Maintaining Network Separations with Trusted Solaris™ 8 Software

By Glenn Faden - Solaris Security Technology Group

Sun BluePrints™ OnLine -

http://www.sun.com/blueprints

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303 USA
650 960-1300 fax 650 969-9131

Part No.: 816-0246-10
Revision 01, mm / dd / 01
Edition: month 2001

Please
Recycle

Adobe PostScript

Maintaining Network Separation with Trusted Solaris™ 8 Software

Multilevel systems like Trusted Solaris Operating Evironment (Trusted Solaris OE) have been around for nearly a decade, but only now are they being widely deployed for web-based environments. The term *multilevel* refers to the property that access control is based on sensitivity levels or *labels*, not just user and group identities. The ability to connect to multiple networks and prevent any unauthorized information flows between them is a capability unique to such systems. By associating labels with each network and applying a policy known as Mandatory Access Control (MAC), Trusted Solaris OE enforces data separation based on the sensitivity of these labels. Since labels have both a hierarchical classification and non-hierarchical compartments or categories, they can be applied to most organizational structures.

This article describes how MAC can be used to provide concurrent access to two isolated networks without compromising that separation. The reader is assumed to be familiar with network administration in the Solaris™ Operating Environment (Solaris OE) and have a general familiarity with trusted systems. However, the reader may need to refer to the bibliography for more information on subjects such as Role-Based Access Control (RBAC), least privilege, and label relationships, which are mentioned in this article, but not explained in great detail.

Polyinstantiation of System Resources

Although the MAC policy in the Trusted Solaris OE is applied pervasively to all subjects and objects, it is usually transparent. In addition, applications do not have to be specially written to take advantage of labeling. Labels can provide complete isolation between separate domains of execution through a feature known as *polyinstantiation*. While the Trusted Solaris 8 OE preserves the Solaris OE application binary interface (ABI), it presents an environment which is similar to running

separate operating environments on one machine. In general, applications running at different labels are unaware of each other and can run concurrently without regard to synchronization or contention issues.

For example, in the Trusted Solaris version of CDE, each window and workspace has an associated label. A telecommuter using the Trusted Solaris OE on a home workstation can connect to his or her company's private intranet through one network interface, and connect to an Internet Broadband carrier through another interface. In this case each CDE workspace corresponds to a separate network. When the Trusted Solaris OE is properly configured, no information flow is permitted between these two networks. However, if authorized, the telecommuter is permitted to cut and paste data between separately labeled windows associated with the two networks. Each such operation must be manually reviewed and approved by the authorized user in a window that pops up for this purpose. Other than this feature, it is as if the telecommuter was using a separate machine for each connection.

This separation also applies to network services. Using the Trusted Solaris OE as a web server allows multiple secured, independent web environments on the same machine. This is useful for situations where multiple organizations are collocating web services. An unmodified web server such as Apache HTTP server can be configured to execute concurrently at separate labels, with one instance corresponding to each network. Clients from the Internet would be served by an instance of the web server which could only provide information dominated by the Internet label. Clients from the intranet would be similarly served by a separate instance of the web server running with the intranet label. The two servers could share the same port number and directories because the Trusted Solaris OE provides separately labeled instances of ports and directories.

In some cases, there are some extra administrative steps required to facilitate this separation. This article describes some of the network configuration parameters that affect the flow of information between the networks. It also describes some of the resources that need to be polyinstantiated.

Sample Configuration

This section looks at a sample configuration consisting of two isolated networks that are connected to a single Trusted Solaris system. The primary network interface, `le0`, is connected via a Virtual Private Network (VPN) gateway router to a private corporate network. In this example, we will assign a single label to all data sent and received through this network. An arbitrary name, PRIVATE INTRANET, is used as the external or long form of the label; for typing convenience, a short form, PRIVATE, is also used.

The second network interface, `hme0`, is connected to a small network with a DHCP server running on a gateway router. The external end of this router is connected to the Internet via a Broadband service. This network is assigned a label called PUBLIC INTERNET, or PUBLIC for short. As a multi-homed host, the Trusted Solaris system has two hostnames. In this example (see FIGURE 1) we will call the primary host `ultra2`, and the secondary host `dhcppc3` (a name which was assigned by the gateway router).

