



An Oracle White Paper
August 2005

ENFORCING THE TWO-PERSON RULE VIA ROLE-BASED ACCESS CONTROL IN THE ORACLE SOLARIS 10 OPERATING SYSTEM

Important note: this paper was originally published before the acquisition of Sun Microsystems by Oracle in 2010. The original paper is enclosed and distributed as-is. It refers to products that are no longer sold and references technologies that have since been re-named.

ENFORCING THE TWO-PERSON RULE VIA ROLE-BASED ACCESS CONTROL IN THE SOLARIS™ 10 OPERATING SYSTEM

Glenn Brunette, Client Solutions

Sun BluePrints™ OnLine — August 2005



© 2005 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, CA 95054 USA

All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California.

Sun, Sun Microsystems, the Sun logo, Solaris, and Sun BluePrints are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries.

UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g)(2)(6/87) and FAR 52.227-19(6/87), or DFAR 252.227-7015(b)(6/95) and DFAR 227.7202-3(a). DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS HELD TO BE LEGALLY INVALID.



Table of Contents

Enforcing the Two-Person Rule Via Role-Based Access Control in the Solaris™ 10 Operating System	1
What is the Two-Person Rule?	1
The Two-Person Rule in IT Security	1
About the Example in This Article	2
Users and Roles	2
Configuration Options	2
Steps for Configuring the Two-Person Rule Using RBAC	3
Step 1: Verify Users	4
Step 2: Verify User Privileges	4
Step 3: Grant the Audit Review Rights Profile to Users	5
Step 4: Create New Roles	5
Step 5: Assign Restricted Operations to the Roles	6
Step 6: Verify Role Configuration	7
Step 7: Verify User Actions in Audit Trails	9
Summary	11
About the Author	11
Acknowledgements	12
References	12
Publications	12
Web Sites	12
Command Manual Pages	12
Ordering Sun Documents	14
Accessing Sun Documentation Online	14

Enforcing the Two-Person Rule Via Role-Based Access Control in the Solaris™ 10 Operating System

This Sun BluePrints™ Cookbook describes how to use Solaris Role-Based Access Control (RBAC) in the Solaris™ 10 Operating System (Solaris OS) to enforce the two-person rule in IT security. For an introduction to RBAC, see “Role-Based Access Control (Overview)” in the *System Administration Guide: Security Services* at <http://docs.sun.com/app/docs/doc/816-4557/6maosrjfi?a=view>.

What is the Two-Person Rule?

The *two-person rule* (sometimes called the “four eyes rule”) has its origins in military protocol, although for quite some time it has been welcomed into the stockpile of IT security controls used by organizations around the world.

The two-person rule specifies that:

- there must be two individuals who need to act in concert in order to perform some action, and
- each individual should have comparable knowledge and skill in order to detect attempts at subversion initiated by the other.

One can certainly see why this is essential for highly sensitive tasks, such as safeguarding a country's nuclear arsenal—a single individual should not have excessive authority.

The Two-Person Rule in IT Security

Whether discussing physical or logical access controls, organizations have for years applied the practice of the two-person rule to help secure IT assets. Using the two-person rule is an optional approach for organizations wanting to protect access to key data sets, or to restrict who may perform sensitive or high impact operations on a system.

In many circumstances, however, more traditional IT security controls are likely appropriate. Using the two-person rule is most often reserved for restricting the most sensitive IT security operations performed within an organization. Whether and where a given organization could apply the two-person rule depends on its policies, architecture, processes, and requirements.

What constitutes a sensitive or high impact operation often varies based on an organization's policies, regulatory pressures, or other factors—such as the industry in which the organization operates. As organizations better understand and document their relative thresholds for risk, certain operations might rise above others, requiring stronger levels of protection such as those described in this article. One can easily envision, for example, that sensitive operations could include tasks related to audit trail or cryptographic key management functions.

About the Example in This Article

The example in this article shows how an organization can implement the two-person rule in Solaris 10 using the Solaris RBAC facility.

