# Global Information Assurance Certification Paper

**Security to Think About for a Beginner in Unix**
By: Jeromy Hackney

## Introduction

In the consulting business the question can be raised regarding the underlying knowledge of a consultant auditing the security of a system. Many times consultants find themselves in situations that require them to evaluate and make recommendations on a system they don't have a good understanding of. If presented with the opportunity to audit a Unix or NT environment, most would probably choose NT. The reasoning is simple: people tend to feel more comfortable in a Windows environment. However, many companies rely on Unix boxes just as much as they rely on an NT box. In fact, the vast majority of environments that are encountered in the real world are a mixture of both. Encountering a Unix environment for the first time can be overwhelming if you don't have the benefit of previous knowledge. From personal experience, if you don't seek this knowledge it won't be offered to you. New consultants are expected to dive right in without the slightest understanding of user IDs, group IDs, permissions, access control lists and general Unix security. Do most new consultants know what access control lists are? Most probably don't. How could anyone possibly evaluate access, integrity and vulnerabilities of a system without some of the above knowledge? What happens next for the consultant is mass confusion, frustration and a missed opportunity to gain valuable knowledge. Moreover, this isn't just a phenomenon among consultants. I've worked with clients that have unequivocally denied users access to their Unix box. The reason for this was they didn't want their users to touch what they didn't understand. The developers coded on NT and used change management software that had ability to move code between machines. Those developers didn't need knowledge about Unix, but a consultant that never knows what job the next day will bring needs some basic knowledge. I would like to focus on the flavor of Unix I find most interesting, Sun Solaris. Hopefully this will serve as a starting point that will allow readers of this paper to seek further in-depth knowledge without the fear of thinking Unix should be handled by the more technically or computer-savvy users.

## Access and Permissions

The first thing to know about a Unix system such as Solaris is how a user gains access to the system. As in most systems users need a username and a password to gain access. They also need a user identification number, group identification number(s), a home directory, a login shell and initialization files. Most experienced users stick to the command line to add or view users in a system, but this can be intimidating. Some versions of Solaris, such as Solaris 2.8, come with a nice GUI interface that can present the same information in a friendlier manner. The tool used to view user information among other things is called the admintool. Before moving on it will be helpful to identify how each of these affect security in regards to integrity and helping to audit

the system. I also will discuss which files track the user information discussed above and how permissions can control access.

Like most systems, a username is typically a combination of a users last name, and the first and middle initial. Passwords can be set a few different ways in Solaris. You can set a password to be changed the 1st time a user logs in or you can assign a password to a user. Each user needs to be assigned a unique user identification number. The number can be as large as 2147483647. Assigning a number over 60,000 can cause compatibility issues with some earlier versions of Solaris. From an integrity standpoint, unique user identification numbers provide the ability to audit the actions of user. If a problem arises it may be necessary to determine who caused the problem. Two users with the same user ID would prohibit this. Additionally, users with the same user ID would have the same permissions to each other's files. We'll discuss permissions to access files later.

The next thing a user needs that affects security is a group identification number. Some group identification numbers come with Solaris when installed. Some examples of these groups are root (0), sys(3), adm(4). Be aware that there are other groups that come defined with Solaris, but overall groups 0 - 99 are reserved for system accounts. The system administrator can define the groups from 100 to 60,000. For example, one group could be for accounting staff that all perform the same duties. So how exactly does a group ID affect system security? For one, you immediately gain uniformity for your users. Different groups of users have needs that are different. When auditing a system it is a good idea to identify what users are in a group. Knowing who is in a group provides a simple way to identify the permissions users have for certain files or directories. The access and permissions a group has on a file or directory tells you the access and permissions that everybody in that group has. Payroll personnel should be the only employees that have access to files or directories that contain salaries, social security numbers and other personnel information.

