Interested in learning more about security?

SANS Institute InfoSec Reading Room

Unix and Sarbanes-Oxley: a management and auditors guide

This document is designed to assist auditors of publicly traded companies and Security Exchange Commission (SEC) registrants to comply with the Sarbanes- Oxley act ("the act") by reviewing the generally accepted IT control framework called Cobit-12 and applying it towards a Unix environment. This document can also be used by management to understand the requirements that their auditors will be looking for when they perform their compliance tests. Although no single document or opinion can ensure full compliance towards...

Unix and Sarbanes-Oxley: a management and auditors guide.

For the title of GIAC Certified Unix Analyst (GCUX)
Practical Version 3

By Micho Schumann
SANS Las Vegas – October 2004

Submitted on March 20th, 2005

Table of contents

"… a skilled defense is one which opponents do not know how to attack."

Sun Tzu (The Art of War)

Abstract

This document is designed to assist auditors of publicly traded companies and Security Exchange Commission (SEC) registrants to comply with the Sarbanes-Oxley act ("the act") by reviewing the generally accepted IT control framework called Cobit-12 and applying it towards a Unix environment. This document can also be used by management to understand the requirements that their auditors will be looking for when they perform their compliance tests. Although no single document or opinion can ensure full compliance towards the act or can guarantee companies to have no deficiencies reported by their auditors, this document will assist in covering all the important bases for Unix operating environments.

Disclaimer

This document should be viewed as the opinion of the author. The author makes no guarantees as to the compliance of companies or operating environments following the use of this document. Furthermore, this document should not be viewed as a comprehensive Unix security guide that will ensure complete Unix security. There are many great books out there for that purpose; some of them can be found in the references section of this document. However, following the guidelines found in this whitepaper should assist management and auditors of Unix environments to appropriately evaluate compliance to the act.

Introduction

In late 2001, while the world is still recovering from the tragic events of September 11th in New-York, one of the largest corporate scandals to ever hit the financial markets strikes: the collapse of Enron.

Following the collapse of Enron and other publicly traded companies, the Sarbanes-Oxley act was created. Sarbanes-Oxley introduced a sweeping reform for corporate governance rules, regulations and standards for public companies and SEC registrants. These new rules make company CEO's & CFO's criminally liable of the financial statements that their companies publish. Therefore, these individuals must ensure that appropriate internal controls are in place, so that their wellbeing, their company's and their investor's is safeguarded.

After the introduction of the act, the PCAOB[1] published auditing standards that included IT controls. These controls have been matched to twelve sections of the well known COBIT IT controls methodology. This document reviews the twelve sections of COBIT and links them, when possible, to the Unix operating environment. Management and auditors will need to examine each of these sections for compliance of the IT based internal control rules for all IT operations, including Unix environments.

This document must be viewed as the author's personal opinion and should not be seen as a "one size fits all" document for compliance to the act. However, if the elements seen in this document are covered for a standard Unix environment, compliance should be attainable. Auditors and management should refer to PCAOB for sample sizes to ensure appropriate testing is performed. Sample sizes are beyond the scope of this document and are not covered.

[1] Public Company Accounting Oversight Board

Cobit 12

The diagram below represents the twelve sections of COBIT, mapped to the PCAOB control sections. The color coding was added by the author of this document and should be read as follows:

- Will be covered by this document
- Will not be covered (out of scope)

Control processes				
	PCAOB IT General Control heading			
COBIT control objective heading	Program development	Program changes	Computer operations	Access to programs and data
1. Acquire or develop application software	•	•	•	•
2. Acquire technology infrastructure	•	•	•	•
3. Develop and maintain policies and procedures	•	•	•	•
4. Install and test application software and technology infrastructure	•	•	•	•
5. Manage changes	•	•		•
6. Define and manage service levels	•	•	•	•
7. Manage third-party services			•	•
8. Ensure systems security			•	•
9. Manage the configuration			•	•
10. Manage problems and incidents			•	
11. Manage data			•	•
12. Manage operations			•	•

Source: IT Governance Institute, 2004 (Color coding added by the author)

To be clear, for Unix operating environments, only some of the categories above will be covered, since the twelve sections do not all apply to an operating system. Sections related to "program development" and "program changes" (the two columns on the left) of the PCAOB will not be addressed, since programming and development is out of the scope of a Unix operating environment audit that is the focus of the GCUX certification. In this paper, we will assume that management and auditors have access to the operating system to make their tests for compliance to the act.

