# Global Information Assurance Certification Paper

# Auditing a Snort Intrusion Detection System: An Auditor's Perspective

## Auditing Networks, Perimeters, and Systems
## GSNA Practical Assignment Version 2.1
## (Amended July 5, 2002)

*Prepared By*
**Brent Zimmerman**

March 25, 2003

# Table of Contents

## Table of Figures



## Table of Tables

## Summary

This paper provides a detailed audit of a Snort distributed intrusion detection system. The paper is broken down into four primary sections: researching, creating an audit checklist, gathering audit evidence, and creating an audit report. Each section builds on the previous section and requires the previous section's information to be completed.

The research portion of this paper involved extensive research utilizing technical books and a multitude of Internet sites. After scouring through all of this information, bits and pieces were pieced together from several resources to help in compiling the audit checklist. This checklist consisted of 27 items and covered all components defined in the scope. These components are the Snort intrusion detection software, the Apache web server, the MySQL database, the ACID analyst console, the SnortCenter front-end interface, and the Red Hat Linux operating system.

Once the complete checklist was completed, the actual audit was performed. The actual audit analyzed each checklist item to determine if it was in compliance; however, output from only ten items is listed in this paper. For these ten items, a table is present that states if that checklist item as well as each individual test for that item was or was not in compliance.

Once the audit was complete, it was possible to create the audit report. This audit report includes the audit findings, recommendations, and costs associated with performing these recommendations. This portion of the paper would be the report that is presented to management. Therefore, it takes the technical information and provides a summary of all the findings that were discovered.

All of this work was performed solely by one auditor, Brent Zimmerman. In addition, the IS administrator was very instrumental in making this a smooth, successful project. The entire audit from research to the audit report was done over a three month period. In the end, it was determined that there were some items that were discovered to not be in compliance. However, overall, the Snort intrusion detection system was configured in a secure manner. The details of this determination and the research to get to that point are detailed in the sections to follow.

# 1. Research in Audit, Measurement Practice, and Control

## 1.1. Identify the System to be Audited

I am auditing a Snort intrusion detection system (IDS) that is part of the perimeter security architecture of a mid-size enterprise. The Snort IDS is distributed in nature; therefore, it consists of two separate devices: the IDS probe and the IDS management console. Both devices are running Red Hat Linux. The IDS probe is where the Snort IDS software is located while the IDS management console contains four primary components. These components are the Apache web server, the MySQL database server, the ACID analyst console, and the SnortCenter front-end interface. Following are the specific versions and function for each component of the IDS:

| IDS Component | Sub-Component | O.S. or Version | Function |
|---|---|---|---|
| Probe | | Red Hat 8.0 | |
| | Snort | 1.9.1 | Detecting Alerts |
| Management Console | | Red Hat 8.0 | |
| | Apache | 2.0.40 | Web Server |
| | MySQL | 3.23.54 | Database to Store Event Information |
| | ACID | v0.9.6b23 | Managing Alerts Stored in the Database |
| | SnortCenter | V0.9.6 | Front-End Interface |

**Table 1: IDS Component Information**

The primary function of the IDS is to serve as one component of the organization's overall perimeter security solution. In addition to the IDS, this company has a firewall and perimeter router in place (see Figure 1). Both the firewall and the perimeter router play a critical role in filtering packets and stopping certain traffic from entering the organization's private network. However, in almost every circumstance, certain traffic still needs to be allowed through the firewall and perimeter router for business continuity. Unfortunately, malicious data can enter through this legitimate traffic. Given this, the primary function of the IDS is to detect intrusions and attempted intrusions that the firewall and perimeter router may not stop.

This is a vital responsibility in today's networks as more and more attacks are being created and performed every day against every type of business. In addition, the skill level needed to perform these attacks has dramatically decreased with automated freeware tools. This has resulted in attacks being performed at an unprecedented frequency. Therefore, it is becoming more essential that organizations have an IDS in place to help prevent against intrusion attempts. With an IDS in place, organizations tend to feel a greater sense of perimeter security due to the fact that they believe that an attack will not go undetected. However, this assumes that the IDS has been configured and updated correctly, and that the IDS itself has not been compromised. Therefore, it is imperative to make sure that the IDS is configured in a secure manner.

To guarantee a secure configuration, each and every component listed in table 1 must be configured securely. This is because a distributed IDS is irreducibly complex; i.e., if you take one component away, it will not function as a true distributed IDS. In addition, the IDS must be configured so that its function coincides with the organization's security policy. If both of these are done, an organization can then have a greater comfort that the IDS is behaving as expected. Therefore, this audit will look at each component of the IDS to make sure that all components are securely configured.

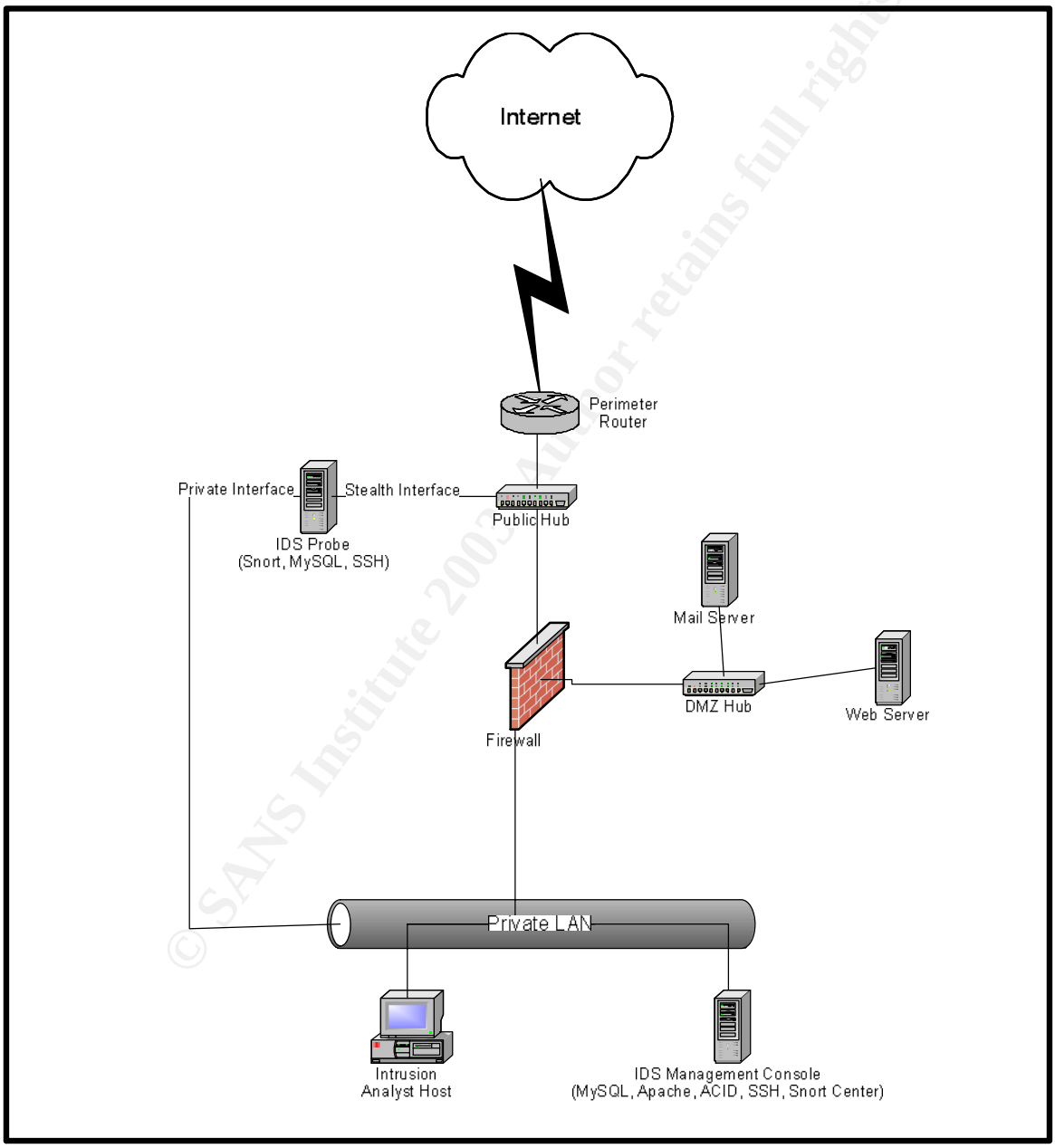



**Figure 1: Perimeter Network Diagram**

### 1.2: Evaluate the Risk to the System

If an Intrusion Detection System is in place, an organization should have a greater feeling that they are accurately able to detect intrusions. However, what happens when an IDS misses a legitimate attack? An organization may not have any idea that an attack was launched. In fact, it may even be accurate to say that at that point they are worse off than if they did not utilize an IDS. The last thing that you want is the feeling of that false sense of security that you get by looking at what you think is happening but really has been modified by an intruder; i.e., a false sense of security is worse than a true sense of insecurity. Even with that in mind, an IDS is an effective, critical component of the perimeter security architecture if it is secured correctly. However, if it is not, there are many risks that could be realized to make the IDS ineffective.

This is because the IDS is composed of many components as described in the previous section. All of these components must be configured securely so that there is not one component that is vulnerable to an attack. Therefore, when looking at the security control objectives, it is important to address each component. Making sure that each component is securely configured will make for an overall secure solution.

The number of involved components also means that there are more avenues for an intruder to attempt to enter through. This creates more complexity in mitigating the risk especially because an IDS can easily become a prime target if an intruder discovers that an IDS exists. This is because when an intruder hacks into a network, the first thing that he/she will usually do is remove any trace (by altering logs and other storage methods) that he/she has been there. However, an IDS makes it very difficult to go undetected unless the IDS can be disabled or altered in some way. Therefore, it is vital to make sure that the IDS does not have one insecure component that allows easy access for an intruder. Following are some potential risks that need to be addressed for each component of the IDS:

#### IDS Probe

- **Blinding the Sensor –** An IDS is usually placed alongside the networking stream, not in the middle. This means that if it cannot keep up with the high rates of traffic, it has no way to throttle the traffic back. Not only will the sensor start dropping packets it cannot process, but high traffic rates can completely shut down the sensor. This is because frame reception and frame analysis are two different activities. Most architectures require the system to capture the packet even when it is too busy to analyze it, which takes even more time away from analysis.[1] In either of these circumstances (dropped packets or a sensor shutting down), an attacker can possibly send packets past the IDS without being detected.

[1] Graham, Robert. "FAQ: Network Intrusion Detection System." 21 March 2000. URL: http://www.robertgraham.com/pubs/network-intrusion-detection.html (3 Jan. 2003).

- **Events on a Different Network –** Unauthorized "back door" connections into a network are common; every machine with a modem has the potential to create a back door.[2] If an attacker can access the network through a back door then he/she is able to perform detrimental activities without the IDS noticing. That is, an IDS cannot inspect packets that it does not see.
- **Encrypted Packets –** If packets are encrypted, an IDS will not be able to analyze the packet for a potential attack.
- **Insertion –** This is where an IDS accepts a packet that an end-system rejects. An IDS that does this makes the mistake of believing that the end-system has accepted and processed the packet when it actually hasn't. An attacker can exploit this condition by sending packets to an end-system that an end-system will reject, but that the IDS will accept. In doing this, the attacker is "inserting" data into the IDS. An attacker can use insertion attacks to defeat signature analysis, allowing him to slip attacks past an IDS.[3]
- **Evasion –** This is where an end-system accepts a packet that an IDS rejects. An IDS that mistakenly rejects such a packet misinterprets its contents entirely. Therefore, entire sessions can be carried forth in packets that evade an IDS in this manner, and blatantly obvious attacks couched in such sessions will happen right under the nose of even the most sophisticated analysis engine.[4] Examples of evasion attacks include fragmentation, slow scans, address spoofing, and pattern changes.
- **Denial of Service –** The numerous protocols that an IDS analyzes leaves it open to crashes when unexpected traffic is seen. Attackers can often buy the same IDS used by their victim, then experiment in many ways in order to find packets that will kill the IDS. Then during the attack, the intruder kills the IDS, and continues undetected.
- **Signature Updates** - This IDS, as with most widely-used IDS's today, utilizes signatures to determine if an attack is occurring. A signature is a pattern that you are looking for in the traffic that attempts to get by the IDS. Each packet is compared to a database of signatures to determine if it matches a known vulnerability or attack. If the signature database is not up-to-date with all known vulnerabilities, then it is not performing at its peak capability. Therefore, it is critical to make sure that the IDS is constantly updated with the most recent signatures.
- **IDS Network Model Different Than Real World –** Different operating systems handle packets differently. For example, if overlapping fragments are sent with different data, some systems prefer the data from the first fragment (WinNT, Solaris), whereas others keep the data from the last fragment (Linux, BSD). The IDS has no way of knowing which packet the end-node will accept, and may

---

[2] Northcutt, Stephen. Network Intrusion Detection: An Analyst's Handbook. Indianapolis, New Riders Publishing, 1999. 35.
[3] Ptacek, Thomas H. and Newsham, Timothy N. "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection." Jan. 1998. URL: http://www.snort.org/docs/idspaper (5 Jan. 2003)
[4] Ptacek, Thomas H. and Newsham, Timothy N. "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection." Jan. 1998. URL: http://www.snort.org/docs/idspaper (5 Jan. 2003)

guess wrong.[5] Therefore, unless an IDS has a way of telling what operating system the packets are intended for, the IDS cannot be certain how the end system will handle the packets.

**IDS Management Console**

- **Manipulation of the Database –** An IDS not only helps a company know when they are being attacked, but also how they are being attacked. This is presented to the customer through the information that is stored in the management database. Therefore, it is critical that the information stored in this database is accurate. Otherwise, an intrusion analyst can be sent on a wild goose chase while the attacker is having fun doing whatever he/she so desires.
- **False Positives -** A risk can be realized when an intrusion analyst receives too many alerts that are not real attacks (false positives). When this happens, a couple of undesirable circumstances can occur. An intrusion analyst could be tracking down a false positive while a real attack is occurring, or the analyst might overlook a real attack because he/she just chalks it up as a false positive.
- **Analyst Limitations –** Sometimes an analyst will evaluate an intrusion attempt and decide it is not worth investigating, or may not report something he/she does not understand.[6] If this is the case, the IDS has reported the issue; but, the attack still goes unnoticed because of human error.

As can be seen, there are a multitude of risks listed here as well as additional risks not listed that if realized can make the IDS ineffective. At the same time, some of these risks have a higher likelihood of being realized than others. This likelihood needs to be taken into account when mitigating risks. Either way, given its critical function within the organization, this audit should be followed to verify that a secure configuration exists for all components.

### 1.3: What is the Current State of Practice, If Any?

After researching a number of Internet sites and technical books, I discovered a considerable amount of information pertaining to testing an IDS, creating benchmarks, and the current state of the practice. Many of these documents provide helpful information in determining what to audit on an IDS (see table 2); however, even these documents seem to contradict each other at times or are not completely thorough. In fact, according to NFR Security, some benchmarks may even provide misleading results that provide an analyst with a feeling that their IDS is more secure than it actually is. After this research, it is my opinion that a true, thorough audit checklist or benchmark for an IDS is difficult to create and does not truly exist. However, I do believe that as intrusion detection systems become more prevalent in today's business, individuals will continue to refine existing checklists and benchmarks that will more

[5] Graham, Robert. "FAQ: Network Intrusion Detection System." 21 Mar. 2000. URL: http://www.robertgraham.com/pubs/network-intrusion-detection.html (3 Jan. 2003).
[6] Northcutt, Stephen. Network Intrusion Detection: An Analyst's Handbook. Indianapolis, New Riders Publishing, 1999. 37.

effectively determine the validity of the IDS configuration. I wanted to partake in this refinement and that is why I chose this specific area for performing my audit.

| Document Title | Document Author | Internet Link |
|---|---|---|
| Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection | Thomas H. Ptacek and Timothy N. Newsham | www.snort.org/docs/idspaper |
| NFR Security: Experiences Benchmarking Intrusion Detection Systems | Marcus J. Ranum | www.snort.org/docs/Benchmarking-IDS-NFR.pdf |
| NSS IDS Group Test | N/A | www.nss.co.uk/ids/edition3 |
| 50 Ways to Defeat Your Intrusion Detection System | Fred Cohen | http://hackersplayground.org/papers/50_Ways_to_Defeate_Your_IDS.txt |
| 1999 DARPA Intrusion Detection Evaluation: Design and Procedures | J.W. Haines, R.P. Lippmann, and D.J. Fried. | www.ll.mit.edu/IST/ideval/pubs/2001/TR-1062.pdf |
| Open Source Security Testing Methodology Manual | N/A | http://www.isecom.org/projects/osstmm.htm |
| State of the Practice of Intrusion Detection Technologies | Julia Allen and Cohorts | www.sei.cmu.edu/pub/documents/99.reports/pdf/99tr028.pdf |
| Snort Configuration Documents | N/A | www.snort.org |

**Table 2: Current IDS Benchmark Documents**

Given this, I would like to discuss why, from my research, I feel that measuring the validity of an IDS is a difficult and consuming task.

1. As the complexity of a system increases, so does the validity of tests and/or benchmarks, as it is difficult to form tests that are not biased or inaccurate.
2. Many people look at external results for a final decision, not realizing that it is quite possible for a complex system to sometimes yield the right result for the wrong reason. As a thought experiment, consider an IDS that does nothing more than beep whenever it detects a packet crossing the network. Such an IDS will score 100% in terms of detecting attacks; i.e., nothing will get past it. However, it will be generating a lot of false positives.[7]
3. It is very difficult to simulate real traffic and attacks to fully test the functionality of an IDS. For example, the use of a synthetic load generator to test the performance of an IDS may not be a completely accurate test. This is because load generators were designed for load-testing routers, switches, and other devices that do not normally examine packet payload. In normal circumstances, traffic does not appear as anything similar to what a load generator provides.

[7] Ranum, Marcus J. "NFR Security: Experiences Benchmarking Intrusion Detection Systems." Dec. 2001. URL: www.snort.org/docs/Benchmarking-IDS-NFR.pdf. (7 Jan. 2003).

Therefore, the use of a load generator could actually make an IDS that is less rigorous in its testing look and perform better.

With this in mind, a good benchmark needs to take all of these concerns into account. In addition, a good benchmark needs to state what will be measured. For this, the answer is that the benchmark needs to measure how well the IDS detects intrusions or does not detect intrusions. In addition, the accuracy of detecting intrusions also needs to be determined. This involves the percentages of false negatives and false positives that occur. Furthermore, a thorough benchmark needs to go beyond this, as a system could be 100% accurate at catching intrusions, but still fail as an IDS as a whole.

This is because a distributed IDS consists of many different components beyond the IDS software listed in table 1. Therefore, in order for an IDS to be completely secure, all components of the IDS must be locked down. Fortunately, for most of these additional components, there are excellent security configurations and benchmarks that currently exist. These configurations and/or checklists are listed in this paper as part of the overall IDS benchmark. However, the main focus here is the functionality of the IDS as a whole, so detailed checklists for each of these components is beyond the scope of this paper.

With this said, the ability to benchmark the functionality of an IDS is a very difficult task, and is a work in progress due to the fact that intrusion detection systems are still so rapidly evolving. In addition, benchmarks that currently exist still need to be constantly updated or they will become out-of-date very quickly. Even so, there have been articles written such as “Fifty ways to bypass an intrusion detection system”, and some researchers report that there are proofs that show how intrusion detection systems will never be accurate. Therefore, creating an accurate and thorough benchmark for testing a distributed IDS is a challenging task that will require proving some people wrong. This is the main motivation for picking this subject for my audit, and a reason why I feel that a thorough audit can provide value to the security community.