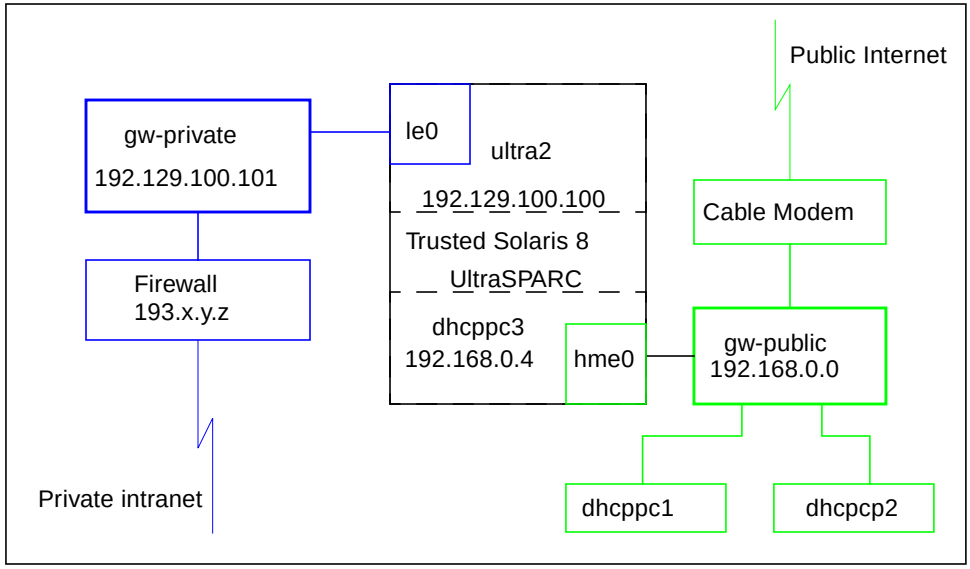

FIGURE 1 Network Block Diagram

The following hosts are defined in `/etc/hosts`.

```
#
# Internet host table
#
127.0.0.1      localhost
0.0.0.0        tsol-default
192.129.100.100 ultra2 loghost  # primary tsol hostname
192.168.0.4    dhcppc3        # 2nd tsol hostname
192.129.100.101 gw-private    # VPN Router
192.168.0.1    gw-public      # DHCP server/router
192.168.0.2    dhcppc1        # home desktop PC
192.168.0.3    dhcpcp2        # home laptop PC
```

The networks are identified in their respective `hostname.xxx` files.

The file `/etc/hostname.le0` contains:

```
ultra2
```

The file `/etc/hostname.hme0` contains:

```
dhcppc3
```

Although DHCP is being used on this interface, the IP address of the interface cannot be changed without corresponding changes in the trusted networking configuration. Therefore the IP address was granted with an indefinite lease by an explicit `ifconfig` request rather than relying on `sysidtool`(1M).

Labels and Accreditation Ranges

In the Trusted Solaris OE, all subjects and objects including files, processes, network endpoints, and windows, are assigned a label representing their sensitivity.

Note – This article does not describe how labels are constructed or maintained in the Trusted Solaris OE. It is sufficient to understand that labels can be compared hierarchically for dominance, equality, or inequality. A label *dominates* another if its classification is greater and it includes all the compartments (categories) of the other.

Any single label can be compared against a range of labels which consist of an upper and lower bound, known as an accreditation range. Accreditation ranges are associated with hosts and network interfaces and are used to restrict the flow of data. The Trusted Solaris OE will route packets based on their labels and the accreditation range of available interfaces.

The Trusted Solaris OE hosts support a range of labels and are termed *multilevel* or sometimes just *labeled* machines. Network packets exchanged between the Trusted Solaris OE hosts are explicitly labeled by the sender. Other hosts are considered to be *unlabeled* and must be assigned a default label by the Trusted Solaris OE. The assignment can be done on an individual host, a subnet, or an entire network.