Also, some of the controls that will be discussed in this document are not uniquely related to the Unix operating environment. For example: backups. As we all know, backups are an essential function for business continuity and good computing practices. Furthermore, the act takes disaster recovery very seriously, so for that reason, a few such items will be covered. For management, it is also good to note that auditors will most probably not look at servers that are not financially significant; basically those that do not host financial applications or process financial data. Of course, if a company were to follow best practices, all servers in a company should be appropriately secured, patched, etc. However, when time is of essence and auditors are on their way, efforts should be put first towards the servers that are financially significant.

The test controls that will be discussed in this document were done, when required, on an OpenBSD version 3.6 server running on i386 architecture. Although there are many other flavors of Unix that exist, most, if not all of the controls that will be discussed in this document should be applicable to them; the only difference may be the location of files, directories or commands.

Before actually getting into the thick of the document, I believe that a few definitions should be presented, especially for those who do not do audit work or have an audit background.

- Control: Policies, procedures, practices and organizational structures designed to provide reasonable assurance that business objectives will be achieved and that undesired events will be prevented or detected and corrected[2]

- Compensating control: An internal control that reduces the risk of an existing or potential weakness resulting in errors and omissions. [3]

- Materiality: An auditing concept regarding the importance of an item of information with regard to its impact or effect on the functioning of the

[2] 2002 : CISA review manual. ISACA
[3] 2002 : CISA review manual. ISACA

entity being audited. An expression of the relative significance or importance of a particular matter in the context of the organization as a whole. [4]

Controls are the foundation of any type of audit work and the focus of this document, since, at the end of the day, the auditor will have to evaluate the quality of the controls for compliance to the act. A control can be as simple as appropriate password settings to as complex as biometric authentication. Of course, the operating system is just one piece of a much larger picture. However, for example, if the operating system has no passwords, have never been patched and have all default services running, serious deficiencies will most likely be raised and may jeopardize the companies overall compliance.

The compensating control is also very important and must be understood by management and auditors. To illustrate a compensating control, consider the following: a Unix server does not require administrators to SU in order to gain Root access. They can authenticate with Root directly. As we all know, this is a cardinal sin in any Unix production environment. However, if we were to say that the server is not network accessible (only via the console) and it is in a secure server room where only 3 people have access and where the access to this room is strictly monitored. I believe that you will agree that the direct Root login issue just got significantly less worrisome. The compensating controls do not mean that it is OK. The compensating control simply reduces (as in the definition) *the risk of an existing or potential weakness.* This being said, all the controls that are mentioned in this document that are not present or insufficient (ex: weak password settings) can be compensated by one or more compensating controls. The validity of a compensating control will be evaluated by the auditor.

Finally, here we discuss the materiality of a weakness. When speaking of materiality of a weakness, we must first have identified a deficiency in a control. Just a deficiency alone may not be *bad.* However, the deficiency could be material if the impacts put the financial information into doubt. So basically, a material deficiency (also called material weakness) could all alone, jeopardize compliance to the act. As an example, a material deficiency could be the following: The financial package of a company is hosted on one main Unix server. This server is accessible to anyone on the network and does not require any passwords. Anyone can access the server and launch the financial package.

There are many other examples possible. Just keep in mind: does this individual weakness mean the financial information could be compromised? Of course, before concluding to the materiality of a deficiency, the auditor should discuss findings with their managers and/or with the financial audit teams. The word individual was underlined above. A material weakness is one deficiency, which taken alone could lead to fraud. However, audit teams may also find materiality

[4] 2002 : CISA review manual. ISACA

with a group of deficiencies. This will not be discussed.

Ensure systems security

Ensure systems security includes both physical and logical security mechanisms. These mechanisms must prevent unauthorized access to the systems. The author will not deal with physical security in this document, since it is out of scope.

*A*uthentication

As with any operating environment, the cornerstone of systems security is authentication. You will want to make sure that the authentication scheme is appropriate and that is will keep intruders out of the operating system.

In OpenBSD, as in any other Unix system, the operating system should require at least 8 characters for passwords. For highly critical servers, this setting could be higher. You will find these settings in /etc/login.conf. Look for the variable *minpasswordlen.* Along with the password, setting, you should make sure that login-backoff is active. This is a sort of intruder lockout that is found in the Windows world that will put login to sleep for a period of time. Also, make sure that the *passwordtime* is set to 60 days or less. If it is more than 60 days, make sure that there is a valid reason for this.