## 2. Create an Audit Checklist

As can be seen above, there are many components that could possibly be vulnerable in an intrusion detection system. This audit needs to address each component of the IDS to make sure they are all configured securely. These components include:

- The Intrusion Detection Software (Snort)
- The IDS Probe and Management Console Operating Systems (Linux Red Hat)
- The Web Server (Apache)
- The Analyst Console (ACID)
- The Storage Database (MySQL)
- The Front-End Interface (SnortCenter)

The primary component that will be audited is the intrusion detection software, Snort. This is because the primary purpose of an IDS is to detect intrusions, and this would not

be possible without the intrusion detection software. However, to ensure the overall integrity of the IDS, it is necessary to audit each sub-component. Keep in mind that each of these sub-components could be an entire audit by themselves so if one of these sub-components seems to be an area of concern, it may be best to investigate that component further via additional documented audits. That is, beyond the IDS, the sub-components have many detailed documents that provide checklists and guidelines for secure configuration. Following are the checklist items that are necessary for the overall security of the distributed Snort IDS. These checklist items are broken down into items pertaining to the IDS probe, items pertaining to the IDS management console, and items pertaining to both devices.

## 2.1. IDS Probe and Management Console Checklist Items

### 2.1.1. Verify the Existence of a Security Policy

| | |
|---|---|
| **References** | 1. Personal Knowledge and Experience<br>2. www.sans.org/resources/policies |
| **Control Objective** | Verify that a comprehensive security policy exists. This security policy should provide the guidelines that the IDS adheres to. |
| **Risk** | A security policy is an absolute necessity for all organizations. Without a defined security policy, an organization has no basis for what is appropriate behavior. This puts an organization at risk by allowing an individual employee to determine what appropriate behavior is. |
| **Test Type** | Whether or not a security policy exists is objective. The details of what the policy contains and whether it is exhaustive enough is subjective. This is because each organization will have a different policy correlating to the organization's specific needs. |

| # | Tests | Compliance |
|---|---|---|
| 1 | Verify that a comprehensive security policy exists by requesting the security policy from the IS administrator. If a security policy exists, verify that the IDS is in compliance with the security policy, and that the security policy is supported from senior management on down.<br><br>In addition, talk to employees at all levels of the organization, and verify that they are aware of the security policy. | The IS administrator should be able to provide a documented security policy. Also, all employees should be aware of the current security policy. If this is not the case, it may be necessary to have a security awareness meeting for all employees.<br><br>**Note:** A specific example of the contents of a security policy is beyond the scope of this paper. This is because a security policy is specific to each organization and varies greatly depending on the organization. However, www.sans.org/resources/policies is a great resource for obtaining information regarding security policies if your organization is in need of creating or |

| | | |
|---|---|---|
| | | improving your security policy. |

### 2.1.2. Verify that the IDS Operating Systems are Regularly Patched

| | |
|---|---|
| **References** | 1. "Linux Benchmark v1.0.0." 16 Feb. 2002. URL: http://www.cisecurity.org/bench_linux.html (8 Feb. 2003).<br>2. Naidu, Krishni. "Auditing Linux." URL: http://www.sans.org/score/checklists/AuditingLinux.doc (18 Jan. 2003).<br>3. Wassom, Darrin. "Auditing a Distributed Intrusion Detection System: An Auditor's Perspective." July 2002. URL: http://www.giac.org/practical/Darrin_Wassom_GSNA.doc (3 Feb. 2003). |
| **Control Objective** | Verify that the IDS probe and management console operating systems are regularly updated with the most recent patches. |
| **Risk** | Installing up-to-date vendor patches and developing a procedure for keeping up with vendor patches is critical for the security and reliability of the system. Vendors will issue operating system updates when they become aware of security vulnerabilities and other serious functionality issues. Therefore, there should be a process in place for updating the operating system on the IDS probe and management console when new patches are released. |
| **Test Type** | All tests are objective. |

| # | Tests | Compliance |
|---|---|---|
| 1 | Enter **'rpm –qa > current_packages.txt'** to verify that both devices have all of the current operating system patches. | The contents of the 'current_packages.txt' file should align with Red Hat's list of current packages located at www.redhat.com/apps/support/errata/. |
| 2 | Ensure that there is a process in place to update the operating systems with the latest patches. Patches can be downloaded a couple of different ways:<br>1. Automatically download the packages by listing the download sites in the file '/etc/autorpm.d/pools/redhat-updates'. Make sure these download sites are secure, trusted sites.<br>2. Run up2date –l if the device has been registered with the Red Hat Network. | One of the listed methods, or an alternate method, should be used to make sure patches are installed when needed. Therefore, verify with the system administrator that a process is in place to update patches. In addition, analyze the IDS probe and management console to verify that the specified process is being performed; i.e., all current patches are installed on both ID systems. |

### 2.1.3. Eliminate Unneeded Services

| | |
|---|---|
| **References** | 1. Personal knowledge and experience |

| | |
|---|---|
| | 2. Naidu, Drishni. "Auditing Linux" URL: http://www.sans.org/score/checklists/AuditingLinux.doc (18 Jan. 2003).<br>3. "Linux Benchmark v1.0.0." 16 Feb. 2002. URL: http://www.cisecurity.org/bench_linux.html (8 Feb. 2003). |
| **Control Objective** | Verify that only services that are absolutely needed for the functionality of the IDS are enabled. |
| **Risk** | Vulnerabilities exist for most every service. The more services that are running on a device, the greater the opportunity for an intruder to penetrate the system in some manner. Therefore, if a service is not needed, it should be disabled or removed. |
| **Test Type** | All tests are objective. |

| # | Tests | Compliance |
|---|---|---|
| 1 | Enter **'less /etc/xinetd.conf'** to verify that the xinetd services are disabled.<br><br>In addition, this file may contain the line: 'includedir /etc/xinetd.d/'. If this line exists, enter **'ls –l /etc/xinetd.d'** to examine the contents of that directory. | The '/etc/xinetd.conf' file should not list any services. In addition, the '/etc/xinetd.d' directory should be empty or the services listed in this directory should be disabled. Therefore, if this directory is not empty, view each file **(less <file-name>)** and verify that there is a line that reads 'disable=yes'. |
| 2 | Turn off all services that are not needed. Enter **'chkconfig –list \| grep :on'** to determine which services are turned on for specific run levels and enter **'ps –ax'** to determine which services are currently running.<br><br>For all services that should be turned off, enter '**chkconfig –delete <*service*>'.**<br><br>**Note:** A service can always be turned back on if it is needed later by entering '**chkconfig –add <*add*>'.** | All services should be turned off unless absolutely needed. Following is a list of services that should be turned off unless absolutely needed:<br><br>apache, apmd, autofs, gated, gpm, httpd, innd, irda, isdn, kdcrotate, lpd, lvs, mars-nwe, named, netfs, nfs, nfslock, oki4daemon, portmap, postfix, postgresql, routed, rstatd, ruserd, rwalld, rwhod, smb, snmpd, squid, webmin, ypbind, yppasswd, and ypserv. |
| 3 | Enter **'netstat –ant'** to verify that only the needed daemons are active and listening on their respective port(s).<br><br>In addition, run NMap to verify that only the necessary ports are visible from the network. To do this, enter '**nmap –sS –p 1-65000 <*ip*>'** once where the ip is the IDS probe and another time where the ip is the IDS management console. | The daemons/services that should be running on the IDS probe are:<br><br>1. SSH (sshd)<br>2. MySQL<br><br>The daemons/services that should be running on the management console are:<br><br>1. SSH (sshd) |

| | | |
|---|---|---|
| | | 2. HTTP (httpd)<br>3. MySQL (mysqld)<br><br>The NMap output for the IDS probe should be:<br><br>Port 22/tcp State open Service ssh<br><br>The NMap output for the IDS management console should be:<br><br>Port 22/tcp State open Service ssh<br>80/tcp open http<br>443/tcp open https<br>3306/tcp open mysql |

### 2.1.4. Determine Vulnerabilities Present on Each Device

| | |
|---|---|
| **References** | 1. Personal Experience and Knowledge<br>2. http://www.nessus.org |
| **Control Objective** | Verify that vulnerabilities do not exist for the ports that need to be open on the IDS probe and management console. |
| **Risk** | If a known vulnerability exists for a service that needs to be open on either the probe or the management console, an attacker has an open door through which to enter. |
| **Test Type** | All tests are objective. |

| # | Test | Compliance |
|---|---|---|
| 1 | Run Nessus with the IDS probe and management console as the target. If a vulnerability is discovered on either device, try to exploit that device via the discovered vulnerability. | No serious or high vulnerability should be discovered. If a high or serious vulnerability is discovered and determined exploitable, it should be corrected immediately. |
| 2 | Verify that a procedure is in place for periodic vulnerability scans to be performed. In addition, verify that a procedure is in place for what corrective actions should be taken when a vulnerability is discovered. | There should be a procedure that an IS administrator can show that states the following:<br>• the interval between vulnerability scans.<br>• the corrective actions that should be taken when a vulnerability is discovered.<br><br>In addition, the IS administrator should be able to produce past vulnerability scans to verify that they are actually being performed as stated in the |

| | | |
|---|---|---|
| | | security policy. |

### 2.1.5. Backup and Recovery Procedure

| | |
|---|---|
| **References** | 1. Personal Experience and Knowledge |
| **Control Objective** | Verify that there is a tested backup of the IDS probe and management console as well as a recovery procedure in case of a system failure. |
| **Risk** | An IDS is a prime target for attackers if they discover that an IDS exists. If an attacker is able to penetrate into a portion of the IDS, he/she may alter files or run programs that could cause irreversible damage. If a tested backup and recovery procedure is not in place, critical information may be completely lost or an organization may be without an IDS for a considerable amount of time. |
| **Test Type** | This test is subjective. |

| # | Tests | Compliance |
|---|---|---|
| 1 | Verify that the IDS probe and management console are backed up on a regular basis and that these backups have been tested. In addition, make sure that there is a detailed procedure for disaster recovery. | Specific results are beyond the scope of this paper as this step is very dependent on the organization's specific security policy. For example, some organizations may require a completely separate backup site with completely redundant servers while other organizations may only require a backup tape or CD. |

### 2.1.6. Physical Security and Single-User Login

| | |
|---|---|
| **References** | 1. Fenzi, Kevin. "Linux Security HOWTO." June 2002. URL: http://www.linuxsecurity.com/docs/LDP/Security-HOWTO/ (22 Dec. 2003)<br>2. Mann, Scott and Mitchell, Ellen L. Linux System Security: The Administrator's Guide to Open Source Security Tools. Indianapolis, Prentice Hall PTR, 2000. 22 – 23. |
| **Control Objective** | Verify that the physical security of the IDS probe and management console is secure so that an intruder cannot physically reboot or shut the system down. |
| **Risk** | If an intruder is physically able to manipulate the IDS, he/she can bring the device down or alter it in some way. |
| **Test Type** | All tests below are objective |

| # | Tests | Compliance |
|---|---|---|
| 1 | Verify that the IDS probe and management console are in a physically secured environment. | Both devices should be located in a locked room and preferably in a locked cabinet with restricted and monitored access. |
| 2 | Make sure neither device is logged in locally at the console. | Physically go to the IDS probe and management console and verify that neither device is logged in. |

| | | |
|---|---|---|
| 3 | Enter '**less /etc/inittab**' to verify that rebooting from the console with the Ctrl+Alt+Del key sequence is disabled. | This file should reveal that the line 'ca::ctrlaltdel:/sbin/shutdown –t3 r now' is commented out; i.e., proceeded with a '#'. |
| 4 | Enter **'less /etc/lilo.conf'** to verify that a password is required when logging into single-user mode. In addition, reboot each device and enter **'linux single'** at the LILO: or boot: prompt to boot each device into single-user mode and verify that a password is required. | The line 'password=<password>' should exist in the '/etc/lilo.conf' file. If this is not the case, use a text editor to add this line. Then, enter **'sbin/lilo'** so the change will take effect.<br><br>When entering single-user mode, a password should be required. |
| 5 | Enter **'ls –l /etc/lilo.conf'** to verify that the /etc/lilo.conf file is only readable and writeable from root. | The output of this command should be:<br><br>-rw------- 1 root root ……<br><br>If this is not the case, other people may be able to read the single-user mode password. Therefore, enter **'chmod 600 /etc/lilo.conf'.**<br><br>In addition, try to read or write to this file while accessing each device via a user other than root. |

### 2.1.7. System Access, Authentication, and Authorization

| | | |
|---|---|---|
| **References** | 1. "Linux Benchmark v1.0.0." 16 Feb 2002. URL: http://www.cisecurity.org/bench_linux.html (8 Feb. 2003).<br>2. Wassom, Darrin. "Auditing a Distributed Intrusion Detection System: An Auditor's Perspective." July 2002. URL: http://www.giac.org/practical/Darrin_Wassom_GSNA.doc (3 Feb. 2003). | |
| **Control Objective** | Verify that access to both ID systems is set up securely. | |
| **Risk** | If an intruder is able to access the IDS, he/she can alter the system to his/her needs. | |
| **Test Type** | All tests are objective. | |
| **#** | **Tests** | **Compliance** |
| 1 | The '/etc/hosts.equiv' file sets up global trust relationships for all accounts on the system, which work in an analogous fashion to .rhosts files in user home directories. Therefore, this file should be removed from the IDS, so enter **'ls /etc/hosts.equiv'** to verify that it does not exist. | The output of this command should be:<br><br>**ls: /etc/hosts.equiv: No such file or directory**<br><br>If this is not the case, enter '**rm –f /etc/hosts.equiv'.** |

| | | |
|---|---|---|
| 2 | Enter **'less /etc/motd'** to verify that appropriate logon warning banners and legal notices are configured for remote access. | The contents of the '/etc/motd' file should contain a warning banner/legal notice. In addition, remotely access both ID systems and verify that the warning banner is displayed. |
| 3 | Enter **'less /etc/passwd'** to verify that only the needed user ID's are present on both devices. | There should not be any unused user ID's in this file. If an unused user ID is found, delete or disable that user ID. |
| 4 | Enter **'less /etc/login.defs'** to verify that minimum password length and maximum password age is being enforced on the IDS probe and management console.<br><br>If possible, use a password-cracking program such as John the Ripper or LC4 to determine if there are any passwords that are easy to crack. | The values of the 'PASS_MIN_LEN' and 'PASS_MAX_DAYS' variables should adhere to the organization's security policy. Therefore, attempt to change a user's password to a shorter length than the minimum length allowed by the security policy. You should not be allowed to do this.<br><br>If a password-cracking program is run, there should not be any passwords discovered. If there are, make sure these passwords are changed. |
| 5 | Enter **'awk –F: '($3==0) {print $1}' /etc/passwd'** to verify that no UID 0 accounts exist other than root. | This command should only return root. If another user is returned, change the UID for that user immediately. |
| 6 | Enter **'awk –F: '($2==""){print $1}' /etc/shadow'** to verify that there are no user accounts with empty password fields. | This command should not return anything. If a user is returned, create a password for that user immediately. |

### 2.1.8. Verify File Integrity via Tripwire

| | |
|---|---|
| **References** | 1. www.tripwiresecurity.com<br>2. Naidu, Krishni. "Auditing Linux" URL: http://www.sans.org/score/checklists/AuditingLinux.doc (18 Jan. 2003).<br>3. "Linux Security Quick Reference Guide." 2000. URL: http://www.linuxsecurity.com/docs/QuickRefCard.pdf (22 Jan. 2003). |
| **Control Objective** | Verify the integrity of the files located on the IDS probe and management console. |
| **Risk** | Tripwire utilizes message digest or other cryptographic checksums for critical files and objects, comparing them to reference values, and flagging differences or changes. The use of cryptographic checksums is important, as attackers often alter system files. Tripwire will allow you to monitor and track these changes so that you are aware when a critical file is altered unbeknownst to you. |
| **Test Type** | All tests are objective. |

| # | Tests | Compliance |
|---|---|---|

| | | |
|---|---|---|
| 1 | Verify that Tripwire is installed on the IDS probe and management console. | A tripwire configuration file (/etc/tripwire/twcfg.txt) should exist as well as a tripwire database (/var/lib/tripwire/host_name.twd). |
| 2 | Enter **'less /etc/tripwire/tw.cfg'** and **'less /etc/tripwire/tw.pol'** to verify that the configuration and policy are set up to comply with the security policy. | The configuration of these files is dependant on an organization's security policy; therefore, make sure these configurations comply with your organization's security policy. |
| 3 | Enter **'less /var/lib/tripwire/host_name.twd'** to verify that the tripwire database contains all critical system files that you wish to monitor. | This database should contain all critical system files that you wish to monitor. This list will vary according to an organization's security policy. |
| 4 | Verify that an initial baseline was run on each device and that this information is stored on a write-protected floppy or CD. | Talk with the security administrator to verify that an initial baseline was run. Ask for verification by requesting a copy of the initial baseline. Examine this baseline yourself to verify its integrity. |
| 5 | Verify that Tripwire is run at regular specified intervals. | There should be a file called tripwire-check in the /etc/cron.daily directory that will automatically run an integrity check once per day. If this is not the case, an integrity check should be manually run at specified intervals by entering '**/usr/sbin/tripwire –check'**. |
| 6 | Verify that the database gets updated when legitimate system changes are made. | Make a change to a critical system file. Then, enter **'/usr/sbin/tripwire –update –twrfile /var/lib/tripwire/report/<name>.twr'.** The database should get updated with this change. |
| 7 | Determine what corrective actions are taken if there are variances; i.e., changed files. | Talk with the security administrator and request to see the policy that states what actions are taken when a variance is discovered. |
| 8 | Ensure that Tripwire sends alerts to the appropriate system administrator if a modification has occurred. | The rule directive section of each rule should have an "emailto=" line. If it does not, use a text editor to make sure that line exists. To make sure that Tripwire's email notification configuration can actually send email correctly, use the following command: "/usr/sbin/tripwire –test –email your@email.address". Verify that you receive an email. |

### 2.1.9. Verify File System Permissions are Set Accordingly

| | |
|---|---|
| **References** | 1. Personal Experience and Knowledge |
| **Control Objective** | Verify that all system configuration files and sensitive user files have restricted permissions. |
| **Risk** | If a critical system file has world rights, these files may be able to be undesirably altered or viewed. |
| **Test Type** | This test is objective. |

| # | Tests | Compliance |
|---|---|---|
| 1 | Verify that all critical system files have the appropriate permissions. | Enter '**ls –l <directory or file name>**'. Verify that the permissions are set up according to the organization's security policy for each critical system file. |

### 2.1.10. Verify Sufficient Logging is being Performed

| | |
|---|---|
| **References** | 1. Poppi, Sandro. "Snort-Setup for Statistics HOWTO." 23 Feb 2002. URL: http://www.tldp.org/HOWTO/Snort-Statistics-HOWTO/configuration.html (19 Feb. 2003).<br>2. Naidu, Krishni. "Auditing Linux." URL: http://www.sans.org/score/checklists/AuditingLinux.doc (18 Jan. 2003).<br>3. "Linux Security Quick Reference Guide." 2000. URL: http://www.linuxsecurity.com/docs/QuickRefCard.pdf (22 Jan. 2003). |
| **Control Objective** | Verify that logging is being done on the necessary facilities, log files are rotated on a regular basis, and log files are regularly checked via an automated or manual method. |
| **Risk** | A good logging system is a record of what an intruder may have done to manipulate the system. If a good logging system is not in place, an organization may have no idea how a system has been compromised. |
| **Test Type** | All tests are objective. |

| # | Tests | Compliance |
|---|---|---|
| 1 | Enter **'less /etc/syslog.conf'** to verify that warnings and errors on all facilities are being logged and that all priorities on the kernel facility are being logged. | The output of this command should show that facilities are being logged at a level that is in compliance with the organization's security policy. If not, use a text editor to modify this file. |
| 2 | Enter **'ls –l <file-name>'** to verify that all log-related files and directories (/var/log, /etc/logrotate.d, /etc/syslog.conf, /etc/logrotate.conf, and /var/log/*log) have the appropriate permissions. | The output will provide the permissions that exist for each file or directory listed. In most cases, a permission of 640 (–rw-r----- or drw-r-----) is sufficient, but there may be a need for certain files to have higher permissions. If so, use 'chmod' to alter the permissions. |
| 3 | Enter **'less /etc/cron.daily'** and **'less /etc/cron.daily'** to verify that logs are | The file 'logrotate.conf' should show that log files are rotated regularly. In |

| | | |
|---|---|---|
| | being rotated on a regular basis and in compliance with the security policy. | addition, 'cron.daily' should list logrotate in its file. |
| 4 | Verify that logs are being reviewed on a regular basis. | There are various options to make sure an organization is in compliance. All of these options fall under automated or manual review. The more preferable method is an automated review such as Swatch. |
| 5 | If Swatch is being used, enter **'less /etc/swatch/swatchrc'** to verify that Swatch is properly configured. | Verify that the swatch configuration file is configured to review the necessary logs and notify the appropriate personnel when needed. |