Excluding Specific Hosts

If the label assigned to a host is outside the accreditation range of a network interface, then that interface cannot be used to communicate with that host. Therefore, it is possible to use labeling for access control in a manner similar to Wiets

Venema's TCP Wrappers (see bibliography for the URL). Like TCP Wrappers, accreditation ranges can be used to limit the hosts which may connect to a system. Machines that are untrusted or unwelcome can be assigned a label which is outside of the accreditation range of one or more network interfaces. Such accreditation failures are logged, and the packets are dropped.

Note – A bug existed in the handling of accreditation failures for undeliverable packets in the Trusted Solaris 8 OE. The bug number is 4401871, and a fix is included in patch #110337-01.

Network interfaces are assigned label ranges using the trusted network interface database, `/etc/security/tsol/tnidb`.

The following `tnidb` entries define the security attributes for the two interfaces.

```
le0:forced_privs=none;min_sl=PRIVATE;\
def_cl=PRIVATE;def_label=PRIVATE;max_sl=PRIVATE
#
hme0:forced_privs=none;min_sl=PUBLIC;\
def_cl=PUBLIC;def_label=PUBLIC;max_sl=PUBLIC
```

In this example, the interfaces are configured with minimum (`min_sl`) and maximum labels (`max_sl`) set equal to each other, so only that one label is allowed. The default label (`def_label`) and default clearance (`def_cl`) specifications are redundant here because defaults are actually set up in the trusted hosts and trusted template databases.

When applying changes to these entries, it is useful to verify that the changes have been pushed into the kernel. The current kernel values can be queried using the `tnifno`(1M) command.

```
ultra2# tninfo -i
==================
Interface Table Entries:
_____________________________
Interface name: le0
min_sl: PRIVATE
hex: 0x00040c0000000000000000000000000000000000000000000000000000
max_sl: PRIVATE
hex: 0x00040c0000000000000000000000000000000000000000000000000000
def_label: PRIVATE
hex: 0x00040c0000000000000000000000000000000000000000000000000000
def_cl: PRIVATE
hex: 0x00040c0000000000000000000000000000000000000000000000000000
forced_privs: none
_____________________________
Interface name: hme0
min_sl: PUBLIC
hex:
0x0001000000000000000000000000000000000000000000000000000000000000
max_sl: PUBLIC
hex:
0x0001000000000000000000000000000000000000000000000000000000000000
def_label: PUBLIC
hex:
0x0001000000000000000000000000000000000000000000000000000000000000
def_cl: PUBLIC
hex: 0x0000000000000000000000000000000000000000000000000000000000000000
forced_privs: none
```

Routing

In the Trusted Solaris OE, each network acknowledges its own router. The routes for these two networks are defined in the file `/etc/tsolgateways`, whose contents are shown below. All intranet machines have the 193.0.0.0 prefix, and are routed through the private router. The public router handles all other traffic.

```
default gw-public 1 -m metric=1,min_sl=PUBLIC,max_sl=PUBLIC
net 193.0.0.0 gw-private 1 \
-m metric=1,min_sl=PRIVATE,max_sl=PRIVATE
```

The label ranges specified in /etc/tsolgateways are not required in this example, but are included for clarity and completeness. The file is interpreted by route(1M) during system initialization. The extended contents of the routing table can be viewed using the -R option to netstat(1M):

```
ultra2# netstat -R

Routing Table:
IPv4
  Destination          Gateway         FlagsRef   Use   Interface
-------------------- --------------- ----- ----- ------ ---------
192.129.100.0        ultra2            U      1     14    le0
192.168.0.0          dhcppc3           U      1     16    hme0
193.0.0.0            gw-private      UG     1    137
  metric=1,minsl=PRIVATE,max_sl=PRIVATE
224.0.0.0            ultra2            U      1      0    le0
default              gw-public        UG     1    182
  metric=1,minsl=PUBLIC,max_sl=PUBLIC
localhost            localhost         UH    66   2480    lo0
```

To help ensure that IP forwarding between interfaces is prohibited, it is recommended that the file /etc/notrouter be created. However, as you will see in the trusted networking configuration files, such routing is actually prevented by MAC.