The auditor will want to test these configurations. Ideally, the auditor will not want to lockout legitimate accounts. Have the Unix administrator create an account and then test the backoff function. As for the password change time, you will have to rely on the setting, since this will not be practical to test. Also, attempt to change the password and to set the password to have less characters that the specified in the system. Of course the company that you are auditing may be using other authentication systems such as tokens or biometric devices. These should of course be evaluated by the auditor and included in the overall assessment.

User accounts & groups

For user accounts, only authorized personnel should have access to the server. This may sound obvious, but from the author's experience, it's definitely not followed in the real world. The first task of the auditor would be to verify that all users who have access to the server are still employees of the organization. A walk over to the HR group to get a listing of employees would be the best way to proceed. If the organization has a large number of employees, have an HR

person look at the /etc/passwd list with you and validate the user list. While the auditor verifies the /etc/passwd file, the use of shadow password should also be verified. This can usually be done by looking at the listing of users and the character(s) after the first colon:

```
bsdbox#  more /etc/passwd
root:*:0:0:Charlie &:/root:/bin/csh
daemon:*:1:1:The devil himself:/root:/sbin/nologin
operator:*:2:5:System &:/operator:/sbin/nologin
```

As we can see above, the accounts are followed by a colon and then a *. The star symbol usually means that shadow password (or an encryption method) is in use. Should there be a hash (long string of characters) instead of the *, then shadow password is most probably not in use and passwords could be cracked if the /etc/passwd file is compromised.

Back to the /etc/passwd file; once the auditor has validated active users, it is important to make sure that only the necessary accounts can login to the system. So make sure that all other accounts, such as system accounts (ex: named) do not have any login privileges. You can check this by looking at the end of the line associated to their name. They should not have a shell; you should see something like /dev/null or /sbin/nologin.

Once the legitimacy of the users who are present on the server is cleared up, you will have to take a look at their UID (User Identifiers)

```
micho:*:1000:1000:micho:/home/micho:/bin/sh
julie:*:1001:1001:Julie Poirier:/home/julie:/bin/sh
```

In the case of the above print screen, the user micho has a UID of 1000 and julie has as UID of 1001. The Root account has a UID of 0. The auditor will want to make sure that no other user accounts have a UID of 0. If others are present, the auditor should enquire with the Unix administrators of the reason for this. Unless a business requirement justifies this (ex: a specific software vendor requires this for its application), there is no reason for this to be and this would be a deficiency under this category.

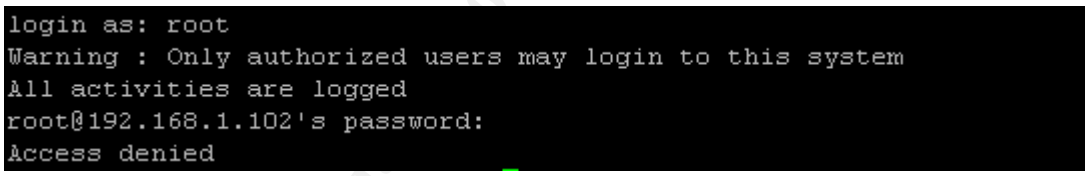
For groups (/etc/group), the auditor should take an inventory of all the available groups and identify the use of each one and challenge administrators about any group that does not appear to serve any purpose. Special attention should be paid to the wheel group. This group allows specific users to *su* (short for *substitute user* or superuser as some believe) to root privileges. Only a limited number of users should be included in the wheel group. The auditor will have to make a judgment call as to what is reasonable while considering the IT group size, the number of Unix administrators, the size of the operations, etc. If no one is in the wheel group and everyone just logs on with Root, a deficiency should definitely be reported.

If the server hosts some sort of application that requires users to have operating systems level accounts (this is especially true with older applications), you will probably have a long listing of users in the /etc/passwd. However, the auditor will need to make sure that these accounts do not have shell access. To check this out, you have to look at the end of the line for each user ID and check that there is no shell such as /bin/csh, /bin/bash or something along those lines. Instead, you should see a path that upon login will automatically send the user to the application.

Root access

As most of the readers of this document will already know, the Root account is the single most important account in the Unix world. It is the all-powerful account that has full control over the operating system. It is also the Holy Grail for hackers and unauthorized users.

There are a few things for Root access that an auditor will want to verify. First, verify that administrators do not login directly to the servers using Root. To test this, simply ask the Unix administrator to login to the Unix server and observe what they do. If they login with Root, you have found a deficiency that they will need to correct. If the server is setup correctly, you should get a result similar to the following print screen where Root was denied login rights.