### 2.1.11. SSH Configuration

| | |
|---|---|
| **References** | 1. Naidu, Krishni. "Auditing Linux" URL: http://www.sans.org/score/checklists/AuditingLinux.doc (18 Jan. 2003). |
| **Control Objective** | Verify that SSH is configured securely and that it is being used for any required remote administration. |
| **Risk** | If remote administration is being handled by another protocol other than SSH, chances are that information is sent in clear text. If this is the case, someone may be able to capture data, obtain passwords, or gather other critical pieces of information. Furthermore, if SSH is not configured securely, an intruder may be able to use this port as an avenue into the IDS. |
| **Test Type** | All tests are objective. |

| # | Tests | Compliance |
|---|---|---|
| 1 | Enter **'netstat –at \| grep ssh'** and/or **'ps –ax \| grep ssh'** to verify that SSH is running. In addition, verify that you can connect to both ID devices using an SSH client. | SSH should be in a state of LISTEN as well as '/usr/sbin/sshd' should be running. You should also be able to connect to both the IDS probe and management console via SSH. |
| 2 | Enter **'ssh –V'** to verify that the most recent and stable version of SSH is running. | At the time of this paper, 3.5p1 is the most recent release. Verify this at ftp://ftp.openbsd.org/pub/OpenBSD/OpenSSH/portable/. |
| 3 | Enter **'less /etc/rc.d/rc.local'** and/or **'ls /etc/rc3.d'** to verify that the SSH daemon is started at boot time. | The 'rc.local' file should contain the line '/usr/local/sbin/sshd' or the 'rc3.d' file should contain S##sshd where ## is the boot up sequence number; e.g., S55sshd. |
| 4 | Enter **'less /etc/hosts.allow'** to verify that this file is set up for SSH access. In addition, attempt to access the ID devices from IP addresses that are not allowed. Verify that you are not able to | This file should only list the IP addresses of the hosts that have permission to remotely access these devices. If IP addresses are not entered or if IP addresses are entered |

| | | |
|---|---|---|
| | connect. | that should not be entered, edit this file. |
| 5 | Enter **'less /etc/ssh/ssh_config'** to verify that this file is properly configured. | Verify that this configuration file is in compliance with your organization's security policy. |
| 6 | Enter **'less /etc/ssh/sshd_config'** to verify that this file is properly configured. | Verify that this configuration file is in compliance with your organization's security policy. |

### 2.1.12. Restrict Root Access

| | |
|---|---|
| **References** | 1. Mann, Scott and Mitchell Ellen L. Linux System Security: The Administrator's Guide to Open Source Security Tools. Indianapolis, Prentice Hall PTR, 2000. 518. |
| **Control Objective** | Verify root access is restricted as much as possible on the IDS probe and management console. |
| **Risk** | If an intruder is able to gain root access, he/she can manipulate the system in any way he/she desires. Therefore, it is critical to make sure that this does not happen. |
| **Test Type** | Objective |

| # | Tests | Compliance |
|---|---|---|
| 1 | Enter **'less /etc/securetty'** to verify that this file is empty. | This file should be empty to ensure that native root logins are entirely disallowed. If this file is not empty, edit the file and remove all entries. |
| 2 | Enter **'less /etc/pam.d/su'** to verify that access to the root account is disabled through su in order to force all root access through sudo. This allows for the best audit trail for all root use and gives the greatest control over the privileges granted to particular users. | This file should contain the following line:<br>**'auth requisite /lib/security/pam_deny.so'**<br>If this line does not exist, use a text editor to add it. |

### 2.1.13. Verify Correct Time

| | |
|---|---|
| **References** | 1. Personal Experience and Knowledge |
| **Control Objective** | Verify that the time is correct on the IDS probe and management console. |
| **Risk** | If an attack against the IDS or another network device is reported, it is critical that the time on the IDS is accurate so that it can be determined when the attack occurred. |
| **Test Type** | All tests are objective. |

| # | Tests | Compliance |
|---|---|---|
| 1 | Enter **'date'** to verify that the system time and date for both devices is correct. | This output should provide the current time and date. If not, use the **date** command to set the correct time. |
| 2 | Enter **'ps –ax \| grep xntpd'** to verify that an xntpd server is being used so | This command should show that an xntpd server is running. If an xntpd |

| | | |
|---|---|---|
| | that there is a process in place to make sure that the time remains accurate. | server is not running, set up xntpd and start the process. |

### 2.1.14. MySQL Database Configuration

| | |
|---|---|
| **References** | 1. Maple, Ryan W. "MySQL Security." 24 Aug. 2000. URL: http://www.linuxsecurity.com/tips/tip-24.html (2 Mar. 2003).<br>2. "MySQL Reference Manual." URL: http://www.mysql.com/doc/en/index.html (21 Feb. 2003).<br>3. Wassom, Darrin. "Auditing a Distributed Intrusion Detection System: An Auditor's Perspective." July 2002. URL: http://www.giac.org/practical/Darrin_Wassom_GSNA.doc (3 Feb. 2003).<br>4. "ACID: Installation and Configuration." 9 Oct. 2002. URL: http://www.andrew.cmu.edu/~rdanyliw/snort/acid_config.html (7 Mar. 2003). |
| **Control Objective** | Verify that the MySQL database is securely configured. |
| **Risk** | The MySQL Database contains critical information pertaining to attacks or potential attacks against your organization. If that database is compromised , the integrity of the database may be eliminated. Therefore, it is vital to make sure all aspects of the MySQL database are configured securely. |
| **Test Type** | All tests are objective. |

| # | Tests | Compliance |
|---|---|---|
| 1 | Enter **'mysql –V'** to verify that MySQL is running the most recent version. | At the time this paper was written, the current MySQL version is 3.51.06. If the output does not coincide with this, make sure MySQL gets upgraded as soon as possible. |
| 2 | Enter **'mysql –u root mysql'** and **'mysql –u root mysql –p'** to verify that the root password is set and protected. | The 'mysql –u root' command should result in the following error message:<br><br>ERROR 1045: Access denied for user: 'root@localhost' (Using password: NO)<br><br>The 'mysql –u root –p' command should prompt for a password. |
| 3 | Enter **'show grants'** within the MySQL database to verify that only the mysql root user has access to the 'user' table. | If the output of 'show grants' reveals that a user other than root has access to the 'user' table, use the 'revoke' command to eliminate this access. |
| 4 | Enter **'SELECT * FROM user WHERE User = '';'** and **'SELECT * FROM db WHERE Host = '%';'** to verify that default users and tables do not exist within the MySQL database. The | These commands should result in zero rows being returned. |

| | | |
|---|---|---|
| | default users are for connecting to the DBMS without specifying a password, and there are also entries so that tables called or starting with 'test' can be world-writable. | |
| 5 | Enter **'show databases'** from a MySQL session to verify that only the necessary databases are present on the IDS management console. | Only the necessary databases should be listed. |
| 6 | Enter **'SELECT * FROM User;'** from a MySQL session to verify that only the required users are configured. | This output should list only the required users. |
| 7 | Enter **'show grants for snort@localhost;'** from a MySQL session to verify that the rights for user 'snort' are sufficient for its needs, but do not exceed what is needed. | The rights that should be listed for user snort are: CREATE, INSERT, SELECT, DELETE, and UPDATE. |
| 8 | Verify that all data that is transferred from the IDS probe to the IDS management console is encrypted. Use a general sniffer such as Ethereal to capture traffic that is sent between the two devices. | The output of the sniffer data capture should not produce any clear text data. |
| 9 | Enter **'echo "SELECT count(*) FROM event" \| mysql snort_db –u root –p;'** to verify that successful logging of alerts is being sent from the sensor to the MySQL database. | This command should return output similar to:<br><br>Count(*)<br>#<br><br>where # is a number greater than zero. |

## 2.2. IDS Probe Checklist Items

### 2.2.1. Make Sure the Probe Interface is in "Stealth" Mode

| | |
|---|---|
| **References** | 1. Personal Knowledge and Experience |
| **Control Objective** | Verify that an IP address is not associated with the interface that is gathering the data to be analyzed. |
| **Risk** | An IDS is quite often placed on a network that is publicly addressed, and, therefore, accessible to the entire world. If this is the case, it is critical that the IDS probe be configured as secure as possible. This is necessary because if an intruder is trying to get into your network and realizes that an IDS is in place, he/she will likely try to take the IDS out of the equation by disabling it in some manner. By placing the IDS into "stealth" mode (not assigning it an IP address) on the data gathering interface, it will be much more difficult for an intruder to |

| | |
|---|---|
| | discover that an IDS is in place. |
| **Test Type** | All tests are objective. |

| # | Tests | Compliance |
|---|---|---|
| 1 | Physically check that there are two separate network interfaces. | The IDS probe should have two Ethernet cables connected to two different network interface cards, one for data gathering and one for management. Each cable should connect to two different hubs or switches. |
| 2 | Enter **'ifconfig'** to verify that the interface designated for data gathering does not have an IP address. | There should be an interface that does not have an IP Address. Most likely this will be either 'eth0' or 'eth1'. |
| 3 | Enter **'less /etc/rc.d/init.d/snortd'** to determine which interface is used by Snort to gather data. | The interface listed for the 'INTERFACE=' variable should be the same interface that '**ifconfig'** reported as not having an IP address in step 2. |
| 4 | Enter **'cat /proc/sys/net/ipv4/ip_forward'** to verify that IP Forwarding is disabled. | The output should be 0. If the output is 1 or something other than 0, enter '**echo 0 > /proc/sys/net/ipv4/ip_forward**'. |

### 2.2.2. Base IDS Info

| | |
|---|---|
| **References** | 1. Personal Experience and Knowledge |
| **Control Objective** | Verify that the Base IDS parameters are configured correctly; i.e., Snort is running, Snort is automatically started at startup, and the latest release of Snort is installed. |
| **Risk** | These three items are critical as this verifies that Snort is running and it is running the latest and greatest release. If either is not the case, this may provide an organization with a sense of security that does not exist. |
| **Test Type** | All tests are objective. |

| # | Tests | Compliance |
|---|---|---|
| 1 | Enter **'ps –ax \| grep snort'** to verify that Snort is running and enter '**ls /etc/rc.d/init.d'** and/or **'chkconfig –list \| grep snortd'** to verify that snort is automatically started if the IDS probe gets rebooted for some reason. | The output of 'ps –ax \| grep snort' should produce a line proving that snort is running. The output of 'ls /etc/rc.d/init.d' should list snortd as one of the files. The output of 'chkconfig –list \| grep snortd' should show runlevel 3 (3:on) as on. |
| 2 | Enter **'snort –V'** to verify the latest and greatest release of Snort is installed by comparing it against the current version listed at www.snort.org/dl. | At the time this paper was written, Snort 1.9.1 is the latest and greatest version of Snort. |

### 2.2.3. Base Snort Configuration

| | |
|---|---|
| **References** | 1. Roesch, Martin, and Green, Chris. "Snort Users Manual Snort Release: 1.9.1." URL: http://www.snort.org/docs/writing_rules/index.html (4 Jan. 2003).<br>2. Poppi, Sandro. "Snort-Setup for Statistics HOWTO." 23 Feb. 2002. URL: http://www.tldp.org/HOWTO/Snort-Statistics-HOWTO/index.html (19 Feb. 2003). |
| **Control Objective** | Verify that Snort is configured correctly and in a secure manner to coincide with the organization's security policy. |
| **Risk** | If Snort is not configured correctly or in the optimal manner for the organization, it may not behave in the expected manner. This may result in an organization feeling that they are more secure than they actually are. For example, if a rule is not defined for an attack specific to an organization, then Snort will have no way of reporting that specific attack occurred. |
| **Test Type** | All tests are subjective. |

| # | Tests | Compliance |
|---|---|---|
| 1 | Enter **'less /etc/snort/snort.conf'** to verify that the 'Variables' section of the Snort configuration file is set up correctly to reflect the network topology being protected. | This section of the configuration should have all variables (HOME_NET, EXTERNAL_NET, DNS_SERVERS, etc.) defined that are necessary. |
| 2 | Enter **'less /etc/snort/snort.conf'** to verify that the 'Preprocessors' section of the Snort configuration file is set up correctly. | This section of the configuration should list all necessary preprocessors that are required for the desired functionality. |
| 3 | Enter **'less /etc/snort/snort.conf'** to verify that the 'Output Modules' section of the Snort configuration file is set up to send alerts and logs to the appropriate server(s) and/or database(s). | This section of the configuration file should have the output log information (database name, database type, database server, etc.) defined that is necessary. |
| 4 | Enter **'less /etc/snort/snort.conf'** to verify that the 'Rule Set' section of the snort configuration file is set up correctly to coincide with the organization's security policy. | This section should have all rules defined, default and customized, that are necessary for the organization. |

### 2.2.4. Snort Performance

| | |
|---|---|
| **References** | 1. Poppi, Sandro. "Snort-Setup for Statistics HOWTO." February 23, 2002. URL: http://www.tldp.org/HOWTO/Snort-Statistics-HOWTO/configuration.html (19 Feb. 2003). |
| **Control Objective** | Verify that the IDS does not cease operation at a certain performance level or does not start dropping packets. |
| **Risk** | If an intruder is able to send a substantial amount of data to the IDS |

| | |
|---|---|
| | and, thereby, shut down the IDS or cause the IDS to drop packets, he/she can then penetrate the network undetected. |
| **Test Type** | This test is objective. |

| # | Tests | Compliance |
|---|---|---|
| 1 | Determine if packets are being dropped by the IDS probe by performing the following steps:<br><br>1. **ps –ax \| grep snort**<br>2. **kill –SIGUSR1 <*pid of snort*>**<br>3. **less /var/log/messages** (look for the line that displays the number of packets that was dropped by Snort). | There should be a line that states 'Snort analyzed x out of y packets' followed by a line that states 'dropping y (z%) packets'. X and y should be equal and y and z should both equal 0. If this is not the case, probe further into determining why or at what point Snort starts dropping packets; otherwise, restart snort. |

### 2.2.5. Vulnerability and Port Scan Recognition

| | |
|---|---|
| **References** | 1. "IDS Group Test." July 2002. URL: http://www.nss.co.uk/download_form.htm (12 Feb. 2003).<br>2. Ranum, Marcus J. "NFR Security: Experiences Benchmarking Intrusion Detection Systems." Dec 2001. URL: http://www.snort.org/docs/Benchmarking-IDS-NFR.pdf (7 Jan. 2003).<br>3. Wassom, Darrin. "Auditing a Distributed Intrusion Detection System: An Auditor's Perspective." July 2002. URL: http://www.giac.org/practical/Darrin_Wassom_GSNA.doc (3 Feb. 2003). |
| **Control Objective** | Verify that the IDS recognizes an acceptable level of all known vulnerabilities. |
| **Risk** | If an IDS does not accurately detect current known vulnerabilities, then an intruder will be able to go undetected while compromising the network. |
| **Test Type** | This test is subjective. |

| # | Tests | Compliance |
|---|---|---|
| 1 | Verify that the IDS accurately reports when port and/or vulnerability scans are run against the network that the IDS is protecting. Scan the network using various scanning tools to make sure that the IDS catches all scans from all tools. For this test, we will utilize NMap and Nessus although there are many additional tools that could be utilized. | The IDS analyst console should report the Nessus vulnerability scan and the NMAP port scan as attacks and display all scans that were attempted. |

### 2.2.6. Regularly Update the Signature Database

| | |
|---|---|
| **References** | 1. Personal Experience and Knowledge |

| | |
|---|---|
| **Control Objective** | Verify that there is a process in place to regularly update the signature database. |
| **Risk** | New vulnerabilities are discovered constantly. If the signature database is not updated frequently, attacks will exist that are not part of the signature database, and an intruder will be able to use these attacks to send traffic past the IDS undetected. |
| **Test Type** | This test is objective. |

| # | Tests | Compliance |
|---|---|---|
| 1 | Verify that there is a process in place for updating signatures (preferably automated) and that signatures are updated within an acceptable time level. | Rules can be downloaded manually from the Snort web page (www.snort.org/dl/rules/), or if you are using Snort Center to manage your IDS, you can automatically update rules by performing the following:<br>• On the management console, make sure that the 'config.php' file contains the parameter: $User_authentication = 2;<br>• Create the following cron job on the management console: 'wget –O – http://localhost/db_pars.php?source=net&push=yes<br><br>In either case, verify that the latest snort rule set is installed. |

### 2.2.7. Snort Defense Against Known Attacks

| | |
|---|---|
| **References** | 1. "IDS Group Test." July 2002. URL: http://www.nss.co.uk/download_form.htm (12 Feb. 2003).<br>2. Ranum, Marcus J. "NFR Security: Experiences Benchmarking Intrusion Detection Systems." Dec 2001. URL: http://www.snort.org/docs/Benchmarking-IDS-NFR.pdf (7 Jan. 2003).<br>3. Roesh, Marty. "SNORT FAQ." 25 Mar. 2002. URL: http://www.snort.org/docs/faq.html (16 Feb. 2003).<br>4. Herzog, Pete. "Open Source Testing Methodology Manual." URL: http://www.isecom.org/projects/osstmm.htm (13 Mar. 2003).<br>5. Graham, Robert. "FAQ: Network Intrusion Detection Systems." URL: http://www.robertgraham.com/pubs/network-intrusion-detection.html (3 Jan. 2003).<br>6. Cohen, Fred. "50 Ways to Defeat Your Intrusion Detection System." 16 Oct. 2002. URL: http://hackersplayground.org/papers/50_Ways_to_Defeate_Your_IDS.txt (26 Feb. 2003). |
| **Control** | Verify that the IDS is not susceptible to any known attacks. |

| Objective | |
|---|---|
| Risk | If an IDS is susceptible to known attacks, an intruder may be able to use these attacks to go unnoticed through the IDS. |
| Test Type | All tests are subjective. This is because these tests rely on an intrusion analyst or engineer to run these tests and determine whether or not the IDS is behaving correctly. As mentioned in the first section, this could result in an analyst wrongly interpreting the results. |

| # | Tests | Compliance |
|---|---|---|
| 1 | Run several common exploits in their normal form followed by the same exploits altered by various insertion or evasion techniques. Examples of possible insertion or evasion attacks include:<br><br>RPC record fragging, inserting spaces in command lines, inserting non-text telnet opcodes in data stream, URL encoding, /./ directory insertion, premature URL ending, and Long URL. | For each of the attacks, note if (1) the attempted attack is detected in any form, and (2) if the exploit is successfully "decoded" to provide an accurate alert relating to the original exploit, rather than alerting purely on anomalous traffic detected as a result of the evasion technique itself.<br><br>Ideally, the exploit should alert as the original exploit; however, alerting as any form of exploit is better than not alerting at all. |
| 2 | Verify that the IDS is not vulnerable to noise generators such as "Stick" and "Snot". Verify this by running each application, and then analyzing the output that Snort generated. | After these applications are executed, the IDS management console should show that a large number of alerts were generated; however, the IDS should not drop any alerts or fail.<br><br>If the stream4 preprocessor is configured properly, these types of applications should be easily defeated. |
| 3 | Verify that the IDS is not susceptible to speed adjustments. This test measures the IDS's sensitivity to scans over definitive time periods. Due to the volume of traffic on the wire, ID systems have difficulty maintaining long-term traffic logs. It is therefore difficult to detect "slow scans" (ping sweeps or port-scans) where intruders scan one port/address every hour. | Use NMAP and/or various other tools to run a slow port scan and a slow ping sweep. The output of these scans/sweeps should be the same as a normal scan/sweep; i.e., Snort should still detect these scans as an attack. |
| 4 | Run several attacks via fragrouter using various IDS fragmentation evasion techniques such as ordered IP fragments of various sizes, out-of-order IP fragments of various sizes, and | Snort should detect these attacks and send them to the appropriate personnel and list them on the ACID console. |

| | | |
|---|---|---|
| | duplicate fragments. | |
| 5 | Perform any additional testing that an organization feels is needed to have a satisfied comfort level of the performance and functionality of the IDS in place. The article listed in the reference section above, 50 ways to defeat your intrusion detection system, is a good place to start for ideas on testing. Some of these tests have been addressed or are not relevant; however, it is always good to do whatever testing an organization feels is needed. | If additional attacks are run through the IDS, the IDS should detect each and every attack. If this is not the case, the organization should determine an alternate method for detecting any attacks that went undetected through the IDS or create a specific rule that will detect this attack. |

