In addition to the cron jobs, each role needs to have a basic set of functionality defined before using the learning mode. This will assist the system administrator during learning mode to use only the commands required to perform that role. Additional access can be granted and policy modified later if needed.

The root role will allow the following:

- System initialization as described above including the changeover to the netfilter firewall configuration.
- Enable the cron jobs to run
- Kernel logging
- Syslog logging
- Management of SSHd authentication

NOTE: Do not start/stop services, access unneeded files or commands, adding/removing users, or any other administrative task that is not required.

The ircd role will allow the following:

- Have users connect to the IRC server and access all the needed services including transferring files and accessing operator.

The achan user will allow the following:

- Login via ssh using public key authentication

- Use basic commands including cd, chmod, chown, chgrp, cp, date, df, env, grep, gzip, kill, ln, ls, mail, mkdir, more, mv, netstat, nice, ping, ps, pwd, rm, rmdir, rpm, sort, tar, touch, traceroute, uname, vi, xargs, zcat and any other user command that might be required.
- Use su to change into root, which would be required to transition to the admin role. This would also be required to access any root owned files as even the admin role still respects the standard linux ACL and cannot override it.

The admin role will become the new super user account. This role could be split up even further to divide up key privileges among multiple roles. This will be investigated and implemented at a later date.

Then reboot the machine and allow it to boot up completely. Let this run through several cycles of the modified cron jobs as well as allow the various roles perform their needed functions as defined above. Try to run through each role requirement at least 4 times to ensure that all of the resources required are recorded.

To stop the learning mode gradm must be shutdown as follows:

```
[root@hermes]# gradm –D
Password: <Type in default password>
grsec: shutdown auth success for /sbin/gradm[gradm:1355] uid/euid:0/0
gid/egid:0/0, parent /bin/bash[bash:1226] uid/euid:0/0 gid/egid:0/0
```

You will have a file called /root/boot-learning.log and you now want to generate an actual policy you can use.

```
[root@hermes]# gradm -F -L /root/boot-learning.log -O /etc/grsec/temp-
policy
Beginning full learning 1st pass...done.
Beginning full learning role reduction...done.
Beginning full learning 2nd pass...done.
Beginning full learning subject reduction for user root...done.
Beginning full learning 3rd pass...done.
Beginning full learning object reduction for subject /...done.
<messages cut>
Beginning full learning final pass...done.
```

Next add an admin role to the policy that will allow that role to access administration commands. Currently the temporary policy is located at /etc/grsec/temp-policy.

```
role admin sA
subject   /     {
          /     rwcdmxi
}
```

Also under the default role add:

```
role root uG
role_transitions admin
```

The admin role's password needs to be set up.

```
[root@hermes]# gradm –P admin
```

Next modify policy settings as follows:

- Modify any role_allow_ip to reflect all of the IP addresses that this role can connect from. This will typically be "0.0.0.0/32".
- Remove specific temporary directories under /tmp and give permissions to entire /tmp directory.
- Allow access to all of the /proc filesystem if specific access was allowed.
- Modify to include all specific user directories that could be needed by the number of users on system. For instance, the subject usr/sbin/sshd requires access to /home/achan/.ssh directory. If other users require access then their home directories would have to be added.
- Modify connect settings to include all of the IP addresses for that server type if more than one. For instance, if 2 DNS servers are on the network ensure that both are included.

Review the temp-policy and rename it to /etc/grsec/policy. Start up RBAC with "gradm –E" and let it run for a few minutes to see if anything starts complaining. If error messages are displayed then the policy needs to be modified to allow access to the resource required.

Replace crontab with original crontab file. Once everything has been running without problems you can modify the /etc/init.d/gradm initialization file to start gradm upon boot.

```
# Use this for initial learning mode
# $GRADM –F –L /root/learning.log

# Use this for production
$GRADM -E
```

SYSCTL Support

In the kernel, sysctl support was enabled which allows some options to be enabled and disabled via the sysctl command. The following kernel configuration settings can be modified via sysctl (only those enabled in kernel have been listed).

SYSCTL option name	Description (additional information found in Appendix B)
linking_restrictions	Linking restrictions (GRKERNSEC_LINK)
fifo_restrictions	FIFO restrictions (GRKERNSEC_FIFO)
exec_logging	Exec logging (GRKERNSEC_EXECLOG)
audit_chdir	Chdir logging (GRKERNSEC_AUDIT_CHDIR)
audit_mount	(Un)Mount logging (GRKERNSEC_AUDIT_MOUNT)
audit_ipc	IPC logging (GRKERNSEC_AUDIT_IPC)
signal_logging	Signal logging (GRKERNSEC_SIGNAL)
forkfail_logging	Fork failure logging (GRKERNSEC_FORKFAIL)

timechange_logging	Time change logging (GRKERNSEC_TIME)
execve_limiting	Enforce RLIMIT_NPROC on execs (GRKERNSEC_EXECVE)
Dmesg	Dmesg(8) restriction (GRKERNSEC_DMESG)
rand_pids	Randomized PIDs (GRKERNSEC_RANDPID)
rand_isns	Truly random TCP ISN selection (GRKERNSEC_RANDISN)
rand_ip_ids	Randomized IP IDs (GRKERNSEC_RANDID)
rand_tcp_src_ports	Randomized TCP source ports (GRKERNSEC_RANDSRC)
rand_rpc	Randomized RPC XIDs (GRKERNSEC_RANDRPC)

In addition, all of the settings relating to chroot jail restrictions would be covered by sysctl if enabled in the kernel. In this case the settings have been disabled.

It is suggested that each of the options above are added to the /etc/sysctl.conf so it is set upon operating system boot.

Remove old kernel

Once everything is working it is suggested to remove the original kernel that does not support grsecurity. If this is not done then an attacker could simply boot into the old kernel and keep all the new features disabled. Remove the kernel as follows:

```
[root@hermes]# rpm –e kernel
```

This will remove the kernel and modify any necessary files.

Design and Implement Ongoing Maintenance Procedures

No server on a network can be immune from ongoing maintenance procedures. This section will go into the five areas of ongoing maintenance the company uses across its network. These are:

- Change Control
- Software Updates / Patch management
- Log Review
- Backup
- Security Audit

Change Control

This server will fall under a change management policy. This provides a process to apply changes, upgrades, or modifications to the server. It covers any and all changes to the hardware, software or applications. Changes to the server may arise from many circumstances, such as:

- Periodic maintenance
- Patch management
- Hardware and/or software upgrades
- Acquisition of new hardware and/or software
- Environmental changes
- User requests
- Unforeseen events

The change management process is as follows:

1. Submission of a change request – the requestor must obtain management approval prior to submitting a request. This ensures that managers are aware of all changes occurring in their areas of responsibility.
2. Submission to Change Management Administrator. It is this persons job to schedule and coordinate with the project managers to make the change a smooth process.
3. The Change Management Administrator properly documents the change and notes any issues that came up.

Emergencies are dealt with on an as-required basis with the approval of the CIO. Documentation is given to Change Management Administrator for his records.

Software Updates / Patch Management

GIAC Enterprises has a heavy dependency on open source software. Red Hat 9.0 is used on most of the network and the company has a subscription to the Red Hat Network for software updates. A RPM archive server is maintained and updated twice daily. The team uses a VMware network to simulate the entire GIAC Enterprises network including firewalls, IPSec servers, hosts, etc. These test servers are updated from the RPM archive server and run for 48 hours. The team is also responsible for

monitoring various mailing lists including BugTraq, Full Disclosure, and several hosted by Red Hat for security issues that might need to be dealt with ASAP. In this case, the updates could be installed in as little as 2 hours with intense testing and the CIOs approval. All software updates and modifications follow the change control policy above and a change request submission is drafted as updates are loaded onto the test network.

This server is running Fedora Core 1, which GIAC Enterprises is evaluating as the next OS upgrade. Along with the YUM service, everything is maintained on the same archive servers and follow the same processes regarding security issues. Since this server is running a custom kernel any kernel update packages will be specifically blocked. If a flaw is discovered in the kernel it will have to be evaluated and possibly updated manually.

The following software has to be maintained separately, as it is not supported by the Fedora project. For this server, the software is monitored and maintained separately.

Software	Location	Description
grsecurity patch	http://www.grsecurity.net/	A security patch required when compiling the kernel. Unless some needed functionality or security vulnerability is found this will not be modified often.
UnrealIRCd	http://www.unrealircd.org/	The IRCd server software

Log Review

Administrators are required to do a periodic log review with two goals in mind.

1. Logs are reviewed and compared to what was recorded by the central log server. This enables the administrators to tune what is sent to the central log server. Any data found in the server logs that should be sent to the central log server can be added and any data that is not necessary can be removed. This keeps the log server from getting overwhelmed by unneeded data.
2. Since additional kernel auditing was enabled on the new kernel the administrator needs to understand what these messages could mean and investigate any suspicious audit entries. A complete understanding of this will be required or it severely limits the value of such utilities.

During this log review, the log rotation schedule will be reviewed and modified as needed. Any changes to the logging configuration files will follow the change control process.

Backup

This server will have key files backed up on a weekly basis and the administrator will use secure copy (scp) to move this data to a central admin server that is backed up to

tape. A complete server image will not be done and will backup the following information:

File/Directory	Purpose
/etc/ssh	SSH configuration and encryption keys
/etc/init.d	Custom initialization scripts
/etc/cron.d /etc/cron.hourly /etc/cron.daily /etc/cron.weekly /etc/cron.monthly	Custom cron scripts
/etc/logrotate.d	Custom logrotate scripts
/etc/yum.conf	Yum configuration file
/etc/grsec	grsecurity ACL policies
/etc/inittab	Custom inittab file
/etc/grub.conf	Custon grub configuration file
/etc/syslog.conf	Custom syslog configuration file
/usr/local/etc/unrealircd.conf	Unreal IRCd configuration file
/usr/local/etc/server.key.pem /usr/local/etc/server.cert.pem	Unreal IRCd encryption keys
/boot/config-2.6.7-grsec	Kernel configuration configuration file
/var/log/*	Backup of complete logs for last 4 weeks

These files were chosen to backup in case the server crashes and has to be rebuilt or in case of compromise, where accurate historical data is required to assess the damage. Any change to the backup procedures will follow the change control process.

Security Audits

GIAC Enterprises conducts a network wide security audit once a week using the Nessus vulnerability assessment tool (http://www.nessus.org). This open source tool is updated daily and provides checks for the most recent security issues. In addition, it provides its own scripting language (NASL), which enables our staff to write custom plugins to test for mis-configurations. The company maintains a standalone workstation in the server room that has the latest version of Nessus and simply plugs it into the subnet that needs testing. While it is in the DMZ its update feature is used to download the latest plugins.

Test and Verify the Setup

Once the system has been setup an initial security audit should be done to ensure the configuration changes made earlier have been correctly implemented. Security mis-configurations can actually account for a good portion of the possible threats to a system and this step minimizes that from happening. For this testing GIAC enterprises Vmware test environment will be used, as it will be identical to our production server. For all of the tests below hermes.GIACenterprises.com is at 192.168.22.20 and test.GIACenterprises.com is at 192.168.22.19.

Port Scanning and OS detection

First a simple nmap TCP portscan of the server will be performed.

```
[root@test]# nmap –P0 192.168.22.20

Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) at 2004-08-12 11:11
EDT
All 1659 scanned ports on 192.168.22.20 are: filtered

Nmap run completed – 1 IP address (1 host up) scanned in 1362.054 seconds
```

A host firewall log entry would be entered for each packet dropped.

```
---snip---
Aug 12 11:12:10 hermes kernel: INPUT DROP: IN=eth0 OUT=
MAC=00:0c:29:6a:3f:00:8c:29:2a:52:d1:08:00 SRC=192.168.22.19
DST=192.168.22.20 LEN=28 TOS=0x00 PREC=0x00 TTL=38 ID=47553 PROTO=TCP
SPT=61653 DPT=21 WINDOW=3072 RES=0x00 SYN URGP=0
---snip---
```

This is good since TCP port 22 is only responding to internal administrator workstations. Also it is known that TCP port 6697 is open for our IRC server.

```
[root@test]# nmap –P0 –p 6697 192.168.22.20

Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) at 2004-08-12 11:32
EDT
Interesting ports on 192.168.22.20
PORT         STATE SERVICE
6697/tcp     open  unknown

Nmap run completed – 1 IP address (1 host up) scanned in 0.316 seconds
```

Next a simple nmap UDP portscan of the server will be performed.

```
[root@test]# nmap –P0 -sU 192.168.22.20

Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) at 2004-08-12 11:41
EDT
```

```
All 1478 scanned ports on 192.168.22.20 are: filtered

Nmap run completed – 1 IP address (1 host up) scanned in 1783.546 seconds
```

A host firewall log entry would be entered for each packet dropped.

```
---snip---
Aug 12 11:42:30 hermes kernel: INPUT DROP: IN=eth0 OUT=
MAC=00:0c:29:6a:3f:00:8c:29:2a:52:d1:08:00 SRC=192.168.22.19
DST=192.168.22.20 LEN=28 TOS=0x00 PREC=0x00 TTL=40 ID=59753 PROTO=UDP
SPT=43984 DPT=214 LEN=8
---snip---
```