Users also must have initialization files that define their login shell or work environment. These files run when a user logs in. Finally, all users need a home directory in Unix. This is where a user keeps files that they create and can allow access to for other users. After doing a **ls -alrt** command on a directory the following is a sample of what might be seen.

```
drwxr-xr-x  4 psoft   psoft      3584 Mar  1 09:51 bin
-rw-r--r--  1 psoft   psoft     10038 Mar  7 10:18 GME070.sqr
```

The first line is a directory, identified by the d at the beginning of a line. The second line is a file, which is represented by the dash at the beginning of the line. The 4 and the one represent links or names the file has. The first 'psoft' is the name of the user who owns the file. The second 'psoft' is the name of the group that owns the file. In this example they happen to be the

same. Size, date and time stamp and the name of the file follow. But what do the combination of r, w and x at the beginning mean? These combinations tell you the access that certain users have. Looking at the second line we start with the second letter. The first three letters portray the permissions of the owner of the file. The second three letters tell the permissions of the group that owns the file. Finally, the last three letters portray everybody else's permissions to the file.

- 'r' means read access
- 'w' means write access
- 'x' means execute access.

So for the file named GME070.sqr the file owner can read and write to the file, while the group and everybody else can only read the file.

**Access Control List**

Access Control Lists (ACL) provides a more granular approach to file permissions. Sometimes normal permissions for a file do not provide the desired results and don't provide the integrity that is needed. If you wanted to grant read access to everyone in a certain group, but wanted one member of the group to have write access, regular permissions could not provide this functionality. Access control lists allow the ability to grant users access based on what they need. Access does not need to be granted based on group, but is based on relevance to individual needs. The list can be defined based on users and groups. The best way to understand ACLs is to take a look at a few commands and describe what they mean. One thing to keep in mind about ACLs is that they can only be used with Solaris 2.5 or higher. The two commands that are important when dealing with ACL are getfacl and setfacl. Like most Unix commands these commands are long and contain many options. The getfacl command will tell you the ACL that a file presently has. All files have an access control list, so running this command on any file will return some kind of output. Before looking at a command it is important to understand what a mask is. A mask is basically the maximum amount of permissions that a user or a group can have. Changing the mask a file has allows you to change permissions a user or group has.

```
#getfacl testacl
#file: testacl
#owner: acl
#group: security
user::rwx
user::testuser:rwx              #effective:r--
group::r--                      #effective:r--
mask::r--
```

other::---

The first line is the format that would be used to view the ACL for 'testacl'. The file name, the owner and the group follow on the next three lines. The 1st user is the owner of the file and that person has read, write and execute permissions. The user 'testuser' also has all permissions, but a mask has been set that only allows users (besides the owner) and groups to have maximum access of read permissions. The effective permissions on the right side are the permissions that the user and group actually have. The ACL on 'testacl' has specifically limited the permissions that a single user, 'testuser' has on a file. This is a very simple example, but it illustrates the importance of access control list when trying to tighten security on files.

In order to set an ACL use the setfacl command. The options that are most commonly used with setfacl are the -m (create or modify an ACL), -s (replace old ACL on a file with a new ACL) and -d (delete an ACL). The symbols r,w,x can be used or octets can be used. In order to set testuser with read and write permissions the following command would be used.

#setfacl -s u::7,g::4,o:0,m:7,u:1006:6 testacl

1006 is testuser's user identification number and can be used instead of the username. The same holds true when setting the permissions for a specific group. In the above command, we replaced the ACL testacl with the new one we just defined. The owner of the file received all permissions. The owner's group received read only permissions. Everyone else received no permissions. The mask set allows users and groups to have a maximum of all permissions. Finally, we gave testuser, identified by user ID, read and write permissions. Using ACL takes practice but can be very useful when tightening permissions and security.

**Automated Security Access Tool**

The Automated Security Access Tool (ASET) is included with Solaris and can assess the current state of the system. It also places the system into one of three security states. The purpose of the tool is to notify the administrator of security problems. The following is a summary of a list from a white paper on the Sun Microsystems homepage. The list provides an example of information that ASET would check for and then notify the administrator with. ASET checks for setuid programs, home directory permissions, file permissions, the size of bin directories and contents of the .rhosts, /etc/passwd and /etc/group files (http://www.sun.com/software/white-papers/wp-security). The three security states that systems can be placed into are low, medium and high.