### 2.2.8. Test For False Positives and False Negatives

| | |
|---|---|
| **References** | 1. Personal Experience and Knowledge |
| **Control Objective** | Verify that Snort is correctly configured so that there is not an excessive number of false positives and/or false negatives. |
| **Risk** | If a high number of false positives occur, an intrusion analyst may start to ignore attacks that are recorded. Then, an attack may go unnoticed even if it is recorded by Snort. On the flip side, if a high number of false negatives occur, attacks are occurring that are going unnoticed by Snort. In either case, there is the risk that an organization will miss attacks even with an IDS in place. |
| **Test Type** | All tests are subjective. This is because they rely on an intrusion analyst or engineer to run these tests and determine whether or not the IDS is behaving correctly. As mentioned in the first section, this could result in an analyst wrongly interpreting the results. |

| # | Tests | Compliance |
|---|---|---|
| 1 | Verify that the number of false positives that occur are within the acceptable level outlined by the organization's security policy. This can be verified by examining the ACID console, database, and logs. | Monitor the detected alerts for a week or two. For each alert that is detected, verify that it is a legitimate attack. If it is not, record that alert as a false positive. Compare the number of generated false positives against the total number of detected alerts. The percentage of false positives should fall within the acceptable level outlined by the organization's security policy. If this is not the case, alter the base configuration and rules examined until this percentage is at an acceptable level. |
| 2 | Verify that false negatives are not occurring. | Unfortunately, there is no way to guarantee that false negatives are not occurring. However, the best way to |

| | | |
|---|---|---|
| | **Note:** This is much harder to verify than false positive occurrence because there is nothing to notify an organization if an attack is missed. | at least be fairly confident that this is not the case is to run various known attacks through the IDS and verify that the IDS is detecting all of these attacks. Periodically, run various attacks through the IDS to make sure that the IDS continues to detect all attacks that you are aware of. |

### 2.2.9. Verify Correct IDS Placement

| | | |
|---|---|---|
| **References** | 1. Northcutt, Stephen and McLachlan, Donald and Novak, Judy. Network Intrusion Detection: An Analyst's Handbook (2nd Edition). Indianapolis, New Riders Publishing, 2000. | |
| **Control Objective** | Verify that the IDS is placed in a position where it will monitor all of the data that the system is intended to monitor. | |
| **Risk** | An IDS cannot catch an alert that it does not see. Therefore, it is critical that the IDS be placed in a position where it will see all intended data. | |
| **Test Type** | This test is objective. | |
| **#** | **Tests** | **Compliance** |
| 1 | Verify that the IDS probe is placed in a position on the network where it will collect all traffic that it is intended to. | The IDS should be placed in a hub or in a switch port that is able to span the necessary ports. In addition, it should be placed between the network(s) that you need to protect and the network(s) that you are trying to protect it from. |

## 2.3. IDS Management Console Checklist Items

### 2.3.1. Apache Security

| | |
|---|---|
| **References** | 1. "Apache Security Tips." URL: http://httpd.apache.org/docs-2.0/misc/security_tips.html (13 Mar. 2003).<br>2. "Using User Authentication." 18 Oct. 1996. URL: www.apacheweek.com/features/userauth (7 Mar. 2003).<br>3. Naidu, Krishni. "Auditing Linux" URL: http://www.sans.org/score/checklists/AuditingLinux.doc (18 Jan. 2003).<br>4. "Linux Security Quick Reference Guide" 2000. URL: http://www.linuxsecurity.com/docs/QuickRefCard.pdf (22 Jan. 2003). |
| **Control Objective** | Verify that the Apache Web Server is configured securely. |
| **Risk** | Web servers are notorious for vulnerabilities; therefore, it is best to make sure that the web server is constantly patched and configured as |

| | |
|---|---|
| | secure as possible. |
| **Test Type** | All tests are objective. |

| # | Tests | Compliance |
|---|---|---|
| 1 | Enter **'httpd –V'** and compare this output with the current release listed at http://httpd.apache.org to verify that the current version of Apache is being used.<br><br>In addition, verify that there is a process in place for keeping track of new updates and for updating the server. | As of this paper, 2.0.44 is the current and best available version.<br><br>There should be a documented process for keeping track of updates and for updating the server when needed. One method for this is to subscribe to the Apache HTTP Server Announcements list where you can keep informed of new releases and security updates. |
| 2 | Enter **'less /etc/httpd/conf/httpd.conf'** to verify that the web server is only listening on the local interface. | This file should have a line that reads:<br>Listen 127.0.0.1:80<br><br>There should be no other lines in the configuration file that start with Listen. |
| 3 | Enter **'less /etc/httpd/conf/httpd.conf'** to verify that access is disabled to the entire file system by default, unless explicitly permitted. This will disable printing of indexes if no index.html exists, server-side includes, and following symbolic links.<br><br>Then, add appropriate directory blocks to allow access only in those areas you wish. | The following output should be seen in the httpd.conf file to disable the entire file system by default:<br><br><Directory /><br>Options None<br>AllowOverride None<br>Order deny, allow<br>Deny from all<br></ Directory><br><br>Then to allow access to all needed directories, the following output should be seen in the httpd.conf file for each needed directory:<br><br><Directory *directory-path*><br>Order deny, allow<br>Allow from all<br></Directory> |
| 4 | Enter **'less /etc/httpd/conf/httpd.conf'** to verify that access to the server is limited to specific addresses or networks. | The following output should be seen in the httpd.conf file:<br><br><Directory /var/www/html><br>Order deny, allow<br>Allow from 127.0.0.1<br>Allow from *x.x.x* |

| | | |
|---|---|---|
| | | Allow from *y.y.y.y*<br>Deny from all<br></Directory><br><br>where *x.x.x* is a specific network and *y.y.y.y* is a specific address that you want to be allowed access. You can enter as many 'Allow from' lines as needed. |
| 5 | Enter **'less /etc/httpd/conf/httpd.conf'** to verify that users are not able to set up .htaccess files. These files can override security features that the administrator has configured.<br><br>In addition, attempt to set up a '.htaccess' file via a user other than root and verify that this is not possible. | In the httpd.conf file, the following output should be seen:<br><br><Directory /><br>AllowOverride None<br></Directory><br><br>A user other than root should not be able to set up a '.htacess' file. |

### 2.3.2. Real-Time Alert Configuration

| | | |
|---|---|---|
| **References** | 1. Personal Experience and Knowledge | |
| **Control Objective** | Verify that Snort is configured to send real-time alerts via some sort of alerting method such as e-mails or pages. | |
| **Risk** | If real-time alerting is not configured correctly, there is no way for an analyst to be notified when an attack is occurring. Therefore, unless an analyst is looking at the management console constantly, an attack may not be noticed. At this point, the IDS is performing in the same manner as a normal logging system. | |
| **Test Type** | This test is objective. | |
| **#** | **Tests** | **Compliance** |
| 1 | Verify that real-time alerting is correctly configured. There are various ways to test this. One way is to run a vulnerability or port scan and verify that the configured pager and/or email address is notified appropriately. | The email address or pager should receive all alerts that align with the Snort configuration. |

### 2.3.3. ACID Configuration

| | |
|---|---|
| **References** | 1. Wassom, Darrin. "Auditing a Distributed Intrusion Detection System: An Auditor's Perspective." July 2002. URL: http://www.giac.org/practical/Darrin_Wassom_GSNA.doc (3 Feb. 2003).<br>2. www.cert.org/kb/acid<br>3. Scott, Steven J. "Snort Installation Manual: Snort, MySQL and ACID on Redhat 7.3." August, 2002. URL: |

| | |
|---|---|
| | http://www.snort.org/docs/snort-rh7-mysql-ACID-1-5.pdf (1 Mar. 2003). |
| **Control Objective** | Verify that ACID is configured securely. |
| **Risk** | If ACID is not configured securely, it may be possible for an intruder to access the ACID web console and make changes to the collected alerts. |
| **Test Type** | This test is objective. |

| # | Tests | Compliance |
|---|---|---|
| 1 | Examine the version seen on the bottom left of the main ACID web page to verify that ACID is running the latest version. | Compare this version to the current version on the ACID web site (www.cert.org/kb/acid). At the time of this paper, the current ACID version is v0.9.6b23. |
| 2 | Enter **'less /var/www/html/acid/acid_conf.php'** to verify that all settings are configured appropriately in the acid_conf.php file. Especially verify that the database and email alert action settings are set up correctly. | All settings should be set up to comply with the organization's security policy. In addition, the database settings should coincide with the settings defined in /etc/snort/snort.conf. |
| 3 | Verify the ACID management console can perform the necessary tasks. The tasks of most interest are the Email Alert feature, the Archive feature, and the Delete feature.<br><br>For the Email Alert feature, select an alert from the ACID console and use the drop-down box to email an alert.<br><br>For the Archive feature, select an alert from the ACID console and attempt to archive it (move and/or copy) to the archive database.<br><br>For the Delete feature, select an alert from the ACID console and attempt to delete it from the database. | All features should be functioning in the desired way. For the Email Alert Feature, an email should be received at each mailbox of the corresponding email that was entered. For the Archive and Delete features, it should be verified that the alert was either archived or deleted. This can be determined because after each action, SUCCESS or FAILURE will be shown on the ACID console web page. Therefore, SUCCESS should be seen after each of these actions. |
| 4 | Enter **'ls /var/www.html/acidviewer'** to verify that there is view-only ACID portal for people who only need to view alerts. (cp –R /var/www/html/acid /var/www/html/acidviewer) | This command should confirm that this directory exists. If not, enter **'cp –R /var/www/html/acid /var/www/html/acidviewer'.** In addition, you should be able to access the ACID console web page via acidviewer. |

| | | |
|---|---|---|
| 5 | Enter **'less /usr/lib/httpd/passwords/passwords'** to verify that both Acid websites are secured with Apache. | There should be a password for admin and acidviewer in this file.<br><br>**Note:** This directory may vary per system. |

### 2.3.4. SnortCenter Console Configuration

| | |
|---|---|
| **References** | 1. "SnortCenter Installation Manual." 2002. URL: http://users.pandora.be/larc/documentation/ (15 Mar. 2003). |
| **Control Objective** | Verify that Snort Center is configured securely. |
| **Risk** | An administrator is able to make configuration changes and updates to an IDS probe from the SnortCenter Console. Unfortunately, if an attacker can gain access to this console, he/she will also be able to make changes. Therefore, it is imperative that SnortCenter is configured securely. |
| **Test Type** | All tests are objective. |

| # | Tests | Compliance |
|---|---|---|
| 1 | Examine the version seen on the top left of the main SnortCenter web page to verify that SnortCenter is running the latest version. | Compare this version to the current version on the SnortCenter web site (http://users.pandora.be/larc/download ). At the time of this paper, the current SnortCenter version is v0.9.6. |
| 2 | Verify that the Login Name and Password have been changed from defaults. The default Login Name is 'admin' and the default Password is 'change'. | You should not be able to login to the SnortCenter console using the default parameters. |
| 3 | Verify that all data that is transferred from the IDS probe to the IDS management console is encrypted. Use a general sniffer such as Ethereal to capture traffic that is sent between the two devices. | The output of the sniffer data capture should not produce any clear text data. |
| 4 | Verify that the configuration file (config.php) is configured correctly. | All parameters should be configured appropriately. |

# 3. Audit Evidence

This section provides the evidence for this audit and whether each checklist item is or is not in compliance. Only ten checklist items are listed in this audit section due to the constraints of this project. It could be argued as to what ten items are most critical, because all checklist items listed in section 2 are critical to the overall security of the IDS. Therefore, a subjective decision had to be made as to what ten items were going

to be listed. It is important to note that even though every checklist item is not listed in this section, the audit was performed in its entirety for the purpose of the audit report. Following is a description of each item that was chosen to be examined in this audit section:

- Elimination of Unneeded Services
- Determine Vulnerabilities Present
- System Access and Authentication
- OpenSSH Configuration Verification
- Stealth Mode Verification
- Base IDS Configuration Verification
- Base Snort Configuration Verification
- Snort Performance
- Vulnerability and Port Scan Recognition
- Apache Security

Each checklist item listed has several tests associated with it. Every test for each checklist item was performed and the results are shown below. At the beginning of each checklist item, there is a table that describes every test associated with the checklist item and whether each test passed or failed. Then, at the bottom of the table, is a row that lists whether the overall checklist item passed or failed. The overall checklist item passed if every single test passed, and it failed if at least one test did not pass. Following is the output from the ten chosen checklist items.

## 3.1. Conduct the Audit

### 3.1.1. Eliminate Unneeded Services (Checklist Item 2.1.3)

The control objective for this checklist item is to verify that only services that are absolutely needed for the functionality of the IDS are enabled. Three specific tests were performed to verify that this control objective was met. These tests must be performed on both the IDS probe and management console. The table below shows the results of each individual test for both devices, as well as the overall result. Following this table is the output for each test.

| Test | Probe Result | Mgmt Result |
|---|---|---|
| Verify xinetd services are disabled | Pass | Pass |
| Turn off services that are not needed | Fail | Fail |
| Verify only needed services/daemons are listening | Pass | Pass |
| **OVERALL RESULT:** | **Fail** | **Fail** |

**Table 3: Test Results for Eliminating Unneeded Services**

#### 3.1.1.1 Verify xinetd services are disabled

The ‘etc/xinetd.conf’ file should not list any services. In addition, if a line stating ‘includedir /etc/xinetd.d’ exists in the ‘/etc/xinetd.conf’ file, then it is necessary to

examine the contents of that directory. If the '/etc/xinetd.d' directory is found to contain files, then you need to verify that each file contains the line 'disable=yes'. This output shows that all 'xinetd' services are disabled.

**IDS Probe Output**

```
[test@probe test]$ less /etc/xinetd.conf
# Simple configuration file for xinetd
# Some defaults, and include /etc/xinetd.d/

defaults
{
        instances               = 60
        log_type                = SYSLOG authpriv
        log_on_success          = HOST PID
        log_on_failure          = HOST
        cps                     = 25 30
}
```

**IDS Management Console Output**

```
[test@mgmt test]$ less /etc/xinetd.conf
# Simple configuration file for xinetd
# Some defaults, and include /etc/xinetd.d/

defaults
{
        instances               = 60
        log_type                = SYSLOG authpriv
        log_on_success          = HOST PID
        log_on_failure          = HOST
        cps                     = 25 30
}
```

### 3.1.1.2. Turn off services that are not commonly needed

Use the 'chkconfig' and 'ps' commands to determine what services are currently running and which run levels specific services will automatically be started upon reboot. This output shows that both the probe and the management console have services running that are not needed.

**IDS Probe Output**

```
[test@probe test]$ chkconfig --list | grep :on
ntpd            0:off   1:off   2:off   3:on    4:off   5:on    6:off
syslog          0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

```
snortd          0:off   1:off   2:on    3:on    4:on    5:on    6:off
netfs           0:off   1:off   2:off   3:on    4:on    5:on    6:off
network         0:off   1:off   2:on    3:on    4:on    5:on    6:off
random          0:off   1:off   2:on    3:on    4:on    5:on    6:off
rawdevices      0:off   1:off   2:off   3:on    4:on    5:on    6:off
xinetd          0:off   1:off   2:off   3:on    4:on    5:on    6:off
apmd            0:off   1:off   2:on    3:on    4:on    5:on    6:off
atd             0:off   1:off   2:off   3:on    4:on    5:on    6:off
gpm             0:off   1:off   2:on    3:on    4:on    5:on    6:off
autofs          0:off   1:off   2:off   3:on    4:on    5:on    6:off
keytable        0:off   1:on    2:on    3:on    4:on    5:on    6:off
kudzu           0:off   1:off   2:off   3:on    4:on    5:on    6:off
sshd            0:off   1:off   2:on    3:on    4:on    5:on    6:off
nfslock         0:off   1:off   2:off   3:on    4:on    5:on    6:off
rhnsd           0:off   1:off   2:off   3:on    4:on    5:on    6:off
crond           0:off   1:off   2:on    3:on    4:on    5:on    6:off
anacron         0:off   1:off   2:on    3:on    4:on    5:on    6:off
xfs             0:off   1:off   2:on    3:on    4:on    5:on    6:off
firstboot       0:off   1:off   2:off   3:off   4:off   5:on    6:off

[test@probe test]$ ps -ax
USER       PID %CPU %MEM   VSZ  RSS TTY      STAT START   TIME
COMMAND
root         1  0.0  0.2  1336  436 ?        S    Jan21   0:03 init
root         2  0.0  0.0     0    0 ?        SW   Jan21   0:00 [keventd]
root         3  0.0  0.0     0    0 ?        SW   Jan21   0:00 [kapmd]
root         4  0.0  0.0     0    0 ?        SWN  Jan21   0:00 [ksoftirqd_CPU0]
root         5  0.0  0.0     0    0 ?        SW   Jan21   0:11 [kswapd]
root         6  0.0  0.0     0    0 ?        SW   Jan21   0:00 [bdflush]
root         7  0.0  0.0     0    0 ?        SW   Jan21   0:00 [kupdated]
root         8  0.0  0.0     0    0 ?        SW   Jan21   0:00 [mdrecoveryd]
root        12  0.0  0.0     0    0 ?        SW   Jan21   0:03 [kjournald]
root        68  0.0  0.0     0    0 ?        SW   Jan21   0:00 [khubd]
root       160  0.0  0.0     0    0 ?        SW   Jan21   0:00 [kjournald]
root       425  0.0  0.3  1400  568 ?        S    Jan21   0:00 syslogd -m 0
root       429  0.0  0.2  1336  412 ?        S    Jan21   0:00 klogd -x
root       514  0.0  0.2  1328  412 ?        S    Jan21   0:00 /usr/sbin/apmd -p 10 -w
5 -W -P /etc/sysconfig/apm-scripts/apmscript
root       590  0.0  0.3  2088  624 ?        S    Jan21   0:00 xinetd -stayalive -reuse -
pidfile /var/run/xinetd.pid
ntp        604  0.0  1.2  1916 1908 ?        SL   Jan21   0:00 ntpd -U ntp
root       613  0.0  0.2  1372  376 ?        S    Jan21   0:00 gpm -t ps/2 -m
/dev/mouse
root       623  0.0  0.3  1516  576 ?        S    Jan21   0:00 crond
xfs        652  0.0  0.3  4420  628 ?        S    Jan21   0:00 xfs -droppriv -daemon
daemon     670  0.0  0.3  1368  496 ?        S    Jan21   0:00 /usr/sbin/atd
```

```
root       680  0.0  0.2  1316  352 tty2     S    Jan21   0:00 /sbin/mingetty tty2
root       866  0.0  1.2  7716 2044 ?        S    Jan21   0:01 /usr/bin/perl
/opt/snortagent/sensor/miniserv.pl /etc/snort/miniserv.
root       772  0.0  0.2  1316  348 tty1     S    Feb20   0:00 /sbin/mingetty tty1
root     16815  0.2 31.6 51624 50176 ?       S    Mar03  20:13 /usr/local/bin/snort -
D -i eth1 -U -o -c /etc/snort/snort.eth1.conf
root     24269  0.0  1.0  6696 1692 ?        S    15:32   0:00 /usr/sbin/sshd
502      24272  0.0  0.8  4132 1420 pts/0    S    15:32   0:00 -bash
502      24332  2.0  0.4  2684  716 pts/0    R    16:05   0:00 ps –aux
```

**IDS Management Console Output**

```
[test@mgmt test]$ /sbin/chkconfig --list | grep :on
ntpd            0:off   1:off   2:off   3:on    4:off   5:on    6:off
syslog          0:off   1:off   2:on    3:on    4:on    5:on    6:off
netfs           0:off   1:off   2:off   3:on    4:on    5:on    6:off
network         0:off   1:off   2:on    3:on    4:on    5:on    6:off
random          0:off   1:off   2:on    3:on    4:on    5:on    6:off
rawdevices      0:off   1:off   2:off   3:on    4:on    5:on    6:off
xinetd          0:off   1:off   2:off   3:on    4:on    5:on    6:off
apmd            0:off   1:off   2:on    3:on    4:on    5:on    6:off
atd             0:off   1:off   2:off   3:on    4:on    5:on    6:off
gpm             0:off   1:off   2:on    3:on    4:on    5:on    6:off
autofs          0:off   1:off   2:off   3:on    4:on    5:on    6:off
isdn            0:off   1:off   2:on    3:on    4:on    5:on    6:off
keytable        0:off   1:on    2:on    3:on    4:on    5:on    6:off
kudzu           0:off   1:off   2:off   3:on    4:on    5:on    6:off
sshd            0:off   1:off   2:on    3:on    4:on    5:on    6:off
sendmail        0:off   1:off   2:on    3:on    4:on    5:on    6:off
iptables        0:off   1:off   2:on    3:on    4:on    5:on    6:off
nfslock         0:off   1:off   2:off   3:on    4:on    5:on    6:off
rhnsd           0:off   1:off   2:off   3:on    4:on    5:on    6:off
crond           0:off   1:off   2:on    3:on    4:on    5:on    6:off
anacron         0:off   1:off   2:on    3:on    4:on    5:on    6:off
xfs             0:off   1:off   2:on    3:on    4:on    5:on    6:off
lpd             0:off   1:off   2:on    3:on    4:on    5:on    6:off
firstboot       0:off   1:off   2:off   3:off   4:off   5:on    6:off
httpd           0:off   1:off   2:on    3:on    4:on    5:on    6:off
mysql           0:off   1:off   2:on    3:on    4:on    5:off   6:off

[test@mgmt test]$ ps -ax
USER       PID %CPU %MEM   VSZ  RSS TTY      STAT START   TIME
COMMAND
root         1  0.0  0.1  1336  476 ?        S    Jan21   0:03 init
root         2  0.0  0.0     0    0 ?        SW   Jan21   0:00 [keventd]
root         3  0.0  0.0     0    0 ?        SWN  Jan21   0:00 [ksoftirqd_CPU0]
```

```
root         4  0.0  0.0     0    0 ?        SW   Jan21   0:09 [kswapd]
root         5  0.0  0.0     0    0 ?        SW   Jan21   0:00 [bdflush]
root         6  0.0  0.0     0    0 ?        SW   Jan21   0:01 [kupdated]
root         7  0.0  0.0     0    0 ?        SW   Jan21   0:00 [mdrecoveryd]
root        13  0.0  0.0     0    0 ?        SW   Jan21   0:00 [scsi_eh_0]
root        14  0.0  0.0     0    0 ?        SW   Jan21   0:00 [scsi_eh_1]
root        18  0.0  0.0     0    0 ?        SW   Jan21   0:00 [kjournald]
root       397  0.0  0.1  1400  536 ?        S    Jan21   0:00 syslogd -m 0
root       401  0.0  0.1  1336  428 ?        S    Jan21   0:00 klogd -x
root       532  0.0  0.3  3276 1464 ?        S    Jan21   0:04 /usr/sbin/sshd
root       546  0.0  0.2  2092  904 ?        S    Jan21   0:00 xinetd -stayalive -reuse -
pidfile /var/run/xinetd.pid
ntp        560  0.0  0.4  1916 1908 ?        SL   Jan21   0:00 ntpd -U ntp
root       604  0.0  0.1  1372  428 ?        S    Jan21   0:00 gpm -t ps/2 -m
/dev/mouse
root       615  0.0  2.1 17792 8444 ?        S    Jan21   0:08 /usr/sbin/httpd
root       624  0.0  0.1  1516  612 ?        S    Jan21   0:00 crond
root       631  0.0  0.3  4012 1200 ?        S    Jan21   0:00 /bin/sh
/usr/bin/safe_mysqld --datadir=/var/lib/mysql --pid-file=/var
mysql      657  0.8  2.3 12952 9124 ?        S    Jan21   7:10 /usr/sbin/mysqld --
basedir=/ --datadir=/var/lib/mysql --user=mysql --
xfs        688  0.0  0.8  4520 3232 ?        S    Jan21   0:00 xfs -droppriv -daemon
daemon     706  0.0  0.1  1368  520 ?        S    Jan21   0:00 /usr/sbin/atd
root       716  0.0  0.1  1316  404 tty2     S    Jan21   0:00 /sbin/mingetty tty2
root     10612  0.0  0.1  1316  404 tty1     S    Feb20   0:00 /sbin/mingetty tty1
apache   31744  0.0  3.0 20332 11660 ?       S    04:02   0:08 /usr/sbin/httpd
501       6498  0.0  0.3  4128 1408 pts/0    S    15:36   0:00 -bash
501       6562  0.0  0.1  2696  728 pts/0    R    15:59   0:00 ps –aux
```

#### 3.1.1.3 Verify only needed daemons listening

This can be done in a variety of ways; however, for this specific test, I utilized the 'netstat' command, and ran NMAP to determine what ports are accessible from the network.

**IDS Probe Output**

```
[test@probe test]$ netstat -ant
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 172.16.1.110:2525       0.0.0.0:*               LISTEN
tcp        0      0 172.16.1.110:32809      172.16.1.101:3306       ESTABLISHED
tcp        0      0 172.16.1.110:22         172.16.1.111:870        ESTABLISHED

[test@scan test]# nmap -sS -p 1-65000 172.16.1.110
```

```
Starting nmap V. 2.53 by fyodor@insecure.org ( www.insecure.org/nmap/ )
Interesting ports on ntsql.itscommunications.com (172.16.1.110):
(The 64998 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     open        ssh

Nmap run completed -- 1 IP address (1 host up) scanned in 7 seconds
```

**IDS Management Console Output**

```
[test@mgmt test]$ netstat -ant
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:3306            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:443             0.0.0.0:*               LISTEN
tcp        0      0 172.16.1.101:22         172.16.1.111:845        ESTABLISHED
tcp        0      0 172.16.1.101:3306       172.16.1.110:32809      ESTABLISHED
tcp        0      0 172.16.1.101:443        172.16.1.45:1461        TIME_WAIT
tcp        0      0 172.16.1.101:443        172.16.1.45:1460        TIME_WAIT

[test@scan test]# nmap -sS -p 1-65000 172.16.1.101

Starting nmap V. 2.53 by fyodor@insecure.org ( www.insecure.org/nmap/ )
Interesting ports on ic001.itscommunications.com (172.16.1.101):
(The 64996 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     open        ssh
80/tcp     open        http
443/tcp    open        https
3306/tcp   open        mysql

Nmap run completed -- 1 IP address (1 host up) scanned in 6 seconds
```

### 3.1.2. Determine Vulnerabilities Present (Checklist Item 2.1.4)

The control objective for this checklist item is to verify that vulnerabilities do not exist for the ports that need to be open on the IDS probe and management console. Two specific tests were performed to verify that this control objective was met. These tests must be performed on both the IDS probe and management console. The table below shows the results of each individual test for both devices, as well as the overall result. Following this table is the output for each test.

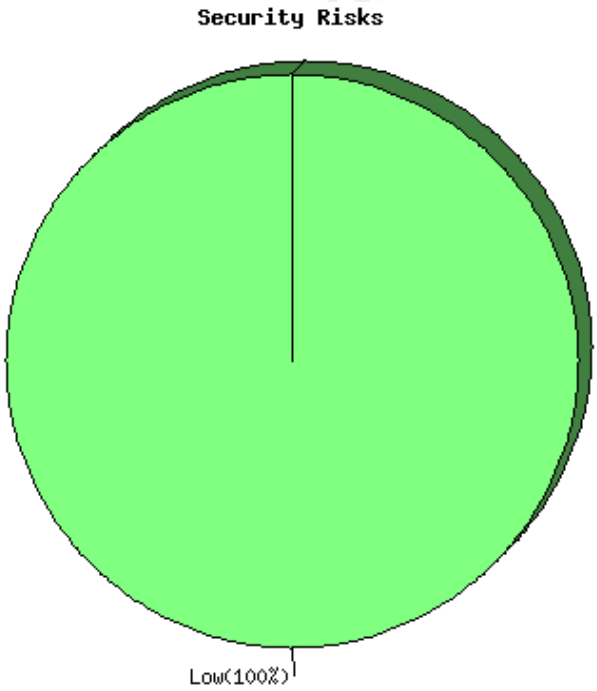
| Test | Probe Result | Mgmt Result |
|---|---|---|
| Verify vulnerabilities are not present from the network | Pass | Pass |
| Verify that a scan & corrective action procedure exists | Fail | Fail |
| **OVERALL RESULT:** | **Fail** | **Fail** |

**Table 4: Test Results for Determining Vulnerabilities Present**

#### 3.1.2.1. Verify vulnerabilities are not present

For this test, I utilized the Nessus vulnerability scanner. Nessus was selected because it has received rave reviews as well as being a likely tool for a hacker given that it is an open-source freeware product. This output shows that there are no serious vulnerabilities that exist on either device.

**IDS Probe Output**




**List of open ports:**

- ssh (22/tcp) (Security warnings found)
- general/udp (Security notes found)
- general/tcp (Security notes found)
- general/icmp (Security warnings found)

**Warning found on port ssh (22/tcp)**

The remote SSH daemon supports connections made using the version 1.33 and/or 1.5 of the SSH protocol.

These protocols are not completely cryptographically safe so they should not be used.

Solution: If you use OpenSSH, set the option 'Protocol' to '2'. If you use SSH.com's set the option 'Ssh1Compatibility' to 'no'

Risk factor: **Low**

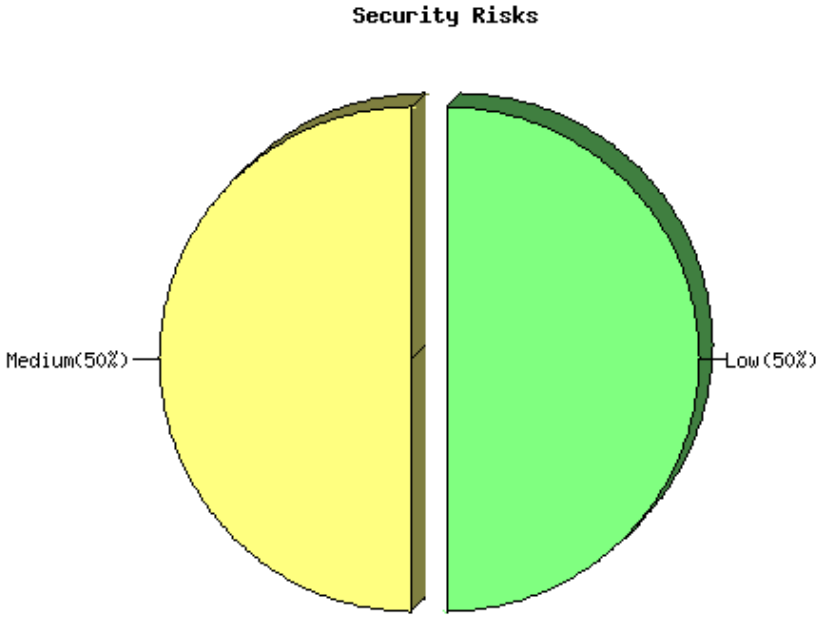
**Information found on port ssh (22/tcp)**

An ssh server is running on this port. Remote SSH version : SSH-1.99-OpenSSH_3.4p1. The remote SSH daemon supports the following versions of the SSH protocol: 1.33, 1.5, 1.99, 2.0.

**Information found on port general/tcp**

Remote OS guess: Linux Kernel 2.4.0 - 2.5.20

**IDS Management Console Output**




**List of open ports:**

- ssh (22/tcp) (Security warnings found)
- http (80/tcp) (Security warnings found)
- https (443/tcp) (Security warnings found)
- mysql (3306/tcp) (Security notes found)
- general/udp (Security notes found)
- general/tcp (Security notes found)

**Warning found on port ssh (22/tcp)**

The remote SSH daemon supports connections made using the version 1.33 and/or 1.5 of the SSH protocol.

These protocols are not completely cryptographically safe so they should not be used.

Solution: If you use OpenSSH, set the option 'Protocol' to '2'. If you use SSH.com, set the option 'Ssh1Compatibility' to 'no'.

Risk factor: **Low**

**Information found on port ssh (22/tcp)**

An ssh server is running on this port. Remote SSH version : SSH-1.99-OpenSSH_3.4p1. The remote SSH daemon supports the following versions of the SSH protocol: 1.33, 1.5, 1.99, 2.0.

**Warning found on port http (80/tcp)**

Your web server supports the TRACE and/or TRACK methods. It has been shown that servers supporting this method are subject to cross-site-scripting attacks, dubbed XST for
'Cross-Site-Tracing', when used in conjunction with various weaknesses in browsers.

An attacker may use this flaw to trick your legitimate web users to give him their credentials.

Solution: Disable these methods.

Risk factor: **Medium**

**Information found on port http (80/tcp)**

The remote web server type is: Apache/2.0.40 (Red Hat Linux)

Solution: You can set the directive 'ServerTokens Prod' to limit the information emanating from the server in its response headers.

**Warning found on port https (443/tcp)**

The SSLv2 server offers 5 strong ciphers, but also 0 medium strength and 2 weak "export class" ciphers. The weak/medium ciphers may be chosen by an export-grade or badly configured client software. They only offer a limited protection against a brute force attack.

Solution: disable those ciphers and upgrade your client software if necessary.

**Information found on port https (443/tcp)**

A TLSv1 server answered on this port

**Information found on port https (443/tcp)**

Here is the list of available SSLv2 ciphers:
RC4-MD5
EXP-RC4-MD5
RC2-CBC-MD5
EXP-RC2-CBC-MD5
DES-CBC-MD5
DES-CBC3-MD5
RC4-64-MD5

**Information found on port https (443/tcp)**

This TLSv1 server also accepts SSLv2 connections.
This TLSv1 server also accepts SSLv3 connections.

**Information found on port mysql (3306/tcp)**

Remote MySQL version: 3.23.54

**Information found on port general/tcp**

Remote OS guess: Linux Kernel 2.4.0 – 2.5.20

### 3.1.2.2. Verify that a scan and corrective action procedure exists

This test is more subjective and requires obtaining a policy from the IS administrator. This policy should state that a vulnerability scan is run on a periodic basis. The time between scans will vary depending on the specific

organization and the nature of critical data. Following are the steps that I performed to determine whether or not this policy was being enforced:

a. I met with the IS administrator and requested a written policy that stated the procedure for running periodic vulnerability scans and the steps that are performed when a vulnerability is discovered. The IS administrator was able to produce this policy. This policy stated that vulnerability scans were to be run every three months, and at anytime a major network change is done. This policy also stated the procedure that was to be followed if a serious vulnerability is discovered.
b. I requested that the IS administrator provide a record of past vulnerability scans and the vulnerabilities that were discovered. However, the IS administrator was not able to produce documentation of the last vulnerability scan.

### 3.1.3. System Access and Authorization (Checklist Item 2.1.7)

The control objective for this checklist item is to verify that system access and authorization is configured securely. Six specific tests were performed to verify that this control objective was met. These tests must be performed on both the IDS probe and management console. The table below shows the results of each individual test for both devices, as well as the overall result. Following this table is the output for each test.

| Test | Probe Result | Mgmt Result |
|---|---|---|
| Verify that the 'hosts.equiv' file does not exist | Pass | Pass |
| Verify that a login warning banner exists | Pass | Pass |
| Verify that only the needed user ID's are present | Pass | Pass |
| Verify that secure password settings are configured | Fail | Fail |
| Verify that no UID 0 accounts exist other than root | Pass | Pass |
| Verify that there are no users with empty passwords | Pass | Pass |
| **OVERALL RESULT:** | **Fail** | **Fail** |

**Table 5: Test Results for System Access and Authorization**

#### 3.1.3.1. Verify that the 'hosts.equiv' file does not exist

This file should be deleted if it exists. The output below shows that this file does not exist on either device.

**IDS Probe and Management Console Output**

[test@probe test]$ ls /etc/hosts.equiv or [test@mgmt test]$ ls /etc/hosts.equiv
ls: /etc/hosts.equiv: No such file or directory

#### 3.1.3.2. Verify that an appropriate login warning banner exists

A login warning banner helps an organization in a situation where a lawsuit is instituted against an attacker. I have heard of situations where an attacker has not been prosecuted because an organization did not have a login warning banner. As a result, there was no warning to tell the attacker that they should not enter. Therefore, it is always in an organization's best interest to have a login warning banner and legal notice configured to be displayed when the device is accessed remotely. This can be done by altering the '/etc/motd' file. This output shows that a warning banner is configured on both devices.

**IDS Probe and Management Console Output**

[test@probe test]# cat /etc/motd or [test@mgmt test]#cat /etc/motd
Unauthorized Access is prohibited. All unauthorized users will be prosecuted.

### 3.1.3.3. Verify that only the needed user ID's are present

It is important to restrict remote access to only specified employees. Therefore, it should be verified that user names only exist for these select employees. It is also important to check that there is a process in place for removing users from this device if they leave the company.

**IDS Probe and Management Console Output**

To determine if this test was in compliance, 'less /etc/passwd' was entered. The results are not shown here due to security concerns. However, for each device, I methodically went through this file with the IS administrator. Each defined user ID was confirmed to be necessary.

### 3.1.3.4. Verify that Password settings are configured securely

There should be a minimum password length and a maximum password age for any passwords that are set on either ID device. The organization's security policy states that password should be eight characters at the minimum; however, the minimum length is set to five characters.

**IDS Probe Output**

```
[root@ic100 root]# less /etc/login.defs
# *REQUIRED*
#   Directory where mailboxes reside, _or_ name of file, relative to the
#   home directory.  If you _do_ define both, MAIL_DIR takes precedence.
#   QMAIL_DIR is for Qmail
#
#QMAIL_DIR      Maildir
MAIL_DIR        /var/spool/mail
#MAIL_FILE      .mail
```

```
# Password aging controls:
#
#      PASS_MAX_DAYS   Maximum number of days a password may be used.
#      PASS_MIN_DAYS   Minimum number of days allowed between password
#                      changes.
#      PASS_MIN_LEN    Minimum acceptable password length.
#      PASS_WARN_AGE   Number of days warning given before a password
#                      expires.
PASS_MAX_DAYS   30
PASS_MIN_DAYS   0
PASS_MIN_LEN    5
PASS_WARN_AGE   7

# Min/max values for automatic uid selection in useradd
#
UID_MIN                 500
UID_MAX                60000

# Min/max values for automatic gid selection in groupadd
#
GID_MIN                 500
GID_MAX                60000

# If defined, this command is run when removing a user.
# It should remove any at/cron/print jobs etc. owned by
# the user to be removed (passed as the first argument).
#
#USERDEL_CMD    /usr/sbin/userdel_local