Note – The techniques described in this example apply to the Solaris 8 and 9 releases as well, because these versions of the operating system also included RBAC.

Users and Roles

This article refers to both normal users and roles. In the Solaris 10 OS, roles are similar to normal user accounts with two important differences:

- A role cannot be accessed directly over the network or from the console. Administrators must use the `su(1M)` command (or `smc(1M)`) to assume a role. In system before attempting to access a role (This is true as of Solaris 10 OS 03/2005; in future versions of the Solaris OS, this condition might be configurable in order to relax this requirement).
- A role can be assumed only by authorized users. Before a given user can assume a role, an administrator must assign that role to the user. Otherwise, attempts to access the role will fail.

Both of these restrictions are important for preserving accountability.

Configuration Options

The example in this article implements the two-person rule for two hypothetical IT security administrators, `joe` and `john`. This implementation could be configured in three different ways, with different implications for each.

- **Option One:** Configure both `joe` and `john` with normal accounts on the system, each able to assume an individual Solaris role.

For example, `joe` would be permitted to assume `roleA`, and `john` would be permitted to assume `roleB`. In this scenario, however, neither `joe` nor `john` would be entrusted with the password for the role that they are permitted to assume. Therefore, while `joe` could in theory assume `roleA`, he cannot do so because he lacks the required password.

This approach has the benefit of allowing either user to assume the role—it does not require that only one user be the initiator of the action. Further, by authorizing both users to be on the system, they have more opportunity to monitor the actions of the other (such as Solaris auditing, `syslog`, and so on).

- **Option Two:** Configure one user on the system with permission to assume one role.

In this case, one user would know only the initial (user) password, while the other user would know only the second (role) password. This case requires only one user and one role on the system.

The disadvantage of this approach is that it does not allow the second user to log directly into the system and participate in critical monitoring opportunities.

Further, because there is only one, non-shared user account and two users, this approach requires that `joe` must be first logged into the system before `john` can assume `roleA`. To ensure a separation of duty, the single user account cannot be shared in any way (doing so violates accountability principles), which means that only one person can be logged in (because there is only one account). In this scenario, one person would know the password to the user account while the other would have the password for the role; this requires that both users work together to perform privileged operations.

- **Option Three:** Configure both `joe` and `john` with normal accounts on the system, but give them access to the same shared role. In contrast with Option One (where the users must assume different roles), in this option, the users both assume the same role.

In this case, neither user would have the password to authenticate to the shared role. In order to access the shared role, both users would be required to work together to obtain the password from an external source, such as a physical safe.

This option provides greater protection because the safe could be placed in an area that requires additional physical controls and provides greater opportunities for monitoring. Further, the authenticator could be monitored and changed by a third party who has no relationship to either `joe` or `john`. This approach also allows both `joe` and `john` to easily monitor the activities of the other.

The example in this article uses the first option, showcasing the capabilities of Solaris RBAC to implement the two-person rule without depending on external factors, such as those described in the third option. A given organization must choose the appropriate option based on its own policies and requirements. Additional security controls (physical and logical) beyond those described in this article may also be required, depending on the level of assurance that an organization requires.

Steps for Configuring the Two-Person Rule Using RBAC

This section describes the following steps to configure the two-person rule using RBAC:

- Step 1: Verify Users
- Step 2: Verify User Privileges
- Step 3: Grant the Audit Review Rights Profile to Users
- Step 4: Create New Roles
- Step 5: Assign Restricted Operations to the Roles
- Step 6: Verify Role Configuration
- Step 7: Verify User Actions in Audit Trails

Step 1: Verify Users

This example begins by verifying that the two user accounts (`joe` and `john`) are already configured on the system.

```
# getent passwd joe john
joe:x:105:1:::/export/home/joe:/bin/pfsh
john:x:106:1:::/export/home/john:/bin/pfsh
```

Note – These users have been configured with profile shells in this example to simplify some of the commands that follow. A user must either have a profile shell or prefix commands with `pfexec(1)` in order to have the commands be evaluated by Solaris RBAC to run with additional privileges.

