An Overview of Solaris 10 Operating System Security Controls

Glenn Brunette
Distinguished Engineer
Sun Microsystems, Inc.
http://blogs.sun.com/gbrunett/

September 25, 2007

Table of Contents

Introduction

The purpose of this document is to extend upon the foundation of security recommendations provided by the Solaris 10 Security Benchmark published by the Center for Internet Security (CIS). The CIS Solaris 10 Security Benchmark is written as a (Level 1) "how to" guide for organizations wanting to rapidly harden their Solaris 10 OS configurations. As such, it does not include content or recommendations focused on the security controls found in the Solaris 10 OS that fall outside of the area of system hardening.

This document is intended to complement the CIS Solaris 10 Security Benchmark by providing a more complete overview of the security features and capabilities found in the Solaris 10 OS, including those that were not appropriate for the Level 1 CIS guide. Specific recommendations and references for more detailed information are provided wherever possible.

This document has been created as a companion to the Solaris 10 Security Benchmark for Solaris 10 11/06 and 8/07. While many of the controls discussed in this document were available in earlier versions of Solaris, some of the functionality discussed may not be present in those older versions.

Acknowledgements

The author would like to extend special thanks to the following people for their significant contributions to this document:

- Bart Blanquart
  Security Architect, Sun Microsystems, Inc.
- Glenn Faden
  Distinguished Engineer, Sun Microsystems, Inc.
- Dan McDonald
  Sr. Staff Engineer, Sun Microsystems, Inc.
- Darren Moffat
  Sr. Staff Engineer, Sun Microsystems, Inc.
- Mark Thacker
  Solaris Security Product Manager, Sun Microsystems, Inc.
- Scott Rotondo
  Sr. Staff Engineer, Sun Microsystems, Inc.
- Larry Wake
  Group Manager, Solaris Software, Sun Microsystems, Inc.

In addition, Sun would also like to thank the following organizations for their contributions, support and feedback without which this document would not have been possible:

- Center for Internet Security (CIS), http://www.cisecurity.org/
- U.S. Defense Information Systems Agency (DISA), http://www.disa.mil/
- U.S. National Institute of Standards and Technology (NIST), http://www.nist.gov/
- U.S. National Security Agency (NSA) , http://www.nsa.gov/

Installation Considerations

The purpose of this section is to discuss security issues present prior to and during the initial installation of an instance of the Solaris Operating System system. Careful consideration of these issues is recommended in order to avoid having to reinstall the operating system. Post-installation considerations and recommendations are covered in the section titled Configuration Considerations starting on page 9.

Disk Partitioning

While there is no single recipe for defining disk partitioning and file system layout, there are a number of considerations that should be taken into account in order to produce a configuration that is more likely to meet an organization's specific requirements. For example:

- How will the system and any running applications be impacted if the available space in a file system is exhausted?
- Does a specific application, data set or directory require special treatment? Should it be mounted read-only, for example? Are programs, set-uid or otherwise, permitted to be run from that location?
- Is there a need to ensure that specific users or services use only a specified amount of storage under a given file system?

The list goes on and on. Each of these considerations may require the use of a specific file system layout (e.g., placing applications and data on separate file systems) or mount options (e.g., read-only, do not honor set-id bit, do not honor executable bit, require quota, etc.). No where is this more critical than when working with security relevant directories, data and services.

For example, if a mail server were configured to have a single root file system (that combined `/usr`, `/var`, `/opt`, etc.), then that system could be significantly impacted if incoming messages caused the file system to be full. In addition to not being able to receive new messages, audit records and log messages would not be able to be recorded either. Given the importance of auditing and log information, organizations often require that they be stored on separate partitions. The same case could occur if audit information was on a partition that included a world writable file. In this case, a malicious user could use all of the available space causing audit records to be lost.

Additionally, disk space set aside for users and often applications is kept separate from partitions designated for the operating system. This too is done in order to help ensure that the operating system or application is not impacted should a user or other application consume all of the available space on a file system since it is often not predictable how an application will behave when confronted with this kind of failure.

As a general recommendation, organizations may want to consider making `/var`, `/var/mail` (if system is a mail server), `/var/audit` (if Solaris auditing is configured), `/opt`, and `/export` (if used for applications or home directories) separate file systems in order to mitigate the risk of issues such as those discussed above. As with any recommendation, organizations are strongly encouraged to evaluate their own requirements and situation to determine the applicability of the recommendation before taking action.

In addition to deciding how to partition disks and create file systems, organizations should also consider how those file systems will be mounted and used on their systems. More information on additional capabilities such as quotas and mount options can be found in the section File System Security, starting on page 9.

Software Installation Clusters

A Solaris OS system is installed using one of the Sun provided (and supported) installation mechanisms (e.g., Graphical or Console Installer, JumpStart, etc.) The initial information of the system is based upon one of the Sun provided (and supported) software installation clusters (i.e., metaclusters). The following is the list of available metaclusters in the Solaris 10 OS:

Metacluster	Description
Reduced Networking `SUNWCrnet`	Contains the packages that provide the minimum code that is required to boot and run a Solaris system with limited network service support. The Reduced Network Support Software Group provides a multiuser text-based console and system administration utilities. This software group also enables the system to recognize network interfaces, but does not activate network services.
Core `SUNWCreq`	Contains the packages from the Reduced Networking Software Group plus those that provide the minimum code that is required to boot and run a networked Solaris system.
End User `SUNWCuser`	Contains the packages from the Core Software Group plus those that provide the minimum code that is required to boot and run a networked Solaris system and the Java Desktop System (CDE and GNOME).
Developer `SUNWCprog`	Contains the packages for the End User Software Group plus additional support for software development. The additional software development support includes libraries, include files, man pages, and programming tools. Compilers are not included.
Entire `SUNWCall`	Contains the packages for the Developer Solaris Software Group and additional software that can be useful for some server deployments.
Entire + OEM `SUNWCXall`	Contains the packages for the Entire Solaris Software Group plus additional hardware drivers, including drivers for hardware that is not on the system at the time of installation.

For more information on Solaris software installation clusters, see:

http://docs.sun.com/app/docs/doc/820-2712/6nea09pgt?a=view#fqnnj

It should be mentioned that organizations should choose the software installation cluster that most closely aligns with their specific requirements – noting that individual software packages and clusters can be added or removed as necessary. There are times, however, when organizations want to more substantively customize the installation of the Solaris OS. For this, see the section on Minimization on page 6.

Minimization

The term minimization as applied to operating system installation goes back many years although there has often been some debate over its true meaning. In order to provide a clear discussion of this topic, the following definitions will be provided:

- Reduced Installation. A reduced installation (or reduced software installation) is a Solaris OS configuration created by installing fewer than all of the software packages delivered by the Solaris OS. This includes installations that are based on any software installation cluster that is not `SUNWCXall` as well as any configuration where individual software packages or clusters have been removed.

- Minimal Installation. A minimal installation is a special case of a reduced installation whereby the only software packages that are installed (or remain) are those that directly contribute to the operational and management requirements of the system. All unnecessary software is removed (or not installed in the first place). Minimal installations are highly dependent on the actual hardware and software configuration being used (including any Sun and third party software and devices).

For purposes of comparison, the following chart illustrates the differences between each of the default Solaris installation clusters with respect to size, number of packages as well as number of set-uid and set-gid programs. This information was collected from a system running Solaris 10 8/07:

Metacluster	Size (MB)	# Pkgs	# Set-UID	# Set-GID
Reduced Networking SUNWCrnet	250	119	27	11
Core SUNWCreq	290	176	34	14
End User SUNWCuser	3000	731	66	22
Developer SUNWCprog	3700	978	67	22
Entire SUNWCall	3800	1035	81	23
Entire + OEM SUNWCXall	3800	1114	87	23

Regardless of the actual approach taken, reduced or minimal installations are only supported if they comply with the requirements defined in Sun's INFODOC #86177 (found at http://sunsolve.sun.com/) and in the document, “Rules of Engagement for the Support of Reduced or Minimal Configurations”. A copy of this document can be found at:

http://www.opensolaris.org/os/community/security/files/minimization-support-rules-ext.pdf

In addition, Sun has published a number of articles and documents related to this topic. While many were developed for older versions of the Solaris OS, their approach remains valid even if the specific list of packages or dependencies has changed. For example:

- Sun Blog: Foundation for Minimal Solaris 10 Systems
  http://blogs.sun.com/gbrunett/entry/foundation_for_minimal_solaris_10
- Sun BluePrint: Minimizing Domains for Sun Fire V1280, 6800, 12K and 15K Systems
  Part 1: http://www.sun.com/blueprints/0903/817-3340.pdf
  Part 2: http://www.sun.com/blueprints/0903/817-3628.pdf
- Sun BluePrint: Minimizing the Solaris OE for Security: Updated for the Solaris 9 OE
  http://www.sun.com/blueprints/1102/816-5241.pdf

In addition, an open-source tool called the Solaris Package Companion has been made available by the OpenSolaris Install Community. This tool can be used to ask interesting questions about an operating system distribution including:

- What clusters or packages are contained in a given metacluster?
- What packages are contained in a given cluster?
- What metacluster or cluster contains a given package?
- On what other packages does a given package or cluster depend?
- Which packages depend on a given package?

Answers to these questions are essential for organizations wanting to build reduced or minimal configurations. For more information on and examples showing the Solaris Package Companion tool, see:

http://www.opensolaris.org/os/project/svr4_packaging/package_companion/

Configuration Considerations

Every organization has its own security policies, standards, and requirements. As a result, there is often no single answer as to how a system should be configured. The goal of this section is to amplify the general recommendations made in the CIS Solaris 10 Security Benchmark while at the same time providing an overview of additional security features and capabilities available in the Solaris 10 operating system. Where appropriate, specific recommendations will also be provided.

Non-Executable Stacks

The concept of non-executable stacks has been introduced earlier in the Benchmark in the section called "Enable Stack Protection". The goal of this section is to provide some additional information for those looking to implement this feature on SPARC, Intel and AMD platforms as well as to provide specific guidance for developers of software that may want to enable this feature in their software by default. A three-part series of Sun blog articles has been devoted to this subject matter and can be found at:

Part 1: http://blogs.sun.com/gbrunett/entry/solaris_non_executable_stack_overview
Part 2: http://blogs.sun.com/gbrunett/entry/solaris_non_executable_stack_continued
Part 3: http://blogs.sun.com/gbrunett/entry/solaris_non_executable_stack_concluded

This capability was introduced as a global kernel tunable in Solaris 2.6 and has been used as a workaround to mitigate the risk of quite a few vulnerabilities in the past. That said, this will not prevent all types of buffer overflow exploits. For example, see:

http://seclists.org/bugtraq/1999/Mar/0004.html

Sun takes a defense in depth approach to mitigating these issues including by: architectural (design) and code level reviews of privileged code, the reduction of privileges where possible using Solaris 10 privileges, the use of privilege bracketing, as well as features such as non-executable stacks. Together, these efforts serve to significantly improve the security services and code in the Solaris OS.

File System Security

Unix Permissions

The first line of defense for protecting objects in a file system are the default Unix permissions assigned to each and every file system object. Unix permissions support assigning unique access rights to the owner of the object, a group assigned to the object as well as anyone else. These rights are generally referred to as user, group and world permissions. For more information on these permissions and how they can be set, refer to the `chmod(1)` manual page.

While the assignment of owners, groups and permissions will vary widely based upon the file system object in question, there are a few common recommendations that should be followed including those covered previously in Section 5: File/Directory Permission/Access of the Benchmark. As a general rule:

- Assign the least amount of privilege that is required for services or users accessing the object. This includes read, write and execute access as well as special permissions such as set-uid and set-gid.