It is known that Unreal IRCd uses its own DNS resolver that binds to a random UDP port. The following can be done to verify this:

```
[root@hermes]# netstat --inet –an
Active Internet connections (servers and established)
Proto Recv-Q Send-Q   Local Address     Foreign Address         State
tcp        0      0   0.0.0.0:6697      0.0.0.0:*               LISTEN
tcp        0      0   0.0.0.0:22        0.0.0.0:*               LISTEN
udp        0      0   0.0.0.0:1024      0.0.0.0:*
```

Next UDP port 1024 can be checked. This port may change each time IRCd is executed so this would have to be checked.

```
[root@test]# nmap –P0 -sU –p 1024 192.168.22.20

Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) at 2004-08-12 11:59
EDT
Interesting ports on 192.168.22.20
PORT            STATE SERVICE
1024/udp        open  unknown

Nmap run completed – 1 IP address (1 host up) scanned in 12.181 seconds
```

Next nmap should be used to attempt OS fingerprinting. To compare how effective grsecurity masks the OS, use nmap against the server with all of the networking enhancements including random IP IDs, RPC XIDs, TCP Initial Sequence Numbers, and altered Ping IDs disabled (via sysctl) and the netfilter firewall down. This scan included one closed and one open port.

```
[root@test]# nmap –P0 –p 20,6697 –O 192.168.22.20

Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) at 2004-08-13 09:57
EDT
Interesting ports on 192.168.22.20
PORT            STATE       SERVICE
20/tcp          closed      ftp-data
6697/tcp        open        unknown
Device type: general purpose
Running: Linux 2.4.X|2.5.X
OS details: Linux Kernel 2.4.18 – 2.5.70 (X86)
```

```
Uptime 1.073 days

Nmap run completed – 1 IP address (1 host up) scanned in 5.074 seconds
```

Next use nmap against the server with all of the networking enhancements enabled (via sysctl) and the netfilter firewall down.

```
[root@test]# nmap –P0 –p 20,6697 –O 192.168.22.20

Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) at 2004-08-13 10:14
EDT
Interesting ports on 192.168.22.20
PORT            STATE       SERVICE
20/tcp          closed      ftp-data
6697/tcp        open        unknown
Device type: general purpose
Running: Linux 2.4.X
OS details: Linux Kernel 2.4.22-ck2 (X86) w/grsecurity.org and HZ=1000
patches
Uptime 1.091 days

Nmap run completed – 1 IP address (1 host up) scanned in 5.245 seconds
```

Finally leave the networking enhancements enabled but also start the netfilter firewall.

```
[root@test]# nmap –P0 –p 20,6697 –O 192.168.22.20

Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) at 2004-08-13 10:19
EDT
Warning: OS detection will be MUCH less reliable because we did not find at
least 1 open and 1 closed TCP port
Interesting ports on 192.168.22.20
PORT            STATE       SERVICE
20/tcp          filtered    ftp-data
6697/tcp        open        unknown
Device type: general purpose
Running: Linux 2.4.X
OS details: Linux Kernel 2.4.20 - 2.4.22 w/grsecurity.org patch, Linux
Kernel 2.4.22-ck2 (X86) w/grsecurity.org and HZ=1000 patches

Nmap run completed – 1 IP address (1 host up) scanned in 6.444 seconds
```

It looks like nmap has added some signatures to assist with the fingerprinting of grsecurity. It looks like the OS was still distorted some since it showed up as a 2.4.X kernel and the server is currently running 2.6.7.

Vulnerability scan using Nessus

In addition to conducting a network wide security audit once a week, one is also done after a new server has been brought online. The company uses a standalone laptop for its scans as documented above. Installing nessus will not be covered in this document but installation information can be found in the following resources:

-

- http://www.nessus.org/nessus_2_0.html
- http://www.linuxsecurity.com/feature_stories/nessusintro-part1.html

Before each new nessus scan the operator should go ahead and ensure that the latest attack plugins have been installed by doing:

```
[root@test]# /usr/local/sbin/nessus-update-plugins
```

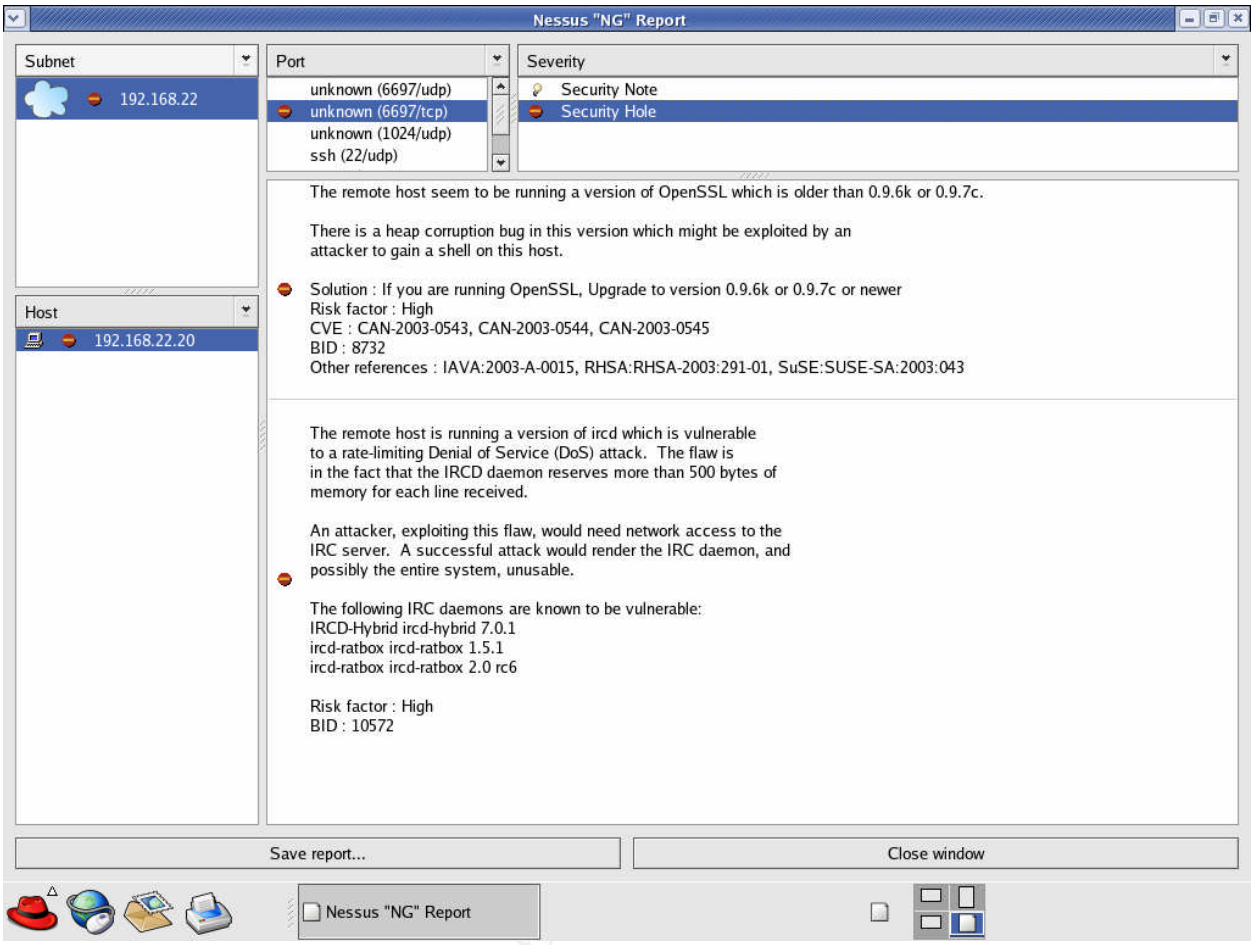
Login to the nessus client and since the test network is being scanned select the "Enable all" button to re-enable services that have the ability to crash remote services. The Preferences page should be modified as follows:

- TCP scanning technique:
  - SYN scan
  - Uncheck Ping the remote host
- Port range: User specified range
- Timing policy: Aggressive
- Ping the remote host: Remove Do a TCP ping

The Scan Options page should be modified as follows:

- Port range: 22, 1024, 6697
- Uncheck Optimize the test
- Uncheck Safe checks
- Port scanner:
  - Add check for Nmap
  - Remove check for Ping the remote host
  - Remove check for tcp connect() scan

Then put in the target IP address under Target selection and then press the "Start the Scan" button.



Nessus listed the following possible issues to look into. The first was running an OpenSSL version older than 0.9.7c. This server is using the OpenSSL-0.9.7a-33.10 rpm for SSH but the vulnerability was fixed in this version. The second was a possible rate-limiting Denial of Service attack on the IRCd. This server is using Unreal IRCd that is a derivative of EliteIRCd and not either IRCD-Hybrid or ircd-ratbox. Just to verify that code wasn't reused the proof of concept exploit located at http://downloads.securityfocus.com/vulnerabilities/exploits/h7kill.c was compiled and tested against the IRCd server. This exploit failed so it seems as this IRCd server is not vulnerable to this attack.

Verifying the PaX Settings

A regression test suite called Paxtest written by Peter Busser will be used to check for possible attack vectors that attackers could use to exploit software bugs. The following is a brief description of what each test does.

- Executable anonymous mapping - Tests if code can be executed in anonymous mappings (stack, brk(), and mmap() controlled heap).
- Executable bss – Tests if code in the .bss segment can be executed.
- Executable data – Tests if code in the .data segment can be executed.
- Executable heap – Tests if code in the heap can be executed.

- Executable stack – Tests if code on the stack can be executed.
- Executable anonymous mapping (mprotect) – Tests if code can be executed in anonymous mappings (stack, brk(), and mmap() controlled heap) after using mprotect() to make it executable.
- Executable bss (mprotect) - Tests if code in the .bss segment can be executed after using mprotect() to make it executable.
- Executable data (mprotect) - Tests if code in the .data segment can be executed after using mprotect() to make it executable.
- Executable heap (mprotect) - Tests if code in the heap can be executed after using mprotect() to make it executable.
- Executable shared library bss (mprotect) - Tests if code in the .bss segment of a shared library can be executed after using mprotect() to make it executable.
- Executable shared library data (mprotect) - Tests if code in the .data segment of a shared library can be executed after using mprotect() to make it executable.
- Executable stack (mprotect) - Tests if code on the stack can be executed after using mprotect() to make it executable.
- Anonymous mapping randomization test - Tests the randomization of anonymous mappings.
- Heap randomization test (ET_EXEC) – Tests the randomization of the heap of ET_EXEC main executable.
- Heap randomization test (ET_DYN) – Tests the randomization of the heap of ET_DYN main executable.
- Main executable randomization (ET_EXEC) – Tests the randomization of ET_EXEC main executable.
- Main executable randomization (ET_DYN) – Tests the randomization of ET_DYN main executable.
- Shared library randomization test – Tests the randomization of shared library loading.
- Stack randomization test (SEGMEXEC) – Tests the randomization of the stack pointer.
- Stack randomization test (PAGEEXEC) – Tests the randomization of the stack pointer.
- Return to function (strcpy) – Tests if return to function exploits work.
- Return to function (strcpy, RANDEXEC) – Tests if return to function exploits work.
- Return to function (memcpy) – Tests if return to function exploits using memcpy() work.
- Return to function (memcpy, RANDEXEC) – Tests if return to function exploits using memcpy() work.
- Executable shared library bss – Tests if code in the .bss segment of a shared library can be executed.
- Executable shared library data – Tests if code in the .data segment of a shared library can be executed.
- Writable text segments – Tests if a .text section can be written to.

The test suite is available from http://pax.grsecurity.net/paxtest-0.9.5.tar.gz.

```
[root@hermes]# Make generic
[root@hermes]# ./paxtest
<snip>
PaXtest – Copyright(c) 2003 by Peter Busser <peter@adamantix.org>
Released under the GNU Public Licence version 2 or later

Executable anonymous mapping                 : Killed
Executable bss                               : Killed
Executable data                              : Killed
Executable heap                              : Killed
Executable stack                             : Killed
Executable anonymous mapping (mprotect)      : Killed
Executable bss (mprotect)                    : Killed
Executable data (mprotect)                   : Killed
Executable heap (mprotect)                   : Killed
Executable shared library bss (mprotect)     : Killed
Executable shared library data (mprotect)    : Killed
Executable stack (mprotect)                  : Killed
Anonymous mapping randomisation test         : 15 bits (guessed)
Heap randomisation test (ET_EXEC)            : 13 bits (guessed)
Heap randomisation test (ET_DYN)             : 23 bits (guessed)
Main executable randomisation (ET_EXEC)      : 15 bits (guessed)
Main executable randomisation (ET_DYN)       : 15 bits (guessed)
Shared library randomisation test            : 15 bits (guessed)
Stack randomisation test (SEGMEXEC)          : 23 bits (guessed)
Stack randomisation test (PAGEEXEC)          : 24 bits (guessed)
Return to function (strcpy)                  : Vulnerable
Return to function (strcpy, RANDEXEC)        : Killed
Return to function (memcpy)                  : Vulnerable
Return to function (memcpy, RANDEXEC)        : Killed
Executable shared library bss                : Killed
Executable shared library data               : Killed
Writable text segments                       : Killed
```

Everything above looks good except for return to function (strcpy) and return to function (memcpy) that still show vulnerable. This is actually expected was included to show the need for a technology such as Stack-Smashing Protector (SSP). SSP is a compiler technology that makes use of canary values by rearranging local variables and function pointers to protect from many types of return-to-libc attacks.