Step 2: Verify User Privileges

```
# roles joe john
joe : No roles
john : No roles

# profiles -l joe john

    All:
        *

    All:
        *

# auths joe john
joe :
solaris.device.cdrw,solaris.profmgr.read,solaris.jobs.users,solaris.mail.
mailq,solaris.admin.usermgr.read,solaris.admin.logsvc.read,solaris.admin.f
smgr.read,solaris.admin.serialmgr.read,solaris.admin.diskmgr.read,solaris.
admin.procmgr.user,solaris.compsys.read,solaris.admin.printer.read,solaris
.admin.prodreg.read,solaris.admin.dcmgr.read,solaris.snmp.read,solaris.proj
ect.read,solaris.admin.patchmgr.read,solaris.network.hosts.read,solaris.a
dmin.volmgr.read
john :
solaris.device.cdrw,solaris.profmgr.read,solaris.jobs.users,solaris.mail.
mailq,solaris.admin.usermgr.read,solaris.admin.logsvc.read,solaris.admin.f
smgr.read,solaris.admin.serialmgr.read,solaris.admin.diskmgr.read,solaris.
admin.procmgr.user,solaris.compsys.read,solaris.admin.printer.read,solaris
.admin.prodreg.read,solaris.admin.dcmgr.read,solaris.snmp.read,solaris.proj
ect.read,solaris.admin.patchmgr.read,solaris.network.hosts.read,solaris.ad
min.volmgr.read
```

As the output shows, neither `joe` nor `john` have been granted access to any roles, rights profiles, or authorizations beyond those available to any normal Solaris user. Several authorizations that are available

to them have been granted by default using the `AUTHS_GRANTED` and `PROFS_GRANTED` parameters in the `/etc/security/policy.conf` file.

Step 3: Grant the Audit Review Rights Profile to Users

Next, grant the `Audit Review` rights profile to each of these users. Doing so allows both `joe` and `john` to review Solaris audit records so that each can monitor the activities of the other. Note that this step is not necessary for implementing the two-person rule if an external third party is able to monitor the activities of these two users.

```
# usermod -P "Audit Review" joe
# usermod -P "Audit Review" john
```

Note – The use of the commands in the `Audit Review` profile depends on the implementation of Solaris Auditing. The example in this article assumes that Solaris Auditing has been enabled and configured. For more information on Solaris Auditing, refer to “Solaris Auditing” in the *Solaris 10 System Administration Guide: Security Services* at <http://docs.sun.com/app/docs/doc/816-4557/6maosrjoj?a=view> and “Auditing in the Solaris™ 8 Operating Environment,” by William Osser and Alex Noordergraaf (Sun BluePrints OnLine, February 2001) at http://www.sun.com/blueprints/0201/audit_config.pdf.

```
# profiles -l joe john

  Audit Review:
    /usr/sbin/praudit      euid=0
    /usr/sbin/auditreduce  euid=0
    /usr/sbin/auditstat    euid=0
  All:
    *

  Audit Review:
    /usr/sbin/praudit      euid=0
    /usr/sbin/auditreduce  euid=0
    /usr/sbin/auditstat    euid=0
  All:
    *
```

Step 4: Create New Roles

Next, create two new roles:

- `roleA` will be assigned to `joe`
- `roleB` will be assigned to `john`

Once created, restricted operations will be assigned to these roles.

```
# roleadd -d /export/home/roleA -m roleA

# passwd roleA
New Password:
Re-enter new Password:
passwd: password successfully changed for roleA

# roleadd -d /export/home/roleB -m roleB

# passwd roleB
New Password:
Re-enter new Password:
passwd: password successfully changed for roleB

# usermod -R roleA joe
# usermod -R roleB john
```

After creating and assigning the roles, verify that the changes have taken effect.

```
# getent passwd roleA roleB
roleA:x:107:1::/export/home/roleA:/bin/pfsh
roleB:x:108:1::/export/home/roleB:/bin/pfsh

# roles joe john
joe : roleA
john : roleB
```