Host Security Attributes

The network security attributes of hosts are defined in named templates. Three such templates are defined for this system. The tsol template (not shown) applies to the Trusted Solaris IP addresses and is the standard template distributed with the system. The custom templates are Internet, which uses the PUBLIC label, and Private, which applies to the PRIVATE label. These templates are maintained in the /etc/security/tsol/tnrhtp table., shown below

```
Public:def_cl=PUBLIC;def_label=[PUBLIC];max_sl=PUBLIC;\
forced_privs=empty;min_sl=PUBLIC;host_type=unlabeled;
#
Private:def_cl=PRIVATE;def_label=[PRIVATE];max_sl=PRIVATE;\
forced_privs=empty;min_sl=PRIVATE;host_type=unlabeled;
```

The hosts and networks are assigned labels in the trusted network host table, `/etc/security/tsol/tnrhdb`.

```
127.0.0.1:tsol
0.0.0.0:Public          # This is the default value
193.0.0.0:Private       # This matches IP address 193.xxx.xxx.xxx
192.168.0.4:tsol
192.129.100.100:tsol
192.129.100.101:Private
```

The trusted network daemon, `tnd`(1M) updates the kernel trusted networking tables. It typically polls the `tnrhdb` and `tnrhtp` databases every 30 minutes. The polling interval can be shortened to one minute as follows:

```
ultra2# tnd -p 60
```

Alternatively, the database can be pushed into the kernel using the `tnctl`(1M) command.

```
ultra2# tnctl -H /etc/security/tsol/tnrhdb
```

Administrative Tools

Trusted Solaris software provides Solaris Management Console™ (SMC) tools (see FIGURE 2) for managing most of the databases discussed in this article. The Security Families tool provides support for the `tnrhdb` and `tnrhtp` tables.