RBAC Security

Since role based access control (RBAC) is a very important protection measure used on the server it should be tested as well. For this test a user will logon as root while RBAC is enabled and attempt a set of commands the root user would normally have access to.

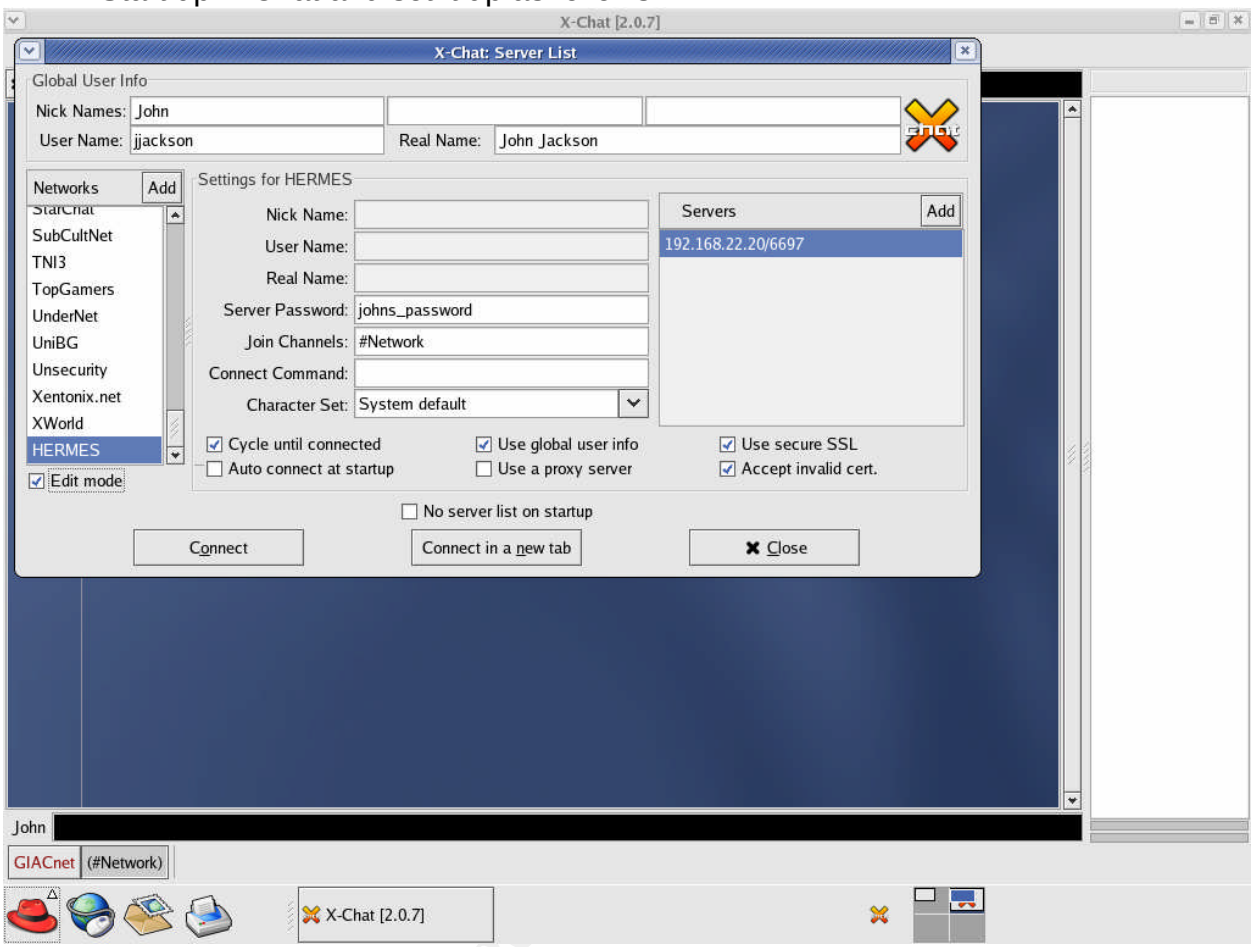
```
[root@hermes root]# ls /etc/grsec
grsec: (root:U:/bin/ls) denied access to hidden file /etc/grsec by
/bin/ls[ls:3436] uid/euid:0/0 gid/egid:0/0, parent /bin/bash[bash:27278]
uid/euid:0/0 gid/egid:0/0
ls: /etc/grsec: No such file or directory
```

```
[root@hermes root]# adduser
grsec: (root:U:/bin/bash) denied access to hidden file /usr/sbin/useradd by
/bin/bash[bash:27278] uid/euid:0/0 gid/egid:0/0, parent
/usr/bin/script[script:18401] uid/euid:0/0 gid/egid:0/0
bash: adduser: command not found

[root@hermes root]# cat /etc/shadow
grsec: (root:U:/) denied access to hidden file /etc/shadow by
/bin/cat[cat:31938] uid/euid:0/0 gid/egid:0/0, parent /bin/bash[bash:27278]
uid/euid:0/0 gid/egid:0/0
cat: /etc/shadow: No such file or directory

[root@hermes root]# ssh 192.168.22.19
grsec: (root:U:/) use of CAP_SETUID denied for /usr/bin/ssh[ssh:15681]
uid/euid:0/0 gid/egid:0/0, parent /bin/bash[bash:27278] uid/euid:0/0
gid/egid:0/0
grsec: (root:U:/) denied access to hidden file /etc/ssh/ssh_config by
/usr/bin/ssh[ssh:15681] uid/euid:0/0 gid/egid:0/0, parent
/bin/bash[bash:27278] uid/euid:0/0 gid/egid:0/0
grsec: (root:U:/) attempted socket(inet,stream,ip) by
/usr/bin/ssh[ssh:15681] uid/euid:0/0 gid/egid:0/0, parent
/bin/bash[bash:27278] uid/euid:0/0 gid/egid:0/0
socket: Permission denied ssh: connect to host 192.168.22.19 port 22:
Permission denied

[root@hermes root]# cat /etc/grub.conf
grsec: (root:U:/) denied open of /boot/grub/grub.conf for reading by
/bin/cat[cat:16333] uid/euid:0/0 gid/egid:0/0, parent /bin/bash[bash:27278]
uid/euid:0/0 gid/egid:0/0
cat: /etc/grub.conf: Permission denied

[root@hermes root]# dmesg
grsec: (root:U:/) use of CAP_SYS_ADMIN denied for /bin/dmesg[dmesg:6864]
uid/euid:0/0 gid/egid:0/0, parent /bin/bash[bash:27278] uid/euid:0/0
gid/egid:0/0
klogctl: Operation not permitted

[root@hermes root]# cd /proc/sys/
grsec: (root:U:/bin/bash) denied access to hidden file /proc/sys by
/bin/bash[bash:27278] uid/euid:0/0 gid/egid:0/0, parent
/usr/bin/script[script:18401] uid/euid:0/0 gid/egid:0/0
grsec: (root:U:/bin/bash) denied access to hidden file /proc/sys by
/bin/bash[bash:27278] uid/euid:0/0 gid/egid:0/0, parent
/usr/bin/script[script:18401] uid/euid:0/0 gid/egid:0/0
bash: cd: /proc/sys/: No such file or directory
```

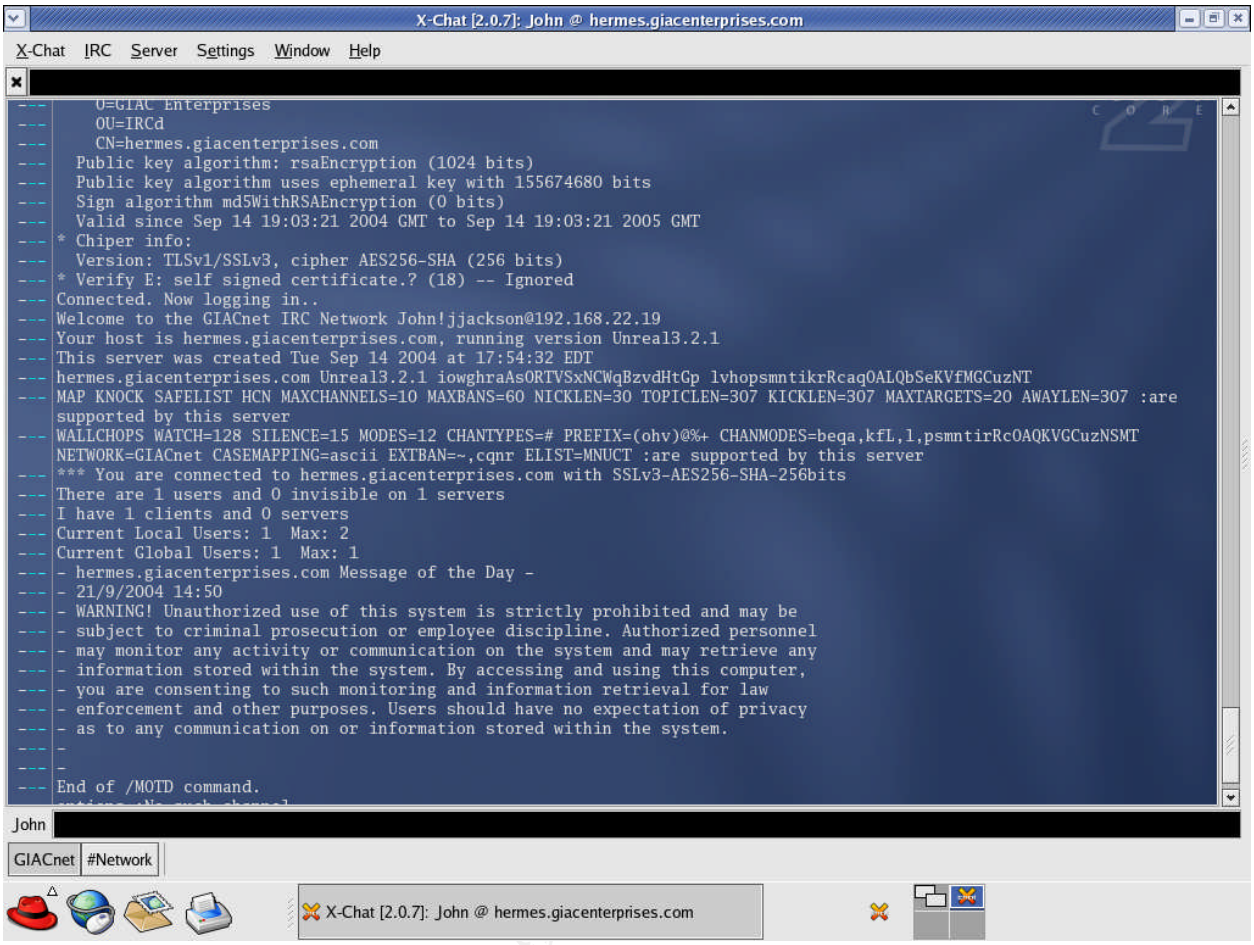
IRCd User Authentication

Since IRC servers are typically wide open for all to use, the user authentication for this server needs to be verified. This one will only allow authorized users with a predefined username and password. Authentication with Unreal IRCd has been tested on both mIRC and X-Chat clients. For this test X-Chat will be used to connect with an authorized user (John Jackson) and then with an unauthorized user.

1. Start up X-Chat and set it up as follows:



2. After connecting the following should display:



If an unauthorized user attempts to log on with either a bogus userid or password, it does not get logged in the ircd.log. If an unauthorized user attempts to connect with a bogus userid, the operators logged on the server will see the following error:

```
-hermes.giacenterprises.com- *** Notice – Unauthorized connection from
User[192.168.22.19].
```

If an unauthorized user attempts to connect with a valid userid but incorrect password, again no error message on the server or ircd.log is shown. This can open up the possibility of an attacker using a brute force method to crack a users password. This IRCd server has 2 main layers as most do, user clients and operators. To become an operator the user must log in as a user first and then asks the server for operator privileges using “/oper <userid> <password>”. All successful attempts are logged in ircd.log:

```
[Thurs Aug 12 16:13:29 2004] – OPER (adamchan) by
(achan!achan@192.168.22.19)
```

As well as all unsuccessful attempts:

```
[Thurs Aug 12 16:15:11 2004] – OPER FAILEDAUTH(adamchan) by
(achan!achan@192.168.22.19)
```

This test validates that only users with valid userids can connect to the server. Using public-key encryption for user authentication is being investigated and will be implemented as soon as possible. This will eliminate the ability of an attacker to do a brute force attack against the server.