Step 5: Assign Restricted Operations to the Roles

Next, assign the restricted operations to the roles so that `joe` and `john` can perform them after they have jointly assumed either `roleA` or `roleB`. This example assigns the `Crypto Management` rights profile to both `roleA` and `roleB`. Doing so allows either of those roles to execute cryptographic administration functions, such as those provided by `cryptoadm(1M)`:

```
# rolemod -P "Crypto Management" roleA
# rolemod -P "Crypto Management" roleB
```

To determine which privileged commands are available to these roles, use the `profiles` command.

```
# profiles -l roleA roleB

Crypto Management:
  /usr/sbin/cryptoadm      euid=0
  /usr/sfw/bin/CA.pl      euid=0
  /usr/sfw/bin/openssl     euid=0
All:
  *

Crypto Management:
  /usr/sbin/cryptoadm      euid=0
  /usr/sfw/bin/CA.pl      euid=0
  /usr/sfw/bin/openssl     euid=0
All:
  *
```

To summarize what has been configured thus far:

- Two users have been defined (`joe` and `john`).
- `joe` is permitted to assume `roleA`.
`john` is permitted to assume `roleB`.
- `joe` does not know the password for `roleA` (although `john` does).
`john` does not know the password for `roleB` (although `joe` does).
- Both of the users have been assigned the `Audit Review` rights profile, allowing them to monitor each other's activities.
- Both of the roles have been assigned the `Crypto Management` rights profile, allowing them to perform cryptographic management operations on the system.

Step 6: Verify Role Configuration

Next, verify that the roles have been configured as expected. In this example, all of the users and roles were created within a zone called `blue`.

The first task is to log into the zone as `joe`:

```

# zlogin -l joe blue
[Connected to zone 'blue' pts/2]
Last login: Tue Apr 12 07:20:19 on pts/2
joe$ id -a
uid=105(joe) gid=1(other) groups=1(other)
joe$ auths
solaris.audit.read,solaris.device.cdrw,solaris.profmgr.read,solaris.jobs.u
sers,solaris.mail.mailq,solaris.admin.usermgr.read,solaris.admin.logsvc.re
ad,solaris.admin.fsmgr.read,solaris.admin.serialmgr.read,solaris.admin.dis
kmgr.read,solaris.admin.procmgr.user,solaris.compsys.read,solaris.admin.pr
inter.read,solaris.admin.prodreg.read,solaris.admin.dcmgr.read,solaris.snm
p.read,solaris.project.read,solaris.admin.patchmgr.read,solaris.network.ho
sts.read,solaris.admin.volmgr.read
joe$ profiles -l

    Audit Review:
        /usr/sbin/praudit      euid=0
        /usr/sbin/auditreduce  euid=0
        /usr/sbin/auditstat    euid=0
    All:
        *
joe$ roles
roleA

```

Because `joe` does know the password for `roleB`, try to assume that role.

```

joe$ su roleB
Password:
Roles can only be assumed by authorized users
su: Sorry

```

Even with a valid password, `joe` is not permitted to assume `roleB`.

Next, try to guess the password for `roleA`.

```

joe$ su roleA
Password:
su: Sorry
joe$ su roleA
Password:
su: Sorry
joe$ su roleA
Password:
su: Sorry

```

Attempts at guessing fail. In order to prevent brute force password attempts, *account lockout* could also be configured so that an account is administratively locked after *n* consecutive failed authentication attempts (where *n* is the value defined by the `RETRIES` parameter in `/etc/default/login`). For more information, see Glenn Brunette's Solaris 10 OS Security article, "Solaris 10 Account Lockout (Three Strikes)," at <http://blogs.sun.com/roller/page/gbrunett/20040923>.

Step 7: Verify User Actions in Audit Trails

Finally, examine what user `john` would see in the audit trails after `joe` executed the above commands. Because `john` has been granted the `Audit Review` rights profile, he is able to look at the Solaris audit records. To further limit what `john` could see in the audit trail, a small wrapper script could be developed that called `praudit(1M)` and `auditreduce(1M)` to filter out unwanted records.

