Global Information Assurance Certification Paper

Monitoring the Security of a Legacy Solaris System
Steve Allanson
Version 1.2e
September 28, 2001

Introduction

My company was new and we put together the entire commercial (production) network, management network and the Network Operating Center in less than a month. It was difficult to keep track of what the various contractors were doing on our behalf. We did a lot of things right and others we did not do such a good job with. One of the areas we needed to improve a bit was the security of our Unix servers during the initial operating system installation.

So, I needed a way to automatically check the 'security configuration' of these servers; there are more than 25 of them in use. I also wanted to build a way to periodically monitor their 'security configuration' health. I dug into the literature and attended a "Security Essentials Class" from SANS to get me on my way.

Background

I was new to the company and tackling a job it seemed no one wanted to take on providing some of the security hardening to our system, both production and management networks. It was absolutely necessary to get this task accomplished in a timely matter. I had more than 25 Solaris servers and over 50 network devices to deal with. What to do?

I decided to attempt the Solaris server task first. I started looking into the various options available. I found many papers and tools describing what needed to be done – as it turned out there was lots to do. There were tools available to harden Unix servers, but they were mainly available for use on newly installed systems, as you were installing the operating system. For example, some of these tools or documents include: "Solaris Security Step by Step" distributed by SANS, Titan, a set of scripts released by several Sun Microsystems engineers, and YASSP (Yet Another Secure Solaris Package) released by Xerox engineers. [2] These packages provide many ideas and many useful ideas of what needs to be done to secure your system.

But, I wanted and needed something for an installed system, a system I could not start from scratch with. There was nothing new in what I wanted to accomplish, it was how I wanted to do it. I decided to read and gather as much information as possible and then create something I could use to monitor systems as well after setting the system up for as secure an environment as possible.

I was unable to find a definitive document detailing all that needed to be done, so I started collecting the available documentation, talked with more experienced security engineers about a fruitful approach to what I wanted to do, and decided a good way to help me do this was to make a list of the most important items, at least the most important to me, and find a way to check these particular items and then monitor them on a regular basis. I decided to set up the computers for as much security as possible without having to start over on their configuration and then have the computers check

themselves periodically by using 'a cron job' to run a script to activate available Unix commands and tools. I will attempt to describe this script, some of the thought that went into creating it, and then its use in this paper.

What needed to be checked?

There are many documents and tools available to start with, far too many to digest easily. I looked at many of the available documents and started building my list. Here is the list, which is by no means all-inclusive, and my reasoning behind most of the items in the list, or at least a short explanation of why I thought it important enough to periodically keep track of that particular item.

1. setuid (SUID) and setgid (SGID) – the 'setuid' essentially sets the effective user id running a program to the owner's user id for the program, and the setgid essentially sets the effective group id to the owner's group of the program. [5]

2. log files – log files are important and should be reviewed regularly for any irregular activities. These log files include:
    a. /var/adm/messages – contains the majority of the system messages.
    b. /var/log/syslog – contains the mail system messages.
    c. /var/log/authlog – important authentication log messages.
    d. /var/adm/sulog – successful and unsuccessful attempts to become superuser.
    e. /var/adm/vold.log – information concerning the various volumes attached to the server.
    f. /var/adm/utmpx – user access and administration information.
    g. var/cron/log – keeps information concerning the cron activities.
    h. /var/adm/wtmpx – history of user access and administration information. This file can be viewed with the 'last' command.
    i. /var/adm/loginlog – unsuccessful login attempts.

3. default mask – this is the default mask created when a user creates a file or directory. This level should be set so the owner can allow as much permission as he desires without a too lenient default level of permission. [9]

4. write permissions in the /etc directory – it is unnecessary for there to be a group write permission for the files in the /etc directory structure. [9]

5. startup files – these files are used to start a service or application. However, with the installation of the operating system there can be many services included and started which are actually unnecessary for the server's operation and purpose; it really depends on the particular server. These files, located generally in the /etc/rc2.d and /etc/rc3.d directories, need to be audited. They should be evaluated and then any unnecessary services should not be started with these start up scripts. Changing the name of the unnecessary files is one way to do

this and at the same time leave the file there in the event it is ever needed in the future. [4]

6. user accounts – the user accounts on a server need to be kept up to date. Also, old accounts need to be removed and any account, which is no longer required, should also be removed.