Appendix A – Server Firewall configuration

/etc/init.d/netfilter

```
#!/bin/bash
#
# chkconfig: 2345 95 95
# (run in init levels 2345 with start priority of 95 and stop priority of 95)
# (since no lockfile stop priority will never be used)
# processname: iptables
#
# description: Startup script for NetFilter firewall for Servers
# Replacement for the default iptables script

# source function library
. /etc/rc.d/init.d/functions

# Setup global variables
RETVAL=0
IPT=/sbin/iptables
LOGLEVEL=notice
DEBUG=0
SERVERIP=<Server IP>
EXTDNSIP=<External DNS/NTP server IP>
ADMINIP=<Admin workstation IP>
ADMIN2IP=<2nd Admin workstation IP>

# Define Interfaces
EXTIF=eth0

# Ensure iptables command is available
if [ ! -x $IPT ]; then
      echo "ERROR: Cannot locate iptables command."
      exit 0
fi

# Ensure ipchains command is not running
if  /sbin/lsmod 2>/dev/null |grep -q ipchains ; then
      echo "ERROR: Cannot run both ipchains and iptables"
      exit 0
fi

start() {

      # Log to screen and syslog the start message
      echo –n $"Starting NetFilter Firewall: "
      logger -p local3.info NetFilter Firewall Started

      # Enable syn cookie support
      echo 1 > /proc/sys/net/ipv4/tcp_syncookies

      # Enable logging of Martians (oddball addresses).
      echo 1 > /proc/sys/net/ipv4/conf/all/log_martians
```

```
# Enable IP Forward support.
echo 1 > /proc/sys/net/ipv4/ip_forward

# Disable response to ping
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all

# Disable response to Broadcasts (to keep from being SMURF amplifier)
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

# Enable Dynamic IP Address support.
echo 1 > /proc/sys/net/ipv4/ip_dynaddr

# Enable anti spoofing support.
echo 1 > /proc/sys/net/ipv4/conf/all/rp_filter

# Disable source routing support.
echo 0 > /proc/sys/net/ipv4/conf/all/accept_source_route

# Disable icmp redirect support.
echo 0 > /proc/sys/net/ipv4/conf/all/accept_redirects

###################################
# Flush,delete and zero existing chains
###################################

$IPT -t filter -F
$IPT -t nat -F
$IPT -t mangle -F

$IPT -X
$IPT -Z

###################################
# Set up default policies
###################################

$IPT -t nat -P PREROUTING ACCEPT
$IPT -t nat -P POSTROUTING ACCEPT
$IPT -t nat -P OUTPUT ACCEPT

$IPT -t mangle -P PREROUTING ACCEPT
$IPT -t mangle -P INPUT ACCEPT
$IPT -t mangle -P FORWARD ACCEPT
$IPT -t mangle -P OUTPUT ACCEPT
$IPT -t mangle -P POSTROUTING ACCEPT

$IPT -t filter -P FORWARD DROP
$IPT -t filter -P INPUT DROP
$IPT -t filter -P OUTPUT ACCEPT

###################################
# nat Module Rules
###################################

# This table is consulted when a packet which creates a new
# connection is encountered. Use PREROUTING to drop unwanted new
# traffic before passing on to FORWARD or INPUT Filters.
```

```
################################
# Mangle Module Rules.
################################

# Rules to mangle TOS values of packets routed through the firewall.
# Based on RFC 1060/1349

# TOS stuff: (type: iptables -m tos -h)
# Minimize-Delay 16 (0x10)
# Maximize-Throughput 8 (0x08)
# Maximize-Reliability 4 (0x04)
# Minimize-Cost 2 (0x02)
# Normal-Service 0 (0x00)

$IPT -t mangle -A PREROUTING -p tcp --dport 22 -j TOS --set-tos 16
$IPT -t mangle -A PREROUTING -p tcp --dport 53 -j TOS --set-tos 16
$IPT -t mangle -A PREROUTING -p udp --dport 53 -j TOS --set-tos 16
$IPT -t mangle -A PREROUTING -p udp --dport 514 -j TOS --set-tos 16

###################################
# Filter Module Rules
###################################

# Setup debugging for tracking traffic flows
if [ $DEBUG = 1 ]; then
      echo "Extra Debug logging enabled"
      $IPT -A INPUT -i lo -j LOG --log-level $LOGLEVEL \
            --log-prefix "local-in: "
      $IPT -A OUTPUT -o lo -j LOG --log-level $LOGLEVEL \
            --log-prefix "local-out: "
      $IPT -A INPUT -p udp --sport 53 --dport 53 -j LOG \
            --log-level $LOGLEVEL --log-prefix "DNS: "
      $IPT -A INPUT -p udp --sport 123 --dport 123 -j LOG \
            --log-level $LOGLEVEL --log-prefix "NTP: "
      $IPT -A INPUT -p udp --sport 514 --dport 514 -j LOG \
            --log-level $LOGLEVEL --log-prefix "SYSLOG: "
      $IPT -A INPUT -p tcp --dport 22 -j LOG \
            --log-level $LOGLEVEL --log-prefix "SSH: "
      $IPT -A INPUT -p tcp --dport 6697 -j LOG \
            --log-level $LOGLEVEL --log-prefix "IRCd: "
fi

## End additional logging

# Allow traffic on loopback interface
$IPT -A INPUT -i lo -j ACCEPT
$IPT -A OUTPUT -o lo -j ACCEPT

# Allow incoming traffic if is part of an previous connection
$IPT -A INPUT -p TCP -m state --state ESTABLISHED,RELATED -j ACCEPT

# DNS traffic between IRCD server and External DNS/NTP server
$IPT -A OUTPUT -o $EXTIF -p udp -s $SERVERIP -d $EXTDNSIP \
      --dport 53 -j ACCEPT
$IPT -A INPUT -i $EXTIF -p udp -s $EXTDNSIP --sport 53 \
      -d $SERVERIP -j ACCEPT
```

```
      # NTP traffic between IRCD server and External DNS/NTP server
      $IPT -A OUTPUT -o $EXTIF -p udp -s $SERVERIP -d $EXTDNSIP \
            --dport 123 -j ACCEPT
      $IPT -A INPUT -i $EXTIF -p udp -s $EXTDNSIP --sport 123 \
            -d $SERVERIP -j ACCEPT

      # SYSLOG traffic between IRCD server and Syslog server
      $IPT -A OUTPUT -o $EXTIF -p udp -s $SERVERIP -d 192.168.24.2 \
            --dport 514 -j ACCEPT

      # SSH traffic between IRCD server and Administration workstation
      $IPT -A INPUT -i $EXTIF -p tcp -s $ADMINIP -d $SERVERIP \
            --dport 22 -j ACCEPT
      $IPT -A OUTPUT -o $EXTIF -p tcp -s $SERVERIP --sport 22 \
            -d $ADMINIP -j ACCEPT
      $IPT -A INPUT -i $EXTIF -p tcp -s $ADMIN2IP -d $SERVERIP \
            --dport 22 -j ACCEPT
      $IPT -A OUTPUT -o $EXTIF -p tcp -s $SERVERIP --sport 22 \
            -d $ADMIN2IP -j ACCEPT

      # IRCD traffic between IRCD server and Everyone
      $IPT -A INPUT -i $EXTIF -p tcp -d $SERVERIP --dport 6697 -j ACCEPT
      $IPT -A OUTPUT -o $EXTIF -p tcp -s $SERVERIP --sport 6697 -j ACCEPT

      # Drop known traffic without logging
      # Micrsoft networking
      $IPT -A INPUT -i $EXTIF -p udp --dport 137 -j DROP
      $IPT -A INPUT -i $EXTIF -p udp --dport 138 -j DROP

      # LOG all packets that are dropped due to default FORWARD POLICY
      $IPT -A FORWARD -j LOG --log-level $LOGLEVEL \
            --log-prefix "FORWARD DROP:"

      # LOG all packets that are dropped due to default INPUT POLICY
      $IPT -A INPUT -j LOG --log-level $LOGLEVEL \
            --log-prefix "INPUT DROP:"

      RETVAL=$?
}

stop() {

      DEBUG=0

      # Send message to screen and syslog
      echo –n $"Stopping NetFilter Firewall: "
      logger -p local3.info NetFilter Firewall Stopped

      ####################################
      # Filter Module Rules
      ####################################

      # Disable syn cookie support
      echo 0 > /proc/sys/net/ipv4/tcp_syncookies

      # Disable logging of Martians (oddball addresses).
```

```
        echo 0 > /proc/sys/net/ipv4/conf/all/log_martians

        # Enable response to ping
        echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_all

        # Disable Dynamic IP Address support.
        echo 0 > /proc/sys/net/ipv4/ip_dynaddr

        # Disable anti spoofing support.
        echo 0 > /proc/sys/net/ipv4/conf/all/rp_filter

        # Enable source routing support.
        echo 1 > /proc/sys/net/ipv4/conf/all/accept_source_route

        # Enable icmp redirect support.
        echo 1 > /proc/sys/net/ipv4/conf/all/accept_redirects

        # Keep IP forwarding enabled
        echo 1 > /proc/sys/net/ipv4/ip_forward


        ###################################
        # Flush,delete and zero existing chains
        ###################################

        $IPT -t filter -F
        $IPT -t nat -F
        $IPT -t mangle -F

        $IPT -X
        $IPT -Z

        ###################################
        # Set up default policies
        ###################################

        $IPT -t nat -P PREROUTING ACCEPT
        $IPT -t nat -P POSTROUTING ACCEPT
        $IPT -t nat -P OUTPUT ACCEPT

        $IPT -t mangle -P PREROUTING ACCEPT
        $IPT -t mangle -P INPUT ACCEPT
        $IPT -t mangle -P FORWARD ACCEPT
        $IPT -t mangle -P OUTPUT ACCEPT
        $IPT -t mangle -P POSTROUTING ACCEPT

        $IPT -t filter -P FORWARD ACCEPT
        $IPT -t filter -P INPUT ACCEPT
        $IPT -t filter -P OUTPUT ACCEPT

        RETVAL=$?
}

debug() {
        # Sets Debug Flag to 1 to turn on Debugging
        echo "Starting NetFilter Firewall with Debug Flag"
        logger -p local3.info NetFilter Firewall Started with Debug Flag
```

```
      DEBUG=1
}

case "$1" in
  start)
      start
      ;;

  stop)
      stop
      ;;

  restart)
      # Basically perform a stop() then start()
      stop
      start
      ;;

  debug)
      # Sets up normal firewall with additional debugging
      stop
      debug
      start
      ;;

  status)

      echo "-----------------------------------"
      echo "NAT table"
      $IPT -t nat -nvxL --line-number
      echo "-----------------------------------"
      echo "Mangle table"
      $IPT -t mangle -vxL --line-number
      echo "-----------------------------------"
      echo "Filter table:"
      $IPT -t filter -nvxL --line-number
      ;;

  panic)

      echo "Initating panic mode"
      echo "Changing target policies to DROP "
      logger -p local3.info NetFilter Firewall initated PANIC mode

      $IPT -t filter -P INPUT DROP
      $IPT -t filter -P FORWARD DROP
      $IPT -t filter -P OUTPUT DROP
      $IPT -t nat -P PREROUTING DROP
      $IPT -t nat -P POSTROUTING DROP
      $IPT -t nat -P OUTPUT DROP
      $IPT -t mangle -P PREROUTING DROP
      $IPT -t mangle -P OUTPUT DROP
      $IPT -t mangle -P POSTROUTING DROP
      $IPT -t mangle -P INPUT DROP
      $IPT -t mangle -P FORWARD DROP

      echo "Flushing all chains"
```

```
      $IPT -t filter -F
      $IPT -t nat -F
      $IPT -t mangle -F

      echo "Removing user defined chains"
      $IPT -X

      # Disable IP Forward support.
      echo 0 > /proc/sys/net/ipv4/ip_forward
      ;;
  *)
      echo " Usage: $0 {start|stop|restart|debug|status|panic} "
      exit 1

esac
exit 0
```

Appendix B – Grsecurity Kernel Configuration Settings

Grsecurity (GRKERNSEC)