Start with `john` logging into the `blue` zone.

```
# zlogin -l john blue
[Connected to zone 'blue' pts/2]
john$ id -a
uid=106(john) gid=1(other) groups=1(other)
john$ auths
solaris.audit.read,solaris.device.cdrw,solaris.profmgr.read,solaris.jobs.u
sers,solaris.mail.mailq,solaris.admin.usermgr.read,solaris.admin.logsvc.re
ad,solaris.admin.fsmgr.read,solaris.admin.serialmgr.read,solaris.admin.dis
kmgr.read,solaris.admin.procmgr.user,solaris.compsys.read,solaris.admin.pr
inter.read,solaris.admin.prodreg.read,solaris.admin.dcmgr.read,solaris.snm
p.read,solaris.project.read,solaris.admin.patchmgr.read,solaris.network.ho
sts.read,solaris.admin.volmgr.read
john$ profiles -l

    Audit Review:
        /usr/sbin/praudit      euid=0
        /usr/sbin/auditreduce  euid=0
        /usr/sbin/auditstat    euid=0
    All:
        *
john$ roles
roleB
```

Next, `john` uses the `praudit` and `auditreduce` commands to look at any attempts to `su` taken by `joe`.

```
john$ auditreduce -m AUE_su -r joe | praudit -s
file,2005-04-12 07:25:06.000 -04:00,
header,97,2,AUE_su,,10.8.31.9,2005-04-12 07:28:30.220 -04:00
subject,gmb,root,other,joe,other,1834,3097759606,12114 22 129.150.66.247
text,bad auth. for user roleB
return,failure,2
header,97,2,AUE_su,,10.8.31.9,2005-04-12 07:28:40.043 -04:00
subject,gmb,root,other,joe,other,1835,3097759606,12114 22 129.150.66.247
text,bad auth. for user roleA
return,failure,2
header,97,2,AUE_su,,10.8.31.9,2005-04-12 07:28:49.940 -04:00
subject,gmb,root,other,joe,other,1836,3097759606,12114 22 129.150.66.247
text,bad auth. for user roleA
return,failure,2
header,97,2,AUE_su,,10.8.31.9,2005-04-12 07:28:59.683 -04:00
subject,gmb,root,other,joe,other,1837,3097759606,12114 22 129.150.66.247
text,bad auth. for user roleA
return,failure,2
file,2005-04-12 07:28:59.000 -04:00,
```

This audit trail shows that user `joe` attempted to assume both `roleA` and `roleB`. In fact, `joe` attempted to assume `roleB` three times, with each attempt ending in failure.

For more information on the audit logs, their format, and options for configuring them, see `auditd(1M)`. For the purposes of this example, Solaris auditing was configured with the following policies:

```
# auditconfig -getpolicy
audit policies = argv,cnt,perzone
```

The users and roles defined in this example were audited as follows.

```
# cat /etc/security/audit_user
#
# Copyright (c) 1988 by Sun Microsystems, Inc.
#
# ident "@(#)audit_user.txt      1.6      00/07/17 SMI"
#
#
# User Level Audit User File
#
# File Format
#
#      username:always:never
#
root:lo:no
joe:lo,ad,ex:
john:lo,ad,ex:
roleA:lo,ad,ex:
roleB:lo,ad,ex:
```

These settings effectively audited login and logout events, administrative actions, and command execution (`exec(2)` calls). Solaris auditing should be configured to comply with an organization's auditing requirements and policies. The configuration described in this article serves as just one example of the type of information that can be collected.

Note that `joe` cannot access the `Crypto Management` rights profile on his own. What would happen if `joe` attempted to simply perform a restricted cryptographic operation on his own?

```
joe$ cryptoadm stop
cryptoadm: failed to stop cryptographic framework daemon - Not owner.
```

As expected, in order to perform those restricted operations, `joe` needs `john` to help him assume `roleA`.