That said, it is important the remember that Sun does not support changing the default file ownership, group or permissions of files shipped in the Solaris OS. If there are questions about the validity or security of a given setting, please contact Sun directly to address any concerns.

Access Control Lists (UFS and ZFS)

In addition to Unix permissions, the Solaris OS supports access control lists (ACLs) on both the UFS and ZFS file systems. Access Control Lists offer organizations the ability to more finely control access to individual or groups of file system objects. ACLs differ between the UFS and ZFS implementations so each will be discussed independently.

- Unix File System (UFS) Access Control Lists. Based upon a version of a POSIX draft for ACLs (POSIX 1003.1e/1003.2c which was withdrawn), UFS access control lists include support for twelve different access modes that can be assigned to specific users or groups. Typically, ACLs are used to restrict or grant read, write or execute access to specific users or groups. In addition, ACLs can be used to assign default permissions for objects created under specified directories (for users, groups, or world). Prior to the Solaris 10 OS, UFS ACLs were set using `setfacl` and queried using `getfacl`. While these commands can be used in Solaris 10, many find that using the enhanced `chmod(1)` and `ls(1)` commands provide a simpler more easy to use method for managing ACLs.

  In the following example, the `chmod(1)` command is used to add an ACL to the file `sample-file` granting read and write access to the user `joe`:

```
$ ls -lv sample-file
-rw-r--r--   1 john john       0 Sep 17 21:42 sample-file
     0:user::rw-
     1:group::r--                #effective:r--
     2:mask:r--
     3:other:r--

$ chmod A+user:joe:rw- sample-file

$ ls -lv sample-file
-rw-r--r--+  1 john john       0 Sep 17 21:42 sample-file
     0:user::rw-
     1:user:joe:rw-              #effective:r--
     2:group::r--                #effective:r--
     3:mask:r--
     4:other:r--
```

  It should be noted that the effective rights for user `joe` are still read only. In cases such as this, it is necessary to recalculate the permissions (ignoring the default permissions mask) in order to ensure that sufficient rights are granted to satisfy the ACLs that have been created. To do this use the following command:

```
$ getfacl sample-file | setfacl -r -f - sample-file
```

  After running this command, the user `joe` has both read and write access to the file `sample-file`:

```
$ ls -lv sample-file
-rw-r--r--+  1 john john       0 Sep 17 21:42 sample-file
     0:user::rw-
     1:user:joe:rw-              #effective:rw-
     2:group::r--                #effective:r--
     3:mask:rw-
     4:other:r--
```

- ZFS Access Control Lists. Based upon the NFSv4 access control list model, ZFS ACLs are set using `chmod(1)`, queried using `ls(1)` and support over fifteen different access modes that can be assigned to specific users or groups. A few such modes not available with UFS ACLs include: `add_file`,

write_xattr, delete, and write_acl. Each of these is described in more detail in chmod(1). Collectively, these modes enable organizations to control access to files and directories with a high degree of precision.

In the following example, a ZFS ACL is used to enforce a policy stating that files created under the archive directory are not permitted to be removed (even by the owner of the file):

```
$ id
uid=101(gbrunett) gid=101(gbrunett)

$ mkdir archive

$ chmod A+everyone@:delete_child/delete:file_inherit/dir_inherit:deny \
  ./archive

$ ls -ldv ./archive
drwxr-xr-x+  2 gbrunett gbrunett       2 Sep 18 22:52 ./archive
     0:everyone@:delete_child/delete:file_inherit/dir_inherit:deny
     1:owner@::deny
     2:owner@:list_directory/read_data/add_file/write_data/add_subdirectory

         /append_data/write_xattr/execute/write_attributes/write_acl
         /write_owner:allow
     3:group@:add_file/write_data/add_subdirectory/append_data:deny
     4:group@:list_directory/read_data/execute:allow
     5:everyone@:add_file/write_data/add_subdirectory/append_data
         /write_xattr /write_attributes/write_acl/write_owner:deny
     6:everyone@:list_directory/read_data/read_xattr/execute
         /read_attributes /read_acl/synchronize:allow

$ cp /etc/motd ./archive/

$ ls -l archive/
total 2
-rw-r--r--+  1 gbrunett gbrunett      53 Sep 18 22:53 motd

$ rm archive/motd
rm: archive/motd not removed: Permission denied
```

This should be considered a representative example only. While the file itself may not be deleted in this example, the content contained in this file can be altered as in:

```
$ echo "Hello World" > archive/motd

$ cat archive/motd
Hello World
```

In addition to the actual ACL modes, ZFS also provides two ACL-related parameters that can be assigned to file systems: aclmode and aclinherit. Respectfully, these parameters define how ACLs can be modified by chmod(2) operations and how ACL entries are inherited when files and directories are created. More information on these settings and their possible values can be found in zfs(1M). These settings can be queried using the zfs(1M) command:

```
$ zfs get aclmode,aclinherit pool/saved
NAME        PROPERTY    VALUE        SOURCE
pool/test   aclmode     groupmask    default
pool/test   aclinherit  secure       default
```

The recommendations from the section on Unix Permissions apply to access control lists as well. For more information and examples on setting or querying access control lists, see `chmod(1)`. Further, a mapping of NFSv4 and POSIX draft ACLs can be found at: http://tools.ietf.org/id/draft-ietf-nfsv4-acl-mapping-03.txt.

Mount Options

Included in the table below are a few useful file system mount options that organizations may want to use for further restricting how their file systems can be used. It is strongly recommended to exercise caution when selecting individual mount options as these restrictions may cause programs relying on the disabled functionality to fail.

Mount Option	Description
`nodevices`	Disallow the opening of device-special files. This option should be used on all file systems that do not contain (or need to support) device special files.
`noexec`	Disallow the execution of programs from the file system. This option should be used on all file systems that do not contain (or need to support) executable files or memory mapped executable data. Note that the use of this option will not prevent the execution of interpreted scripting code (e.g., Bourne shell, Perl, etc.) when the interpreter itself exists on a file system where execution is permitted.
`nosetuid`	Disallow the execution of programs in the file system that have the set-uid bit set. This option is highly recommended whenever the file system is shared by way of NFS using the `root=` option. Further, this option should be used on all file systems that do not contain (or need to support) set-uid execution.
`nosub`	Disallow clients from mounting sub-directories of shared directories.
`nosuid`	This is a convenience option combining both `nodevices` and `nosetuid`.
`noxattr`	`ZFS` or `tmpfs` only. Disallow the use of extended file attributes on the file system. Solaris does not use extended file attributes by default. This option can be used on all file systems that do not contain (or need to support) extended file attributes.
`ro`	Disallow write access to the file system, making it read-only.
`size`	`tmpfs` only. Define the maximum size of the file system. This option can be used to force the size of the file system to be less than the amount of virtual memory on the system.

For more information on each of these options including examples of how they can be used, see: `mount(1M)`, `mount_ufs(1M)`, `mount_tmpfs(1M)`, and `zfs(1M)` respectively. In addition, for more information on the potential dangers of extended file attributes, see:

Hiding within the Trees
http://www.usenix.org/publications/login/2004-02/pdfs/brunette.pdf

Quotas and Reservations

While both UFS and ZFS support the notion of file system quotas to control how much available storage space can be used on a given file system, their implementations differ. Given this, each is discussed separately:

- UFS Quotas. Quotas on a UFS file system specify the number of blocks and inodes that can be consumed by a user on a file system. The number of blocks and inodes is often specified with both "soft" and "hard" values. Soft limits may be temporarily exceeded and the user will be warned about the violation. Hard limits are those that cannot be exceeded and attempts to use more blocks or inodes beyond a user's hard limit will be denied. For more information on UFS quotas, see: `quota(1M)`, `edquota(1M)`, `repquota(1M)`, `quotacheck(1M)` and `quotaon(1M)`.

  In the following example, a quota is set on the `/export/home` file system for the user `gmb`:

  [manually configure `/export/home` to use quotas by editing `/etc/vfstab` as follows]

```
# grep "/export/home" /etc/vfstab
/dev/dsk/c0d0s7 /dev/rdsk/c0d0s7 /export/home ufs     2       yes     rq

# mount -o remount,quota /export/home

# cd /export/home

# touch quotas

# chmod root:root quotas

# edquota gmb
```

  [manually enter the quota information for `gmb` using form displayed using `/usr/bin/vi`]

```
# quotaon -v /export/home
/export/home: quotas turned on

# quota -v gmb
Disk quotas for gmb (uid 102):
Filesystem     usage  quota  limit    timeleft  files  quota  limit ...
/export/home       0      0   5000                  0      0   2000
```

  After these steps have been completed, the user `gmb` will receive an error message if the defined quota has been exceeded as in the following example:

```
$ mkfile 100m gmb
quota_ufs: over hard disk limit (pid 5618, uid 102, inum 25822, fs
/export/home)
gmb: initialized 5079040 of 104857600 bytes: Disc quota exceeded
```

  In addition, administrators can use the repquota(1M) command to get a report showing usage for file systems with quotas enabled:

```
$ repquota /export/home
                     Block limits                       File limits
User           used   soft   hard    timeleft    used   soft   hard ...
gmb       --   4992      0   5000                   4      0   2000
```

- ZFS Quotas. Quotas are managed differently on ZFS file systems. They are set on a file system or volume and are not directly tied to specific users. ZFS quotas are defined and queried using the `zfs(1M)` command as in the following example:

```
# zfs set quota=1G pool/zones

# zfs get quota pool/zones
NAME        PROPERTY  VALUE       SOURCE
pool/zones  quota     1G          local
```

A ZFS quota can also be reported using the following command:

```
# zfs list -o name,mountpoint,quota,used
NAME            MOUNTPOINT       QUOTA   USED
pool            /pool             none  4.36G
pool/ws         /pool/ws          none  1.72G
pool/zones      /pool/zones         1G   157M
```

Per-user ZFS file systems can be used if an organization needs to enforce quotas on a per-user basis. In the following example, two user home directories are created on a ZFS pool that is then mounted as `/export/home`. Once completed, the home directories will exist under `/export/home` and each will have its own quota assigned.

```
# zfs create -p -o quota=500M pool/home/gbrunett

# zfs create -p -o quota=200M pool/home/gmb

# zfs set mountpoint=/export/home pool/home

# zfs list -ro name,mountpoint,quota,used pool/home
NAME                MOUNTPOINT             QUOTA   USED
pool/home           /export/home            none    57K
pool/home/gbrunett  /export/home/gbrunett   500M    18K
pool/home/gmb       /export/home/gmb        200M    18K
```

In addition to quotas, ZFS also supports the notion of reservations. Reservations set aside an allotted amount of storage capacity whereas quotas typically enforce a maximum level of utilization. ZFS reservations are also set using the `zfs(1M)` command as in the following example:

```
# zfs set reservation=512M pool/zones

# zfs get reservation pool/zones
NAME        PROPERTY     VALUE        SOURCE
pool/zones  reservation  512M         local
```

The impact of this reservation can be best seen in the before and after views of the storage pool itself:

```
# zfs list pool
NAME   USED  AVAIL  REFER  MOUNTPOINT
pool  4.15G  7.36G    23K  /pool          (Before)
pool  4.65G  6.86G    23K  /pool          (After)
```

For more information, see:

- Sun BigAdmin: ZFS, Sun's Cutting-Edge File System
  http://www.sun.com/bigadmin/features/articles/zfs_part1.scalable.html
- Sun Blog: ZFS End to End Data Integrity
  http://blogs.sun.com/bonwick/entry/zfs_end_to_end_data

Universal Serial Bus (USB) Support

In some cases, it may be useful or necessary to disable USB support on a system either for all devices or for some specific classes of devices. This is most often done, in concert with restrictions placed on volume management, to help prevent infiltration of malicious or otherwise unauthorized content as well as to limit the potential for exfiltration of sensitive information.