7. various system files – these includes the following:
    a. /etc/rc2.d/S72inetsvc – this is the file where a domain name serve is started, if it exists, and also where the 'inetd' is started.
    b. etc/inetd.conf – this is the file, which starts some of the services and generally starts many more services than are required for the server to be secure in its operation.
    c. /etc/ftpusers – this file limits the access to the 'ftp' service by the account user id's listed in this file.
    d. /etc/default/telnetd – this is the message which shows when a person is accessing a server via telnet.
    e. /etc/default/ftpd – this is the message which shows when a person is accessing a server via ftp.
    f. /etc/passwd – this file contains information pertaining to the registered users.
    g. /etc/shadow – very similar to the /etc/passwd file except that it contains encrypted password information and also has restricted access.
    h. /etc/group – this file stores system group information.
    i. /etc/default/login – this file contains information that defines several default parameters.
    j. /etc/inetinit – sets up the routing capabilities of the system during startup.
    k. /etc/hosts.allow – lists the ip addresses allowed access to the server using various processes available to remote users.
    l. /etc/hosts.deny – lists those ip addresses with no access, or limited access to the server.

8. cron activities – the cron is a way to activate policies automatically in a very regular time pattern.

9. kernel parameters – this mainly concerns the TCP/IP parameters of the operating system. The way to adjust these parameters is with the 'ndd' command. But any changes here are temporary in that if the server is rebooted the parameters are again defaulted. These changes must be reset after startup and this can be accomplished using a script set to run during startup. [3,6,7] Note: any changes made here must be reviewed when the operating system is upgraded. [10]

10. patch levels – applying patches is probably the easiest and surest way to keep up with the many security enhancements and fixes from the operating system manufacturer or from an application manufacturer. It is absolutely necessary for

the system administrator to monitor and apply the new patches for known security holes in a system. There are really two types of patches available from the manufacturers. They are security patches, which correct known security problems with the software or operating system. Next are normal patches that correct known software deficiencies, but these normal patches fit into two basic categories. There are patches that may be classified as enhancements and may or may not be necessary for system operation and on the other hand there are manufacturer recommended patches, which should be installed as they affect system operation o some extent.

How to check these items

In this section I will describe in more detail what I hoped to accomplish by checking and monitoring each of the areas described in the previous section. When appropriate I will give the necessary command and an example of the output from the command.

1. setuid (SUID) and setgid (SGID) – it is necessary to check the existing files for these permissions and at the same time verify there are not new setuid and setgid permissions being created. A possible result of a "new" setuid or setgid is a back door into the system. I should also remind myself that it is necessary to check each of the files listed as to whether the setuid or the setgid is necessary for the operation of the associated package.

   The following command will print out a listing of all the files on the system that have the setuid or setgid bits set.

   ```
   find / -type f \( -perm –u+s –o –perm –g+s \) –ls
   ```

   The following command will list all of the setuid files and also list the details concerning each of the packages they belong to. This is especially helpful for legacy systems because it helps narrow down the files that actually need to have the setuid or setgid turned on. If the package associated with the file is not being used the bits may be changed or the package may even be removed.

   ```
   find / -perm –u+s –exec pkgchk –l {} \;  [1]
   ```

   This command is not in the script and I use it to manually check the individual files and remove the setuid and setgid permissions when it is possible to do so.

2. log files – basically log files need to be checked or any new valuable information. But, they must exist before they can be useful. So, in most cases I am checking that the log file exists and has not for some reason or another disappeared.

3. default masks – the default mask on the servers I am monitoring have been modified from the default value to be 077. This value gives the owner

permissions in the created directories and files, but no one else. The owner is responsible for allowing others to access and use his directories and files.