To restrict direct Root login, make sure that the /etc/default/login has the #CONSOLE=/dev/console line commented out. Be careful, this is not the only place you have to enter this. If administrators are logging in with Telnet (we'll address that later), you are covered. However, if they are using SSH, the mention *permitrootlogin* must be uncommented and should be set to *no* in the /etc/ssh/sshd_config file; this is often overlooked.

Segregation of duties

The segregation of duties is defined as following: "A basic control that prevents or detects errors and irregularities by assigning responsibility to initiating transactions, recording transactions and custody of assets to separate

individuals. Commonly used in large IT organizations, so that no single person is in a position to introduce fraudulent or malicious code without detection."[5]

In a financial package, segregation of duties is relatively simple. A payables clerk (who creates payments that are mailed to suppliers for example) should not have access to the supplier's master file and be able to create new supplier; this person could then create a phony company and send out payments to that bogus company and perform fraud.

For the Unix operating environment purposes, the segregation of duties cannot be as clearly defined. Someone HAS to be Root and have all of the privileges in the operating systems; that's a fact of life. The auditor will have to make a judgment call and decide if there is sufficient segregation of duties to ensure that there is no fraud and that no single person can cause significant damage.

Here are two scenarios that should give you appropriate guidance for your audit.

1. A small company with production Unix systems and only one systems administrator who has Root. Since there is only one person operating the systems and no one can check up on what they are doing, there may be an issue. Basically, this person could be performing fraudulent activities and no one would ever know about it. Even worse, since no one else knows the systems, the person could one day black-mail his/her employer by planting a logic bomb or something along those lines. To an auditor, the fact that only one person holds all the keys to the kingdom should ring some bells. Of course, there are situations where the situation is that the company is simply too small for additional Unix administrators.

2. In a large company, the problem is sometimes the exact opposite. The auditor will want to ensure that there are not too many people that can have Root access or some form or powerful OS access. For example, in an IT group of 10 employees, if all 10 can logon as Root, there may be an issue. The author would not expect more that 2-3 individuals in an IT group of that size to have Root access. The other administrators should probably only have limited access or only have access to specific commands via SUDO for example.

[5] 2002 : CISA review manual. ISACA

Manage the configuration

This section ensures that applications and systems are configured so that only authorized personnel are allowed to use systems. Also, this section also aims to ensure that applications are appropriately configured and that they are not victims of a virus or worm attack.

Authorized personnel

It is important to ensure that only authorized users are able to actually get to the server. One of the great ways to ensure that only authorized personnel have access to the server is TCP Wrappers. TCP Wrappers allow administrators to either specifically permit users to connect to a server (connect, not authenticate) or specifically deny users to connect to a server. These files can be found at /etc/hosts.allow & /etc/hosts.deny. If a reduced number of individuals require access to a server (say only the administrators), then the hosts.allow should contain their IP addresses or their administrative subnet IP range, should their workstations they be in DHCP. In the case where a large number of individuals need access to the server, such as in our *group* example, where users have a local account, then whole internal network (for example) could be in the hosts.allow. The hosts.deny is only for when there are specific IP's that should never talk to a specific server. For example, the DMZ servers that are never supposed to talk to this server could be included.

Compensating for this section controls could be various. Controls such as permitting only direct console access (the person must have physical access to login), VLAN's, biometric authentication and other such mechanisms can be considered.

Configured to prevent unauthorized access

In order prevent unauthorized access to production Unix servers, administrators must *harden* servers to make them less vulnerable to internal (employees) and external (hackers via the Internet) attacks. The company may have certain standards (see next section) or simply configure one server at a time in a more or less uniform manner. The auditor should examine a reasonable sample of servers in order to examine the security level.

First off, the auditor should carefully examine the /etc/inetd.conf file. This file contains and manages many services. This file should be reviewed with a Unix administrator, line by line. Services that are running (those not commented out) should all be linked to a business requirement, such as being required by a specific application. The auditor should pay special attention to the lines that

contain *telnet, tftp, rlogin, rexec* and *rsh.* If any or all of these services are running, the auditor must enquire as to any business requirements that require any or all of those services. Should they be running when there are no legitimate business requirements justifying their presence, the auditor may have to declare a deficiency. Should some of the more *dangerous* services (ex: telnet) be running, the auditor should examine if any compensating controls are in place. An example of compensating controls for telnet could be a switched network environment, the use of TCP Wrappers, token authentication, etc.