To disable all USB support on a system, each of the USB related packages should be removed (or just not installed in the first place.) This approach will cause a system to no longer recognize USB devices. Care should be taken with this approach especially on systems with USB keyboards or mice. This approach should only be used in cases where no USB support is needed or expected. For Solaris 10 11/06, the list of packages includes those defined in the `SUNWCusb` package cluster:

```
SUNWlibusbugen          SUN libusb ugen plugin
SUNWuedg                USB Digi Edgeport serial driver
SUNWugen                USB Generic Driver
SUNWugenu               UGEN Headers
SUNWuksp                USB Keyspan serial driver
SUNWukspfw              USA49WLC firmware for USB Keyspan serial driver
SUNWuprl                Prolific PL2303 USB-to-serial driver
SUNWusb                 USB Device Drivers
SUNWusbs                USB generic serial module
SUNWusbu                USB Headers
```

As package names and contents may change between releases of the Solaris OS, it is recommended that you verify the list against the actual release of the operating system being used. This list can be easily generated using the Solaris Package Companion tool discussed in the section on Minimization starting on page 6. A similar approach can be used to remove device driver support from Solaris for other mechanisms such as IEEE 1394 (Firewire) and Infiniband.

In addition, it is possible to disable USB support for specific classes of devices. For more information, see:

> Solaris 9 System Administrator Collection: How to Disable Specific USB Drivers
> http://docs.sun.com/app/docs/doc/817-3814/6mjcp0qrj?a=view

In order to best leverage this approach, refer to the `usba(7D)` manual page for a mapping of USB client driver names to descriptions. For example, if an organization wanted to disable support for USB printers, the `usbprn` driver class would be used:

> [Before starting, be sure to save a backup copy of the `/etc/driver_aliases` file.]
>
> ```
> # grep usbprn /etc/driver_aliases
> usbprn "usbif,class7.1"
>
> # update_drv -d -i '"usbif,class7.1"' usbprn
> ```
>
> [The system must be restarted to complete this operation.]

A tool called `usbsecure.pl` has been made freely available at the Sun Developer Network to help automate the process of locking down USB devices:

> http://developers.sun.com/solaris/driverdev/reference/codesamples/usb_security/index.html

Pluggable Authentication Modules (PAM)

Originally developed by Sun in 1995 and first exposed in Solaris 2.6, the Pluggable Authentication Module (PAM) framework provides organizations with the ability to customize the user authentication experience as well as account, session and password management functionality in the Solaris OS. `login` and other system-entry services use the PAM architecture to ensure that all entry points for the system have been secured. This architecture enables the replacement or modification of authentication modules in the field to secure the system against any newly found weaknesses without requiring changes to any of the system services (that use the PAM architecture).

By default, the PAM policy (as defined in `/etc/pam.conf`) is configured to use password-based authentication although support for Kerberos V5 authentication can be configured if required. Sample configurations for Kerberos can be found in `pam_krb5(5)`. `rhosts` authentication is also configured using PAM and is covered earlier in the Benchmark in the section titled, "Disable `.rhosts` Support in `/etc/pam.conf`". In addition to authentication, PAM also includes modules which provide support for functionality such as password complexity checks, password history, account lockout and much more. These controls are discussed in more detail in the Benchmark in the section titled "Set Strong Password Creation Policies" and "Set Retry Limit for Account Lockout".

For more information on the Pluggable Authentication Modules functionality in the Solaris OS, see:

- OpenSolaris Community Project: Pluggable Authentication Modules (PAM)
  http://www.opensolaris.org/os/community/security/projects/pam/
- Sun Developer Network: User Authentication in the Solaris OS
  Part 1: http://developers.sun.com/solaris/articles/user_auth_solaris1.html
  Part 2: http://developers.sun.com/solaris/articles/user_auth_solaris2.html
- Sun BluePrint: Extending Authentication in the Solaris 9 OS Using PAM
  Part 1: http://www.sun.com/blueprints/0902/816-7669-10.pdf
  Part 2: http://www.sun.com/blueprints/1002/816-7670-10.pdf
- Solaris 10 System Administrator Collection: Authentication Services and Secure Communication
  http://docs.sun.com/app/docs/doc/816-4557/6maosrjio?a=view

Password Security

The Solaris 10 OS has a number of features that can be used to promote strong user passwords and help defend against attacks involving brute force guessing. Many of these features have been discussed earlier in the Benchmark in sections:

- "Set Strong Password Creation Policies". This section covers the minimum password length, required password composition (e.g., alphabetic, numeric, special characters, etc.), maximum number of consecutive repeating characters, whether passwords may contain white space or be a rotation of the user's name, or found in a banned word list. This section also covers how soon a password can be reused (i.e., password history).
- "Set Password Expiration Parameters on Active Accounts". This section defines for how long a password may remain valid before it must be changed and how soon after a password change that a user can initiate a subsequent password change.
- "Verify Delay between Failed Login Attempts Set to 4". This section defines the time interval that must pass after a failed authentication attempt before the user can be prompted again to authenticate to the system.
- "Set Retry Limit for Account Lockout". This section enables account lockout and sets the maximum number of consecutive failed authentication attempts before an account is locked.

Pluggable Crypt

In addition to these actions, organizations may want to consider altering the default algorithm used to store user passwords. By default, the Solaris OS uses the traditional UNIX crypt algorithm. This algorithm limits passwords to a maximum of eight characters and is not considered sufficiently secure for current systems. This algorithm is enabled primarily for backwards compatibility. Rather, Sun recommends that organizations consider using one of the following algorithms that are also provided by default in the Solaris 10 OS:

- Sun MD5. The `crypt_sunmd5` module is a one-way password hashing module for use with `crypt(3C)` that uses the MD5 message hash algorithm. The algorithm identifier for `crypt.conf(4)` and `policy.conf(4)` is `md5`. This module is designed to make it difficult to crack passwords that use brute force attacks based on high speed MD5 implementations that use code inlining, unrolled loops, and table lookups. The maximum password length for `crypt_sunmd5` is 255 characters. This module also includes a `rounds` parameter that specifies the number of additional rounds of MD5 to use in generation of the salt; the default number of rounds is 4096. For more information on this module, see `crypt_sundm5(5)`.

- BSD MD5. The `crypt_bsdmd5` module is a one-way password hashing module for use with `crypt(3C)` that uses the MD5 message hash algorithm. The algorithm identifier for `crypt.conf(4)` and `policy.conf(4)` is `1`. The output is compatible with `md5crypt` on BSD and Linux systems. The maximum password length for `crypt_bsdmd5` is 255 characters. For more information on this module, see `crypt_bsdmd5(5)`.

- Blowfish. The `crypt_bsdbf` module is a one-way password hashing module for use with `crypt(3C)` that uses the Blowfish cryptographic algorithm. The algorithm identifier for `crypt.conf(4)` and `policy.conf(4)` is `2a`. The maximum password length for `crypt_bsdbf` is 255 characters. For more information on this module, see `crypt_bsdbf(5)`.

To enable one of these alternate algorithms, the `/etc/security/policy.conf` file is modified. In particular, the `policy.conf` file contains three parameters related to this functionality. They include:

- `CRYPT_ALGORITHMS_ALLOW`. This parameter specifies the algorithms that are allowed to be used for new passwords.

- `CRYPT_ALGORITHMS_DEPRECATE`. This parameter will deprecate the use of specific algorithms.

- `CRYPT_DEFAULT`. This parameter defines the default algorithm to be used.

For example, to set Sun MD5 as the default password algorithm, the following change should be made in the `/etc/security/policy.conf` file:

```
CRYPT_DEFAULT=md5
```

If an organization wants to deprecate the use of the traditional `crypt` algorithm at the same time, this additional change should be made:

```
CRYPT_ALGORITHMS_DEPRECATE=__unix__
```

If an organization wanted to increase the number of rounds completed by the Sun MD5 module from 4000 to 8000, the following line would need to exist in `/etc/security/crypt.conf`:

```
md5     crypt_sunmd5.so.1       rounds=8000
```


Note that simply making these changes will not change existing user accounts. These values are inspected and used only during a password change event. So, if these values are changed, they will not take effect for a user until that user changes his or her password. To force a user to change his or her password at next login, use the command:

```
# passwd -f <user_name>
```

Organizations should carefully consider their existing environment before implementing this change. The MD5 and Blowfish algorithms are not available on older releases of the Solaris OS (prior to Solaris 9 02/02) and some Solaris services will not properly authenticate users whose passwords are computed using these algorithms. An example of one such service is the Solaris Management Console which is covered by the following bug report:

> 4760846 smc and the enhanced crypt(3c) seem to be incompatible

Further, environments leveraging a networked naming service (e.g., NIS, NIS+, LDAP) should also determine if there exists any interoperability issues with any of their non-Solaris 10 clients.

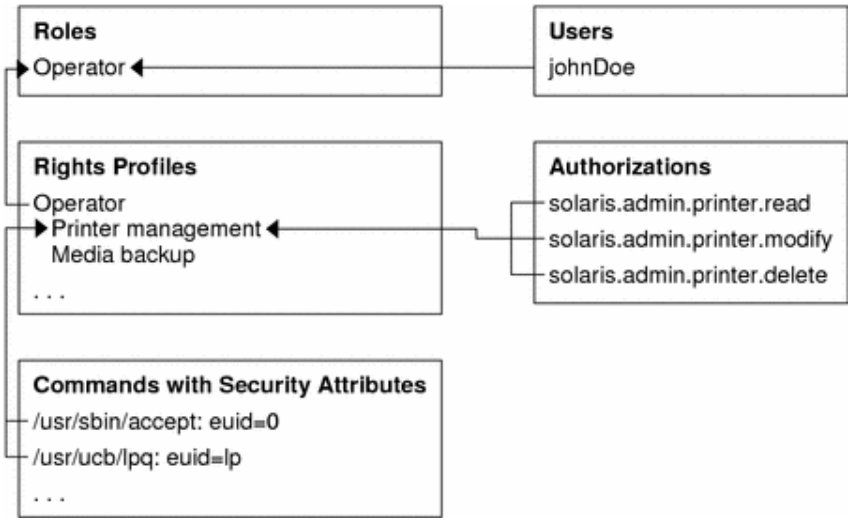
For more information on the pluggable `crypt` functionality, see:

- Solaris 10 System Administrator Collection: Changing the Default Algorithm for Password Encryption
  http://docs.sun.com/app/docs/doc/816-4557/6maosrjc9?a=view
- Solaris Password Security
  http://www.cuddletech.com/blog/pivot/entry.php?id=778
- OpenSolaris, Pluggable Crypt, and the Sun MD5 Password Hash Algorithm
  http://www.crypticide.com/dropsafe/article/1389
- Directory Server 6.1 and Unix Crypt
  http://blogs.sun.com/Ludo/entry/directory_server_6_1_and

Role-based Access Control (RBAC)

Solaris Role-based Access Control (RBAC) is an alternative to the all-or-nothing superuser model. RBAC is in keeping with the security principle of least privilege by allowing organizations to selectively grant privileges to users or roles based upon their unique needs and requirements. In general, organizations are strongly encouraged to use Solaris RBAC to restrict access to privileged operations rather than granting users complete access to the `root` account.

Solaris RBAC was introduced in the Solaris 8 operating system and has been enhanced and expanded with each new release of Solaris. Solaris RBAC functionality contains several discrete elements that can be used individually or together including authorizations, rights profiles and role designations. The relationship between these elements and descriptions of each is included below:



Authorizations

An authorization is a permission that can be assigned to a role or user (or embedded in a rights profile) for performing a class of actions that are otherwise prohibited by security policy. Very often, authorizations are used in concert with privileged programs or services for the purpose of access control. For example, consider this reference from `crontab(1)`:

> Access to `crontab` is denied:
>
> - if `/etc/cron.d/cron.allow` exists and the user's name is not in it.
> - if `/etc/cron.d/cron.allow` does not exist and user's name is in `/etc/cron.d/cron.deny`.
> - if neither file exists, only a user with the `solaris.jobs.user` authorization is allowed to submit a job.

In this case, the `solaris.jobs.user` authorization can be used to grant access to the `cron` facility (when other access control mechanisms are not present). Authorizations are defined in the `/etc/security/auth_attr` (or an equivalent naming service table). Similarly, authorizations are assigned to users and roles in the `/etc/user_attr` file (or an equivalent naming service table) and queried using the `auths(1)` command. Other examples illustrating the use of authorizations are covered below in the section titled Service Management Facility (SMF) on page 25.

Rights Profiles

A rights profile is a collection of overrides that can be assigned to a role or user. A rights profile can consist of authorizations, individual commands and other rights profiles. Each of the commands stored in a rights profile can define security attributes that determine how the program will be run. The following is the list of security attributes that can be assigned to commands in a rights profile:

- `uid (euid)`. The `euid` and `uid` attributes contain a single user name or a numeric user ID. Commands designated with `euid` run with the effective UID indicated, which is similar to setting the `setuid` bit on an executable file. Commands designated with `uid` run with both the real and effective UIDs.

- gid (egid). The `egid` and `gid` attributes contain a single group name or a numeric group ID. Commands designated with `egid` run with the effective GID indicated, which is similar to setting the `setgid` bit on a file. Commands designated with `gid` run with both the real and effective GIDs.
- privs. The `privs` attribute contains a privilege set which will be added to the inheritable set prior to running the command. A listing of available privileges can be found in `privileges(5)`.
- limitprivs. The `limitprivs` attribute contains a privilege set which will be assigned to the limit set prior to running the command. A listing of available privileges can be found in `privileges(5)`.

To determine which rights profiles have been assigned to a given user or role, use the `profiles(1)` command:

```
# profiles lp
Printer Management
Basic Solaris User
All
```

To further determine exactly which commands are included in these rights profiles the `-l` option to the `profiles(1)` command can be used:

```
# profiles -l lp

      Printer Management:
          /usr/lib/lp/local/lpadmin    uid=lp, gid=lp
          /usr/sbin/lpfilter    euid=lp, uid=lp
          /usr/sbin/lpforms    euid=lp
          /usr/sbin/lpusers    euid=lp
          /usr/sbin/ppdmgr    euid=0

      Basic Solaris User:
          /usr/bin/cdda2wav.bin
                  privs=file_dac_read,sys_devices,proc_priocntl,
                        net_privaddr
          /usr/bin/cdrecord.bin
                  privs=file_dac_read,sys_devices,proc_lock_memory,
                        proc_priocntl,net_privaddr
          /usr/bin/readcd.bin
                        privs=file_dac_read,sys_devices,net_privaddr

      All:
          *
```

Users and Roles

A Solaris role is a special identity for running privileged applications that can be assumed by assigned users only. A role is similar to a normal user in that it has its own UID, GID, home directory, shell and password. A role differs from a normal user in two ways:

- a role cannot be used to (initially) log directly into a system either at the console or by any remote access service. Users must first log into the system before assuming a role using either `su(1M)` or `smc(1M)`.
- A role can only be accessed by a user who has previously been authorized to assume that role. Further, roles are not permitted to assume other roles.

Most often, roles are used for administrative accounts to restrict access to sensitive operations as well as for service accounts (e.g., web server or application server UID) since it is important to ensure that actions taken by such accounts be attributable back to a specific user (who accessed the role). It should also be noted that delayed jobs (e.g., `cron` or `batch`) are independent of role assumption.

To determine if a given account is a role, query the `/etc/user_attr` file (or naming service table) and check for the `type=` parameter. If this parameter is set to `role`, then the account is a Solaris role:

```
$ grep "^root:" /etc/user_attr
root::::type=role;auths=solaris.*,solaris.grant;profiles=Web Console
Management,All;lock_after_retries=no;clearance=admin_high;
min_label=admin_low
```

To determine what roles are assigned to a given user, use the `roles(1)` command:

```
$ roles gbrunett
root
```

Converting the `root` Account to a Role

Taken together, authorizations, rights profiles and roles offer organizations the ability to delegate access to administrative functions with a level of detail that can be customized based upon the organization's policies and requirements. One of the most often cited examples of RBAC is the conversion of the `root` account to a role.

By implementing this change, `root` no longer will be able to directly log into the system, and `root` will only be able to be accessed by those possessing the correct credentials and explicit approval to assume that role. It is critical therefore that at least one user account be assigned to the `root` role – otherwise the role itself would no longer be able to be accessed. Note that the risk of administrators being unable to log in and assume the `root` role to perform privileged operations can be reduced by ensuring that their accounts have account lockout disabled, are stored in the local ("files") password tables, and have home directories that are mounted locally rather than over NFS. In addition, booting the Solaris system into single user mode will enable administrators to log into the system directly as `root`, thereby providing a worst-case mechanism to access a privileged shell.

To convert the `root` account to a role, take the following actions:

```
# usermod -K type=role root

# usermod -R root joe
```

The first command is responsible for converting `root` to a role. The second command assigns the user `joe` to the `root` role. This can be verified by looking at their respective entries in `/etc/user_attr`:

```
# egrep "^root|^joe" /etc/user_attr
root::::type=role;[...omitted for brevity...]
joe::::type=normal;roles=root
```

In addition, there are a number of other rights profiles provided in the Solaris OS by default including:

- Primary Administrator. Provides all of the capabilities of "superuser" in one profile. This profile grants rights that are equivalent to `root`.
- System Administrator. Provides a profile that can do most of the "superuser" tasks but fewer connected with security administration. For example, this role can create accounts but it cannot set or reset user passwords.

- Operator. Provides limited capabilities to manage files and offline media.

A complete listing of profiles can be found in `smc(1M)` or in the `/etc/security/prof_attr` file (or naming service table). For more information on each of these profiles, see:

http://docs.sun.com/app/docs/doc/816-4557/6maosrjga

For more information on Solaris RBAC, see:

- OpenSolaris Community Project: Role-based Access Control (RBAC)
  http://www.opensolaris.org/os/community/security/projects/rbac/
- Roles, Rights Profiles and Privileges
  http://docs.sun.com/app/docs/doc/816-4557/prbactm-1?a=view
- Sun Developer Network: Authorization Infrastructure in Solaris
  http://developers.sun.com/solaris/articles/ais.html
- Sun Blog: Using Solaris RBAC to only allow scp / sftp
  http://blogs.sun.com/darren/entry/using_solaris_rbac_to_only
- Sun Blog: 5 Cent Tour of Solaris RBAC (Video)
  http://blogs.sun.com/security/entry/slotd_the_5_cent_tour
- Sun Blog: Guide Book for Solaris Role-based Access Control
  http://blogs.sun.com/security/entry/spotd_the_guide_book_to
- Sun BluePrint: Enforcing the Two-Person Rule using RBAC in the Solaris 10 OS
  http://www.sun.com/blueprints/0805/819-3164.pdf

Process Rights Management (Privileges)

In the Solaris 10 OS, the traditional concept of a Unix super-user was eliminated. It was replaced with a fine-grained approach to privilege delegation that enables organizations to limit what privileges are granted to services and processes running on their systems. Traditionally, Unix-based systems have relied on the concept of a super-user called `root` (that was required to have a special user identifier – `0`). In the Solaris 10 OS, this dependency has been in large part removed. It has been replaced with the ability to grant one or more specific privileges that enable processes to perform otherwise restricted operations.

Privileges Overview

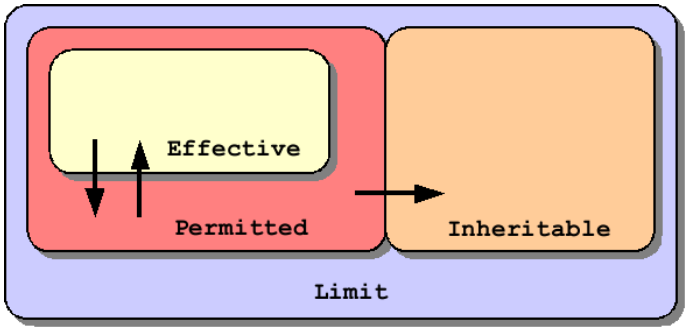
The single, all-powerful UID 0 has been replaced with over 60 discrete privileges that can individually assigned to processes using the Service Management Facility (SMF), Role-based Access Control (RBAC), or using the command-line program, `ppriv(1)`. More information on the use of privileges with SMF and RBAC are covered in their respective sections. To obtain a list of privileges as well as their individual uses on the system, see `privileges(5)` or the output of the command "`ppriv -lv`". For example:

```
# ppriv -vl file_dac_read
file_dac_read
        Allows a process to read a file or directory whose permission
        bits or ACL do not allow the process read permission.
```

Solaris supports the notion of four distinct privilege sets:

- Effective. The set of privileges that are current active (in effect) in the process.
- Permitted. The maximum set of privileges available to the process.
- Inherited. The set of privileges that will be inherited by a child process upon `exec(2)`.
- Limit. The upper bound of all privileges that a process or its children can obtain.

A graphical representation for the relationship between privilege sets is included below:



For more information on these privileges and privilege sets, see:

- `privileges(5)`
- Sun Blog: Solaris Privileges
  http://blogs.sun.com/casper/date/20040722
- Process Rights Management Tutorial
  http://iforce.sun.com/protected/solaris10/adoptionkit/tech/least/tutorial.html

Privilege Bracketing

The implementation of Solaris privileges empowers application developers to control how privileges are used within their programs. Using a technique called privilege bracketing, developers can write their programs such that they are only running with privileges when they are needed. Even more importantly, programs can not only enable or disable their privileges, but they can also drop any privileges granted to them (assuming they will not be needed) and even relinquish them (so they can no longer be used) when there is no longer a need for the privilege. Just as importantly, programs can also restrict which of their privileges can be passed along to their children (e.g., programs that they execute themselves).

It is worth noting that there are many set-uid programs and system services that use this technique including (but not limited to):

- set-uid programs: `/usr/bin/rmformat`, `/usr/lib/fs/ufs/quota`, `/usr/sbin/ping`, `/usr/sbin/traceroute`, `/usr/lib/fs/ufs/ufsdump`, `/usr/bin/rsh`, etc.
- system services: `/usr/sbin/in.ftpd`, `/usr/sbin/rpcbind`, `/usr/lib/nfs/nfsd`, `/usr/lib/rmvolmgr`, `/usr/sbin/nis_cachemgr`, `/usr/lib/nfs/mountd`, etc.

The use of privileges by each of these programs can be seen and verified by looking at the source code for these programs using the OpenSolaris Source Code Browser:

http://src.opensolaris.org/source/

For more information on privilege bracketing, see:

- Sun BluePrint: Privilege Bracketing in the Solaris OS
  http://www.sun.com/blueprints/0406/819-6320.pdf
- Sun Developer Network: Programming in the Solaris OS with Privileges
  http://developers.sun.com/solaris/articles/program_privileges.html

Privilege Debugging

There are often times when organizations may not already know what privileges are needed by an application. This can be a result of unfamiliarity with Solaris privileges or even the lack of source code for the application in question. In such circumstances, an OpenSolaris tool called `privdebug.pl` can be used to profile an application's use of privileges. The `privdebug.pl` command, when run from a Solaris 10 global zone, will enable organizations to see exactly which privileges were being used or attempted.