4. write permissions in the /etc directory [4] – initially it is a good idea to run the following command to set all files to not have group write permissions, this makes it easier to determine which if any of the files have had their permissions changed to allow for the group write permission.

   chmod –R g-w /etc

   The following command will provide a listing of all files with group write permissions. This information will allow for the easy monitoring of any changes in the group write permissions of the files in the /etc directory.

   find /etc –type f –perm –g+w –ls

5. startup files – this is another good check for existing systems. Many times applications are installed and then not used or packages installed and not used. So, in order to eliminate unused services you need to know what is there. These startup files can be removed or you can change the name of the file and leave it in place in the event you may need the service in the future. I also like to use this check to see if there are unauthorized applications being started during a reboot.

   ls –l /etc/rc[2-3].d/S*

   I try to remove all of the unnecessary startup files and reduce the number of services to the minimum necessary for the server operation.

6. user accounts - it is important to verify each of the accounts periodically. This command will provide information on each of the accounts and can be compared to previous data to monitor changes in accounts or additions of new accounts:

   logins –x, this command provides the following information:
   - home directory
   - login shell
   - password aging information and also whether the account is active or locked

7. various system files
   a. /etc/rc2.d/S72inetsvc – verify the inetd daemon is being started with logging turned on. The logging is turned on with a '-t' added to the startup command for the inetd service. This causes all TCP connections to be logged showing the client ip address and TCP port. This is checked using the following form.

cat /etc/rc2.d/S*inetsvc | grep /usr/sbin/inetd

the result should be:

/usr/sbin/inetd –s –t &

b. etc/inetd.conf – verify the services being initialized are set as desired and have not been changed. The services started with this file can be checked easily. For example here is a command that will check the file for services started, i.e., lines that are not commented out.

   cat /etc/inetd.conf | grep –v "^ \{0,\}#"

c. /etc/ftpusers – verify the file exists, exists by default in Solaris 8, and verify the accounts contained in the file.
d. /etc/default/telnetd – verify the telnet login screen is set as desired with the necessary information to the user logging into the system. This file should contain information similar to MOTD file and will provide information to the person attempting to telnet to the system. All references to operating system should be avoided.
e. /etc/default/ftpd – verify the ftp login screen is set as desired with the necessary information to the user logging into the system. This file should contain information similar to MOTD file and will provide information to the person attempting to ftp to the system. All references to operating system should be avoided.
f. /etc/passwd – assure the file permissions on this file are as desired.
g. /etc/shadow – assure the file permissions on this file are as desired. Only the 'root' user has access to this file.
h. /etc/group – assure the file permissions on this file are as desired.
i. /etc/fefault/login – check this file to verify there is not telnet access for the 'root' user. Any user desiring to use the 'su' command, via telnet, must first login as an authorized user and then 'su' to the superuser. The following command will query the file and check for the correct entry.

   grep '^CONSOLE' /etc/default/login

   The output of this command should show an uncommented line entry.

   CONSOLE=/dev/console.

j. /etc/inetinit – check this file to verify the setting of the 'TCP_STRONG_ISS' variable.

   grep '^TCP_STRONG_ISS' /ec/default/inetinit

The output of this command should show the value to be equal to:

TCP_STRONG_ISS=2, for the strongest protection

k. /etc/hosts.allow – verify the existence and contents of this file.
l. /etc/hosts.deny – verify the existence and contents of this file.

8. cron activities – basically here I like to check for unnecessary items in the cron listing. I use the command:

   crontab –l

   to list out root's crontab and look for new or unnecessary entries.

9. kernel parameters – this is an area which is really configuration dependant. The items to be monitored and checked need to be evaluated based on the purpose of each server and operating system version. It is best to utilize a script on startup to assure the correct information is set into the configuration. A good starting point is a tool from Sun Microsystems called 'nddconfig'. This tool may be downloaded from Sun Microsystems at the following location:

   http://www.sun.com/blurprints/tools

   This is a shell script, which can be used to initialize the network driver parameters. This shell script can be modified to meet current conditions and then be used during a reboot to reinitialize the parameters. In order to automatically reinitialize the parameters the suggested location for the file is to copy the shell script to /etc/init.d/nddconfig and then set up a link to /etc/rc2.d/S70nddconfig. This will have the shell script run each time the server is rebooted.