Setting	Description	Comments
Security Level		
Low (GRKERNSEC_LOW)	If you choose this option, several of the grsecurity options will be enabled that will give you greater protection against a number of attacks, while assuring that none of your software will have any conflicts with the additional security measures. If you run a lot of unusual software, or you are having problems with the higher security levels, you should say Y here. With this option, the following features are enabled: - linking restrictions - fifo restrictions - random pids - enforcing nproc on execve() - restricted dmesg - random ip ids - enforced chdir("/") on chroot	**Not Selected.**
Medium (GRKERNSEC_MEDIUM)	If you say Y here, several features in addition to those included in the low additional security level will be enabled. These features provide even more security to your system, though in rare cases they may be incompatible with very old or poorly written software. If you enable this option, make sure that your auth service (identd) is running as gid 10 (usually group wheel). With this option the following features (in addition to those provided in the low additional security level) will be enabled: - random tcp source ports - failed fork logging - time change logging - signal logging - deny mounts in chroot - deny double chrooting - deny sysctl writes in chroot - deny mknod in chroot - deny access to abstract AF_UNIX sockets out of chroot - deny pivot_root in chroot - denied writes of /dev/kmem, /dev/mem, and /dev/port - /proc restrictions with special gid set to 10 (usually wheel) - address space layout randomization - removal of addresses from /proc/<pid>/[maps|stat]	**Not Selected.**
High (GRKERNSEC_HIGH)	If you say Y here, many of the features of grsecurity will be	**Not Selected.**

	enabled, that will protect you against many kinds of attacks against your system. The heightened security comes at a cost of an increased chance of incompatibilities with rare software on your machine. Since this security level enables PaX, you should view <http://pax.grsecurity.net> and read about the PaX project. While you are there, download chpax and run it on binaries that cause problems with PaX. Also remember that since the /proc restrictions are enabled, you must run your identd as group wheel (gid 10). This security level enables the following features in addition to those listed in the low and medium security levels: - additional /proc restrictions - chmod restrictions in chroot - no signals, ptrace, or viewing processes outside of chroot - capability restrictions in chroot - deny fchdir out of chroot - priority restrictions in chroot - segmentation-based implementation of PaX - mprotect restrictions - kernel stack randomization - mount/unmount/remount logging - kernel symbol hiding	
Custom (GRKERNSEC_CUSTOM)	If you say Y here, you will be able to configure every grsecurity option, which allows you to enable many more features that aren't covered in the basic security levels. These additional features include TPE, socket restrictions, and the sysctl system for grsecurity. It is advised that you read through the help for each option to determine its usefulness in your situation.	**Selected.**
Address Space Protection		
Deny writing to /dev/kmem, /dev/mem, and /dev/port (GRKERNSEC_KMEM)	If you say Y here, /dev/kmem and /dev/mem won't be allowed to be written to via mmap or otherwise to modify the running kernel. /dev/port will also not be allowed to be opened. If you have module support disabled, enabling this will close up four ways that are currently used to insert malicious code into the running kernel. Even with all these features enabled, we still highly recommend that you use the ACL system, as it is still possible for an attacker to modify the running kernel through privileged I/O granted by ioperm/iopl. If you are not using XFree86, you may be able to stop this additional case by enabling the 'Disable privileged I/O' option. Though nothing legitimately writes to /dev/kmem, XFree86 does need to write to /dev/mem, but only to video memory, which is the only writing we allow in this case. If /dev/kmem or /dev/mem are mmaped without PROT_WRITE, they will not be allowed to mprotect it with PROT_WRITE later. Enabling this feature could make certain apps like VMWare stop working, as they need to write to other locations in /dev/mem. It is highly recommended that you say Y here if you meet all the conditions above.	**Selected.** Do not need to run Xfree86 or Vmware.
Disabled privileged I/O (GRKERNSEC_IO)	If you say Y here, all ioperm and iopl calls will return an error. Ioperm and iopl can be used to modify the running kernel. Unfortunately, some programs need this access to operate properly, the most notable of which are XFree86 and hwclock. hwclock can be remedied by having RTC support in the kernel, so CONFIG_RTC is enabled if this option is enabled, to ensure that hwclock operates correctly. XFree86 still will not operate correctly with this option enabled, so DO NOT CHOOSE Y IF YOU USE XFree86. If you use XFree86 and you still want to protect your kernel against modification, use the ACL system.	**Selected.** Are not using XFree86 and CONFIG_RTC is enabled.
Remove addresses from /proc/<pid>/[maps|stat] (GRKERNSEC_PROC_MEMMAP)	If you say Y here, the /proc/<pid>/maps and /proc/<pid>/stat files will give no information about the addresses of its mappings if PaX features that rely on random addresses are enabled on the task. If you use PaX it is greatly recommended that you say Y here as it closes up a hole that makes the full ASLR useless for suid binaries.	**Selected.**
Deter exploit bruteforcing	If you say Y here attempts to bruteforce exploits against	**Selected.** Signal

(GRKERNSEC_BRUTE)	forking daemons such as apache or sshd will be deterred. When a child of a forking daemon is killed by PaX or crashes due to an illegal instruction, the parent process will be delayed 30 seconds upon every subsequent fork until the administrator is able to assess the situation and restart the daemon. It is recommended that you also enable signal logging in the auditing section so that logs are generated when a process performs an illegal instruction.	logging has also been selected.
Hide kernel symbols (GRKERNSEC_HIDESYM)	If you say Y here, getting information on loaded modules, and displaying all kernel symbols through a syscall will be restricted to users with CAP_SYS_MODULE. This option is only effective provided the following conditions are met: 1) The kernel using grsecurity is not precompiled by some distribution 2) You are using the ACL system and hiding other files such as your kernel image and System.map 3) You have the additional /proc restrictions enabled, which removes /proc/kcore If the above conditions are met, this option will aid to provide a useful protection against local and remote kernel exploitation of overflows and arbitrary read/write vulnerabilities.	**Selected.**
Role Based Access Control Options		
Hide kernel processes (GRKERNSEC_ACL_HIDEKERN)	If you say Y here, when the ACL system is enabled via gradm -E, an additional ACL will be passed to the kernel that hides all kernel processes. These processes will only be viewable by the authenticated admin, or processes that have viewing access set.	**Selected.**
Maximum tries before password lockout (GRKERNSEC_ACL_MAXTRIES)	This option enforces the maximum number of times a user can attempt to authorize themselves with the grsecurity ACL system before being denied the ability to attempt authorization again for a specified time. The lower the number, the harder it will be to brute-force a password.	**Use the default of 3 tries.**
Time to wait after max password tries, in seconds (GRKERNSEC_ACL_TIMEOUT)	This option specifies the time the user must wait after attempting to authorize to the ACL system with the maximum number of invalid passwords. The higher the number, the harder it will be to brute-force a password.	**Change to 3 minutes or 180 seconds.**
Filesystem Protections		
Proc restrictions (GRKERNSEC_PROC)	If you say Y here, the permissions of the /proc filesystem will be altered to enhance system security and privacy. Depending upon the options you choose, you can either restrict users to see only the processes they themselves run, or choose a group that can view all processes and files normally restricted to root if you choose the "restrict to user only" option. NOTE: If you're running identd as a non-root user, you will have to run it as the group you specify here.	**Selected.**
Restrict /proc to user only (GRKERNSEC_PROC_USER)	If you say Y here, non-root users will only be able to view their own processes, and restricts them from viewing network-related information, and viewing kernel symbol and module information.	**Selected.** Only available after selecting 'Proc restrictions.'
Allow special groups (GRKERNSEC_PROC_USERGROUP)	If you say Y here, you will be able to select a group that will be able to view all processes, network-related information, and kernel and symbol information. This option is useful if you want to run identd as a non-root user.	**Not Selected.** Only available if 'Restrict /proc to user only' is not selected.
Linking restrictions (GRKERNSEC_LINK)	If you say Y here, /tmp race exploits will be prevented, since users will no longer be able to follow symlinks owned by other users in world-writable +t directories (i.e. /tmp), unless the owner of the symlink is the owner of the directory. users will also not be able to hardlink to files they do not own. If the sysctl option is enabled, a sysctl option with name "linking_restrictions" is created.	**Selected.**
FIFO restrictions (GRKERNSEC_FIFO)	If you say Y here, users will not be able to write to FIFOs they don't own in world-writable +t directories (i.e. /tmp), unless the owner of the FIFO is the same owner of the directory it's held in. If the sysctl option is enabled, a sysctl option with name "fifo_restrictions" is created.	**Selected.**
Chroot jail restrictions	If you say Y here, you will be able to choose several options	**Not Selected.** Not

(GRKERNSEC_CHROOT)	that will make breaking out of a chrooted jail much more difficult. If you encounter no software incompatibilities with the following options, it is recommended that you enable each one.	using chroot.
Deny mounts (GRKERNSEC_CHROOT_MOUNT)	If you say Y here, processes inside a chroot will not be able to mount or remount filesystems. If the sysctl option is enabled, a sysctl option with name "chroot_deny_mount" is created.	**Not Selected.** Only available after selecting 'Chroot jail restrictions.'
Deny double_chroot (GRKERNSEC_CHROOT_DOUBLE)	If you say Y here, processes inside a chroot will not be able to chroot again outside of the chroot. This is a widely used method of breaking out of a chroot jail and should not be allowed. If the sysctl option is enabled, a sysctl option with name "chroot_deny_chroot" is created.	**Not Selected.** Only available after selecting 'Chroot jail restrictions.'
Deny pivot_root in chroot (GRKERNSEC_CHROOT_PIVOT)	If you say Y here, processes inside a chroot will not be able to use a function called pivot_root() that was introduced in Linux 2.3.41. It works similar to chroot in that it changes the root filesystem. This function could be misused in a chrooted process to attempt to break out of the chroot, and therefore should not be allowed. If the sysctl option is enabled, a sysctl option with name "chroot_deny_pivot" is created.	**Not Selected.** Only available after selecting 'Chroot jail restrictions.'
Enforce chdir ("/") on all cheroots (GRKERNSEC_CHROOT_CHDIR)	If you say Y here, the current working directory of all newly-chrooted applications will be set to the the root directory of the chroot. The man page on chroot(2) states: Note that this call does not change the current working directory, so that `.' can be outside the tree rooted at `/'. In particular, the super-user can escape from a `chroot jail' by doing `mkdir foo; chroot foo; cd ..'. It is recommended that you say Y here, since it's not known to break any software. If the sysctl option is enabled, a sysctl option with name "chroot_enforce_chdir" is created.	**Not Selected.** Only available after selecting 'Chroot jail restrictions.'
Deny (f)chmod+s (GRKERNSEC_CHROOT_CHMOD)	If you say Y here, processes inside a chroot will not be able to chmod or fchmod files to make them have suid or sgid bits. This protects against another published method of breaking a chroot. If the sysctl option is enabled, a sysctl option with name "chroot_deny_chmod" is created.	**Not Selected.** Only available after selecting 'Chroot jail restrictions.'
Deny fchdir out of chroot (GRKERNSEC_CHROOT_FCHDIR)	If you say Y here, a well-known method of breaking chroots by fchdir'ing to a file descriptor of the chrooting process that points to a directory outside the filesystem will be stopped. If the sysctl option is enabled, a sysctl option with name "chroot_deny_fchdir" is created.	**Not Selected.** Only available after selecting 'Chroot jail restrictions.'
Deny mknod (GRKERNSEC_CHROOT_MKNOD)	If you say Y here, processes inside a chroot will not be allowed to mknod. The problem with using mknod inside a chroot is that it would allow an attacker to create a device entry that is the same as one on the physical root of your system, which could range from anything from the console device to a device for your harddrive (which they could then use to wipe the drive or steal data). It is recommended that you say Y here, unless you run into software incompatibilities. If the sysctl option is enabled, a sysctl option with name "chroot_deny_mknod" is created.	**Not Selected.** Only available after selecting 'Chroot jail restrictions.'
Deny shmat() out of chroot (GRKERNSEC_CHROOT_SHMAT)	If you say Y here, processes inside a chroot will not be able to attach to shared memory segments that were created outside of the chroot jail. It is recommended that you say Y here. If the sysctl option is enabled, a sysctl option with name "chroot_deny_shmat" is created.	**Not Selected.** Only available after selecting 'Chroot jail restrictions.'
Deny access to abstract AF_UNIX sockets out of chroot (GRKERNSEC_CHROOT_UNIX)	If you say Y here, processes inside a chroot will not be able to connect to abstract (meaning not belonging to a filesystem) Unix domain sockets that were bound outside of a chroot. It is recommended that you say Y here. If the sysctl option is enabled, a sysctl option with name "chroot_deny_unix" is created.	**Not Selected.** Only available after selecting 'Chroot jail restrictions.'
Protect outside processes (GRKERNSEC_CHROOT_FINDTASK)	If you say Y here, processes inside a chroot will not be able to kill, send signals with fcntl, ptrace, capget, setpgid, getpgid, getsid, or view any process outside of the chroot. If the sysctl option is enabled, a sysctl option with name "chroot_findtask" is created.	**Not Selected.** Only available after selecting 'Chroot jail restrictions.'
Restrict priority changes (GRKERNSEC_CHROOT_NICE)	If you say Y here, processes inside a chroot will not be able to raise the priority of processes in the chroot, or alter the priority of processes outside the chroot. This provides more security	**Not Selected.** Only available after selecting 'Chroot