- Low - Verify file permissions are set to default values and security checks are performed.
- Medium - Some permission on system files are changed to restrict access and additional security checks are performed.
- High - System files are given minimum access and IP forwarding is disabled.

**Security Files**

Three files that are rather important when understanding security in Solaris are the etc/group, etc/passwd and etc/shadow files. These files can be used to determine what groups users are in, group Ids and user Ids.

The etc/group file contains all the groups that exist on a Solaris system. This file contains the group name, the group identification number and the members of the group. The first column is the group name, the second is a place holder that previously was used for a password, the third column is the group ID and the last column are the users that are members of the group separated by commas. Users can be a member of more than one group. A user that is auditing a system can 'more' this file to view what users are in what groups.

```
#more /etc/group
group4::1000:user1,user3,user12,user78
```

The etc/passwd file also provides ways for auditors to identify the group users are in and where the files they own or create are located. User identification numbers can also be determined using this file.

```
user12:x:119:1000:example user:/export/home/user12:/bin/ksh
```

The first column contains the username, the second column is a place holder. The third column is the user identification number, while the fourth column contains the group identification number. The fifth column contains the users real name, the sixth column is the location of the user's home directory and the seventh column contains the user's shell.

The etc/shadow file contains each user's encrypted password. The purpose of this file is to keep passwords isolated from user information. The first column is the username and the second column is the encrypted password. Users with access to these files should be tightly controlled.

user12:x63hjkswe398hj:::::::

There are seven other columns that deal with such things as password aging and expiration. The first two fields are the most critical.

**Conclusion**

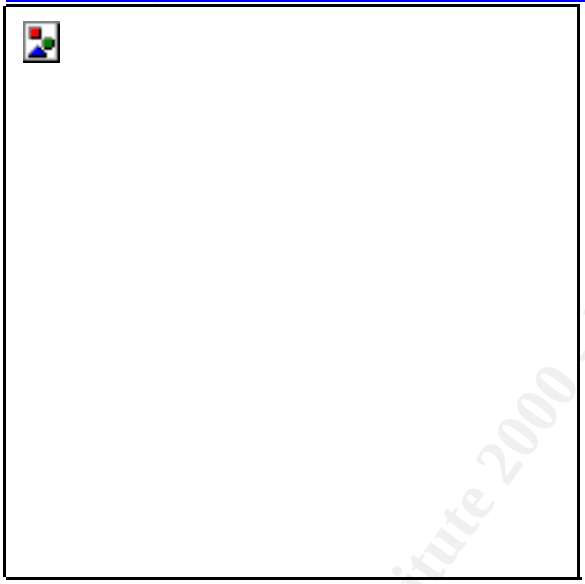
Securing a Unix systems can be very complicated. In this paper, I merely scratched the surface. The only way to feel totally comfortable is to gain hands on experience either from a live system or a lab set-up by your company. Having some basic knowledge will gain the respect of your clients and the respect of your co-workers. Security is ever-changing in Unix. Remember to start with the basics, such as common files that need to be checked, and then move forward with permissions and access control lists. There are many other files and concepts that need to be checked and understood that weren't covered in this paper, but can be obtained over the Internet and through books at your local bookstore.

## References


Flynn, Hal. Role Based Access Control - A distribution of power. Aug 21 2000. http://www.securityfocus.com/frames/?focus=sun&content=/focus/sun/articles/rbac1.html.

Galvin, Peter. Controlling ACLs Down and dirty with the new access control list facility in Solaris 2.5. http://www.unixinsider.com/swol-08-1996/swol-08-security.html.

SecurityFocus Inc. Solaris Access Control Lists. March 6, 2000. www.securityfocus.com/focus/sun/articles/solacls.html.



Software Whitepapers Sun. TM SolarisTM Security. http://www.sun.com/software/white-papers/wp-security/.

Sun Educational Services. Sun Microsystems Student Guide: Solaris 8 Operating Environment System Administration I. Broomfield, Colorado: Sun Microsystems, August 2000.