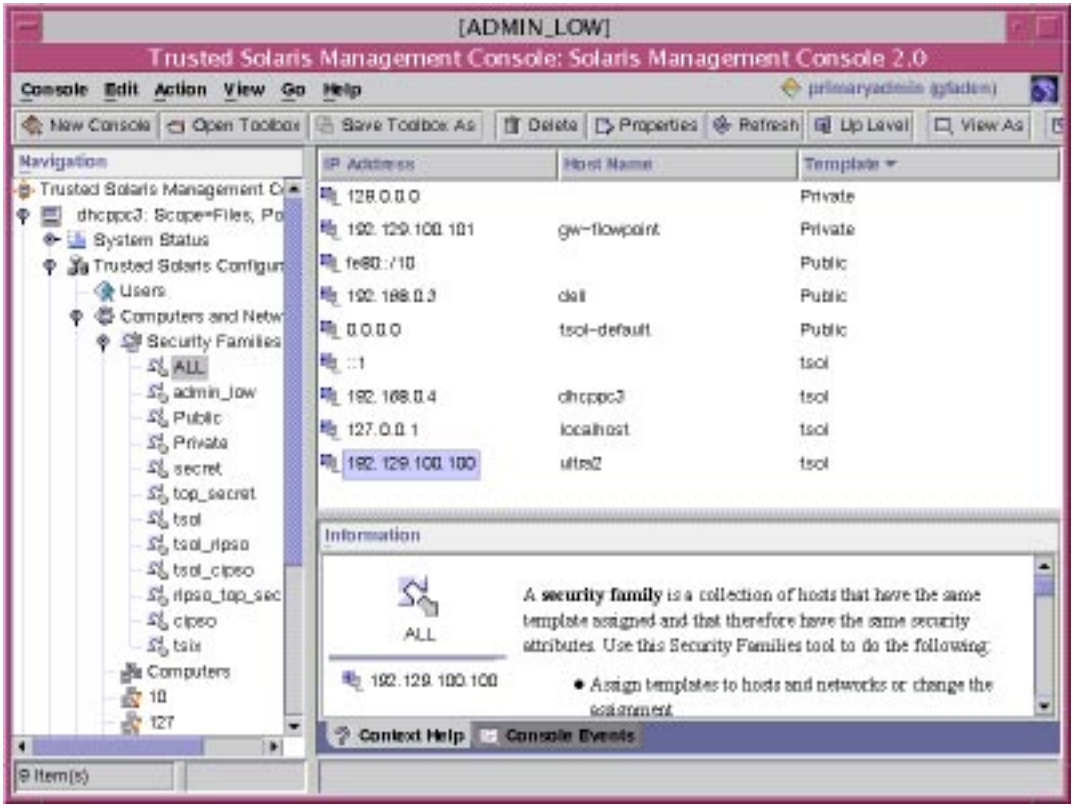

FIGURE 2 The Solaris Management Console Security Families tool

Domains and Name Resolution

A separate Internet domain is associated with each network. Normally a Solaris OE machine has a single local domain, but a Trusted Solaris system may have a local domain for each label, e.g. one per network. The mapping of a hostname to an IP address depends on the label at which the request is being made. While the file `/etc/resolv.conf` provides a list of nameservers that will be contacted for name resolution, each network provides its own set of nameservers, and hostnames must be qualified by the appropriate domains.

A separate copy of `/etc/resolv.conf` is required for each network.

Note – In the Trusted Solaris OE, any pathname can be polyinstantiated. A special directory type known as a Multilevel Directory (MLD) provides a namespace in which the actual directory associated with a pathname depends on the label in effect when the pathname is looked up.

This is accomplished by creating a symlink for `/etc/resolv.conf` which points to a file in a multilevel directory, as follows:

```
ultra2# mkdir -M /etc/resolvd.d
ultra2# ln -s /etc/resolv.conf /etc/resolv.d/resolv.conf
```

The `-M` option to `mkdir`(1) creates an MLD, and the symbolic link will resolve to a unique instance of that directory corresponding to the label in effect when the file is referenced. Each instance of `resolv.conf` is actually in a unique directory, and the instance at the PUBLIC INTERNET label looks like this:

```
domain marin1.sfba.home.com.
nameserver 192.168.0.1
```

By contrast, the instance of `resolv.conf` at the PRIVATE INTRANET label looks like this:

```
domain mycompany.com.
nameserver 193.150.111.9
nameserver 193.150.111.19
```

DNS is required by both networks, so the `/etc/nsswitch.conf` file is modified to include the DNS service for hostname resolution:

```
hosts: files dns
```

Since the set of hosts to IP addresses is different depending on the label and network, the hosts should not be cached in the nameservice. This is done by modifying the `nscd` configuration file, `/etc/nscd.conf`.

```
enable-cache hosts no
```

There is no need to set any domain in `/etc/defaultdomain`.

Polyinstantiating Servers

The labels of a client and server must be equal to enable communication. The Trusted Solaris OE provides system calls which can be used by a privileged server to determine the label of its client and to set its own connection label to match. Furthermore, the Trusted Solaris OE provides special network privileges that will allow unmodified servers to respond at matching labels automatically. However, in either case the server must be trusted not to abuse its privilege.

In this section, we discuss another approach which does not require trusting the server. If such a server is compromised, it has no effect on the rest of the system. As was done for DNS resolution, we rely on polyinstantiation, running a separate server instance for each label at which clients can connect. We will use the Apache HTTP server software as the example since it is bundled with Trusted Solaris software and is widely used. We will run the Apache HTTP server as the user `nobody`.

The Trusted Solaris OE provides two mechanisms for starting servers at specified labels. The first depends on `inetd`, which has been extended to automatically start servers at the label of the requesting client. However, since Apache is normally started from an `rc` script, we will focus on supplying attributes to such boot scripts.

▼ Using MLDs for Polyinstantiation

The first step is to polyinstantiate any resources that must be written by the server or are unique to a particular server instance. The port number used by Apache HTTP server is automatically polyinstantiated by the Trusted Solaris OE, so it can be used for all instances of the Apache HTTP server. As before, we use MLDs to separate the private files for each label. Apache HTTP server typically uses the path `/var/apache/logs` to store its error and access logs. This directory can be turned into a MLD as follows:

```
ultra2# rm /var/apache/logs/*
ultra2# setfattrflag -m /var/apache/logs
```

The directory must be emptied before it can be turned into a MLD. If it is desirable to change the ownership of the directory, this change should be done prior to making it an MLD; however, it is initially owned by `nobody` which is what we want.

The Apache HTTP server keeps track of its existence by writing to the file `/var/run/httpd.pid`. This directory is already an MLD in the Trusted Solaris OE, but, as in the Solaris OE, is only writable by `root`. In the Solaris OE, this is sufficient because the `httpd` is started as `root` and changes to `nobody` after initialization. However, in the Trusted Solaris OE, there is no superuser, so an unprivileged server could not change its user id to `nobody`. Instead, we create a subdirectory, `/var/run/apache` for this purpose. This path must be updated in three files. The variable to modify in each file is included in parenthesis.

- `/etc/apache/http.conf` (`PidFile`)
- `/usr/apache/bin/apachectl` (`PIDFILE`)
- `/etc/init.d/apache` (`PIDFILE`)

▼ Starting Services with the System Shell

Since we are starting Apache at boot time, it will be started by the system shell, `sysh`(1M). The system shell uses the RBAC profile databases to specify the process attributes of programs it runs. The name of the profile is specified using the `setprof` built-in command. In this case, we use the name of the shell script as the name of the profile. The convention is to create start-up scripts in the `/etc/rc`*n*`.d` file by hard linking them to the `/etc/init.d` file; this convention allows us to provide a unique profile for each linked copy. The start-up script needs a few modifications:

```
#!/sbin/sysh
APACHE_HOME=/usr/apache
CONF_FILE=/etc/apache/httpd.conf
PIDFILE=/var/run/apache/httpd.pid
#
# Set profile for command attributes
#
setprof ‘basename $0‘
PIDFILEDIR=‘/bin/mldrealpath /var/run‘/apache
if [ ! -d ${PIDFILEDIR} ]; then
      /bin/mkdir -m 700 ${PIDFILEDIR}
fi
```

The `setprof` command specifies that subsequent command attributes will be looked up in a profile whose name matches the shellscript, e.g. `S50apache`. In this example, four commands are assigned special attributes.

Note – Profiles are normally maintained using the Rights Manager tool in the SMC.

The profile is declared in /etc/security/prof_attr.

```
S50apache:::Start Apache at PUBLIC INTERNET label.:help=none
```

The command attributes are specified in /etc/security/exec_attr.

```
S50apache:tsol:cmd:::/usr/bin/mldrealpath:label=PUBLIC;\
clearance=PUBLIC
#
S50apache:tsol:cmd:::/usr/bin/mkdir:uid=nobody;gid=nobody;\
label=PUBLIC;clearance=PUBLIC;privs=file_dac_write
#
S50apache:tsol:cmd:::/usr/bin/rm:uid=nobody;gid=nobody;\
label=PUBLIC;clearance=PUBLIC
#
S50apache:tsol:cmd:::/usr/apache/bin/apachectl:uid=nobody;\
privs=net_privaddr,proc_owner;gid=nobody;\
label=PUBLIC;clearance=PUBLIC
```

The command mldrealpath(1) returns the pathname of the specific instance of the MLD at the label specified in this profile. The commands mkdir(1), rm(1), and /usr/apache/bin/apachectl are specified to run at the label of the Apache HTTP server, and the user and group IDs nobody. The mkdir command is given the privilege file_dac_write because the parent directory, /var/run, is not writable by the user nobody. In this example, the apachectl command is also given two privileges, net_privaddr, and proc_owner. The net_privaddr privilege is only required if the Apache server is going to be bound to a privileged port, such as 80. The proc_owner privilege is only needed to stop or restart the server if it has this or any other privileges.

Note that the Apach HTTP server is explicitly started with the user and group IDs set to nobody. The Solaris OE convention is to specify these using the User and Group directives in the httpd.conf file. However, in the Trusted Solaris OE the server is not privileged to change its user or group IDs, so the system shell does this instead.