Once the inetd.conf is completed, the auditors should verify if the Sendmail daemon is running. This is often an overlooked item and Sendmail is well known for security weaknesses. To verify if Sendmail is running, the auditor must simply telnet to the server on port 25. If a prompt appears, the auditor will know it's alive. If Sendmail is running, enquire about its use and if its necessity. If Sendmail is required for a business function, examine the version number. Sendmail is notorious for security weaknesses and should always be patched and at the most recent version possible. The author has often seen well configured servers with a three year old version of Sendmail running quietly on port 25.

Finally, for highly critical environment, the auditor may want to perform a port scan on the server or even run Nessus (or a similar tool) to gain a good assessment of the services running. Having netstat outputted will also tell the auditor a story about the services running.

Configured to company standards

Many large companies have specific policies for the configuration of their production servers. The auditor should take these into consideration. Furthermore, these policies may also help the auditor understand the strategic importance of certain servers. The auditor should take these standards into consideration, but they must not override the basic security principals. For example, should the standard mention that it is OK to connect with Root directly and not use SU, the auditor should obviously ignore that guideline! However, should the standards require, for example, a token based authentication because of the criticality of the data stored on the servers, the auditor should examine if the company is compliant with it's own standards.

For example, should a company policy require all servers to have certain applications installed (e.g.: an IDS sensor), then the auditor will definitely want to sample a few servers and make sure that the servers are all compliant to this policy. We can only assume that the policy was written with the intention of protecting critical and/or sensitive data.

Virus protection & patches

To prevent processing errors and intrusion, a server should be appropriately patched to the vendors recommended level.

Having the patches is a good thing, but should not be the only thing. Indeed, administrators may do a "patching blitz" before the auditors visit. The auditor must find evidence that there is a process that is used by administrators to patch their systems on an ongoing basis, and not just when auditors visit for the year-end audit. Subscription to the vendor's mailing list, vendors patch CD's, vulnerability assessments and other such items are good signs that patches and vulnerabilities are a concern. Once a process has been established, the auditor will want to see what has actually been done on the servers.

For example, in OpenBSD, after much researching, there does not seem to exist a simple way to examine what patches have been applied. In this case, the auditor should look for evidence that the patch files are stored on the machine (OpenBSD patches are .patch files) or for some form of documentation such as a log to show that the individual patches were applied by administrators.

Some operating systems, such as Solaris, will list what patches were applied. The auditor will then be able to compare that list to the vendors listing that can usually be found on the Internet. Any patches that have not been applied should be discussed with administrators before being considered a deficiency.

For viruses, although this is a small risk in a Unix environment, it should still be examined. If anti-virus software exists for a particular platform, the auditor should enquire as to its use and implementation. Also, the auditor, along with the standard general controls[6] that will be tested in the course of an audit, should verify that the Unix administrator's workstations (which are usually Windows machines) are equipped with appropriate anti-virus software and the definitions are automatically updated.

Application level security

For a Unix operating environment, application level security is also critical. Before auditing applications on a server, the overall understanding of the company and operations is essential. Financial packages and other accounting systems are out of scope for this document and will not be discussed. This document will not discuss any specific applications such as web server daemons, since producing an exhaustive list would be impossible. The

[6] IT General controls include data backups, virus protection, system access and other such items.

important thing for the auditor is to pinpoint what applications are supporting the financial information. Immediately, the Apache Web server comes to mind, as does SSH. If the auditor judges that the compromise of these applications could lead to financial information being manipulated, stolen or destroyed, then they should be audited. The audit should be performed to ensure that the application is appropriately configured, that only authorized users may access it, that passwords are appropriate, that chroot is used when appropriate and so on. Most of the applications that an auditor may come across are probably covered in the SANS reading room (http://www.sans.org/rr). If not, most large audit firms usually have audit guides for the most popular Unix applications.

Manage data

This section aims to ensure that data is stored, transmitted and recorded in a secure manner.

Protection during data transmission

Unix servers may have many different types of data transmissions. The auditor must inventory these transmissions and identify any of these that could contain either login credentials or sensitive financial data. Once these transmissions have been identified, the auditor should enquire about the encryption that is used.

Too many times, this author has seen critical Unix servers being accessed with telnet. By experience, this is by far the most neglected encryption mechanism. SSH is definitely catching on, but many administrators have still not implemented it for lack of time or organizational will. Many Unix flavors now come standard with an SSH daemon, so unless there is a documented business justification for telnet of other unencrypted protocols, the auditor should raise a deficiency. Should there indeed be a business justification, management will most probably have to identify compensating controls to offset this deficiency.