# If useradd should create home directories for users by default
# On RH systems, we do. This option is ORed with the -m flag on
# useradd command line.
#
CREATE_HOME     yes

[test@probe test]$ passwd
Changing password for user test.
Changing password for test
(current) UNIX password:
New password:
BAD PASSWORD: it is too short
```

**IDS Management Console Output**

```
[test@mgmt test]# less /etc/login.defs
```

```
# *REQUIRED*
#   Directory where mailboxes reside, _or_ name of file, relative to the
#   home directory.  If you _do_ define both, MAIL_DIR takes precedence.
#   QMAIL_DIR is for Qmail
#QMAIL_DIR      Maildir
MAIL_DIR        /var/spool/mail
#MAIL_FILE      .mail

# Password aging controls:
#       PASS_MAX_DAYS   Maximum number of days a password may be used.
#       PASS_MIN_DAYS   Min # of days allowed between password changes.
#       PASS_MIN_LEN    Minimum acceptable password length.
#       PASS_WARN_AGE   # of days warning given before a password expires.
#
PASS_MAX_DAYS   30
PASS_MIN_DAYS   0
PASS_MIN_LEN    5
PASS_WARN_AGE   7

#
# Min/max values for automatic uid selection in useradd
UID_MIN                 500
UID_MAX                60000

# Min/max values for automatic gid selection in groupadd
GID_MIN                 500
GID_MAX                60000

# If defined, this command is run when removing a user.
# It should remove any at/cron/print jobs etc. owned by
# the user to be removed (passed as the first argument).
#USERDEL_CMD    /usr/sbin/userdel_local

# If useradd should create home directories for users by default
# On RH systems, we do. This option is ORed with the -m flag on
# useradd command line.
#
CREATE_HOME     yes

[test@mgmt test]$ passwd
Changing password for user test.
Changing password for test
(current) UNIX password:
New password:
BAD PASSWORD: it is too short
```

#### 3.1.3.5. Verify that no UID 0 accounts exist other than root

Do this by entering **'awk –F: '($3==0) {print $1}' /etc/passwd'.** The only account that should appear with a UID of 0 is the root account.

**IDS Probe and Management Console Output**

```
[test@probe test]# awk -F: '($3==0){print $1}' /etc/passwd
root
```

#### 3.1.3.6. Verify that there are no user accounts with empty password fields

Do this by entering **'awk –F: '($2==””){print $1}' /etc/shadow'.** This command should not return anything.

**IDS Probe and Management Console Output**

```
[test@mgmt test]# awk -F: '($2==""){print $1}' /etc/shadow
[test@mgmt test]#
```

### 3.1.4. SSH Configuration (Checklist Item 2.1.11)

The control objective for this checklist item is to verify that remote access to the IDS probe and management console is set up securely. Five specific tests were performed to verify that this control objective was met. These tests must be performed on both devices. The table below shows the results of each individual test for both devices, as well as the overall result. Following this table is the evidence and output for each test.

| Test | Probe Result | Mgmt Result |
|---|---|---|
| Verify that SSH is running | Pass | Pass |
| Verify that the Most Recent Version is Running | Fail | Fail |
| Verify that the SSH daemon is started at Boot Time | Pass | Pass |
| Verify that /etc/hosts.allow is set up for SSH access | Fail | Fail |
| Verify that the SSH configuration files are configured securely | Pass | Pass |
| **OVERALL RESULT:** | **Fail** | **Fail** |

**Table 6: Test Results for SSH Configuration**

#### 3.1.4.1. Verify that SSH is Running

Verify that SSH is running via the 'netstat –at | grep ssh' and 'ps –ax | grep ssh' commands. In addition, verify that you can actually connect to the device. This output shows that SSH is running and it is possible to connect to both devices via SSH.

**IDS Probe Output**

```
[test@probe test]$ netstat -at | grep ssh
tcp        0      0 *:ssh                   *:*                     LISTEN

[test@probe test]$ ps -ax | grep ssh
  576 ?        S      0:11 /usr/sbin/sshd
3644 pts/0    S      0:00 grep ssh

[test@client test]$ ssh -l test 192.168.1.10
test@192.168.1.10's password:
[test@probe test]$
```

**IDS Management Console Output**

```
[test@mgmt test]$ netstat -at | grep ssh
tcp        0      0 *:ssh                   *:*                     LISTEN

[test@mgmt test]$ ps -ax | grep ssh
  532 ?        S      0:08 /usr/sbin/sshd
19049 pts/0    S      0:00 grep ssh

[test@client test]$ ssh -l test 192.168.1.9
test@192.168.1.9's password:
[test@mgmt test]$
```

### 3.1.4.2. Verify that the Most Recent Version is Running

This output shows that both devices are currently running 3.4p1 while the current version is 3.5p1. Therefore, SSH should be upgraded on both devices.

**IDS Probe and Management Console Output**

```
[test@probe test]$ ssh -V
OpenSSH_3.4p1, SSH protocols 1.5/2.0, OpenSSL 0x0090602f

[test@mgmt test]$ ssh -V
OpenSSH_3.4p1, SSH protocols 1.5/2.0, OpenSSL 0x0090602f
```

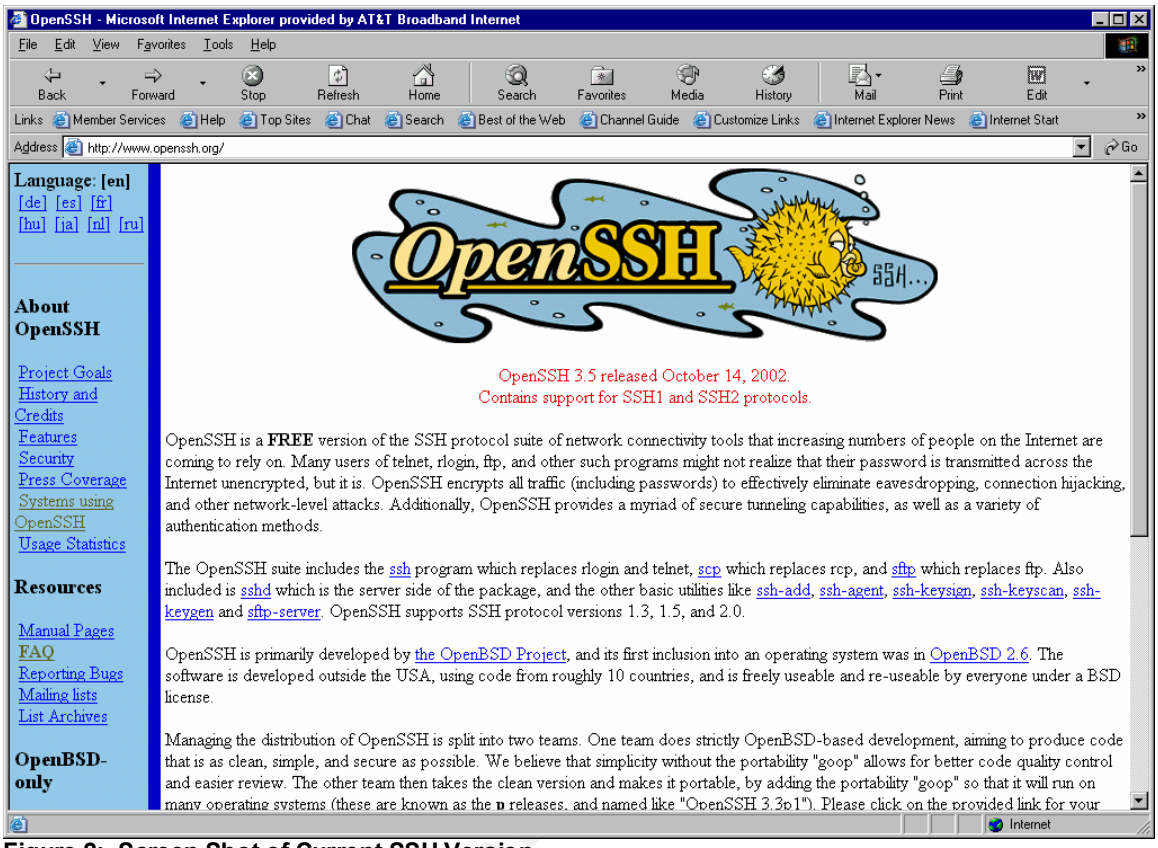



**Figure 2: Screen Shot of Current SSH Version**

### 3.1.4.3. Verify that the SSH daemon is started at Boot Time

This can be accomplished in various ways. For this test, I accomplished this by confirming that S##sshd exists in '/etc/rc3.d' where ## is the boot up sequence number. In addition, I rebooted each device and verified that I could still remotely connect to each device via ssh.

### IDS Probe and Management Console Output

The output for both devices was 'S55sshd'. Therefore, for run level 3, sshd will automatically start.

### 3.1.4.4. Verify that the 'hosts.allow' file is set up for SSH access

The '/etc/hosts.allow' file should only list IP addresses of the hosts that should have permission to remotely access the device. The output below shows that no IP addresses are listed in this file. Therefore, it is not possible to test if devices are not allowed.

### IDS Probe and Management Console Output

[test@probe test]# cat /etc/hosts.allow or [test@mgmt test]# cat /etc/hosts.allow
#
# hosts.allow  This file describes the names of the hosts which are
#  allowed to use the local INET services, as decided
#  by the '/usr/sbin/tcpd' server.
#

#### 3.1.4.5. Verify that the SSH configuration files are configured appropriately

The files 'etc/ssh/ssh_config' and '/etc/ssh/sshd_config' should be configured according to the organization's security policy. One item to take special note of is the 'Host' parameter. This parameter should list specific IP address and not just have a '*', which will allow entry of all IP addresses.

These configuration files are not listed in this report for security purposes; however, they were thoroughly analyzed. Both configuration files adhere strictly to the organization's security policy and are configured with security in mind.

### 3.1.5. Stealth Mode Verification (Checklist Item 2.2.1)

The control objective for this checklist item is to verify that the probe interface of the IDS sensor is configured in "stealth" mode. This means that there is no IP address assigned to the interface that is collecting data. Given that this interface predominantly is connected to a public network, it is critical that this is the case so a potential attacker is not able to easily detect that an IDS is in place. Four specific tests were performed to verify that this control objective was met. The table below shows the results of each individual test as well as the overall result. Following this table is the output for each test.

| Test | Result |
|---|---|
| Physically check that there are two separate interfaces | Pass |
| Make sure the probe interface does not have an IP address | Pass |
| Determine which interface is used by Snort to gather data | Pass |
| Verify IP forwarding is disabled | Pass |
| **OVERALL RESULT:** | **Pass** |

**Table 7: Test Results for Stealth Mode Verification**

#### 3.1.5.1. Physically check that there are two separate interfaces

I physically went to the IDS probe and looked at the actual number of network interface cards (NICs) installed. There were two NICs (both Ethernet) and both NICs had cables connected. One NIC led to a hub that was positioned between the perimeter router and the firewall, and the other NIC led to a switch that connected to the internal private LAN.

### 3.1.5.2. Make sure the data gathering interface does not have an IP address

For this, I entered '**ifconfig**' and obtained the results below. Notice that eth0 has an IP address assigned to it (192.168.1.10). However, eth1 does not have an IP address assigned to it. Therefore, eth1 should be the interface that is being utilized for gathering data.

```
[test@probe test]$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:50:04:61:08:17
inet addr:192.168.1.10  Bcast:192.168.1.255  Mask:255.255.255.0
UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
RX packets:8894665 errors:0 dropped:0 overruns:0 frame:0
TX packets:261284 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:100
RX bytes:1808045778 (1724.2 Mb)  TX bytes:47505598 (45.3 Mb)
  Interrupt:10 Base address:0xe400

eth1      Link encap:Ethernet  HWaddr 00:A0:CC:57:F2:FB
UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
RX packets:56641893 errors:1 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:100
RX bytes:1843630555 (1758.2 Mb)  TX bytes:0 (0.0 b)
Interrupt:11 Base address:0xd000

lo        Link encap:Local Loopback
inet addr:127.0.0.1  Mask:255.0.0.0
UP LOOPBACK RUNNING  MTU:16436  Metric:1
RX packets:10 errors:0 dropped:0 overruns:0 frame:0
TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:1028 (1.0 Kb)  TX bytes:1028 (1.0 Kb)
```

### 3.1.5.3. Determine Which Interface is Used by Snort to Gather Data

The network interface 'eth1' should be listed in this file as the interface to use for capturing data for Snort. Following is the output of the 'snortd' file, which verifies that this is the case.

```
[test@probe test]$ less /etc/rc.d/init.d/snortd
#!/bin/sh
# snortd        Start/Stop the snort IDS daemon.
# chkconfig: 2345 40 60
# description:  snort is a lightweight network intrusion detection tool that currently
detects #                   more than 1100 host and network vulnerabilities,
portscans, backdoors
```

```
#                   and more.
# June 10, 2000 -- Dave Wreski <dave@linuxsecurity.com>
#   - initial version
# July 08, 2000 Dave Wreski <dave@guardiandigital.com>
#   - added snort user/group
#   - support for 1.6.2

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

# Specify your network interface here
INTERFACE=eth1

# See how we were called.
case "$1" in
  start)
echo -n "Starting snort: "
        ifconfig eth1 up
        ifconfig eth1 promisc
        daemon /usr/local/bin/snort -U -o -i $INTERFACE -d -D \
        -c /etc/snort/snort.conf
        touch /var/lock/subsys/snort
        sleep 3
        rm /var/log/snort/alert
        echo
        ;;
        stop)
        echo -n "Stopping snort: "
        killproc snort
        rm -f /var/lock/subsys/snort
        ifconfig eth1 -promisc
        echo
        ;;
        restart)
        $0 stop
        $0 start
        ;;
        status)
        status snort
            ;;
        *)
        echo "Usage: $0 {start|stop|restart|status}"
        exit 1
esac

exit 0
```

#### 3.1.5.4. Verify IP Forwarding is disabled

IP Forwarding should be disabled on the IDS Probe. This can be verified by examining the 'ip_forward' file and verifying that this file's contents are a 0.

```
[test@probe test]$ cat /proc/sys/net/ipv4/ip_forward
0
```

### 3.1.6. Base IDS Setup (Checklist Item 2.2.2)

The control objective for this checklist item is to verify that Snort is running, Snort automatically starts at power on or reboot, and that Snort is running the current version. Two specific tests were performed to verify that this control objective was met. The table below shows the results of each individual test as well as the overall result. Following this table is the output for each test.

| Test | Result |
|---|---|
| Verify snort is running and started automatically | Pass |
| Verify the latest and greatest release of Snort is installed | Pass |
| **OVERALL RESULT:** | **Pass** |

**Table 8: Test Results for Base IDS Setup**

#### 3.1.6.1. Verify Snort is running and started automatically

To verify that snort is running, enter 'ps –ax | grep snort'. To verify that snort is automatically started when the IDS probe is powered on or rebooted, enter 'ls /etc/rc.d/init.d' and 'chkconfig –list | grep snortd'. The output below shows that snort is running on the eth1 interface and that it is configured to start automatically for run levels 2 through 5.

```
[test@probe test]# ps –ax | grep snort
 16815 ?  S  55:28  /usr/local/bin/snort –D –I eth1 –U –o –c /etc/snort/snort.conf
32347 pts/0  R  0:00  grep snort

[test@probe test]#ls –l /etc/rc.d/init.d/snortd
-rwxr-xr-x    1       root    root    1265   Nov 1  15:00  /etc/rc.d/init.d/snortd

[test@probe test]#chkconfig –list | grep snortd
snortd 0:off    1:off    2:on    3:on    4:on    5:on    6:off
```

#### 3.1.6.2. Verify the latest and greatest release of Snort is installed

This is done by comparing the output of 'snort –V' with the latest release listed on the snort web page. The output here shows that Snort is running 1.9.1 and the latest release is 1.9.1.

```
[test@probe test]#snort –V

--*> Snort! <*--
Version 1.9.1 (Build 231)
By Martin Roesch (roesch@sourcefire.com, www.snort.org)
```

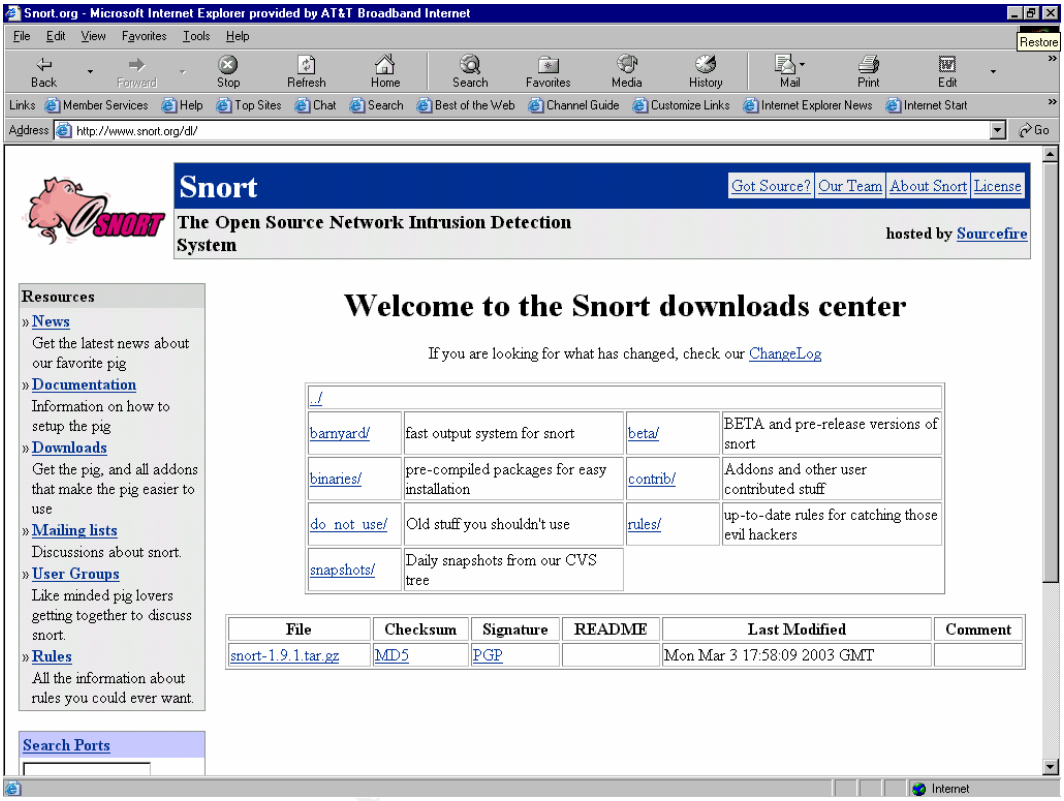



**Figure 3: Screen Shot of Current Snort Version**

### 3.1.7. Base Snort Configuration (Checklist Item 2.2.3)

The control objective for this checklist item is to verify that Snort is configured correctly and in a secure manner. Four specific tests were performed to verify that this control objective was met. The table below shows the results of each individual test as well as the overall result. Following this table is the output for each test.

| Test | Result |
|---|---|
| 'Variables' Section Verification | Pass |
| 'Output Modules' Section Verification | Pass |
| 'Preprocessors' Section Verification | Pass |
| 'Rules' Section Verification | Pass |
| **OVERALL RESULT:** | **Pass** |

**Table 9: Test Results for Base Snort Configuration**

### 3.1.7.1. 'Variables' Section Verification

The 'Variables' section should be set up to reflect the network topology that is being protected. The output below was altered for security purposes; however, the 'variables' section was configured in compliance with the security policy.

```
[root@ic100 root]# less /etc/snort/snort.eth1.conf
#--------------------------------------------------------------------------
# Snort Configuration file for < ic100 >
# Created with SnortCenter v0.9.6 < http://users.pandora.be/larc/ >
# $Id: snort.conf, Monday 17th of March 2003 02:10:35 PM
#--------------------------------------------------------------------------
var HOME_NET 1.2.3.0/27
var EXTERNAL_NET ![1.2.3.0/27]
var HTTP_PORTS 80
var SHELLCODE_PORTS !80
var ORACLE_PORTS 1521
var AIM_SERVERS [1.1.1.1, 2.2.2.2]
var RULE_PATH /etc/snort/rules
var TELNET_SERVERS $HOME_NET
var SQL_SERVERS $HOME_NET
var HTTP_SERVERS $HOME_NET
var SMTP_SERVERS $HOME_NET
var DNS_SERVERS $HOME_NET
#
```

### 3.1.7.2. 'Output Modules' Section Verification

This section allows you to either send alerts to syslog (using the syslog module alert_syslog) or log to a MySQL database (using the output module database). The output below shows that alerts will be sent to a MySQL database on 192.168.1.9.

```
output database: log, mysql, user=test password=test1 dbname=snort
host=192.168.1.9
```

### 3.1.7.3. 'Preprocessors' Section Verification

Preprocessor directives give added functionality by allowing programmers to use modular "plugins" with Snort fairly easily. The preprocessors that currently exist have gone a far way in making Snort a much more robust solution by protecting against many common attacks such as fragmentation, insertion, and evasion. Therefore, it is important that these preprocessors are configured correctly to make Snort as robust as possible.

One thing to remember is that the more preprocessors that are enabled, the greater the cost of performance. This is because all preprocessors must examine each packet in turn; i.e., there is no mechanism for a preprocessor to maturely cut short packet processing and bypass the remaining ones. This means that if a large number of CPU-intensive preprocessors are loaded then it is likely to have an adverse effect on overall performance. Therefore, be careful when choosing to enable a preprocessor.

```
preprocessor http_decode: 80 unicode iis_alt_unicode double_encode
                iis_flip_slash full_whitespace
preprocessor portscan2: scanners_max 3200, targets_max 5000, target_limit 5,
port_limit 20, timeout 60
preprocessor stream4: detect_scans, disable_evasion_alerts
preprocessor stream4_reassemble
preprocessor conversation: allowed_ip_protocols all, timeout 60,
               max_conversations 32000
preprocessor telnet_decode
preprocessor frag2
preprocessor rpc_decode: 111 32771
preprocessor bo: -nobrute
```

#### 3.1.7.4. 'Rules' Section Verification

This section should have rules that address all necessary attacks. The rule set was not listed here as it is considerable in size; however, I analyzed this rule set with the IS administrator and it was determined that the rule set aligned with the necessary requirements.

### 3.1.8. Snort Performance (Checklist Item 2.2.4)

The control objective for this checklist item is to verify that the IDS does not cease operation at a certain performance level, or does not start dropping packets. One specific test was performed to verify that this control objective was met. The table below shows the results of this test as well as the overall result. Following this table is the output for this test.

| Test | Result |
|---|---|
| Verify that packets are not being dropped by the IDS probe | Pass |
| **OVERALL RESULT:** | **Pass** |

**Table 10: Test Results for Snort Performance**

#### 3.1.8.1. Verify that packets are not being dropped by the IDS Probe

```
[test@probe test]# ps –ax | grep snort
  405 ?        S      1:45  /usr/local/bin/snort –D –i eth1 –U –o-c /etc/snort/s
```

```
[test@probe test]#kill –SIGUSR1 405

[test@probe test]#less /var/log/messages
Mar 17 14:17:40 ic100 snort:
==========================================================
======
Mar 17 14:17:40 ic100 snort: Snort analyzed 20644654 out of 20644654 packets,
Mar 17 14:17:40 ic100 snort: dropping 0(0.000%) packets
Mar 17 14:17:40 ic100 snort: Breakdown by protocol:            Action Stats:
Mar 17 14:17:40 ic100 snort:     TCP: 15895153   (76.994%)      ALERTS:
18553
Mar 17 14:17:40 ic100 snort:     UDP: 688059     (3.333%)       LOGGED: 4717
Mar 17 14:17:40 ic100 snort:    ICMP: 20350      (0.099%)       PASSED: 0
Mar 17 14:17:40 ic100 snort:     ARP: 43245      (0.209%)
Mar 17 14:17:40 ic100 snort:   EAPOL: 0          (0.000%)
Mar 17 14:17:40 ic100 snort:    IPv6: 0          (0.000%)
Mar 17 14:17:40 ic100 snort:     IPX: 0          (0.000%)
Mar 17 14:17:40 ic100 snort:   OTHER: 3997831    (19.365%)
Mar 17 14:17:40 ic100 snort: DISCARD: 0          (0.000%)
Mar 17 14:17:40 ic100 snort:
==========================================================
======
```

### 3.1.9. Vulnerability and Port Scan Recognition (Checklist Item 2.2.5)

The control objective for this checklist item is to verify that the ACID Console is recording when scan attacks are occurring. One specific test was performed to verify that this control objective was met. The table below shows the results of this individual test as well as the overall result. Following this table is the output for this test.

| Test | Result |
|---|---|
| Verify that ACID is recording when network scans are occurring | Pass |
| **OVERALL RESULT:** | **Pass** |

**Table 11: Test Results for Attack Recognition**

#### 3.1.9.1. Verify That ACID is recording when network scans are occurring

For this test, I ran Nessus and NMap from the Internet against a device located on the target's network. These scans were run to determine if Snort was detecting these scans and properly displaying them on the ACID console. The output below is a portion of the table that I copied from the ACID Console. As can be seen, the scans were detected from the attacker's device. All output was not listed here because there were over 2800 alerts recorded; however, it was verified that ACID alerted on all scans that took place.

FQDN: **1.1.1.1.client.xyz.com** ( local whois )

| # | Occurances as Src. | Occurances as Dest. | 1st Occurance | Last Occurance |
|---|---|---|---|---|
| 1 | 2859 | 21 | 2003-03-19 05:06:24 | 2003-03-19 05:36:23 |

---

**256 unique alerts detected among 2880 alerts on 1.1.1.1/32**

| Signature | Total Occurrences | Num of Sensors | First Occurrence | Last Occurrence |
|---|---|---|---|---|
| cve cve cve[snort] SNMP public access udp | 3 | 1 | 2003-03-19 05:06:40 | 2003-03-19 05:06:41 |
| cve cve[snort] SNMP private access udp | 6 | 1 | 2003-03-19 05:06:40 | 2003-03-19 05:06:41 |
| cve cve[snort] SNMP request udp | 742 | 1 | 2003-03-19 05:06:41 | 2003-03-19 05:26:21 |
| arachnids[snort] ICMP PING NMAP | 14 | 1 | 2003-03-19 05:06:24 | 2003-03-19 05:33:29 |
| arachnids[snort] SMTP expn root | 2 | 1 | 2003-03-19 05:10:08 | 2003-03-19 05:10:13 |
| [snort] SMTP vrfy root | 1 | 1 | 2003-03-19 05:10:08 | 2003-03-19 05:10:08 |
| nessus[snort] WEB-MISC robots.txt access | 4 | 1 | 2003-03-19 05:10:08 | 2003-03-19 05:26:09 |
| [snort] WEB-MISC /CVS/Entries access | 4 | 1 | 2003-03-19 05:10:08 | 2003-03-19 05:16:34 |
| [snort] WEB-MISC Admin_files access | 2 | 1 | 2003-03-19 05:10:13 | 2003-03-19 05:10:13 |
| [snort] WEB-IIS _mem_bin access | 2 | 1 | 2003-03-19 05:10:21 | 2003-03-19 05:10:21 |
| [snort] ATTACK | 21 | 1 | 2003-03-19 | 2003-03- |

| | | | | |
|---|---|---|---|---|
| RESPONSES 403 Forbidden | | | 05:10:22 | 19 05:22:47 |
| [snort] WEB-FRONTPAGE /_vti_bin/ access | 95 | 1 | 2003-03-19 05:10:23 | 2003-03-19 05:27:27 |
| nessus bugtraq bugtraq[snort] WEB-MISC mod-plsql administration access | 2 | 1 | 2003-03-19 05:10:28 | 2003-03-19 05:10:29 |
| nessus[snort] WEB-MISC Oracle Dynamic Monitoring Services (dms) access | 2 | 1 | 2003-03-19 05:10:59 | 2003-03-19 05:10:59 |
| [snort] WEB-IIS iisadmin access | 3 | 1 | 2003-03-19 05:11:17 | 2003-03-19 05:17:43 |

**Table 12: Detected Alerts from Vulnerability and Port Scan**

### 3.1.10. Apache Security (Checklist Item 2.3.1)

The control objective for this checklist item is to verify that the Apache Web Server is configured securely. Six specific tests were performed to verify that this control objective was met. The table below shows the results of each individual test as well as the overall result. Following this table is the output for each test.

| Test | Result |
|---|---|
| Verify the version of Apache being used is current | Fail |
| Verify the web server is only listening on the local interface | Pass |
| Verify that access is disabled to the entire file system by default | Pass |
| Verify that access to the server is limited to specific addresses | Pass |
| Verify File Permissions are set up appropriately | Pass |
| **OVERALL RESULT:** | **Fail** |

**Table 13: Test Results for Apache Security**

#### 3.1.10.1. Verify the version of Apache being used is current

To determine if the IDS management console was running the current version, I ran 'httpd –V' and compared the results to the current release listed on httpd://httpd.apache.org. This output shows that the IDS management console is running Apache 2.0.40, while the best available version is Apache 2.0.44. Therefore, the Apache web server should be upgraded as soon as possible.

```
[test@mgmt test]# httpd -V
Server version: Apache/2.0.40
Server built:   Oct  9 2002 08:01:13
```

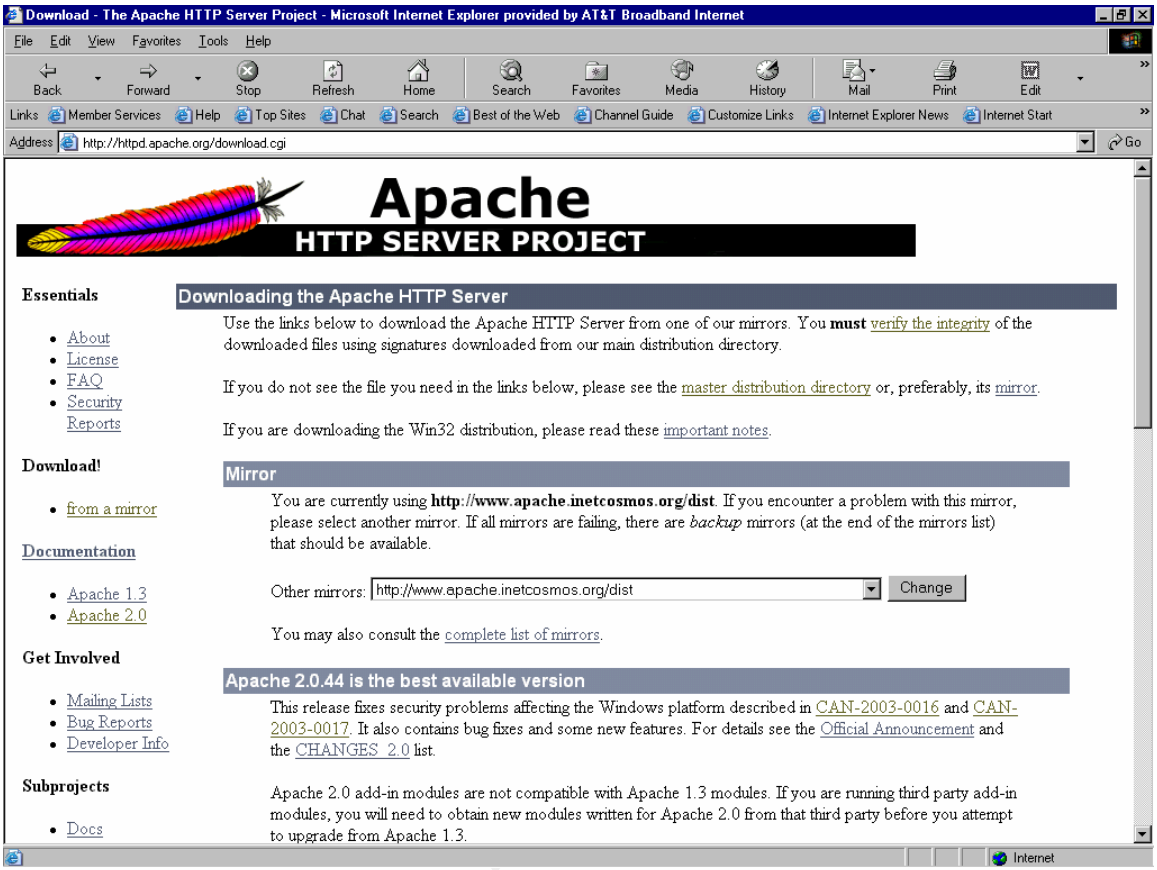



**Figure 4: Screen Shot of Current Apache Version**

### 3.1.10.2. Verify the Web Server is only listening on the local interface

The httpd.conf file should be configured so that the web server is only listening on the local interface. This eliminates the possibility of listening on multiple interfaces or on an undesired interface. The output here shows that this is correctly set up.

```
[test@mgmt test]# less /etc/httpd/conf/httpd.conf
# Listen: Allows you to bind Apache to specific IP addresses and/or
# ports, in addition to the default. See also the <VirtualHost>
# directive.
#
# Change this to Listen on specific IP addresses as shown below to
# prevent Apache from glomming onto all bound IP addresses (0.0.0.0)
#
#Listen 12.34.56.78:80
Listen 127.0.0.1:80
```

### 3.1.10.3. Verify that access is disabled to the entire file system by default

The httpd.conf file should be configured so that access to the entire file system is disabled by default. This is done by configuring a <Directory /> section that specifies 'deny from all'. Then, it is necessary to add appropriate directory blocks to allow access to those file systems that you desire.

There are additional directories listed in the directory blocks; however, for brevity, I have not included each directory block. After analyzing this portion of the configuration with the IS administrator, it was determined that this portion is configured correctly. In addition, I tried to access directories that were not listed in the directory section and I got an error stating "cannot display web page".

```
[test@mgmt test]# less /etc/httpd/conf/httpd.conf
# Each directory to which Apache has access can be configured with respect
# to which services and features are allowed and/or disabled in that
# directory (and its subdirectories).
#
# First, we configure the "default" to be a very restrictive set of
# features.
#
<Directory />
    Options None
    AllowOverride None
    Order deny, allow
    Deny from all
</Directory>

<Directory /var/www/html>
    Options None
    AllowOverride None
    Order deny, allow
    Allow from all
</Directory>
```

### 3.1.10.4. Verify that access to the server is limited to specific addresses

The httpd.conf file should be configured so that access to the web server is limited to specific addresses or networks. This is done by configuring a <Directory /var/www/html> section that specifies specific addresses or networks.

From this output, it can be seen that three devices (IP addresses) are allowed access to the web server. These addresses are statically defined to three members of the security team. However, I also verified that I was not able to access the web server from other IP addresses.

```
[test@mgmt test]# less /etc/httpd/conf/httpd.conf
# Controls who can get stuff from this server.
```

```
#
   <Directory /var/www/html>
   Order deny, allow
   Allow from 127.0.0.1
   Allow from 192.168.1.51
   Allow from 192.168.1.27
   Allow from 192.16.1.32
   Deny from all
</Directory>
```

#### 3.1.10.5. Verify File Permissions are set up appropriately

The files in /etc/httpd should only be writable by root. Try to write to these files while logged in as a user other than root.

```
[test@mgmt test]# ls -l /etc/httpd/
total 8
drwxr--r--   7 root    root       4096 Jan 21 11:23 conf
drwxr--r--   2 root    root       4096 Jan 20 15:30 conf.d
lrwxr--r--   1 root    root         19 Jan 20 15:30 logs -> ../../var/log/httpd
lrwxr--r--   1 root    root         27 Jan 20 15:30 modules ->
../../usr/lib/httpd/modules
lrwxr--r--   1 root    root         13 Jan 20 15:30 run -> ../../var/run
```

### 3.2. Measure Residual Risk

As stated by SANS, residual risk = exposure – controls. Given this, I have never seen a situation where exposure – controls = 0. This implies that there is always some residual risk with a network component. In fact, the risk to a system can never be totally eliminated – that would entail ceasing operations. Risk mitigation means finding out what level of risk the enterprise can safely tolerate and still continue to function effectively.[8]

Given this, let's take a look at the two variables, exposure and controls, in a little more detail to determine the residual risk to the Snort IDS. Exposure for this IDS is two-fold; i.e., what can be seen from the internal network and what can seen from the external Internet. Given that the IDS is configured correctly with a "stealth" interface, external visibility is fairly low. On the other hand, it is very difficult to reduce visibility on the internal network. Therefore, the system could be at risk from an internal attack or from an external intruder that is able to gain access into a different internal device and perform attacks from that device. Therefore, overall exposure is fairly high especially due to the fact that an IDS is a prime target if an intruder realizes that one exists. That

[8] Krutz, Ronald L. and Vines, Russell. The CISSP Prep Guide: Mastering the Ten Domains of Computer Security. Wiley Computer Publishing, 2001. 15.

is why controls are needed to mitigate the residual risk and bring that value as close to zero as possible.

The controls were determined by the checklist items above and used to determine how much the residual risk was mitigated. Therefore, for the controls here, I looked directly at the result of the audit. This audit showed that there were about ten checklist items that were not in compliance. However, most, if not all, of these checklist items can be corrected fairly easily and with minimal costs involved. In fact, most corrections do not require any additional purchase of equipment; they just require manual labor to make configuration changes and upgrades. Therefore, given that the recommendations in the audit report are performed, the control objectives should be achieved to an acceptable level. This would result in a fairly low residual risk.

However, there are a couple items from the control objectives that may result in a higher residual risk if they are not handled properly. These items are false negatives, real-time alerting, and performance, and are discussed in more detail in the next section (3.3). These items address threats where true preventative controls could not be implemented. Given that these control objectives are achieved, the overall residual risk will be fairly low.

### 3.3. Is the System Auditable?

This audit encompassed several months of research and analysis, and was extremely time consuming given the complexity of the system and the number of components involved. However, it is a necessity to analyze each component given that this system is irreducibly complex as explained in section one. Therefore, this system is difficult to completely audit, but it can be audited with careful planning.

Taking each component individually, most components already have excellent checklists and benchmarks in place. Therefore, these components could be audited with pretty good assurance that everything was taken into account. These components included the Apache Web Server, the MySQL Database, SSH, the ACID analyst console, and the SnortCenter front-end interface; i.e., pretty much every component except for the intrusion detection software, Snort.

The biggest reason for the uncertainty with the Snort software has to do with the unknown; i.e., false negatives. A false negative, by definition, means that an attack has occurred that has gone undetected by the IDS unless there are other security measures in place such as a host-based intrusion detection system. Unfortunately, false negatives are a great possibility with the number of attacks that exist and the number of attacks that are constantly being created. I included a section in the above for testing against all known attacks to verify that Snort was properly alerting on all attacks. However, I did not attempt even close to the number of attacks that exist for evading intrusion detection systems. To do this would be a daunting task that would take much more time than was allotted for this audit. Therefore, to a certain extent, trust is needed to believe what the vendor says is true concerning what attacks their system can protect

against. Beyond this trust, testing against all attacks is truly not feasible for one individual audit.

Another item that is truly difficult to audit is the Snort software performance. I did test the number of packets that were dropped during a time frame of a couple months. This showed that no packets were dropped during that time frame. However, what is to say that an intruder will not launch an attack tomorrow that will cause the IDS to start dropping packets or shut down. That is, because of the multitude of attacks that can be thrown at the IDS, it is very difficult to know for sure that the IDS will withstand everything.

Another item that was not possible to audit was real-time alerting. This item is auditable; however, it was not set up appropriately at the time of auditing. Therefore, there is no way to accurately tell if an analyst will correctly identify all attacks when notified by e-mail or pager. If this identification does not take place, legitimate attack may be missed.

Finally, the last item to note is that an audit is a snapshot of a system at one moment in time. This means that it is necessary to constantly stay on top of any new vulnerabilities, attacks, and findings that are discovered for this particular system. If these items are not taken into account, the system may be in a vulnerable state. That is, this audit needs to dynamically change as the industry and world changes.

Given this, overall, the Snort IDS can be audited. However, the concern areas noted above should be considered as the security of the IDS is considered. There are either additional security components that can be implemented to provide greater security or additional steps to provide greater testing. Either way, the deficiencies listed are by no means a reason to not include an IDS as part of the overall security architecture.

# 4. Audit Report

## 4.1. Executive Summary

For this engagement, I performed a detailed audit on the Snort intrusion detection system located between the firewall and the perimeter router. This audit was designed to identify possible security weaknesses and/or vulnerabilities that exist on this system. To do this, it was necessary to analyze each component of the IDS to verify that all components were configured in a secure manner. Given the complexity of this system and the number of different components, it was very difficult to create a specific scope for this project. Therefore, I would like to thank John Berry for working with me during the planning stage to make sure that a clear scope was defined. Following are the items that were defined as the scope for this audit:

- Intrusion Detection Software (Snort)
- The Underlying Operating System (Linux Red Hat)

- The Web Server (Apache)
- The Analyst Console (ACID)
- The Storage Database (MySQL)
- The Front-End Interface (SnortCenter)

Given these scope items, the audit was performed in its entirety. Overall, the findings show an IDS that has been set-up and configured in a secure manner. There were some items that were found to not be in compliance. However, overall, good security measures have been put into place and the configuration is very secure. Therefore, I will start with the high points that were discovered during this audit. These high points are:

- Policies and Procedures
- Up-to-Date Operating System Patches
- File System Security
- Backup Recovery Procedure
- Snort Configuration
- Snort Performance
- Overall Integration of all Components
- Response Time when Changes are Required

In addition to these high points, there were about ten items that were not in compliance. These items are listed below and discussed in more detail in the audit findings section. It is noted that the non-compliance of these items should not be looked as shortcomings of the individual responsible for maintaining this system. This is because this individual has a multitude of responsibilities. The procedures and processes were in place for most of these items; however, maintaining the security on all systems is a daunting task for one individual. Therefore, it may be necessary to offload some of these responsibilities to other individuals or hire additional personnel.

- Unneeded Services are Running
- Vulnerability Scan and Procedure Needs to be Addressed
- Automated File Integrity System Needs to be Implemented
- SSH Needs to be Upgraded
- Eliminate Remote SSH Access using Root
- Default MySQL Users and Tables Exist
- Real-Time Alerts Need to be Configured
- Apache Needs to be Upgraded
- Minimum Password Length Needs to be Increased
- Regular Signature Update Procedure Needs to be Implemented

As mentioned, overall, the IDS security is strong and Steve Daniels, the IS administrator, should be commended for this. In addition, I would like to add that during this audit, I received a SANS alert email from the SANS Institute announcing a critical vulnerability with all Snort versions from 1.8 to 1.9. This vulnerability was a buffer

overflow in the Snort Remote Procedure Call (RPC) normalization routines that could possibly give local and remote users almost complete control of the machine. As soon as I received this email, I made Steve aware of this vulnerability due to the fact that the Snort box was running 1.9. Within a half hour, Steve let me know that the Snort box had been upgraded to 1.9.1. I felt that this was excellent response time and should be acknowledged.

### 4.2. Audit Findings

This audit entailed several steps to ensure that all necessary areas were thoroughly audited. Initially, extensive research was performed to determine what areas should be audited, as well as determine the best process for auditing these areas. Once this research was completed, the detailed checklist was created to outline exactly what needed to be performed to ensure a secure configuration. At that time, the actual audit could then be performed.

The actual audit was performed over a period of one month. This involved extensive testing and detailed analysis of configuration files and system settings. This was due to the complexity of the IDS and the need to perform a variety of tests to confirm the compliance or non-compliance of all items. After all of these tests were performed, all output was thoroughly analyzed to determine the outcome of each test.

After all tests were analyzed, it was determined that the overall IDS security was at a good level. I have never seen a security audit performed that does not find at least some areas that need improvements, and this audit was no exception. However, the findings from this audit definitely show an organization, and an IS administrator, that keeps security as a priority and is diligent in maintaining systems. At the same time, it is important to discuss the items that were determined to not be in compliance so they can be corrected and the IDS can be made even more secure. Therefore, following are the items that were determined to not be in compliance. Any item that is not listed below was found to be in compliance so no additional improvements are needed.

- **Unneeded Services –** The 'chkconfig' and 'ps' commands discovered that some unneeded services were running (see output from 3.1.2.2).
    - **Background/Risk –** Any service that is running provides a possible avenue for an attacker to enter the system. Once in the system, the attacker may be able to alter files or services and thereby corrupt the integrity of the system. This could result in an IDS behaving in an unexpected manner, and not providing the functionality that is desired.
- **Vulnerability Scan and Procedure –** The vulnerability scan that was run against the IDS probe and management console produced several warning (see output from 3.1.2.1). Although no serious vulnerability was discovered, it is best to mitigate any deficiency if possible. In addition, although a security scan procedure is in place that provides a process for performing regular vulnerability scans, the IS administrator was not able to produce documentation of any previous scans that had been performed.

    - **Background/Risk –** Any results that are returned from a vulnerability scan should be analyzed and corrected if possible. Even simple informational data can provide an attacker with critical information to launch an attack. For example, if an attacker is aware of a particular application or operating system that is running, he/she can launch attacks specific to that application or operating system. In addition, the fact that regular vulnerability scans are not being performed should be addressed. This is because new vulnerabilities are constantly being discovered. Therefore, if a vulnerability effects either the IDS probe or management console, an intruder may have an open door into the device unless the vulnerability is eliminated. If a vulnerability is not eliminated, an organization is at risk of an intruder potentially gaining full access to that device. At that point, the intruder may have the capability to manipulate the system in any way he/she wishes.
- **File Integrity (Tripwire) –** A file integrity software application is not being used on either the IDS probe or management console. Therefore, a process is not in place to ensure the integrity of critical system files.
    - **Background/Risk –** The two previous bullet items show methods in which an attacker may possibly be able to gain access into the IDS in one way or another. If this happens, it is very likely the attacker will attempt to alter critical system files. A file integrity software system will allow an organization to determine if files have changed unexpectedly and what changes were made to these files. Without this in place, if an intruder is able to gain access and manipulate files, there may not be a good way to determine if and how files have been changed. This possibly could result in not being able to accurately determine what needs to be done to properly correct the IDS, or not being able to provide ample evidence when attempting to prosecute a discovered attacker.
- **SSH Version –** The current version running on the IDS probe and management console is not the latest and most stable software version (see output from 3.1.4.2).
    - **Background/Risk –** New software versions are often released to either correct discovered vulnerabilities or add additional features. Therefore, it should be standard procedure to upgrade to the latest and most stable software version at the earliest opportunity. Otherwise, you may be vulnerable to well-known vulnerabilities specific to the older release and/or not fully utilizing additional features that might provide greater security.
- **Root Access –** It is possible to gain login remotely via ssh using the root user name.
    - **Background/Risk –** Gaining root access to a machine is the ultimate goal for an attacker. With this access, an intruder has the ability to perform or alter anything he/she wishes. With this in mind, it is possible to login remotely via ssh using the root login name. This allows an attacker to attempt a brute-force attack into the IDS via the root login name. Therefore, access by this method should be rendered inoperative.

- **MySQL Default Users and Tables –** The default users and tables still exist in the MySQL database.
    - **Background/Risk –** Default items are often included with software packages for testing purposes. These default items are usually not configured in a secure manner. Therefore, they should be removed when the system is put into operation. If not, an attacker may be able to gain access into the MySQL database. Manipulation of this database could result in inaccurate attack results being listed on the ACID console.
- **Real-Time Alerts -** The IDS is currently not configured to send real-time alerts to specified individuals. There is no specific output for this item; however, this was confirmed by the IS administrator.
    - **Background/Risk –** If an attack is being attempted through the system, the Snort IDS is not set up to automatically defend against the attack. Therefore, it is critical that an analyst be informed that this attack is occurring. If not, the attacker will have already done the intended damage before the organization is aware. At this point, the IDS is serving as nothing more than a logging server.
- **Apache Version -** The current version running is not the latest and most stable software version (see output from 3.1.10.1).
    - **Background/Risk -** New software versions are often released to either correct discovered vulnerabilities or add additional features. Therefore, it should be standard procedure to upgrade to the latest and most stable software version at the earliest opportunity. Otherwise, you may be vulnerable to well-known vulnerabilities specific to the older release and/or not fully utilizing additional features that might provide for greater security.
- **Minimum Password Length –** The minimum password length is currently set at five characters (see output from 3.1.3.4). The security policy states that passwords should be no less than eight characters in length.
    - **Background/Risk –** Password cracking tools work by trying all combinations of characters at each length. For example, they start with a length of 1 and try all permutations, then a length of 2, and so on. Therefore, the smaller the length of the password, the smaller number of permutations that are required to discover a password and the greater the chance that a password will be discovered. At that point, an attacker will most likely be able to gain access into the device in some manner and perform undesirable commands.
- **Signature Updates –** Currently, there is no process in place to regularly update Snort signatures. There is no specific output for this item; however, this was confirmed by the IS administrator.
    - **Background/Risk –** Snort detects attacks based on the intrusion signatures that are installed. If a new attack is created, Snort will most likely not be able to detect this attack until a new signature is created and downloaded. Given that new attacks are created regularly, it is critical that a process be in place to regularly update Snort signatures. Otherwise, an attacker will be able to utilize a newly created attack to pass through the IDS undetected.

### 4.3. Audit Recommendations

Given the items listed above that were not in compliance, I recommend the following additional control objectives be put into place:

- **Eliminate Unneeded Services –** Examine the list of services that are currently running. Look at each service one by one and decide if that service is absolutely needed. If it is not, turn off that service.
- **Vulnerability Scan Improvements –** All vulnerabilities and information found during each vulnerability scan should be addressed. That is, analyze each item and determine, according to your organization's security policy, if that item can be corrected. If so, take the necessary actions to make this correction. In addition, vulnerability scans need to be regularly performed. A policy was in place stating that regular scans need to be performed; however, it did not specify who was responsible for performing these scans. Therefore, this task should be assigned to a specific person so that these scans get performed regularly.
- **Implement File Integrity Monitoring –** Install Tripwire software so an automated file integrity software application is in place. This will help to maintain the integrity of all critical system files.
- **Upgrade SSH –** Initially, SSH should be upgraded to the latest, stable software version. In addition, the process for maintaining the latest release should be revisited. More specifically, it should be better defined who is responsible for this process. Currently, the IS administrator has a multitude of responsibilities that keep him considerably busy. Therefore, it may be best to assign this task to one of the technicians to ensure that there is someone that has enough time to stay on top of this.
- **Root Remote Login –** The ability to login via ssh using the root login name should be eliminated.
- **Default MySQL Users and Tables –** Remove all users and tables that exist by default in the MySQL database.
- **Real-Time Alerting –** Implement a fully functional real-time alerting system. This involves several items: a real-time alerting method, specific individuals that will be notified, and a process for defending against attacks. The key for real-time alerting is the specific individual that will be notified. That is, if an individual does not respond when notified so that he/she can analyze the alert and defend against it if necessary, then there is no need to notify people in real time.
- **Upgrade Apache -** Initially, Apache should be upgraded to the latest, stable software version. In addition, the process for maintaining the latest release should be revisited. More specifically, it should be better defined who is responsible for this process. Currently, the IS administrator has a multitude of responsibilities that keep him considerably busy. Therefore, it may be best to assign this task to one of the technicians to ensure that there is someone that has enough time to stay on top of this.
- **Increase Minimum Password Length –** The minimum password length for user accounts should be increased from five characters to eight characters.

- **Regularly Update Signatures –** A process should be defined so that signatures are updated at regular intervals. Currently, there is an easy process to update signatures utilizing the SnortCenter web interface; however, when discussing this item with the IS administrator, he was not aware of the last time that signatures had been updated. Therefore, the problem lies more with the fact that this task needs to be assigned to a specific individual that has this as a priority.

### 4.4. Costs

Given that all of the software applications that are in use or suggested to be used are open-source freeware, the additional costs that would be incurred to correct the items that are non-compliant would not be substantial. In fact, most of these items only involve making configuration changes, upgrading software, adding additional features, or something else that is somewhat simple to correct. This would involve approximately four to eight hours for each non-compliant item. It is true that time is money and an engineer's time is not always readily available. However, given the importance of security, it would be in the organization's best interest to make these items a priority.

There are a couple items that are a little more involved in making corrections. These two items are implementing file integrity controls and real-time alert controls. For implementing file integrity controls, no additional software cost would be incurred given that Tripwire is an open-source freeware application. The primary cost incurred is setting up the software application to behave in the desired way. Therefore, once again, this is prioritizing the engineer's time to allow for this to be set up. This would most likely take more than 4 to 8 hours to fully set up as with the other tasks; however, this should not take more than a week to set up, tweak, and fully test.

The other item that needs to be addressed more extensively is fully implementing a real-time alert process. This inquires several variables: a functional alerting method, an analyst or analysts to examine the alerts when they are received, and a process in place to respond to alerts when they occur. The first variable, a functional alerting method, is pretty straightforward and should require minimal time to set up and configure. The ability for an analyst or analysts to monitor received alerts is where things become a little more in-depth. For example, what happens when an alert is received at three in the morning? Is there an analyst that will be available to analyze this alert and react is the desired way if required? If not, then real-time alerting will not provide any added benefits because by the time the alert is analyzed, the attacker will already have entered and done what he/she was attempting to do. However, if there is an analyst available, then this means that most likely the organization has taken the measures to have one or more analysts solely dedicated to intrusion analysis. This possibly could incur substantial costs if the organization is required to hire one or more analysts for this purpose.

### 4.5. Compensating Controls

Looking at the costs discussed above, there are really only two items that require a considerable amount of time and/or money: file integrity and real-time alerting. There is really no alternate method to put in place for file integrity that would incur less costs and at least help mitigate the risk in some manner. One could probably monitor critical files manually, but this would have to be done on a regular basis. Initially, this would probably be less complex and take less time. However, over time, this process would be much more costly because Tripwire is an automated process and, after the initial set-up, does not require much overhead.

For real-time alerting, there are several steps in between where you are currently at and full real-time alerting. The steps in between will help to mitigate the risks without incurring as heavy of a cost. One step would be to have a full-time analyst from 8 to 5 and after hours would be handled by only sending e-mails. To go a little further, after hours could be handled by paging an analyst to respond instead of having a full-time analyst available. Another option would be to have a third-party handle this. This may be less expensive then hiring multiple full-time intrusion analysts. All of these methods, as well as several others, could be implemented to mitigate this risk to a certain extent. I would be more than happy to discuss this in further detail to determine the best plan for your organization.

### 4.6. Conclusion

This audit entailed performing detailed testing on all components of the intrusion detection system. After completing this audit, it was determined that the IDS is configured in a secure manner. However, most, if not all audits, discover some items that are not in compliance. In this case, approximately ten items were discovered to not be in compliance. Recommendations are listed above that address these items. I would suggest following these recommendations to greater enhance the overall security of the Snort intrusion detection system.

## References


- Allen, Julia. "State of the Practice of Intrusion Detection Technologies" URL: www.sei.cmu.edu/pub/documents/99.reports/pdf/99tr028.pdf (17 Feb. 2003).
- "ACID: Installation and Configuration." 9 Oct. 2002. URL: http://www.andrew.cmu.edu/~rdanyliw/snort/acid_config.html (7 Mar. 2003).
- "Apache Security Tips." URL: http://httpd.apache.org/docs-2.0/misc/security_tips.html (13 Mar. 2003).
- Cohen, Fred. "50 Ways to Defeat Your Intrusion Detection System." 16 Oct. 2002. URL: http://hackersplayground.org/papers/50_Ways_to_Defeate_Your_IDS.txt (26 Feb. 2003).
- Fenzi, Kevin. "Linux Security HOWTO." June 2002. URL: http://www.linuxsecurity.com/docs/LDP/Security-HOWTO/ (22 Dec. 2003)
- Graham, Robert. "FAQ: Network Intrusion Detection System". 21 March 2000. URL: http://www.robertgraham.com/pubs/network-intrusion-detection.html (3 Jan, 2003).
- Haines, J.W. "1999 DARPA Intrusion Detection Evaluation: Design and Procedures" URL: www.ll.mit.edu/IST/ideval/pubs/2001/TR-1062.pdf (18 Feb, 2003).
- Herzog, Pete. "Open Source Testing Methodology Manual." URL: http://www.isecom.org/projects/osstmm.htm (13 Mar. 2003).
- "IDS Group Test." July 2002. URL: http://www.nss.co.uk/download_form.htm (12 Feb. 2003).
- Krutz, Ronald L. and Vines, Russell. The CISSP Prep Guide: Mastering the Ten Domains of Computer Security. Wiley Computer Publishing, 2001.
- "Linux Benchmark v1.0.0." 16 Feb. 2002. URL: http://www.cisecurity.org/bench_linux.html (8 Feb. 2003).
- "Linux Security Quick Reference Guide." 2000. URL: http://www.linuxsecurity.com/docs/QuickRefCard.pdf (22 Jan. 2003).
- Mann, Scott and Mitchell, Ellen L. Linux System Security: The Administrator's Guide to Open Source Security Tools. Indianapolis, Prentice Hall PTR, 2000.
- Maple, Ryan W. "MySQL Security." 24 Aug. 2000. URL: http://www.linuxsecurity.com/tips/tip-24.html (2 Mar. 2003).
- "MySQL Reference Manual." URL: http://www.mysql.com/doc/en/index.html (21 Feb. 2003).
- Naidu, Krishni. "Auditing Linux." URL: http://www.sans.org/score/checklists/AuditingLinux.doc (18 Jan. 2003).
- Northcutt, Stephen. Network Intrusion Detection: An Analyst's Handbook. Indianapolis, New Riders Publishing,1999.
- Northcutt, Stephen and McLachlan, Donald and Novak, Judy. Network Intrusion Detection: An Analyst's Handbook (2nd Edition). Indianapolis, New Riders Publishing, 2000.

- Poppi, Sandro. "Snort-Setup for Statistics HOWTO." 23 Feb 2002. URL: http://www.tldp.org/HOWTO/Snort-Statistics-HOWTO/configuration.html (19 Feb. 2003).
- Ptacek, Thomas H. and Newsham, Timothy N. "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection". URL: http://www.snort.org/docs/idspaper (5 Jan. 2003).
- Ranum, Marcus J. "NFR Security: Experiences Benchmarking Intrusion Detection Systems" December, 2001. URL: www.snort.org/docs/Benchmarking-IDS-NFR.pdf (7 Jan. 2003).
- Roesch, Martin, and Green, Chris. "Snort Users Manual Snort Release: 1.9.1." URL: http://www.snort.org/docs/writing_rules/index.html (4 Jan. 2003).
- Roesh, Marty. "SNORT FAQ." 25 Mar. 2002. URL: http://www.snort.org/docs/faq.html (16 Feb. 2003).
- Scott, Steven J. "Snort Installation Manual: Snort, MySQL and ACID on Redhat 7.3." August, 2002. URL: http://www.snort.org/docs/snort-rh7-mysql-ACID-1-5.pdf (1 Mar. 2003).
- Snort Configuration Documents URL: www.snort.org/docs (14 Jan. 2003).
- "SnortCenter Installation Manual." 2002. URL: http://users.pandora.be/larc/documentation/ (15 Mar. 2003).
- "Using User Authentication." 18 Oct. 1996. URL: www.apacheweek.com/features/userauth (7 Mar. 2003).
- Wassom, Darrin. "Auditing a Distributed Intrusion Detection System: An Auditor's Perspective." July 2002. URL: http://www.giac.org/practical/Darrin_Wassom_GSNA.doc (3 Feb. 2003).