For more information on `privdebug.pl` or privilege debugging in general, see:

- OpenSolaris Community Project: Privilege Debugging
  http://www.opensolaris.org/os/community/security/projects/privdebug/
- Sun BluePrint: Privilege Debugging in the Solaris OS
  http://www.sun.com/blueprints/0206/819-5507.pdf

Service Management Facility (SMF)

In the Solaris 10 OS, the Service Management Facility is used to add, remove, configure, and manage services in the Solaris OS. It is important therefore that SMF provide organizations with a way to control access to services as well as a means to control how those services are started, stopped, etc.

Access Control

The Service Management Facility leverages the use of the Solaris Role-based Access Control (RBAC) facility in order to control access to service management functions on the system. In particular, SMF uses Authorizations to determine who may manage a service and what functions that person may perform. Many of the services shipped by default in the Solaris have authorizations defined. For example:

```
$ svcprop -p general/action_authorization dns/server
solaris.smf.manage.bind
```

SMF supports three primary types of authorizations:

- `action_authorization`. This authorization allows a user or role to manipulate the state of a service (e.g., restart, temporarily disable or enable, set to maintenance mode, etc.)
- `modify_authorization`. This authorization allows a user or role to add, modify or remove any property associated with a given service. This is by far the most powerful authorization and should be granted with care.

- `value_authorization`. This authorization allows a user or role to manipulate existing service properties. Unlike the `modify_authorization`, this authorization does not grant the ability to add or remove properties. Only the values of existing properties can be changed.

For more information on the access control capabilities of SMF, see:

- `smf_security(5)`
- Sun BluePrint: Restricting Service Administration in the Solaris 10 OS http://www.sun.com/blueprints/0605/819-2887.pdf
- Sun Blog: Securing MySQL using SMF http://blogs.sun.com/bobn/entry/securing_mysql_using_smf_the
- Center for Internet Security BIND Security Benchmark http://www.cisecurity.org/bench_bind.html

Execution Contexts

In addition to access control, SMF also enables organizations to more carefully define the context under which programs are run. Unlike previous service management models in the Solaris OS, SMF enables organizations to specifically define how services are to be started, stopped, restarted, etc. For each of the service methods, organizations can define which user, group, or even privileges will be assigned when a service method is invoked. The full list of method context parameters can be found in `smf_method(5)`, but a few key security-relevant items include:

- `use_profile`. A boolean that specifies whether the profile should be used instead of the `user`, `group`, `privileges`, and `limit_privileges` properties.
- `environment`. Environment variables to insert into the environment of the method, in the form of a number of `NAME=value` strings.
- `profile`. The name of an RBAC (role-based access control) profile which, along with the method executable, identifies an entry in `exec_attr(4)`.
- `user`. The user ID in numeric or text form.
- `group`. The group ID in numeric or text form.
- `supp_groups`. An optional string that specifies the supplemental group memberships by ID, in numeric or text form.
- `privileges`. An optional string specifying the privilege set as defined in `privileges(5)`.

The execution context assigned to a service can be queried using `svcprop(1)`, as in the following example:

```
$ svcprop -p start dns/server
start/exec astring /lib/svc/method/dns-server\ %m\ %i
start/group astring root
start/limit_privileges astring :default
start/privileges astring
basic,!proc_session,!proc_info,!file_link_any,net_privaddr,file_dac_read,
file_dac_search,sys_resource,proc_chroot
start/project astring :default
start/resource_pool astring :default
start/supp_groups astring :default
```



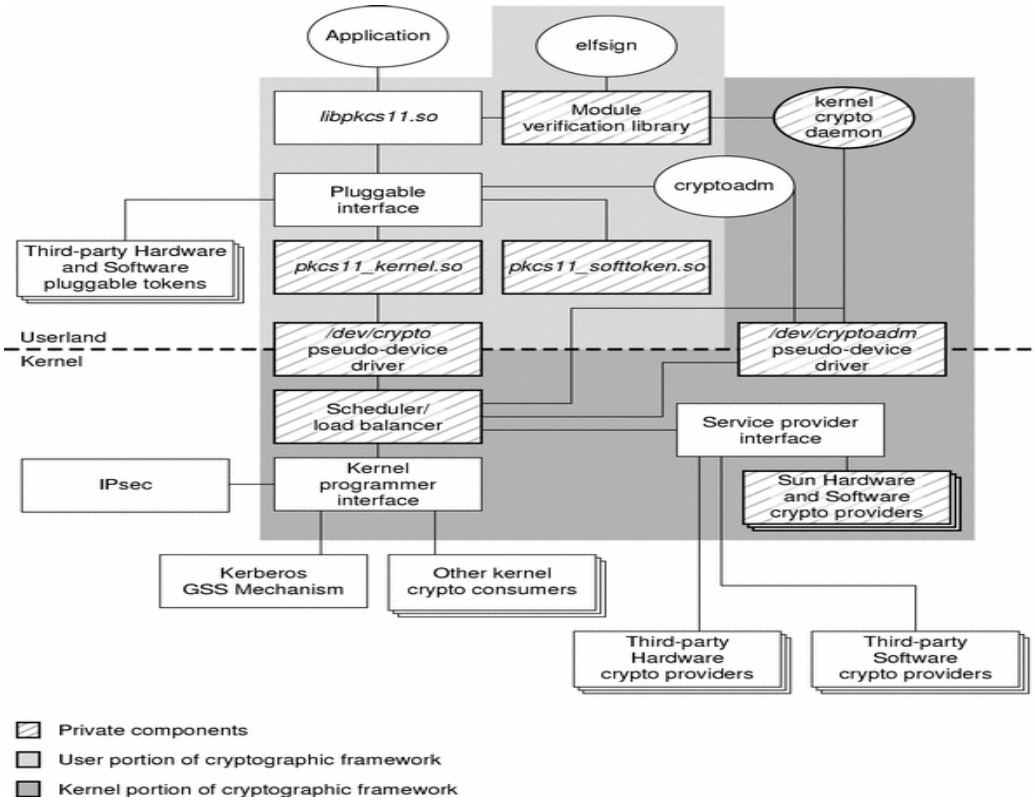
```
start/timeout_seconds count 60
start/type astring method
start/use_profile boolean false
start/user astring root
start/working_directory astring :default
```

For more information on SMF execution contexts, see:

- `smf_method(5)`
- Sun BluePrint: Limiting Service Privileges in the Solaris 10 OS
  http://www.sun.com/blueprints/0505/819-2680.pdf
- Sun Blog: Securing MySQL using SMF
  http://blogs.sun.com/bobn/entry/securing_mysql_using_smf_the
- Center for Internet Security BIND Security Benchmark
  http://www.cisecurity.org/bench_bind.html

Cryptographic Services Management

The Solaris Cryptographic Framework provides cryptographic services to users and applications through commands, a user-level programming interface, a kernel programming interface, and user-level and kernel-level frameworks. The Solaris Cryptographic Framework provides these cryptographic services to applications and kernel modules in a manner seamless to the end user, and brings direct cryptographic services, like encryption and decryption for files, to the end user. The following diagram is an illustration of the Solaris Cryptographic Framework:



Command-line Utilities

The Solaris Cryptographic Framework includes a number of command-line utilities such as:

- `encrypt` (`decrypt`). This utility encrypts (decrypts) the given file or content from stdin using the algorithm specified. For more information on these utilities, see `encrypt(1)` and `decrypt(1)`.
- `digest`. This utility calculates the message digest of a given file, set of files, or from `stdin` using the algorithm specified. For more information on this utility, see `digest(1)`.
- `mac`. This utility calculates the message authentication code (MAC) of a given file, set of files, or from `stdin` using the algorithm specified. For more information on this utility, see `mac(1)`.

Administration

In addition, the Solaris Cryptographic Framework includes the `cryptoadm(1M)` command that displays cryptographic provider information for a system, configures the mechanism policy for each provider, and installs (or uninstalls) a cryptographic provider. The cryptographic framework supports three types of providers: a user-level provider (a PKCS#11 shared library), a kernel software provider (a loadable kernel software module), and a kernel hardware provider (a cryptographic hardware device).

In the following example, the software and hardware implementations of the MD5 algorithm are disabled. The first command impacts the kernel MD5 provider whereas the second command impacts the user-land software MD5 provider:


```
# cryptoadm disable provider=md5 mechanism=CKM_MD5

# cryptoadm disable \
      provider=/usr/lib/security/\$ISA/pkcs11_softtoken_extra.so \
      mechanism=CKM_MD5

# digest -v -a md5 /usr/bin/ls
digest: no cryptographic provider was found for this algorithm - md5

# cryptoadm list -p

User-level providers:
=====================
/usr/lib/security/$ISA/pkcs11_kernel.so: all mechanisms are enabled.
/usr/lib/security/$ISA/pkcs11_softtoken_extra.so: all mechanisms are
enabled, except CKM_MD5. random is enabled.

 Kernel software providers:
==========================
des: all mechanisms are enabled.
aes256: all mechanisms are enabled.
arcfour2048: all mechanisms are enabled.
blowfish448: all mechanisms are enabled.
sha1: all mechanisms are enabled.
sha2: all mechanisms are enabled.
md5: all mechanisms are enabled, except CKM_MD5.
rsa: all mechanisms are enabled.
swrand: random is enabled.

Kernel hardware providers:
==========================
```

This type of policy configuration could be used, for example, to enforce the use of FIPS 180-2 algorithms by prohibiting the Solaris Cryptographic Framework from supporting algorithms such as MD5.

For more information, see:

- OpenSolaris Community Project: Cryptographic Framework
  http://www.opensolaris.org/os/community/security/projects/ef/
- OpenSolaris Community Project: Key Management Framework
  http://www.opensolaris.org/os/project/kmf/
- Solaris Cryptographic Framework
  http://www.sun.com/bigadmin/features/articles/crypt_framework.html
- Sun BluePrint: Using the Cryptographic Accelerator of the UltraSPARC T1 Processor
  http://www.sun.com/blueprints/0306/819-5782.pdf

Compartmentalization (Zones)

Zones are an operating system abstraction for partitioning systems, allowing multiple applications to run in isolation from each other on the same physical hardware. This isolation prevents processes running within a zone from monitoring or affecting processes running in other zones, seeing each other's data, or manipulating

the underlying hardware. Zones also provide an abstraction layer that separates applications from physical attributes of the machine on which they are deployed, such as physical device paths and network interface names.

General Zones Recommendations

Organizations are strongly encouraged to leverage Solaris zones for compartmentalization whenever possible. This includes cases where there may only be one exposed service on a given system. The rationale for this is sample. Zones offer a number of security protections that would otherwise not be available. Deploying services within a zone enables organizations to take advantage of capabilities such as:

- Access to raw kernel memory is not permitted in a non-global zone. As a result, the installation of kernel root kits or attempts to unload or manipulate kernel modules will fail.
- Direct access to devices is not permitted unless explicitly authorized by the global zone.
- Non-global zones operate with inherently fewer privileges than a global zone. While the actual set of privileges are configurable, there are still quite a few (potentially dangerous) privileges that are restricted from being offered in a non-global zone.
- Resource management controls can be implemented on a per-zone basis.
- Large portions of the operating system are mounted read-only (when using sparse root configurations discussed below). This has the effect of limiting the scope and impact of quite a large set of user-land root kits, trojans, and other malicious programs.
- Non-global zones can have their own users, groups, hardening configurations, etc. based upon the types of services that they need to expose.
- Security monitoring can occur in the global zone thereby ensuring that activities in non-global zones are not only monitored but attackers in non-global zones will not know that they are being monitored nor will they have access to monitoring services, configurations, and data.