Items to consider when examining data transmission:

- Web servers using http
- Remote consoles such as telnet
- Web based management consoles such as HP/Compaq's Insight manager
- Databases that interface with other servers
- Applications that are accessed by a remote session
- FTP servers
- X-Windows
- Etc.

Auditors must ensure that they are not bullied by administrators for the lack of SSH or encrypted data transmission protocols. Many times, the author has seen administrators invoke the fact that the data transmission is only on the internal network and not via the Internet. This does not have much merit and should still be identified as a deficiency. The fact that the data is only traveling on the internal network does mean that the data is less exposed to malicious individuals. However, as we all know, internal threats are also significant. Therefore, any sensitive financial data that is traveling from one system to another must have some form of data encryption or protection method. Compensating controls could be a direct link by a leased line (non Internet), a

switched environment (only partially compensates), isolated LAN segments, VLAN's, etc.

Data backups

Although data backups are not Unix specific, the author believes that this issue should be covered considering its strategic importance.

First of all, the auditor should enquire about the backup strategy of the organization and specifically of the Unix production servers. Once the strategy has been explained, the auditor may have to enquire further. Should backups be done less than once a day (for every business day; this may vary depending on the nature of the business), the auditor should get more explanations from management and systems administrators. Should the justification not be satisfactory, a deficiency may be declared. Justifications for extended intervals for backups could be a risk analysis that was performed and approved by management, a data classification scheme that indicates that the data on specific servers is not mission critical, etc. The auditor must take all of these into consideration and declare a deficiency or not.

Most companies will have their data tapes/disks shipped externally to a third party or to a secondary business location. Once again, the auditor should challenge management and systems administrators when tapes/disks they are shipped externally less than once every business day. As previously mentioned, a risk analysis, data classification scheme or other such document could justify why data is not externally stored, but the auditor will have to be the judge of this. Also, to prove that tapes/disks are actually sent to an external location, the auditor should verify the presence of receipts from when they are picked up by the 3rd party. If the tapes/disks are not with a 3rd party but sent to another office location, the auditor should enquire about a log or some sort of tracking mechanism.

Finally, since backups are only as good as the media on which they are stored, the auditor should enquire about restore tests. These restore tests should be performed on a regular basis in order to verify that the media is appropriately backing up the company's critical data. Restore tests should be done a few times a year on randomly selected backup tapes. These tests should be appropriately documented. Should restore tests not be performed, a compensating control could be a well documented restore process for when users require lost or accidentally deleted documents to be restored. This process, if frequently used and well documented by management, should include the user's request, the data tape name or number, the restored document name and the *completed* or *successfully restored* mention should be sufficient. Of course, the auditor will be the judge of this.

Manage problems and incidents

This section aims to ensure that any incidents or problems are responded to and investigated in an appropriate manner.

Logs

The auditor should enquire as to what types of audit logs are collected and more importantly, which ones are revised on a regular basis by administrators. First, the auditor should enquire about what exactly is being logged on the production Unix servers. There is no one correct answer for this, but below is what the author considers the minimal acceptable level.

Unix servers should be logging, at a minimum, users access (authentication, intruder lockout, etc), the use of root and SU (and the failed use of Root and SU) and adding and subtracting user ID's. An Integrity checker or an IDS should be able to do the rest (see the relevant section in this document).

Finally, should the company have a centralized log collector that is reviewed regularly by the Unix administrators; this would be the ideal situation. However, this is not accessible to all organizations. In the case that no centralized log system is present in a large Unix production environment, the auditor will have to find evidence that logs are indeed looked at and that critical events do not go unnoticed for long periods of time. Of course, if no logging in performed or if no one appears to be viewing them, a deficiency should be declared.

Monitoring

Since availability is an issue and must be addressed, the auditor will have to enquire about what type of monitoring is being performed. For critical servers that handle financial information that is time sensitive, the auditor should expect to see some form of automated monitoring that will track critical elements such as CPU load, disk space, memory use, application level failure (especially applications that deal with financial data) and other key parameters. Unusual CPU load could also mean that the server has been compromised and that a hacker is running tools that should not be there. An IDS or integrity checker could have picked that up, but is addressed in another section. Of course, the auditor is not required to test the monitoring functions for failures. However, the auditor should review policies with regards to monitoring & response times. The auditor should also examine what types of alerts are sent to administrators, what actions are taken to resolve any problems and if there is any formal alert follow-ups that allow the company to track problems and identify recurring issues.