Working in concert, `joe` and `john` together can successfully assume `roleA`.

```
joe$ su roleA
Password:
roleA$ id -a
uid=107(roleA) gid=1(other) groups=1(other)
roleA$ profiles -l

    Crypto Management:
        /usr/sbin/cryptoadm      euid=0
        /usr/sfw/bin/CA.pl       euid=0
        /usr/sfw/bin/openssl     euid=0
    All:
        *
```

With these privileges, `joe` and `john` can perform such operations as starting and stopping the Solaris cryptographic framework.

```
roleA$ echo "foo" | digest -a md5
d3b07384d113edec49eaa6238ad5ff00
roleA$ cryptoadm stop
roleA$ echo "foo" | digest -a md5
digest: failed to initialize PKCS #11 framework: CKR_GENERAL_ERROR
roleA$ cryptoadm start
roleA$ echo "foo" | digest -a md5
d3b07384d113edec49eaa6238ad5ff00
```

Summary

The two-person rule is enforced because neither `joe` nor `john` can perform any cryptographic management operations on the system without the knowledge and support of the other. Further, should either user attempt to access their assigned role, the other user can be alerted through the presence of a Solaris audit record that neither user can modify.

About the Author

Glenn Brunette is a Sun Distinguished Engineer with nearly 15 years' experience in information security. Glenn works in the Client Solutions division as the Chief Security Architect for the Global Data Center Practice. In this role, Glenn is responsible for global security strategy, as well as for improving the quality and security of consulting solutions delivered to Sun's customers.

Glenn is the co-founder of the Sun Solaris Security Toolkit software and a frequent author and contributor to the Sun BluePrints program. Glenn works closely with teams across Sun on the development of security strategy, products, services, methodologies, best practices, training, certifications, and tools.

Externally, Glenn is currently the Vice-Chair of the Enterprise Grid Alliance Security Working Group and has served as Champion for the Common Configurations Working Group of the National Cyber Security Partnership's Technical Standards and Common Criteria Task Force. Glenn is also an active contributor to the Center for Internet Security's Unix Benchmark team.

Glenn is a Certified Information Systems Security Professional (CISSP) and has been trained in the National Security Agency's INFOSEC Assessment Methodology (IAM).

Acknowledgements

The author would like to thank Collin Sampson and Scott Rotondo for their inspiration, technical feedback, and overall support in the development of this article.

References

Publications

- Sun Microsystems, Inc. "Roles, Rights Profiles, and Privileges." Solaris 10 Product Documentation.
<http://docs.sun.com/app/docs/doc/816-4557/6maosrjfe?a=view>
- Sun Microsystems, Inc., "Solaris Auditing." Solaris 10 Product Documentation.
<http://docs.sun.com/app/docs/doc/816-4557/6maosrjoj?a=view>
- <http://www.sun.com/blueprints/0603/817-3062.pdf>
BluePrints OnLine, June 2003.
- Osser, William and Noordergraaf, Alex, "Auditing in the Solaris™ 8 Operating Environment," Sun BluePrints OnLine, February 2001.
http://www.sun.com/blueprints/0201/audit_config.pdf

Web Sites

- Glenn Brunette's Solaris 10 OS Security Weblog
<http://blogs.sun.com/gbrunett?catname=Solaris%2010%20Security>

Command Manual Pages

- Sun Microsystems, Inc. "auditconfig(1M) Manual Page", Sun Solaris 10 OS Product Documentation
<http://docs.sun.com/app/docs/doc/816-5166/6mbb1kptv?a=view>
- Sun Microsystems, Inc. "auditd(1M) Manual Page", Sun Solaris 10 OS Product Documentation
<http://docs.sun.com/app/docs/doc/816-5166/6mbb1kpu0?a=view>
- Sun Microsystems, Inc. "auditreduce(1M) Manual Page", Sun Solaris 10 OS Product Documentation
<http://docs.sun.com/app/docs/doc/816-1055/6m7gh31eh?l=ko&a=view>
- Sun Microsystems, Inc. "auths(1) Manual Page", Sun Solaris 10 OS Product Documentation
<http://docs.sun.com/app/docs/doc/816-5165/6mbb0m9bq?a=view>