When non-global zones are used to host users and services, the global zone can be used effectively as a system controller. The global zone would be responsible for zone configuration, resource management and monitoring whereas individual (non-global) zones would be responsible for supporting business or operational services and functions.

For more information on Solaris Zones, see:

- Sun How To Guide: Eliminating Web Page Hijacking in the Solaris 10 OS
  http://www.sun.com/software/solaris/howtoguides/s10securityhowto.jsp

Sparse and Whole Root Zones

By default, Solaris zones are created in a sparse root configuration. Essentially, this means that the `/usr`, `/lib`, `/platform`, and `/sbin` directories are all mounted from the global zone (into the non-global zone) as a read-only loopback mount. In addition to consuming less disk space, the use of sparse root configurations is recommended because file systems mounted in this way are immutable from the viewpoint of a non-global zone. The `zonecfg(1M)` command can be used to determine if a zone is configured in this manner:

```
$ zonecfg -z <zone name> info inherit-pkg-dir
inherit-pkg-dir:
        dir: /lib
inherit-pkg-dir:
```



```
        dir: /platform
inherit-pkg-dir:
        dir: /sbin
inherit-pkg-dir:
        dir: /usr
```

In addition to being read-only, file systems mounted in a sparse root configuration are also mounted with the `nodevices` and `nosub` mount options that were mentioned earlier in the section on File System Security starting on page 9.

IP Instances for Zones

Introduced in Solaris 10 8/07, IP Instances for Zones extends the virtualization capabilities of Solaris zones deeper into the networking area. In particular, Solaris can now be configured to assign an exclusive IP network stack to individual non-global zones. To accomplish this, each non-global zone, using an exclusive IP stack, is assigned its own "data link name" (and today is assigned a dedicated physical network interface). Not only does this capability result in greater network isolation between zones, it also enables zones to manage their own network configuration (e.g., IP addresses, routes, etc.) in addition to supporting functionality such as IP Filter or IPsec within the zone itself.

Traditionally, Solaris zones have used a shared IP stack and therefore have been separated (at the network level) by their IP address. As a result, security capabilities such as IP Filter and IPsec had to be configured and managed from within the Solaris global zone.

By default, a Solaris non-global zone is configured to use a shared IP stack configuration. To configure a zone to use an exclusive IP stack, set the `ip-type` parameter in `zonecfg(1M)` as in:

```
$ zonecfg -z myzone info ip-type
ip-type: exclusive
```

In addition, a dedicated network interface should be assigned to the zone:

```
# zonecfg -z myzone info net
net:
        address not specified
        physical: bge0
```

Note that a zone configured with an exclusive IP stack should not assign a network address to the dedicated interface using `zonecfg(1M)` as would have been done when using a shared IP configuration. The IP address should be set manually or using a zone specific `sysidcfg(4)` file.

In order to leverage IP Instances functionality, a Generic Network Driver (GLD) version 3 interface must be used. To determine if a particular network interface is using GLDv3, use the `dladm(1M)` command:

```
# dladm show-link
ce0             type: legacy    mtu: 1500       device: ce0
ce1             type: legacy    mtu: 1500       device: ce1
```

If a particular interface reports its type as `legacy`, then it is not using the GLDv3 framework and therefore cannot be used by IP Instances. Attempts to use such an interface as part of an exclusive IP stack in a non-global zone will result in an error message such as:

```
zoneadm: zone 'myzone': WARNING: unable to hold network interface 'ce1'.:
Invalid argument
```


For more information on IP Instances for Zones, see:

- Solaris 10 System Administrator Guide: Exclusive IP Non-Global Zones
  http://docs.sun.com/app/docs/doc/817-1592/6mhahuooo?a=view#geprv
- OpenSolaris Community Project: Crossbow - IP Instances Design Document
  http://www.opensolaris.org/os/project/crossbow/Docs/si-design.pdf
- OpenSolaris Community Project: Crossbow – Network Virtualization/Resource Control FAQ
  http://www.opensolaris.org/os/project/crossbow/faq

Cross-zone Network Communication

Solaris zones are only permitted to communicate with one another over the network which is only possible if they exist on the same network or they are able to reach one another through some external router. When two non-global zones exist on the same logical network, they communicate over a loopback interface in Solaris for performance reasons (rather than passing packets out to the NIC). Prior to Solaris 10 11/06, there was no way to limit or filter communications flowing over this pathway. Starting with Solaris 10 11/06, IP Filter is now able to filter communication occurring over the loopback interface by setting the following parameter at the top of `/etc/ipf/ipf.conf`:

```
set intercept_loopback true;
```

Care should be taken when enabling this functionality as all loopback (including `lo0`) network traffic will be subject to the IP Filter policy defined on the system.

Configurable Privileges

Starting in Solaris 10 11/06, it is now possible to adjust the Solaris privileges that are available to non-global zones. This is often necessary in order to enable certain types of applications to run in a non-global zone. In order to promote strong security protections, the privileges available to non-global zones is still limited. A listing of privileges that are prohibited from being used in a non-global zone can be found in Appendix B of the Configurable Privileges for Zones project proposal at:

http://www.opensolaris.org/os/community/arc/caselog/2006/124/proposal/

To adjust the default privileges granted to a non-global zone, the `limitpriv` option to `zonecfg(1M)` should be used as in the following example:

```
$ zonecfg -z myzone info limitpriv
limitpriv: default,sys_time
```

It is also possible to query a running zone to verify that the privilege is in fact available:

```
$ zlogin myzone ppriv -l zone | grep sys_time
sys_time
```

In this case, the `sys_time` privilege was granted to the zone `myzone` allowing that non-global zone to set the system clock in the global zone (and all other non-global zones since there is only one clock on a Solaris system). This could be useful in a case where a non-global zone was acting as a Network Time Protocol (NTP) server, for example.

Integrity Management

This section discusses several capabilities in the Solaris OS that can be used to inspect and validate the integrity of file system objects. Each of these capabilities can be used together or in isolation depending on the goals or requirements of an organization.

Signed ELF Objects

Starting in the Solaris 10 OS, the majority of compiled (ELF objects) code has been cryptographically signed by Sun. This includes ELF objects such as binaries, libraries, shared objects, device drivers, kernel modules, and even plugins for the Solaris Cryptographic Framework. The `elfsign(1)` tool is used to both sign ELF objects as well as verify their signatures:

```
$ elfsign verify -e /usr/bin/ls
elfsign: verification of /usr/bin/ls passed.
```

Using the `-v` argument, information such as the format (of the signature) and the signer can also be inspected:

```
$ elfsign verify -v -e /usr/bin/ls
elfsign: verification of /usr/bin/ls passed.
format: rsa_md5_sha1.
signer: CN=SunOS 5.10, OU=Solaris Signed Execution, O=Sun Microsystems Inc.
```

Currently, these signatures can only be inspected manually using the `elfsign(1)` command. The only exception to this is related to the Solaris Cryptographic Framework. Cryptographic plugins (both user-land and kernel) must be signed by Sun and must pass a signature verification check prior to the ELF object being used.

For more information on signed ELF objects in the Solaris OS, see:

- `elfsign(1)`
- Sun Blog: Signed Solaris 10 Binaries
  http://blogs.sun.com/darren/entry/signed_solaris_10_binaries
- Sun Blog: Integrity Checking in Depth
  http://blogs.sun.com/davew/entry/integrity_checking_in_depth

Basic Audit Reporting Tool (BART)

The Basic Audit Reporting Tool is a program used to capture and record file integrity information. The BART utility supports two primary modes of operation: `create` and `compare`. When run in its `create` mode, BART will create a manifest for each file processed that contains information regarding the attributes of the files. Information collected includes the object name, object type, owner (`uid`), group (`gid`), permissions and access control lists, as well as an MD5 fingerprint of the contents (if a file). For example:

```
# find /etc/security | bart create -I
! Version 1.0
! Tuesday, September 18, 2007 (11:24:33)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
```



```
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/etc/security D 512 40755 user::rwx,group::r-x,mask:r-x,
      other:r-x 46dcb5b6 0 3
/etc/security/audit D 512 40755 user::rwx,group::r-x,mask:r-x,
      other:r-x 46dc91e4 0 3
/etc/security/audit/localhost D 512 40755 user::rwx,group::r-x,
      mask:r-x,other:r-x 46dc91e4 0 3
/etc/security/audit/localhost/files L 21 120777 - 46dc91e4 0 0
       ../../../../var/audit
/etc/security/audit_class F 2399 100644 user::rw-,group::r--,mask:r--,
      other:r-- 46d105cb 0 3 0d1c8c3cdfa3a874dc877ba1bba8380a
/etc/security/audit_control F 1014 100644 user::rw-,group::r--,mask:r--,
      other:r-- 46d105cb 0 3 0e2909e9477baa8dd973f57394b6779e
/etc/security/audit_event F 22330 100644 user::rw-,group::r--,mask:r--,
      other:r-- 46d105cb 0 3 3922a92f76c93d59785214045107a37d
/etc/security/audit_startup F 1116 100744 user::rwx,group::r--,mask:r--,
      other:r-- 46d105cb 0 3 944d347e1c6ac495ebdb4a66d50e690d
/etc/security/audit_user F 1053 100644 user::rw-,group::r--,mask:r--,
      other:r-- 46d105cb 0 3 2913855b950db00b595cf7b5dd1fff31
/etc/security/audit_warn F 7501 100740 user::rwx,group::r--,mask:r--,
      other:--- 46d10907 0 3 ae7290b8d5d63af3703f33f7e7010e46
[...]
```

In this example, BART was used to catalog the `/etc/security` directory. The manifest are directed to standard output but could have been easily saved in a file. BART can also be used in a `compare` mode where it takes two existing manifests and reports any differences. For example:

```
# bart compare manifest-before manifest-after
/etc/security/exec_attr:
  size  control:29654  test:29664
  mtime  control:46e8a76e  test:46efee70
  contents  control:caf727ccd4de989974c2c81fa7cfdf29
      test:071e79250e05b2363415ee5f05d25324
```

In this case, after comparing two manifests, `manifest-before` and `manifest-after`, BART detected a change to the `/etc/security/exec_attr` file. The changes to this file included its size, its modification time and its MD5 fingerprint. BART can be used on both the global zone and non-global zones to detect changes occurring across file systems.

Organizations are encouraged to develop their own policy for BART and use it to regularly take point-in-time snapshots of their system configuration and compare those with a known good baseline so that unauthorized changes can be more quickly and easily detected. When used in concert with Solaris Auditing, these changes can often be traced back to specific users resulting in more effective incident response.