   I have included several of the kernel parameters in my script because these particular parameters are important to our operation. I use a command of the following form to obtain the value of the parameter while not changing the value for my evaluation:

   ndd –get /dev/ip <parameter name>. i.e.,

   ndd –get /dev/ip/ ip_forwarding

10. patch levels – there are several different Solaris commands and tools available to check the current level of patches installed on the serve. A nice tool with a very readable output is called 'patchdiag'. The output provides information concerning installed patches verses the current patch version available, Sun Microsystems' recommended patches that are currently not installed along with a listing of not

installed security patches. This tool is available to contracted users of the Sun support process. It is a perl script that uses a flat data file to evaluate the current level of installed patches against the data file. The data file is periodically updated, bi-weekly, and therefore, should be updated regularly on the server. This data file may be downloaded from the following location:

ftp://sunsolve.sun.com/pub/patches/patchdiag.xref

Explain action

As described above there are many items that need to be checked regularly for security purposes. There are far too many to check manually if there are more than one or two servers involved. So, I enlisted the assistance of a colleague to help create a script from which to work. The result is the attached script 'sec-check-v2b.sh'. It is still a work in progress and it will more than likely always be so, especially with the advent of new tools becoming available to utilize in this security adventure. I have changed it and hopefully improved it in several places while writing this paper. Everything described above is not included in the script, because I do check several of the item manually, but most of the items are included.

I have the script run once a week with the output file e-mailed to me. I take care to update the reference file for the patch level on a regular basis as Sun Microsystems updates it periodically. I then do some post processing on the script's output to extract the items that should not be changing or other items, which need to be analyzed. These areas include the 'setuid' and 'setgid' files, the information included in the /etc/inetd.conf file, the various log files and the patch information.

On a regular basis as well I also run the command:

```
/usr/sbin/sysdef
```

to obtain the actual system information. This includes the installed drivers, loadable objects, information concerning swap, various tunable parameters, IPC semaphores, and IPC shared memory. These items should not be changing and if the output of the command is sent to a file and then the file compared to a baseline it is possible to see any changes made to the system. At least any changes to the items covered with the output of the sysdef command.

Conclusions

Setting up and maintaining a secure server environment is only the beginning of a never-ending task. There is always something to be checked and always something that can be improved upon. I spend many hours each month looking at the various

servers, reading the available security information, and training others to be security conscious when they are working on a server or installing an application. I pass out all of the necessary information to anyone who will listen. I am making progress and I hope you are too. Also, I hope this paper was of some use to you and if you choose to use the attached script, feel free to make changes and improvements as necessary to make it a bit more valuable to you.

References

1. YASSP, Post Installation Steps (Updated for beta#12).

2. Brumley, D. (Oct. 2000) "Solaris Security Recommendations from SANS Step by Step Guide, Titan, and YSSAP." http://www.theorygroup.com/Theory/matrix.pdf

3. Dubrawsky, I. (20 Dec. 2000) "Solaris Kernel Tuning for Security."

4. Orebaugh, A. (2 Oct. 2000) "Securing Solaris." http://www.sans.org/infosecFAQ/unix/sec_solaris.htm

5. Noordergraaf, A. and Watson, K. (Jan. 2000) "Solaris Operating Environment Security." SunBlueprints OnLine. http://www.sun.com/software/solutions/blueprints/browsedate.html#0100

6. Watson, K. and Noordergraaf, A. (Dec. 2000) "Solaris Operating Environment Network Settings for Security." SunBlueprints OnLine. http://www.sun.com/software/solutions/blueprints/browsedate.html#1200

7. Spitzner, L. (22 Oct. 2000) "Armoring Solaris, Preparing solaris for a firewall." http://www.enteract.com/~lspitz

8. Wilson, J. (16 Feb. 2001) "Securing your Solaris Server – Harden from the inside, protect from the outside." Unix Insider ITworld.com http://www.itworld.com/Comp/2377/UIR010216hardening

9. Galvin, P. B. (1 Jan. 2001) "The Solaris Security FAQ." Unix Insider ITworld.com http://www.itworld.com/Comp/2377/security-faq

10. Sun Microsystems, Inc. (Jan. 2001, Jul. 2001) "Solaris Tunable Parameters Manual." http://docs.sun.com:80/ab2/coll.707.1/@Ab2CollView?Ab2Lang=C&Ab2Enc=iso-8859-1