	than simply removing CAP_SYS_NICE from the process' capability set. If the sysctl option is enabled, a sysctl option with name "chroot_restrict_nice" is created.	jail restrictions.'
Deny sysctl writes (GRKERNSEC_CHROOT_SYSCTL)	If you say Y here, an attacker in a chroot will not be able to write to sysctl entries, either by sysctl(2) or through a /proc interface. It is strongly recommended that you say Y here. If the sysctl option is enabled, a sysctl option with name "chroot_deny_sysctl" is created.	**Not Selected.** Only available after selecting 'Chroot jail restrictions.'
Capability restrictions (GRKERNSEC_CHROOT_CAPS)	If you say Y here, the capabilities on all root processes within a chroot jail will be lowered to stop module insertion, raw i/o, system and net admin tasks, rebooting the system, modifying immutable files, modifying IPC owned by another, and changing the system time. This is left an option because it can break some apps. Disable this if your chrooted apps are having problems performing those kinds of tasks. If the sysctl option is enabled, a sysctl option with name "chroot_caps" is created.	**Not Selected.** Only available after selecting 'Chroot jail restrictions.'
Kernel Auditing		
Single group for auditing (GRKERNSEC_AUDIT_GROUP)	If you say Y here, the exec, chdir, (un)mount, and ipc logging features will only operate on a group you specify. This option is recommended if you only want to watch certain users instead of having a large amount of logs from the entire system. If the sysctl option is enabled, a sysctl option with name "audit_group" is created.	**Not Selected.**
Exec logging (GRKERNSEC_EXECLOG)	If you say Y here, all execve() calls will be logged (since the other exec*() calls are frontends to execve(), all execution will be logged). Useful for shell-servers that like to keep track of their users. If the sysctl option is enabled, a sysctl option with name "exec_logging" is created. WARNING: This option when enabled will produce a LOT of logs, especially on an active system.	**Selected.**
Resource logging (GRKERNSEC_RESLOG)	If you say Y here, all attempts to overstep resource limits will be logged with the resource name, the requested size, and the current limit. It is highly recommended that you say Y here.	**Selected.**
Log execs within chroot (GRKERNSEC_CHROOT_EXECLOG)	If you say Y here, all executions inside a chroot jail will be logged to syslog. This can cause a large amount of logs if certain applications (eg. djb's daemontools) are installed on the system, and is therefore left as an option. If the sysctl option is enabled, a sysctl option with name "chroot_execlog" is created.	**Not Selected.** Not using chroot.
Chdir logging (GRKERNSEC_AUDIT_CHDIR)	If you say Y here, all chdir() calls will be logged. If the sysctl option is enabled, a sysctl option with name "audit_chdir" is created.	**Selected.**
(Un)Mount logging (GRKERNSEC_AUDIT_MOUNT)	If you say Y here, all mounts and unmounts will be logged. If the sysctl option is enabled, a sysctl option with name "audit_mount" is created.	**Selected.**
IPC logging (GRKERNSEC_AUDIT_IPC)	If you say Y here, creation and removal of message queues, semaphores, and shared memory will be logged. If the sysctl option is enabled, a sysctl option with name "audit_ipc" is created.	**Selected.**
Signal logging (GRKERNSEC_SIGNAL)	If you say Y here, certain important signals will be logged, such as SIGSEGV, which will as a result inform you of when a error in a program occurred, which in some cases could mean a possible exploit attempt. If the sysctl option is enabled, a sysctl option with name "signal_logging" is created.	**Selected.**
Fork failure logging (GRKERNSEC_FORKFAIL)	If you say Y here, all failed fork() attempts will be logged. This could suggest a fork bomb, or someone attempting to overstep their process limit. If the sysctl option is enabled, a sysctl option with name "forkfail_logging" is created.	**Selected.**
Time change logging (GRKERNSEC_TIME)	If you say Y here, any changes of the system clock will be logged. If the sysctl option is enabled, a sysctl option with name "timechange_logging" is created.	**Selected.**
/proc/<pid>/ipaddr support (GRKERNSEC_PROC_IPADDR)	If you say Y here, a new entry will be added to each /proc/<pid> directory that contains the IP address of the person using the task. The IP is carried across local TCP and AF_UNIX stream sockets. This information can be useful for IDS/IPSes to perform remote response to a local attack. The entry is readable by only the owner of the process (and root if he has CAP_DAC_OVERRIDE, which can be removed via the	**Selected.**

	RBAC system), and thus does not create privacy concerns.	
Executable Protections		
Enforce RLIMIT_NPROC on execs (GRKERNSEC_EXECVE)	If you say Y here, users with a resource limit on processes will have the value checked during execve() calls. The current system only checks the system limit during fork() calls. If the sysctl option is enabled, a sysctl option with name "execve_limiting" is created.	**Selected.**
Dmesg(8) restriction (GRKERNSEC_DMESG)	If you say Y here, non-root users will not be able to use dmesg(8) to view up to the last 4kb of messages in the kernel's log buffer. If the sysctl option is enabled, a sysctl option with name "dmesg" is created.	**Selected.**
Randomized PIDs (GRKERNSEC_RANDPID)	If you say Y here, all PIDs created on the system will be pseudo-randomly generated. This is extremely effective along with the /proc restrictions to disallow an attacker from guessing pids of daemons, etc. PIDs are also used in some cases as part of a naming system for temporary files, so this option would keep those filenames from being predicted as well. We also use code to make sure that PID numbers aren't reused too soon. If the sysctl option is enabled, a sysctl option with name "rand_pids" is created.	**Selected.**
Trusted Path Execution (TPE) (GRKERNSEC_TPE)	If you say Y here, you will be able to choose a gid to add to the supplementary groups of users you want to mark as "untrusted." These users will not be able to execute any files that are not in root-owned directories writable only by root. If the sysctl option is enabled, a sysctl option with name "tpe" is created.	**Not Selected.**
Partially restrict non-root users (GRKERNSEC_TPE_ALL)	If you say Y here, All non-root users other than the ones in the group specified in the main TPE option will only be allowed to execute files in directories they own that are not group or world-writable, or in directories owned by root and writable only by root. If the sysctl option is enabled, a sysctl option with name "tpe_restrict_all" is created.	**Not Selected.**
Network Protections		
Larger entropy pools (GRKERNSEC_RANDNET)	If you say Y here, the entropy pools used for many features of Linux and grsecurity will be doubled in size. Since several grsecurity features use additional randomness, it is recommended that you say Y here. Saying Y here has a similar effect as modifying /proc/sys/kernel/random/poolsize.	**Selected.**
Truly random TCP ISN selection (GRKERNSEC_RANDISN)	If you say Y here, Linux's default selection of TCP Initial Sequence Numbers (ISNs) will be replaced with that of OpenBSD. Linux uses an MD4 hash based on the connection plus a time value to create the ISN, while OpenBSD's selection is random. If the sysctl option is enabled, a sysctl option with name "rand_isns" is created.	**Selected.**
Randomized IP IDs (GRKERNSEC_RANDID)	If you say Y here, all the id field on all outgoing packets will be randomized. This hinders os fingerprinters and keeps your machine from being used as a bounce for an untraceable portscan. Ids are used for fragmented packets, fragments belonging to the same packet have the same id. By default linux only increments the id value on each packet sent to an individual host. We use a port of the OpenBSD random ip id code to achieve the randomness, while keeping the possibility of id duplicates to near none. If the sysctl option is enabled, a sysctl option with name "rand_ip_ids" is created.	**Selected.**
Randomized TCP source ports (GRKERNSEC_RANDSRC)	If you say Y here, situations where a source port is generated on the fly for the TCP protocol (ie. with connect()) will be altered so that the source port is generated at random, instead of a simple incrementing algorithm. If the sysctl option is enabled, a sysctl option with name "rand_tcp_src_ports" is created.	**Selected.**
Randomized RPC XIDs (GRKERNSEC_RANDRPC)	If you say Y here, the method of determining XIDs for RPC requests will be randomized, instead of using linux's default behavior of simply incrementing the XID. If you want your RPC connections to be more secure, say Y here. If the sysctl option is enabled, a sysctl option with name "rand_rpc" is created.	**Selected.**
Socket restrictions (GRKERNSEC_SOCKET)	If you say Y here, you will be able to choose from several options. If you assign a GID on your system and add it to the supplementary groups of users you want to restrict socket	**Not Selected.**

	access to, this patch will perform up to three things, based on the option(s) you choose.	
Deny any sockets to group (GRKERNSEC_SOCKET_ALL)	If you say Y here, you will be able to choose a GID of whose users will be unable to connect to other hosts from your machine or run server applications from your machine. If the sysctl option is enabled, a sysctl option with name "socket_all" is created.	**Not Selected.**
Deny client sockets to group (GRKERNSEC_SOCKET_CLIENT)	If you say Y here, you will be able to choose a GID of whose users will be unable to connect to other hosts from your machine, but will be able to run servers. If this option is enabled, all users in the group you specify will have to use passive mode when initiating ftp transfers from the shell on your machine. If the sysctl option is enabled, a sysctl option with name "socket_client" is created.	**Not Selected.**
Deny server sockets to group (GRKERNSEC_SOCKET_SERVER)	If you say Y here, you will be able to choose a GID of whose users will be unable to run server applications from your machine. If the sysctl option is enabled, a sysctl option with name "socket_server" is created.	**Not Selected.**
Sysctl Support		
Sysctl support (GRKERNSEC_SYSCTL)	If you say Y here, you will be able to change the options that grsecurity runs with at bootup, without having to recompile your kernel. You can echo values to files in /proc/sys/kernel/grsecurity to enable (1) or disable (0) various features. All the sysctl entries are mutable until the "grsec_lock" entry is set to a non-zero value. All features are disabled by default. Please note that this option could reduce the effectiveness of the added security of this patch if an ACL system is not put in place. Your init scripts should be read-only, and root should not have access to adding modules or performing raw i/o operations. All options should be set at startup, and the grsec_lock entry should be set to a non-zero value after all the options are set. *THIS IS EXTREMELY IMPORTANT*	**Selected.**
Logging Options		
Seconds in between log messages (minimum) (GRKERNSEC_FLOODBURST)	This option allows you to choose the maximum number of messages allowed within the flood time interval you chose in a separate option. The default should be suitable for most people, however if you find that many of your logs are being interpreted as flooding, you may want to raise this value.	**Leave default of 10 seconds.**
Number of messages in a burst (maximum) (GRKERNSEC_FLOODTIME)	This option allows you to enforce the number of seconds between grsecurity log messages. The default should be suitable for most people, however, if you choose to change it, choose a value small enough to allow informative logs to be produced, but large enough to prevent flooding.	**Leave default of 4 messages.**

Enable various PaX features (PAX)

Setting	Description	Comments
PaX Control		
Support soft mode (PAX_SOFTMODE)	Enabling this option will allow you to run PaX in soft mode, that is, PaX features will not be enforced by default, only on executables marked explicitly. You must also enable PT_PAX_FLAGS support, as it is the only way to mark executables for soft mode use. Soft mode can be activated by using the “pax_softmode=1" kernel command line option on boot. Furthermore you can control various PaX features at runtime via the entries in /proc/sys/kernel/pax.	**Not Selected.** PaX needs to be enforced by default.
Use legacy ELF header marking (PAX_EI_PAX)	Enabling this option will allow you to control PaX features on a per executable basis via the 'chpax' utility available at http://pax.grsecurity.net/. The control flags will be read from an otherwise reserved part of the ELF header. This marking has numerous drawbacks (no support for soft-mode, toolchain does not know about the non-standard use of the ELF header) therefore it has been deprecated in favour of PT_PAX_FLAGS support. You should enable this option only if your toolchain does not yet support the new control flag location (PT_PAX_FLAGS) or you still have applications not marked by PT_PAX_FLAGS. Note that if you enable PT_PAX_FLAGS marking support as well, it will override the legacy	**Selected.** Some applications not marked by PT_PAX_FLAGS.

	EI_PAX marks.	
Use ELF program header marking (PAX_PT_PAX_FLAGS)	Enabling this option will allow you to control PaX features on a per executable basis via the 'paxctl' utility available at http://pax.grsecurity.net/. The control flags will be read from a PaX specific ELF program header (PT_PAX_FLAGS). This marking has the benefits of supporting both soft mode and being fully integrated into the toolchain (the binutils patch is available from http://pax.grsecurity.net). Note that if you enable the legacy EI_PAX marking support as well, it will be overridden by the PT_PAX_FLAGS marking.	**Selected.**
MAC system integration None (PAX_NO_ACL_FLAGS) Direct (PAX_HAVE_ACL_FLAGS) Hook (PAX_HOOK_ACL_FLAGS)	Mandatory Access Control systems have the option of controlling PaX flags on a per executable basis, choose the method supported by your particular system. - "none": if your MAC system does not interact with PaX, - "direct": if your MAC system defines pax_set_flags() itself, - "hook": if your MAC system uses the pax_set_flags_func callback. NOTE: this option is for developers/integrators only.	**Selected DIRECT.**
Non-executable pages		
Enforce non-executable pages (PAX_NOEXEC)	By design some architectures do not allow for protecting memory pages against execution or even if they do, Linux does not make use of this feature. In practice this means that if a page is readable (such as the stack or heap) it is also executable. There is a well known exploit technique that makes use of this fact and a common programming mistake where an attacker can introduce code of his choice somewhere in the attacked program's memory (typically the stack or the heap) and then execute it. If the attacked program was running with different (typically higher) privileges than that of the attacker, then he can elevate his own privilege level (e.g. get a root shell, write to files for which he does not have write access to, etc). Enabling this option will let you choose from various features that prevent the injection and execution of 'foreign' code in a program. This will also break programs that rely on the old behaviour and expect that dynamically allocated memory via the malloc() family of functions is executable (which it is not). Notable examples are the XFree86 4.x server, the java runtime and wine.	**Selected.**
Paging based non-executable pages (PAX_PAGEEXEC)	This implementation is based on the paging feature of the CPU. On i386 it has a variable performance impact on applications depending on their memory usage pattern. Depreciated due to SEGMEXEC. [7]	**Not Selected.** Only available after selecting 'Enforce non-executable pages.'
Segmentation based non-executable pages (PAX_SEGMEXEC)	This implementation is based on the segmentation feature of the CPU and has little performance impact, however applications will be limited to a 1.5 GB address space instead of the normal 3 GB.	**Selected.** Only available after selecting 'Enforce non-executable pages.'
Emulate trampolines (PAX_EMUTRAMP)	There are some programs and libraries that for one reason or another attempt to execute special small code snippets from non-executable memory pages. Most notable examples are the signal handler return code generated by the kernel itself and the GCC trampolines. If you enabled CONFIG_GRKERNSEC_PAX_PAGEEXEC or CONFIG_GRKERNSEC_PAX_SEGMEXEC then such programs will no longer fork under your kernel. As a remedy you can say Y here and use the 'chpax' or 'paxctl' utilities to enable trampoline emulation for the affected programs yet still have the protection provided by the non-executable pages. On parisc and ppc you MUST enable this option and EMUSIGRT as well, otherwise your system will not even boot. Alternatively you can say N here and use the 'chpax' or 'paxctl' utilities to disable CONFIG_GRKERNSEC_PAX_PAGEEXEC and CONFIG_GRKERNSEC_PAX_SEGMEXEC for the affected files. NOTE: enabling this feature *may* open up a loophole in the protection provided by non-executable pages that an attacker could abuse. Therefore the best solution is to not have any files on your system that would require this option. This can be achieved by not using libc5 (which relies on the kernel signal handler return code) and not using or rewriting programs that make use of the nested function implementation of GCC. Skilled users can just fix GCC itself so that it implements nested function calls in a way that does not interfere with PaX.	**Not Selected.** Only available after selecting 'Segmentation based non-executable pages.' Will disable affected files as needed.

Restrict mprotect() (PAX_MPROTECT)	Enabling this option will prevent programs from - changing the executable status of memory pages that were not originally created as executable, - making read-only executable pages writable again, - creating executable pages from anonymous memory. You should say Y here to complete the protection provided by the enforcement of non-executable pages. NOTE: you can use the 'chpax' utility to control this feature on a per file basis. chpax is available at http://pax.grsecurity.net>	**Selected.** Only available after selecting 'Segmentation based non-executable pages.'
Disallow ELF text relocations (PAX_NOELFRELOCS)	Non-executable pages and mprotect() restrictions are effective in preventing the introduction of new executable code into an attacked task's address space. There remain only two venues for this kind of attack: if the attacker can execute already existing code in the attacked task then he can either have it create and mmap() a file containing his code or have it mmap() an already existing ELF library that does not have position independent code in it and use mprotect() on it to make it writable and copy his code there. While protecting against the former approach is beyond PaX, the latter can be prevented by having only PIC ELF libraries on one's system (which do not need to relocate their code). If you are sure this is your case, then enable this option otherwise be careful as you may not even be able to boot or log on your system (for example, some PAM modules are erroneously compiled as non-PIC by default). NOTE: if you are using dynamic ELF executables (as suggested when using ASLR) then you must have made sure that you linked your files using the PIC version of crt1 (the et_dyn.zip package referenced there has already been updated to support this).	**Selected.** Only available after selecting 'Restrict mprotect().'
Address Space Layout Randomization		
Address Space Layout Randomization (PAX_ASLR)	Many if not most exploit techniques rely on the knowledge of certain addresses in the attacked program. The following options will allow the kernel to apply a certain amount of randomization to specific parts of the program thereby forcing an attacker to guess them in most cases. Any failed guess will most likely crash the attacked program which allows the kernel to detect such attempts and react on them. PaX itself provides no reaction mechanisms, instead it is strongly encouraged that you make use of grsecurity's built-in crash detection features or develop one yourself. By saying Y here you can choose to randomize the following areas: - top of the task's kernel stack - top of the task's userland stack - base address for mmap() requests that do not specify one (this includes all libraries) - base address of the main executable It is strongly recommended to say Y here as address space layout randomization has negligible impact on performance yet it provides a very effective protection. NOTE: you can use the 'chpax' or 'paxctl' utilities to control most of these features on a per file basis.	**Selected.**
Randomize kernel stack base (PAX_RANDKSTACK)	By saying Y here the kernel will randomize every task's kernel stack on every system call. This will not only force an attacker to guess it but also prevent him from making use of possible leaked information about it. Since the kernel stack is a rather scarce resource, randomization may cause unexpected stack overflows, therefore you should very carefully test your system. Note that once enabled in the kernel configuration, this feature cannot be disabled on a per file basis.	**Selected.** Only available after selecting 'Address Space Layout Randomization.'
Randomize user stack base (PAX_RANDUSTACK)	By saying Y here the kernel will randomize every task's userland stack. The randomization is done in two steps where the second one may apply a big amount of shift to the top of the stack and cause problems for programs that want to use lots of memory (more than 2.5 GB if SEGMEXEC is not active, or 1.25 GB when it is). For this reason the second step can be controlled by 'chpax' or 'paxctl' on a per file basis.	**Selected.**Only available after selecting 'Address Space Layout Randomization.'
Randomize mmap() base (PAX_RANDMMAP)	By saying Y here the kernel will use a randomized base address for mmap() requests that do not specify one themselves. As a result all dynamically loaded libraries will appear at random addresses and therefore be harder to exploit by a technique where an attacker attempts to execute library code for his purposes (e.g. spawn a shell from an exploited program that is running at an elevated privilege	**Selected.** Only available after selecting 'Address Space Layout Randomization.'

	level). Furthermore, if a program is relinked as a dynamic ELF file, its base address will be randomized as well, completing the full randomization of the address space layout. Attacking such programs becomes a guess game. You can find an example of doing this at <http://pax.grsecurity.net/et_dyn.zip> and practical samples at <http://www.grsecurity.net/grsec-gcc-specs.tar.gz>. NOTE: you can use the 'chpax' or 'paxctl' utilities to control this feature on a per file basis.	
Randomize ET_EXEC base (PAX_RANDEXEC)	By saying Y here the kernel will randomize the base address of normal ET_EXEC ELF executables as well. This is accomplished by mapping the executable in memory in a special way which also allows for detecting attackers who attempt to execute its code for their purposes. Since this special mapping causes performance degradation and the attack detection may create false alarms as well, you should carefully test your executables when this feature is enabled. This solution is intended only as a temporary one until you relink your programs as a dynamic ELF file. NOTE: you can use the 'chpax' or 'paxctl' utilities to control this feature on a per file basis.	**Selected.** Only available after selecting 'Randomize mmap() base.'
Disable the vsyscall page (PAX_NOVSYSCALL)	The Linux 2.6 kernel introduced a new feature that speeds up or simplifies certain operations, such as system calls or returns from signal handlers. Unfortunately the implementation also gives a powerful instrument into the hands of exploit writers: the so-called vsyscall page exists in every task at the same fixed address and it contains machine code that is very useful in performing the return-to-libc style attack. Since this exploit technique cannot in general be protected against via kernel solutions, this option will allow you to disable the use of the vsyscall page and revert back to the old behaviour.	**Selected.** Only available after selecting 'Address Space Layout Randomization.'

Appendix C – Unreal IRCd configuration

/usr/local/etc/unrealircd.conf

```
/*
 * unrealircd.conf
 */

/*
 * Modules
 * Loading the commands module and a cloaking module is required:
 */

loadmodule "/usr/local/lib/commands.so";
loadmodule "/usr/local/lib/cloak.so";

/*
 * You can also include other configuration files.
 * help.conf contains all the /helpop text.
 */

include "help.conf";
include "users.conf";

/*
 * me {} defines the name, description and unreal server numeric for
 * this server:
 */

me {
```

```
      name "hermes.giacenterprises.com";
      info "IS Communication Server";
      numeric 1;
};

/*
 * Admin gives information on the server admin.
 */

admin {
      "Adam Chan";
      "achan@giacenterprises.com";
};

/*
 * These define settings for classes. A class is a group setting for
 * connections. Example, server connections, instead of going to a client's
 * class, you direct it to the server class. Syntax is as follows
 * class (class name)
 * {
 *     pingfreq (how often to ping a user/server in seconds);
 *     maxclients (how many connections for this class);
 *     sendq (maximum send queue from a connection);
 *     recvq (maximum receive queue from a connection [flood control]);
 *  };
 */

class clients {
      pingfreq 90;
      maxclients 500;
      sendq 100000;
      recvq 8000;
};

/*
 * Defines an IRC Operator
 * IRC operators are there to keep sanity to the server and usually keep it
 * maintained and connected to the network.
 * The syntax is as follows:
 * oper (login) {
 *     class (class to put them in, if different from I, moves them to new
 *                class);
 *     from {
 *        userhost (ident@host);
 *        userhost (ident@host);
 *     };
 *     flags
 *     {
 *       (flags here*);
 *     };
 *     OR
 *     flags "old type flags, like OAaRD";
 * };
 */

oper adamchan {
      class           clients;
```

```
      from {
            userhost achan@*;
      };
      password "oper"; /* Password changes as needed */
      flags
      {
            netadmin;
            global;
      };
};

/*
 * This defines a port for the ircd to bind to, to
 * allow users/servers to connect to the server.
 * Syntax is as follows:
 * listen (ip number):(port number)
 */

listen *:6697 {
      options {
            ssl;
            clientsonly;
      };
};

/*
 * NEW: log {} OLD: N/A Tells the ircd where and what to log(s). You can have
 * as many as you wish.
 *
 * FLAGS: errors, kills, tkl, connects, server-connects, kline, oper
 *
 * Syntax:
 * log "log file"
 * {
 *    flags
 *    {
 *        flag;
 *        flag;
 *        etc..
 *    };
 * };
 */

log "/var/log/ircd.log" {
      flags {
            oper;
            kline;
            connects;
            server-connects;
            kills;
            errors;
            sadmin-commands;
            chg-commands;
            oper-override;
            spamfilter;
      };
};
```

```
/*
 * Network configuration
 */

set {
      network-name            "GIACnet";
      default-server          "hermes.giacenterprises.com";
      services-server         "hermes.giacenterprises.com";
      stats-server            "hermes.giacenterprises.com";
      help-channel            "#GIACnet Help";
      hiddenhost-prefix "giac";
      /* prefix-quit          "no"; */
      /* Cloak keys should be the same at all servers on the network.
       * They are used for generating masked hosts and should be kept secret.
       * The keys should be 3 random strings of 5-100 characters
       * (10-20 chars is just fine) and must consist of lowcase (a-z),
       * upcase (A-Z) and digits (0-9) [see first key example].
       */
      cloak-keys {
            "aoAr1HnR6gl3sJ7hVz4Zb7x4YwpW";
            "adfskkwejrewki82j33kdjdkKK33";
            "893KJLSlksdjflkjekkej3kjkj4k";
      };
      /* on-oper host */
      hosts {
            local       "hermes.giacenterprises.com";
            global            "hermes.giacenterprises.com";
            coadmin           "hermes.giacenterprises.com";
            admin       "hermes.giacenterprises.com";
            servicesadmin     "hermes.giacenerprises.com";
            netadmin    "hermes.giacenterprises.com";
            host-on-oper-up "no";
      };
};

set {
      kline-address *@*;
      auto-join #Network;
      options {
            hide-ulines;
      };
      maxchannelsperuser 10;
      dns {
            nameserver 127.0.0.1;
            timeout 1s;
            retries 1;
      };
};

/*
 * Official Channels Block
 *
 * official-channels {
 *          "#channel" { topic "The default topic"; };
 * };
 */
```

```
official-channels {
      "#Network" { topic "Network Issues"; };
      "#Incident" { topic "Security Incident Issues"; };
      "#Malware" { topic "Malware Issues"; };
      "#Management" { topic "Management Issues"; };
};
```

Appendix D – Unreal IRCd init file

/etc/init.d/ircd

```
  #!/bin/bash
  #
  # chkconfig: 2345 49 49
  # (run in init levels 2345 & start priority of 49 and stop priority of 49)
  # description: Init file for Unreal IRCd
  # processname: iptables
  # config: /usr/local/etc/unrealircd.conf
  # pidfile: /usr/local/etc/ircd.pid

  # source function library
  . /etc/rc.d/init.d/functions

  RETVAL=0
  IRCD=/usr/local/sbin/ircd
  Lockfile=/var/lock/subsys/ircd


  start()
  {
      echo -n $"Starting Unreal IRCd: "
      touch "$lockfile" && success || failure
      RETVAL=$?
   }

  stop()
  {
      echo -n $"Stopping Unreal IRCd: "
      killproc $IRCD -TERM
      rm -f "$lockfile" && success || failure
       RETVAL=$?
   }

  reload()
  {
      echo -n $"Reloading Unreal IRCd: "
      killproc $IRCD -HUP
      RETVAL=$?
  }

  case "$1" in
      start)
```

```
            start
            ;;
      stop)
            stop
            ;;
      restart)
            stop
            start
            ;;
      reload)
            reload
            ;;
      status)
            status $IRCD
            RETVAL=$?
            ;;
      *)
            echo $"Usage: $0 {start|stop|restart|reload|status}"
            RETVAL=1
  esac
  exit $RETVAL
```

Appendix E – RBAC Structure

This information was pulled from several sources and is NOT official. It was combined to assist with this paper as no formal documentation had been released at the time. Please visit http://www.grsecurity.net for the latest up-to-date information and official documentation.

```
role <role name> <role flags>
role_allow_ip <ip>/<netmask>
role_transitions <special role1><special role 2>…<special role n>
<path of subject process> <optional subject modes> {
    <file object> <optional object modes>
    [+|-]<capability>
    <resource name> <soft limit> <hard limit>
    connect <ip>/<netmask>:<low port>-<high port> <type> <proto>
    bind <ip>/<netmask>:<low port>-<high port> <type> <proto>
}
```

Role Flag	Description
A	This role is an administrative role, thus it has special privilege normal roles do not have. In particular, this role bypasses the additional ptrace restrictions.
N	Don't require authentication for this role. To access the role, use gradm -n <rolename>
s	This role is a special role, meaning it does not belong to a user or group, and does not require an enforced secure policy base to be included in the ruleset (Lowercase S)
u	This role is a user role (Lowercase U)
g	This role is a group role (Lowercase G)
G	This role can use gradm to authenticate to the kernel A policy for

	gradm will automatically be added to the role
T	Enable TPE for this role
l	Enable learning for this role (Lowercase L)

Subject Flags	Description
h	This process is hidden and only viewable by processes with the v mode. (Lowercase H)
v	This process can view hidden processes. (Lowercase V)
p	This process is protected; it can only be killed by processes with the k mode. (Lowercase P)
k	This process can kill protected processes. (Lowercase K)
l	Enables learning mode for this process. (Lowercase L)
o	Override ACL inheritance for this process. (Lowercase O)
r	Relax ptrace restrictions (allows process to ptrace processes other than its own descendants). (Lowercase R)
P	Disables the PAGEEXEC(Paging based non-executable pages) feature of PaX on this subject.
S	Disables the SEGMEXEC(Segmentation based non-executable pages)
M	Disables the MPROTECT (Restrict mprotect()) feature of PaX on this subject.
R	Disables the RANDMAP (Randomize mmap() base) feature of Pax on this subject.
G	Enables the EMUTRAMP (Emulate trampolines) feature of PaX on this subject.
X	Enables the RANDEXEC (Randomize ET_EXEC base) feature of PaX on this subject.
O	Override the mmap() and ptrace() checks for this subject.
A	Protect the shared memory of this subject. None except the processes contained within the subject may access the shared memory of this subject.
K	When processes belonging to this subject generate an alert, kill the process.
C	When processes belonging to this subject generate an alert, kill the process and all processes belonging to the IP of that attacker (if there was an IP attached to the process)
T	Deny execution of binaries or scripts that are writable by any other subject in the policy. Ensures this process can never execute any trojaned code.

Object Flags	Description
r	This object can be opened for reading. This is the default if no flags are given. (Lowercase R)
w	This object can be opened for writing or appending. (Lowercase W)
x	This object can be executed. (Lowercase X)
a	This object can be opened for appending. (Lowercase A)
h	This object is hidden. (Lowercase H)
t	This object can be ptraced, but cannot modify the running task. This is referred to as a read-only ptrace. (Lowercase T)
i	This mode only applies to binaries. When the object is executed, it inherits the ACL of the subject in which it was contained. (Lowercase I)
m	Allow creation of setuid/setgid files/directories and modification of files/directories to be setuid/setgid. (Lowercase M)

c	Allow creation of the file/directory. (Lowercase C)
d	Allow deletion of the file/directory. (Lowercase D)
p	Reject all ptraces to this object. (Lowercase P)
R	Audit successful reads to this object.
W	Audit successful writes to this object.
X	Audit successful execs of this object.
A	Audit successful appends to this object.
F	Audit successful finds of this object.
I	Audit successful ACL inherits of this object.
M	Audit the setuid/setgid creation/modification.
C	Audit the creation of file/directory.
D	Audit the deletion of file/directory.
s	Suppress logs of denied access for this path (Lowercase S)

Capability Name	**Description**
CAP_ALL	Includes all capabilities.
CAP_CHOWN	In a system with the [_POSIX_CHOWN_RESTRICTED] option defined, this overrides the restriction of changing file ownership and group ownership.
CAP_DAC_OVERRIDE	Override all DAC access, including ACL execute access if [_POSIX_ACL] is defined. Excluding DAC access covered by CAP_LINUX_IMMUTABLE.
CAP_DAC_READ_SEARCH	Overrides all DAC restrictions, regarding read and search on files and directories, including ACL restrictions, if [_POSIX_ACL] is defined. Excluding DAC access covered by CAP_LINUX_IMMUTABLE.
CAP_FOWNER	Overrides all restrictions about allowed operations on files, where file owner ID must be equal to the user ID, except where CAP_FSETID is applicable. It doesn't override MAC and DAC restrictions.
CAP_FSETID	Overrides the following restrictions that the effective user ID shall match the file owner ID when setting the S_ISUID and S_ISGID bits on that file; that the effective group ID (or one of the supplementary group IDs) shall match the file owner ID when setting the S_ISGID bit on that file; that the S_ISUID and S_ISGID bits are cleared on successful return from chown(2) (not implemented).
CAP_KILL	Overrides the restriction that the real or effective user ID of a process sending a signal must match the real or effective user ID of the process receiving the signal.
CAP_SETGID	Allows setgid(2) manipulation; Allows setgroups(2); Allows forged gids on socket credentials passing.
CAP_SETUID	Allows set*uid(2) manipulation (including fsuid); Allows forged pids on socket credentials passing.
CAP_SETPCAP	Transfer any capability in your permitted set to any pid, remove any capability in your permitted set from any pid.
CAP_LINUX_IMMUTABLE	Allow modification of S_IMMUTABLE and S_APPEND file attributes.
CAP_NET_BIND_SERVICE	Allows binding to TCP/UDP sockets below 1024; Allows binding to ATM VCIs below 32.

CAP_NET_BROADCAST	Allow broadcasting, listen to multicast.
CAP_NET_ADMIN	Allow interface configuration; Allow administration of IP firewall, masquerading and accounting; Allow setting debug option on sockets; Allow modification of routing tables; Allow setting arbitrary process / process group ownership on sockets; Allow binding to any address for transparent proxying; Allow setting TOS (type of service); Allow setting promiscuous mode; Allow clearing driver statistics; Allow multicasting; Allow read/write of device specific registers; Allow activation of ATM control sockets.
CAP_NET_RAW	Allow use of RAW sockets; Allow use of PACKET sockets.
CAP_IPC_LOCK	Allow locking of shared memory segments; Allow mlock and mlockall (which doesn't really have anything to do with IPC).
CAP_IPC_OWNER	Override IPC ownership checks.
CAP_SYS_MODULE	Insert and remove kernel modules modify kernel without limit; Modify cap_bset.
CAP_SYS_RAWIO	Allow ioperm/iopl access; Allow sending USB messages to any device via /proc/bus/usb.
CAP_SYS_CHROOT	Allow use of chroot().
CAP_SYS_PTRACE	Allow ptrace() of any process.
CAP_SYS_PACCT	Allow configuration of process accounting.
CAP_SYS_ADMIN	Allow configuration of the secure attention key; Allow administration of the random device; Allow examination and configuration of disk quotas; Allow configuring the kernel syslog (printk behaviour); Allow setting the domainname; Allow setting the hostname; Allow calling bdflush(); Allow mount() & umount(), setting up new smb connection; Allow some autofs root ioctls; Allow nfsservctl; Allow VM86_REQUEST_IRQ; Allow to read/write pci config on alpha; Allow irix_prctl on mips (setstacksize); Allow flushing all cache on m68k (sys_cacheflush); Allow removing semaphores; Used instead of CAP_CHOWN to "chown" IPC message queues, semaphores and shared memory; Allow locking/unlocking of shared memory segment; Allow turning swap on/off; Allow forged pids on socket credentials passing; Allow setting readahead & flushing buffers on block devices; Allow setting geometry in floppy driver; Allow turning DMA on/off in xd driver; Allow administration of md devices (mostly the above, but some extra ioctls); Allow tuning the ide driver;

	Allow access to the nvram device; Allow administration of apm_bios, serial and bttv (TV) device; Allow manufacturer commands in isdn CAPI support driver; Allow reading nonstandardized portions of pci configuration space; Allow DDI debug ioctl on sbpcd driver; Allow setting up serial ports; Allow sending raw qic117 commands; Allow enabling/disabling tagged queuing on SCSI controllers and sending arbitrary SCSI commands; Allow setting encryption key on loopback filesystem.
CAP_SYS_BOOT	Allow use of reboot().
CAP_SYS_NICE	Allow raising priority and setting priority on other (different UID) processes; Allow use of FIFO and roundrobin (realtime) scheduling on own processes and setting the scheduling algorithm used by another process.
CAP_SYS_RESOURCE	Override resource limits. Set resource limits; Override quota limits; Override reserved space on ext2 filesystem; Modify data journaling mode on ext3 filesystem (uses journaling resources); NOTE: ext2 honors fsuid when checking for resource overrides, so you can override using fsuid too; Override size restrictions on IPC message queues; Allow more than 64hz interrupts from the realtime clock; Override max number of consoles on console allocation; Override max number of keymaps.
CAP_SYS_TIME	Allow manipulation of system clock; Allow irix_stime on mips; Allow setting the realtime clock.
CAP_SYS_TTY_CONFIG	Allow configuration of tty devices; Allow vhangup() of tty.
CAP_MKNOD	Allow the privileged aspects of mknod().
CAP_LEASE	Allow taking of leases on files.

Resource	**Description**
RES_CPU	CPU time in milliseconds.
RES_FSIZE	Maximum file size in bytes.
RES_DATA	Maximum data size in bytes.
RES_STACK	Maximum stack size in bytes.
RES_CORE	Maximum core size in bytes.
RES_RSS	Maximum resident set size in bytes.
RES_NPROC	Maximum number of processes.
RES_NOFILE	Maximum number of open files.
RES_MEMLOCK	Maximum locked-in-memory in bytes.

References

[1] Committee on National Security Systems. National Information Assurance (IA) Glossary. CNS Instruction No. 4009. NSA. May 2003. 11 July 2004. <http://www.ntissc.gov/Assets/pdf/4009.pdf>

[2] Meier, J.D., et al. 2003. Improving Web Application Security: Threats and Countermeasures Roadmap. Microsoft Security Developer Center, Microsoft Corporation. June 2003. 1 August 2004. <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetsec/html/ThreatCounter.asp>

[3] Howard, Michael and David LeBlanc. Writing Secure Code, 2nd edition. Redmond: Microsoft Press, 2003.

[4] Spengler, Brad. Detection, Prevention, and Containment: A Study of grsecurity. July 2002. 26 July 2004. <http://www.grsecurity.com/grsecurity-slide_files/frame.htm>

[5] NIST/ITL. An Introduction to Role-Based Access Control. December 1995. 1 August 2004. <http://csrc.nist.gov/rbac/NIST-ITL-RBAC-bulletin.html>

[6] Spengler, Bradley. Increasing Performance and Granularity in Role-Based Access Control Systems. 1 August 2004. <http://grsecurity.net/~spender/researchpaper.pdf>

[7] Spengler, Brad. PaX: The Guaranteed End of Arbitrary Code Execution. October 2003. 26 July 2004. <http://www.grsecurity.com/PaX-presentation_files/frame.htm>

[8] PaX Team. Documentation for the PaX project. November 2003. 26 July 2004. <http://pax.grsecurity.net/docs/index.html>

[9] SANS Institute. Incident Handling Book I. August 2001. 22 August 2004.

Bauer, Mick. “Stealthful Sniffing, Intrusion Detection and Logging.” Linux Journal October 2002. 28 August 2004.

Dunston, Duane. “Remote Syslog with MySQL and PHP.” LinuxSecurity.Com. February 2003. 28 August 2004. <http://www.linuxsecurity.com/feature_stories/feature_story-138.html>

Bauer, Michael. Building Secure Servers with Linux. Cambridge: O’Reilly Media, Inc. 2002.

Lockhart, Andrew. Network Security Hacks. Cambridge: O’Reilly Media, Inc. 2004.

Moore, Andrew, Robert Ellison and Richard Linger. Attack Modeling for Information Security and Survivability. March 2001. 20 July 2004. <http://www.sei.cmu.edu/pub/documents/01.reports/pdf/01tn001.pdf >

Spengler, Brad. grsecurity ACL Documentation v1.5. April 2003. 5 August 2004. <http://www.grsecurity.com/gracldoc.htm>

Spengler, Brad. grsecurity Quickstart Guide. 8 August 2004. <http://www.grsecurity.com/quickstart.pdf>

Cknight^ and Syzop. UnrealIRCd – 3.2 – Official Documentation. July 2004. 30 July 2004. <http://www.vulnscan.org/UnrealIrcd/unreal32.docs.html>

Fox, Michael, John Giordano, Lori Stotler, and Arun Thomas. SELinux and grsecurity: A Case Study Comparing Linux Security Kernel Enhancements. University of Virginia. 5 August 2004. <http://www.cs.virginia.edu/~jcg8f/GrsecuritySELinuxCaseStudy.pdf>

Vermeulen, Sven. Gentoo Linux Grsecurity Guide for 1.9.x. September 7, 2003. 5 August 2004. <http://gentoo.org/proj/en/hardened/grsecurity.xml>

Murdoch, Don. Building a Secured OS for a Root Certificate Authority. February 28, 2004. 1 July 2004. <http://www.giac.org/practical/GCUX/Don_Murdoch_GCUX.pdf>

Walker, George. GIAC Enterprises. GIAC Practical Assignment, Version 1.9. June 22, 2003. 29 June 2004. <http://www.giac.com/practical/GCFW/Danny_Walker_GCFW.pdf>

Heilman, Marshall. Implementing a Shorewall Firewall and BIND DNS Server on a Hardened Fedora Core 2 OS. August 2, 2004. 29 August 2004. <http://www.giac.org/practical/GCUX/Marshall_Heilman_GCUX.pdf>

Provos, Niels. Privilege Separated OpenSSH. August 9, 2003. 28 August 2004. <http://www.citi.umich.edu/u/provos/ssh/privsep.html>

Ippolito, Greg. YoLinux Tutorial: Linux System Administration. 2 August 2004. <http://www.yolinux.com/TUTORIALS/LinuxTutorialSysAdmin.html>