For more information on the Basic Audit Reporting Tool, see:

- `bart(1M)`
- Sun BluePrint: Automation Solaris 10 File Integrity Checks
  http://www.sun.com/blueprints/0305/819-2259.pdf
- Sun BluePrint: Integrating BART and the Solaris Fingerprint Database in the Solaris 10 OS
  http://www.sun.com/blueprints/0405/819-2260.pdf

Solaris Fingerprint Database

The Solaris Fingerprint Database is a collection of MD5 fingerprints for files included with the Solaris OS (directly or within a patch). This database can be queried using a web-based interface or using a tool such as the Solaris Fingerprint Database Companion (sfpC) to determine if a given MD5 fingerprint belongs to a file shipped by Sun. Often used for integrity validation and forensics, the Solaris Fingerprint Database can quickly and easily help organization determine the integrity of files as well as detect various conditions such as upgrade and downgrade attacks. For example, the fingerprint generated by the following command:

```
$ digest -a md5 -v /usr/bin/ls
md5 (/usr/bin/ls) = b526348afd2d57610dd3635e46602d2a
```

generated the following response from the Solaris Fingerprint Database thereby validating the object as a binary shipped by Sun, `/usr/bin/ls`, included in the `SUNWcsu` package as part of Solaris 10 for the SPARC platform:

```
b526348afd2d57610dd3635e46602d2a - - 1 match(es)
      * canonical-path: /usr/bin/ls
      * package: SUNWcsu
      * version: 11.10.0,REV=2005.01.21.15.53
      * architecture: sparc
      * source: Solaris 10/SPARC
```

When a BART content comparison fails, it is a good idea to submit the MD5 fingerprint to the Solaris Fingerprint Database. This technique will enable organizations to determine if the file failing the check is in fact a valid file that was produced by Sun. Very often, failures can occur when performing BART manifest comparisons after a set of patches has been applied to a system. Clearly, a number of files will be replaced by those patches. In this case, the use of BART with the Solaris Fingerprint Database will help an organization to validate that the new files did in fact come from Sun patches.

For more information on the Solaris Fingerprint Database, see:

- The Solaris Fingerprint Database
  http://www.sun.com/blueprints/0306/816-1148.pdf
- Sun BluePrint: Integrating BART and the Solaris Fingerprint Database in the Solaris 10 OS
  http://www.sun.com/blueprints/0405/819-2260.pdf
- OpenSolaris Community Project: Solaris Fingerprint Database Tools
  http://www.opensolaris.org/os/community/security/projects/sfpdb/

Auditing

Solaris auditing helps to detect potential security breaches by revealing suspicious or abnormal patterns of system usage. Solaris auditing also provides a means to trace suspect actions back to a particular user, thus serving as a deterrent. Users who know that their activities are being audited are less likely to attempt malicious activities. The CIS Solaris 10 Security Benchmark enabled and configured Solaris Auditing in a section titled “Enable Kernel Level Auditing”.

Audit Policy Configuration

In addition to selecting which events to audit, globally or for specific users, Solaris Auditing also supports a number of other configuration options. These configuration options are specified using the `auditconfig(1M)` program and stored in the `/etc/security/audit_startup` file. A few of the most often used policy settings are listed in the following table. For more information on each of these options as well as how they can be set, see `auditconfig(1M)`:

Audit Policy Variable	Description
`ahlt`	Halt the machine if an asynchronous audit event occurs that cannot be delivered because the audit queue has reached the high-water mark or because there are insufficient resources to construct an audit record. By default, this option is not set and audit records are dropped (when either of these conditions occurs) and a count is kept of the number of dropped records. This count can be queried using the `auditstat(1M)` command.
`arge`	Include the `execv(2)` system call environment arguments to the audit record. This information is not included by default in Solaris, but it is enabled by the CIS item titled “Enable Kernel Level Auditing”.
`argv`	Include the `execv(2)` system call parameter arguments to the audit record. This information is not included by default in Solaris, but it is enabled by the CIS item titled “Enable Kernel Level Auditing”.
`public`	Audit public files. By default, read- type operations are not audited for certain files which meet public characteristics: owned by `root`, readable by all, and not writable by all. This option, when enabled, helps to reduce some of the non-security relevant events or “noise” commonly found in audit trails.
`seq`	Include the sequence token as part of every audit record. By default, the sequence token is not included. The sequence token attaches a sequence number to every audit record.
`zonename`	Include the zonename token as part of every audit record. By default, the zonename token is not included. The zonename token gives the name of the zone from which the audit record was generated.

Audit Record Selection and Display

Once Solaris Auditing has been enabled, organizations can use the audit trails to better understand what actions are being taken on their systems. In particular, the commands `auditreduce(1M)` and `praudit(1M)` can be used to select and display events respectively. For example, the following audit record shows that the user `gbrunett` successfully used the `su` command to access a `root` shell in the global zone on `sec-b1600-0`:

```
# auditreduce -u gbrunett -m AUE_su | praudit -s
header,104,2,AUE_su,,sec-b1600-0,2007-09-23 15:29:00.248 -04:00
subject,gbrunett,root,gbrunett,gbrunett,gbrunett,692,2868335681,0 0
      sec-b1600-0
text,success for user root
return,success,0
zone,global
```

This same information can also be represented in XML in the Solaris 10 OS:


```
# auditreduce -u gbrunett -m AUE_su | praudit -x
<?xml version='1.0' encoding='UTF-8' ?>
<?xml-stylesheet type='text/xsl'
      href='file:///usr/share/lib/xml/style/adt_record.xsl.1' ?>

<!DOCTYPE audit PUBLIC '-//Sun Microsystems, Inc.//DTD Audit V1//EN'
      'file:///usr/share/lib/xml/dtd/adt_record.dtd.1'>

<audit>
<file iso8601="2007-09-23 15:29:00.000 -04:00"></file>
<record version="2" event="su" host="sec-b1600-0" iso8601="2007-09-23
15:29:00.248 -04:00">
<subject audit-uid="gbrunett" uid="root" gid="gbrunett" ruid="gbrunett"
rgid="gbrunett" pid="692" sid="2868335681" tid="0 0 sec-b1600-0"/>
<text>success for user root</text>
<return errval="success" retval="0"/>
<zone name="global"/>
</record>
<file iso8601="2007-09-23 15:29:00.000 -04:00"></file>
</audit>
```

Similarly, in the Solaris 10 OS, Solaris Auditing can be configured to forward audit events to `syslog`. Regardless of whether the use of `syslog` has been configured, audit events are always also recorded on the local system. When using the `syslog` plugin, the above audit event would be recorded as:

```
Sep 23 15:29:00 sec-b1600-0 audit: [ID 702911 audit.notice] su ok session
2868335681 by gbrunett as root:gbrunett in global from sec-b1600-0 text
success for user root
```

For more information on Solaris Auditing, see:

- OpenSolaris Community Project: Auditing
  http://www.opensolaris.org/os/project/audit/
- Solaris 10 System Administrator Collection: Auditing and Solaris Zones
  http://docs.sun.com/app/docs/doc/816-4557/6maosrjqv?a=view
- Sun BluePrint: Enforcing the Two-Person Rule using RBAC in the Solaris 10 OS
  http://www.sun.com/blueprints/0805/819-3164.pdf
- Sun Blog: Defining your own audit class
  http://blogs.sun.com/martin/entry/defining_your_own_audit_class
- Sun Blog: Measuring the impact of auditing
  http://blogs.sun.com/martin/entry/measuring_the_impact_of_auditing
- Sun Blog: How system calls are audited
  http://blogs.sun.com/martin/entry/how_system_calls_are_audited
- Sun Blog: ZFS the perfect file system for audit trails
  http://blogs.sun.com/martin/entry/zfs_the_perfect_file_system
- Sun Blog: How to determine if there are audit records in a crash dump
  http://blogs.sun.com/martin/entry/how_to_determine_if_there

- Sun Blog: Adding auditing to your application
  http://blogs.sun.com/martin/entry/adding_auditing_to_your_application
  http://blogs.sun.com/martin/entry/adding_auditing_to_your_application2

Packet Filtering

IP Filter

Solaris IP Filter integrates the open source IP Filter software into Solaris 10. IP Filter provides stateful packet filtering capabilities and can filter packets by IP address or network, port, protocol, network interface, and traffic direction. In addition, it also has the ability to perform network address translation (NAT) and port address translation (PAT). IP Filter supports both IPv4 and IPv6, and is configured using a simple firewall rules policy language. Information and examples on the syntax of the policy language can be found in `ipf(4)`.

IP Filter can be enabled using the SMF service `svc:/network/ipfilter:default` or `ipfilter`, for short:

```
$ svcs ipfilter
STATE          STIME    FMRI
disabled       Sep_16   svc:/network/ipfilter:default

$ svcadm enable -r ipfilter

$ svcs ipfilter
STATE          STIME    FMRI
online         14:39:44 svc:/network/ipfilter:default
```

By default, the IP Filter configuration is stored in `/etc/ipf/ipf.conf`. Using the `ipfstat(1M)` command, the list of incoming and outgoing rules being enforced can be displayed:

```
# ipfstat -io
pass out quick all keep state keep frags
block in quick from any to any port = 137
block in quick from any to any port = 138
block in quick from any to any port = 139
pass in quick proto udp from any to any port = ike
pass in quick proto udp from any to any port = 4500
pass in quick proto esp from any to any
pass in log quick proto tcp from 192.168.1.0/24 to any port = ssh
block in log all
block in from any to 255.255.255.255/32
block in from any to 127.0.0.1/32
```

The example shown above illustrates an example IP Filter rule set that could be used on a desktop or laptop since it does not restrict outbound communication but blocks nearly everything attempting to communicate to the system. The above example also includes support for IPsec/IKE and incoming Secure Shell (from a specific network). The `ipfstat(1M)` command can also be used to collect valuable information and statistics regarding how IP Filter is functioning.

IP Filter can logs information to the `syslog` facility. The following example illustrates a blocked `telnet` connection attempt from 192.168.0.1 to 192.168.0.2:

```
Sep 18 14:47:50 blackhole ipmon[7237]: [ID 702911 local0.warning]
14:47:50.075431 ip.tun0 @0:12 b 192.168.0.1,52854 -> 192.168.0.2,23 PR tcp
len 20 52 -S IN
```

For more information on IP Filter, see:

- Solaris 10 Adoption Kit: IP Filter
  http://partneradvantage.sun.com/protected/solaris10/adoptionkit/general/features/ip_filter.html
- Open Source IP Filter
  http://coombs.anu.edu.au/~avalon/

TCP Wrappers

TCP Wrappers is a collection of programs and libraries that enable organizations to restrict access to TCP-based network services. As originally discussed in the CIS Solaris 10 Benchmark section called "Configure TCP Wrappers", access policy is configured using a pair of files: `/etc/hosts.allow` and `/etc/hosts.deny`. In the Solaris 10 OS, both Secure Shell and sendmail(1M) are configured to always use TCP Wrappers if either of the `/etc/hosts.allow` or `/etc/hosts.deny` files exist. In addition, the RPC port mapping daemon, `rpcbind`, and `inetd` can be optionally configured to restrict access using TCP Wrappers as in the following example for `rpcbind`:

```
# svccfg -s rpc/bind setprop config/enable_tcpwrappers=true

# svcadm refresh rpc/bind
```

The `inetd` service can be configured using a similar SMF property. In addition, services started by the `inetd` service can be configured to individually use TCP Wrappers using per-service configuration parameter as shown in the following example based on the `telnet` service:

```
$ inetadm -l telnet | grep tcp_wrappers
default  tcp_wrappers=FALSE

$ pfexec inetadm -m telnet tcp_wrappers=TRUE

$ inetadm -l telnet | grep tcp_wrappers
         tcp_wrappers=TRUE
```

In the CIS Benchmark section titled "Configure TCP Wrappers", the example illustrated how to configure a single, default deny policy for all services. It should be noted that TCP Wrappers supports a rich configuration policy language that enables organizations to specify policy not only globally but on a per-service basis. Further access to services can be permitted or restricted based upon host name, IPv4 or IPv6 address, netgroup name, network, and even DNS domain. More information and examples ofthe syntax of this access control language can be found in `host_access(4)`.