- Sun Microsystems, Inc. “cryptoadm(1M) Manual Page”, Sun Solaris 10 OS Product Documentation
<http://docs.sun.com/app/docs/doc/816-5166/6mbb1kpvn?a=view>
- Sun Microsystems, Inc. “exec(2) Manual Page”, Sun Solaris 10 OS Product Documentation
<http://docs.sun.com/app/docs/doc/816-5167/6mbb2jafk?a=view>
- Sun Microsystems, Inc. “getent(1M) Manual Page”, Sun Solaris 10 OS Product Documentation
<http://docs.sun.com/app/docs/doc/816-0211/6m6nc66rj?a=view>
- Sun Microsystems, Inc. “id(1M) Manual Page”, Sun Solaris 10 OS Product Documentation
<http://docs.sun.com/app/docs/doc/816-0211/6m6nc66s5?a=view>
- Sun Microsystems, Inc. “passwd(1) Manual Page”, Sun Solaris 10 OS Product Documentation
<http://docs.sun.com/app/docs/doc/816-0210/6m6nb7mgn?a=view>
- Sun Microsystems, Inc. “praudit(1M) Manual Page”, Sun Solaris 10 OS Product Documentation
<http://docs.sun.com/app/docs/doc/816-5166/6mbb1kqc0?a=view>
- Sun Microsystems, Inc. “pfexec(1) Manual Page”, Sun Solaris 10 OS Product Documentation
<http://docs.sun.com/app/docs/doc/816-5165/6mbb0m9o5?a=view>
- Sun Microsystems, Inc. “profiles(1) Manual Page”, Sun Solaris 10 OS Product Documentation
<http://docs.sun.com/app/docs/doc/816-0210/6m6nb7mi9?a=view>
- Sun Microsystems, Inc. “roleadd(1M) Manual Page”, Sun Solaris 10 OS Product Documentation
<http://docs.sun.com/app/docs/doc/816-0211/6m6nc6754?a=view>
- Sun Microsystems, Inc. “rolemod(1M) Manual Page”, Sun Solaris 10 OS Product Documentation
<http://docs.sun.com/app/docs/doc/816-5166/6mbb1kqdn?a=view>
- Sun Microsystems, Inc. “roles(1) Manual Page”, Sun Solaris 10 OS Product Documentation
<http://docs.sun.com/app/docs/doc/816-0210/6m6nb7mjn?a=view>
- Sun Microsystems, Inc. “smc(1M) Manual Page”, Sun Solaris 10 OS Product Documentation
<http://docs.sun.com/app/docs/doc/816-5166/6mbb1kqfk?a=view>
- Sun Microsystems, Inc. “su(1M) Manual Page”, Sun Solaris 10 OS Product Documentation
<http://docs.sun.com/app/docs/doc/816-5166/6mbb1kqhg?a=view>
- Sun Microsystems, Inc. “usermod(1M) Manual Page”, Sun Solaris 10 OS Product Documentation
<http://docs.sun.com/app/docs/doc/816-5166/6mbb1kqjj?a=view>
- Sun Microsystems, Inc. “zlogin(1) Manual Page”, Sun Solaris 10 OS Product Documentation
<http://docs.sun.com/app/docs/doc/816-5165/6mbb0ma1v?a=view>



Enforcing the Two-Person Rule Via
Role-Based Access Control in the
Oracle Solaris 10 Operating System

August 2005
Author: Glenn Brunette

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200

oracle.com



Oracle is committed to developing practices and products that help protect the environment

Copyright © 2011, Oracle and/or its affiliates. All rights reserved. This document is provided for information purposes only and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. UNIX is a registered trademark licensed through X/Open Company, Ltd. 1010

Hardware and Software, Engineered to Work Together