Manage operations

Manage operations mainly aims at end-user computing which is slightly out of scope. However, the part that tells us that we must ensure that authorized programs are *executed as planned* entices me to discuss integrity checkers and intrusion detection.

Integrity checkers and intrusion detection

Some audit professionals may argue that integrity checkers and intrusion detection mechanisms are not required. These professionals may say that sufficient logging and live or periodic log checking is sufficient. The author of this document begs to defer!

As with any part of an audit, the auditor has to use judgment in this case. However, for large companies and highly sensitive environments, logging is most probably not enough to ensure that system administrators have all the information they need. Some form of intrusion detection or integrity checking should be present. The auditor should find out what type of protection is in place on the servers and what types of alerts are sent to administrators.

Good compensating controls for the lack of an integrity checker or intrusion detection system would be the use of TCP Wrappers, a filtering router or firewall in front of the server or only physical access to the server.

Crontabs

The crontabs, which are in charge of batch jobs and all sorts of scheduled activities, should be adequately secured. This item could have been covered in a previous section, but considering the *application execution* involved, the author prefers to address it here.

First of all, the auditor should make sure that the crontabs are appropriately secured and not available to all system users. Privileges for these files should be to the tightest possible setting. 600 (Read/write for Root and nothing for other users) for these files is the ideal setting, as seen below. However, if users have read (644) access it is not as good, but still acceptable.

```
bsdbox# ls -l
total 10
-rw-------  1 root  crontab     0 Sep 17 13:59 at.deny
drwxrwx--T  2 root  crontab   512 Sep 17 13:59 atjobs
-rw-------  1 root  crontab     0 Sep 17 13:59 cron.deny
-rw-------  1 root  wheel    1836 Jan 22 04:30 log
-rw-------  1 root  wheel    1237 Jan 21 01:00 log.0.gz
-rw-------  1 root  wheel    1188 Jan 10 14:00 log.1.gz
drwx-wx--T  2 root  crontab   512 Jan 22 03:20 tabs
```

Since this section is especially designed for managing the operations, as in letting administrators know when things work (or don't work), the auditor should enquire about what happens when the cron fails to execute a batch or an application. Many different possibilities arise. It may alert syslog for example or there may be some more sophisticated 3rd party software that sends emails, pager alerts or other such warnings to administrators. Mainly, the auditor should find out if there are any financially relevant batches of jobs that are executed by cron at specified times. If it appears that there is no mechanism to warn administrators of a failure, then this means that critical jobs may have failed, but no one was warned about it. If this is the case, the auditor should enquire about whether there is a documented procedure for administrators to verify cron failures or a regular basis (e.g.: daily); this could be an adequate compensating control.

Conclusion

Although this document is not designed to be a *one stop Unix security guide*, it is designed to assist auditors and management to understand what needs to be done so that compliance to the Sarbanes-Oxley act is attainable for a Unix environment that handles financial data.

For management: It is essential that an ongoing compliance program be implemented. It is useless to only review security once a year before the visit of your auditors; most seasoned auditors will be able to pick up on the lack of *security culture* in the company and will dig further to gain the real truth about the security of your information systems. Furthermore, ongoing security is an excellent practice and will make life easier for in-house technical personnel as well as for your auditors. A well maintained and secure IT setup is easier to audit and everyone (usually) comes out smiling. Not to mention that the stress and time consumed by your technical staff assisting auditors should be reduced. Finally, you should remember that the information systems contain critical financial data for whose accuracy and integrity you are personally responsible for under the act; this makes information security a whole lot more important.

For auditors: Following this guide should allow you to get a good picture of the compliance for Unix computing environment of the company you are auditing. Of course, each company is different and you will want to get a general overview before jumping into your audit. Should all or most (barring compensating controls) of the controls discussed on this document be in place, you can reasonably say that the environment you audited is in compliance. Keep in mind that the final verdict will be taken by the audit partner or an equivalent in your firm.

For non-audit Unix professionals: The audit world may seem strange to you. However, its not all that complicated! At the end of the day, your auditors will want to know if that widget that your company sold in Montreal for 3$ was correctly captured into the financial package that sits on your Unix servers. So basically, they want to know where could someone manipulate, delete or void that 3$ sale? THAT is what Sarbanes-Oxley is all about, and yes ladies and gentleman, your Unix servers are right in the middle of it!

References

1. Garfinkle, Simson & Spafford, Gene. Practical Unix & Internet security, 2nd edition. O'Reilly & Associates, Inc. 1998.