For more information on configuring and using TCP Wrappers, see:

- `tcpd(1M)`, `tcpdchk(1M)`, `tcpd-match(1M)`, and `hosts_access(4)`
- Sun Blog: Enabling TCP Wrappers in the Solaris 10 OS
  http://blogs.sun.com/gbrunett/date/20050406

Remote Access Security

Internet Protocol Security (IPsec)

Solaris 10 supports IP Security (IPsec) for both IPv4 and IPv6. Originally introduced in the Solaris 8 OS, IPsec and its key exchange protocol, IKE, have been enhanced in Solaris 10 to provide complete support for tunnel mode, IKE NAT traversal (RFC 3947 and RFC 3948), and the Solaris Cryptographic Framework.

In particular, the Solaris Cryptographic Framework provides a `softtoken` keystore for applications that use the `metaslot`. When IKE is configured to use the `metaslot`, organizations have the option of storing the keys on disk, on an attached board, or in the `softtoken` keystore. For more information on the `softtoken` keystore, see `cryptoadm(1M)`.

For more information, see:

- OpenSolaris Community Project: IPsec Tunnel Reform
  http://www.opensolaris.org/os/project/tref/
- Solaris 10 Administrator Collection: IP Security
  http://docs.sun.com/app/docs/doc/816-4554/6maoq0217?a=view
- Solaris 10 Administrator Collection: Protecting Traffic with IPsec
  http://docs.sun.com/app/docs/doc/816-4554/6maoq021r?a=view

Secure Shell

Originally based upon a fork of the OpenSSH software, Solaris Secure Shell enables users or services to access or transfer files between remote systems over an encrypted communications channel. In Solaris Secure Shell, authentication is provided by the use of passwords, public keys, or both. All network traffic is encrypted. Thus, Solaris Secure Shell prevents a would-be intruder from being able to read an intercepted communication. Solaris Secure Shell also prevents an adversary from spoofing the system. Solaris Secure Shell can also be used as an on-demand virtual private network that can forward X Window system traffic or can connect individual port numbers between the local machines and remote machines over an encrypted network link.

The configuration of Solaris Secure Shell is contained in the following files:

- `/etc/ssh/ssh_config`. This file provides the system-wide default settings for the client portion of the Solaris Secure Shell software, `ssh(1)`. For more information on what settings are available, see `ssh_config(4)`. These settings can also be overridden on a per-user basis by creating a file `$HOME/.ssh/config`.
- `/etc/ssh/sshd_config`. This file provides the system-wide default settings for the server portion of the Solaris Secure Shell software, `sshd(1M)`. For more information on what settings are available, see `sshd_config(4)`.

Secure Shell configuration recommendations are covered in the CIS Solaris 10 Benchmark section called “Configure SSH”.

For more information on Solaris Secure Shell, see:

- OpenSolaris Community Project: Secure Shell (SSH)
  http://www.opensolaris.org/os/community/security/projects/SSH/

- Solaris 10 Administrator Collection: Using Secure Shell
  http://docs.sun.com/app/docs/doc/816-4557/sshuser-34?a=view
- Secure Remote Access in the Solaris 9 Operating Environment
  http://www.sun.com/software/whitepapers/solaris9/secureaccess.pdf

Kerberos

The Kerberos service is a client-server architecture that provides secure transactions over networks. The service offers strong user authentication, as well as integrity and privacy. Authentication guarantees that the identities of both the sender and the recipient of a network transaction are true. The service can also verify the validity of data being passed back and forth (integrity) and encrypt the data during transmission (privacy). Using the Kerberos service, you can log in to other machines, execute commands, exchange data, and transfer files securely. Additionally, the service provides authorization services, which allows administrators to restrict access to services and machines. Moreover, as a Kerberos user, you can regulate other people's access to your account.

The Kerberos service is a single-sign-on system, which means that you only need to authenticate yourself to the service once per session, and all subsequent transactions during the session are automatically secured. After the service has authenticated you, you do not need to authenticate yourself every time you use a Kerberos-based command such as `ftp` or `rsh`, or to access data on an NFS file system. Thus, you do not have to send your password over the network, where it can be intercepted, each time you use these services.

The Solaris Kerberos service is based on the Kerberos V5 network authentication protocol that was developed at the Massachusetts Institute of Technology (MIT). Because the Kerberos V5 protocol is a de facto industry standard for network security, the Solaris version promotes interoperability with other systems. In other words, because the Solaris Kerberos service works with systems that use the Kerberos V5 protocol, the service allows for secure transactions even over heterogeneous networks. Moreover, the service provides authentication and security both between domains and within a single domain.

The Kerberos service allows for flexibility in running Solaris applications. You can configure the service to allow both Kerberos-based and non-Kerberos-based requests for network services such as the NFS service, `telnet`, `ftp`, `rlogin` and Secure Shell. As a result, current Solaris applications still work even if they are running on systems on which the Kerberos service is not enabled. Of course, you can also configure the Kerberos service to allow only Kerberos-based network requests.

The Kerberos service provides a security mechanism which allows the use of Kerberos for authentication, integrity, and privacy when using applications that use the Generic Security Service Application Programming Interface (GSS-API). However, applications do not have to remain committed to the Kerberos service if other security mechanisms are developed. Because the service is designed to integrate modularly into the GSS-API, applications that use the GSS-API can utilize whichever security mechanism best suits their needs.

For more information configuring and using Kerberos, see:

- OpenSolaris Community Project: Kerberos
  http://www.opensolaris.org/os/community/security/projects/kerberos/
- Solaris 10 Administrator Collection: Kerberos
  http://docs.sun.com/app/docs/doc/816-4557/ezlsz?a=view

Solaris Trusted Extensions

Solaris Trusted Extensions is an optionally-enabled layer of secure labeling technology that allows data security policies to be separated from data ownership. While it has its roots in the multilevel Trusted Solaris 8 OS, it has been integrated into the standard Solaris 10 Operating System. This new approach allows the Solaris operating system to support both traditional Discretionary Access Control (DAC) policies based on ownership, as well as label-based Mandatory Access Control (MAC) policies. The label-based policies for file systems and networks are light-weight and have been implemented within the standard Solaris 10 kernel, services and utilities. Unless the Trusted Extensions layer is enabled, all labels are equal so the kernel is not configured to enforce the MAC policies. For efficiency, a boolean value is maintained in the kernel to indicate whether labeling comparisons should be used in policy enforcement.

When the label-based MAC policies are enabled, all data flows are restricted based on a comparison of the labels associated with the subjects requesting access and the objects containing the data. Like other multilevel operating systems, Trusted Extensions meets the requirements of the Common Criteria Labeled Security Protection Profile (LSPP), the Role-Based Access Protection Profile (RBACPP) and the Controlled Access Protection Profile (CAPP). However, the Trusted Extensions implementation is unique in its ability to provide high assurance, while maximizing compatibility and minimizing overhead.

For more information, see:

- OpenSolaris Community Project: Trusted Extensions
  http://www.opensolaris.org/os/community/security/projects/tx/
- Architectural Overview of Solaris Trusted Extensions
  http://www.opensolaris.org/os/community/security/projects/tx/TrustedExtensionsArch.pdf
- Solaris Trusted Extensions Developer's Guide
  http://docs.sun.com/app/docs/doc/819-7312
- Sun Blog: Remote Multilevel Desktop Sessions
  http://blogs.sun.com/gfaden/entry/remote_multilevel_desktop_sessions
- Sun Blog: Label Aware Web Services
  http://blogs.sun.com/gfaden/entry/label_aware_web_services

Management Considerations

Solaris Secure by Default

The Solaris Secure by Default project reduces this attack surface of the Solaris OS by disabling as many network services as possible while still leaving a useful system. In this way, the number of exposed network services (in a default configuration) is dramatically reduced. This project changes the default configuration of the Solaris OS such that `ssh` is the only network-listening service. Other network services are either disabled or configured to accept requests only from the local system. This project was integrated into Solaris 10 11/06 (Update 3) and has been discussed earlier in the Benchmark in the section titled, "Establish a Secure Baseline".

The following services are impacted by the Solaris Secure by Default "local only" policy. When running in a secure by default configuration, the following services are set to local only. The following table lists each service, its respective FMRI, as well as the SMF property that controls the local only behavior and its possible values. The value highlighted in bold is the value used in a secure by default configuration:

Service	FMRI	Property	Values
rpcbind	svc:/network/rpc/bind	config/local_only	**true**, false
syslog	svc:/system/system-log	config/log_from_remote	true, **false**
sendmail	svc:/network/smtp:sendmail	config/local_only	**true**, false
smcwebserver	svc:/system/webconsole:console	options/tcp_listen	true, **false**
wbem	svc:/application/management/wbem	options/tcp_listen	true, **false**
X11	svc:/application/x11/x11-server	options/tcp_listen	true, **false**
CDE	svc:/application/graphical-login/cde-login	dtlogin/args	[null], **-udpPort 0**
ToolTalk	svc:/network/rpc/cde-ttdbserver:tcp	proto	tcp, **ticotsord**
calendar	svc:/network/rpc/cde-calendar-manager	proto	tcp, **ticlts**
BSD printing	svc:/application/print/rfc1179:default	bind_addr	[null], **localhost**

To read more about the Solaris Secure by Default project, see:

- OpenSolaris Community Project: Secure by Default
  Project Page: http://www.opensolaris.org/os/community/security/projects/sbd/
  Training: http://www.opensolaris.org/os/community/security/projects/sbd/sbd_toi.pdf
- Sun Blog: Solaris Secure by Default
  Part 1: http://blogs.sun.com/gbrunett/?entry=solaris_secure_by_default_part
  Part 2: http://blogs.sun.com/gbrunett/?entry=solaris_secure_by_default_part1
  Part 3: http://blogs.sun.com/gbrunett/?entry=solaris_secure_by_default_part2

Solaris Security Toolkit (JASS)

The Solaris Security Toolkit, formerly known as the JumpStart Architecture and Security Scripts (JASS) toolkit, provides a flexible and extensible mechanism to harden and audit Solaris Operating System systems. The Solaris Security Toolkit simplifies and automates the process of securing Solaris OS systems and is

based on proven security best practices and practical customer site experience gathered over many years. Originally released in 1999, this toolkit can be used to secure systems running Solaris 2.5.1 through Solaris 10 on SPARC, Intel or AMD platforms, is freely available and supported under the Solaris support contract.

The Solaris Security Toolkit leverages the work of the Solaris Secure by Default project and extends upon it by offering a more complete platform hardening (and audit) capability whereas the Solaris Secure by Default project is only focused on limiting externally exposed network services.

To download a copy of the Solaris Security Toolkit or view its documentation, see:

http://www.sun.com/software/security/jass/

In addition, a few good articles on customizing the Solaris Security Toolkit can be found at:

http://blogs.sun.com/DanX/entry/solaris_security_toolkit_customization
http://blogs.sun.com/jimlaurent/entry/using_the_solaris_security_toolkit

Additional References

- Sun Security Home
  http://www.sun.com/security/
- Sun Security Community Blog
  http://blogs.sun.com/security/
- Sun Systemic Security (Sun BluePrint)
  http://www.sun.com/blueprints/0206/819-5605.pdf
- Solaris Security Home
  http://www.sun.com/software/solaris/security.jsp
- Solaris Security Learning Center
  http://www.sun.com/software/solaris/security_learning_center.jsp
- Solaris Security Certifications
  http://www.sun.com/software/security/securitycert/
- OpenSolaris Security Community
  http://www.opensolaris.org/os/community/security/
  - Library (Sun BluePrints, White Papers, etc.)
    http://www.opensolaris.org/os/community/security/library/
  - Presentation Library
    http://www.opensolaris.org/os/community/security/preso/
- Sun Security BluePrints
  http://blogs.sun.com/security/entry/reference_security_blueprints