Since the Apache start-up script has another instance in the /etc/rc2.d file for use when changing init states, another profile, identical to S50apache, but named K16apache is required for this purpose.

▼ Starting Another Server Instance

Starting a second Apache HTTP server at the PRIVATE INTRANET label involves just two extra steps. We first create another set of hard links for the starting and stopping scripts.

```
ultra2# ln /etc/init.d/apache /etc/rc3.d/S51apache
ultra2# ln /etc/init.d/apache /etc/rc2.d/K17apache
```

Then we create a similar profile called `S51apache` (and `K17apache`), whose `exec_attr` entry looks like this:

```
S51apache:tsol:cmd:::/usr/bin/mldrealpath:
label=PRIVATE;clearance=PRIVATE
#
S51apache:tsol:cmd:::/usr/bin/mkdir:uid=nobody;gid=nobody;\
label=PRIVATE;clearance=PRIVATE;\
privs=file_dac_write
#
S51apache:tsol:cmd:::/usr/bin/rm:uid=nobody;gid=nobody;\
label=PRIVATE;clearance=PRIVATE
#
S51apache:tsol:cmd:::/usr/apache/bin/apachectl:uid=nobody;\
privs=net_privaddr,proc_owner;gid=nobody;\
label=PRIVATE;clearance=PRIVATE
```

Since the Apache HTTP server has no MAC privileges, it will only be able to read files whose labels are dominated by its process label.

Conclusion

The Trusted Solaris OE provides an environment in which multiple domains of execution can operate at different labels concurrently and independently. The MAC policy used to provide this separation is a powerful tool which is not subject to compromise. The techniques described in this article can be applied to many types of applications and services. For example, a single workstation can be used to remotely administer multiple domains, each with a unique label, while maintaining community separation.

While multilevel networking can be managed in a manner which is transparent to most applications, some types of functionality require special development efforts. For example, the Apache HTTP server could be modified to return content based on other attributes such as the user's authorization and/or clearance. Examples of such programming are described in the Trusted Solaris Programmer's Guide.

For a more complete description of the Trusted Solaris OE concepts, take a look at the Administrative Overview in the Trusted Solaris Answerbook.

Bibliography

Glenn Faden, *RBAC in UNIX Administration*, Proceedings of the fourth ACM workshop on RBAC, October 28 - 29, 1999, Fairfax, VA, http://www.acm.org/pubs/citations/proceedings/commsec/319171/p95-faden

Alex Noordegraaf, Keith Watson, *Solaris Operating Environment Network Settings for Security*, http://www.sun.com/blueprints/1299/network.html

Sun Microsystems, *Trusted Solaris™ 8 Operating Environment A Technical Overview*, December 200, http://www.sun.com/software/white-papers/wp-ts8/

Sun Microsystems, *Trusted Solaris™ 8 Answer Book*, November 2000, http://docs.sun.com/ab2/coll.175.4/

Sun Microsystems, *Trusted Solaris™ 8 Reference Manual*, November 200, http://docs.sun.com/ab2/coll.475.2/

Wietse Venema, *TCP Wrappers tool*, ftp://ftp.porcupine.org.pub/security/index.html

Apache HTTP Server Project, http://apache.org

Author's Bio: Glenn Faden

Glenn Faden has worked as an architect and technical contributor in the Trusted Solaris group at Sun Microsystems for over 12 years. His emphasis has been on user interfaces and window systems. He designed the multilevel versions of OpenWindows™ and the Common Desktop Environment, and the trusted administration tools used in the Trusted Solaris OE. Recently he has been focused on Role-Based Access Control (RBAC) and remote administration. The results of his efforts can be seen in the common RBAC framework between the Solaris and Trusted Solaris environments, and the new Solaris Management Console tools: User Account Manager, Administrative Role Manager and Rights Manager, which support mixed Solaris and Trusted Solaris environments.

Prior to joining Sun, Glenn did user interface design at Qubix Systems and Omicad Corportation, and operating systems developement at Gould Computer Systems.