2. IT Governance institute. IT control objectives for Sarbanes-Oxley. 2004.

3. Information systems audit and control association (ISACA). 2002 CISA review manual. 2002.

4. Cheswick, William R. Bellovin, Steven M & Rubin, Aviel D. Firewalls and Internet security, Second edition. Addison-Wesley. 2003.

5. Hatch, Brian. James, Lee. Kurtz, George. Hacking Linux Exposed: Linux security secrets & solutions. Osborne/McGraw-Hill. 2001.

6. Sosinsky, Barrie. Tanielu, Carol. Mastering Solaris 8. Sybex. 2001.

7. Bauer, Michael D. Linux server security : 2nd edition. O'Reilly. 2003.

Appendix A - Auditors field checklist

The following checklist is designed to summarize the contents of this document so that auditors or management can assess Unix servers for compliance to the act according to this document. Of course, no one checklist can guarantee compliance (or non compliance). However, the author believes that the below listing will be able to give a good view of the level of security of Unix servers in the spirit of the legislation.

Sarbanes-Oxley Unix audit checklist			
Server name _____	**Verified (•) and relevant**	**Deficiency (Yes/NO)**	**Comments**
Authentication settings (password, lockout, password changes, etc)			
User accounts and groups (/etc/passwd & /etc/group)			
Root access (SU, console, etc)			
Segregation of duties			
TCP Wrappers			
Open services (Inetd.conf, Sendmail, etc)			
Check for a policy			
Virus protection			

System patches (log, binder, evidence)			
Application level security (http, ssh, etc)			
Secure data transmission (SSL, SSH, encryption)			
Backups			
Logs (SU log, logon/logoff, etc)			
Monitoring (CPU, Disk space, etc)			
Integrity checkers			
Crontabs			

Last Updated: March 15th, 2016

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Secure Singapore 2016	Singapore, SG	Mar 28, 2016 - Apr 09, 2016	Live Event
SANS Northern Virginia - Reston 2016	Reston, VAUS	Apr 04, 2016 - Apr 09, 2016	Live Event
SANS Atlanta 2016	Atlanta, GAUS	Apr 04, 2016 - Apr 09, 2016	Live Event
SANS Secure Europe 2016	Amsterdam, NL	Apr 04, 2016 - Apr 16, 2016	Live Event
Threat Hunting and Incident Response Summit	New Orleans, LAUS	Apr 12, 2016 - Apr 19, 2016	Live Event
SANS Secure Canberra 2016	Canberra, AU	Apr 18, 2016 - Apr 23, 2016	Live Event
SANS Pen Test Austin	Austin, TXUS	Apr 18, 2016 - Apr 23, 2016	Live Event
SANS SEC301 London '16	London, GB	Apr 18, 2016 - Apr 22, 2016	Live Event
ICS Amsterdam 2016	Amsterdam, NL	Apr 18, 2016 - Apr 23, 2016	Live Event
SANS Copenhagen 2016	Copenhagen, DK	Apr 25, 2016 - Apr 30, 2016	Live Event
SANS Security West 2016	San Diego, CAUS	Apr 29, 2016 - May 06, 2016	Live Event
SANS SEC542 Budapest	Budapest, HU	May 02, 2016 - May 07, 2016	Live Event
SANS FOR508 Hamburg in German	Hamburg, DE	May 09, 2016 - May 14, 2016	Live Event
SANS Baltimore Spring 2016	Baltimore, MDUS	May 09, 2016 - May 14, 2016	Live Event
SANS Houston 2016	Houston, TXUS	May 09, 2016 - May 14, 2016	Live Event
SANS Prague 2016	Prague, CZ	May 09, 2016 - May 14, 2016	Live Event
SANS Stockholm 2016	Stockholm, SE	May 09, 2016 - May 14, 2016	Live Event
SANS Melbourne 2016	Melbourne, AU	May 16, 2016 - May 21, 2016	Live Event
Beta 2 Cincinnati - ICS456	Covington, KYUS	May 16, 2016 - May 20, 2016	Live Event
Security Operations Center Summit & Training	Crystal City, VAUS	May 19, 2016 - May 26, 2016	Live Event
SANS SEC401 Luxembourg en francais	Luxembourg, LU	May 30, 2016 - Jun 04, 2016	Live Event
SANSFIRE 2016	Washington, DCUS	Jun 11, 2016 - Jun 18, 2016	Live Event
NetWars @ NSM Conference '16	OnlineNO	Mar 16, 2016 - Mar 17, 2016	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced