Server Virtualization with Trusted Solaris™ 8 Operating Environment

By Glenn Faden - Solaris Security Technology Group

Sun BluePrints™ OnLine - February 2002

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.: 816-2928-10
Revision 1.0, 11/06/01
Edition: February 2002

Server Virtualization with Trusted Solaris™ 8 Operating Environment

This article builds on the concepts presented in our previous Sun BluePrints™ OnLine article, *Maintaining Network Separation With Trusted Solaris™ 8 Operating Environment*[1], which provided an introduction to the configuration of labeled networks. In this article, we expand on those techniques to show how the Trusted Solaris™ Operating Environment (OE) can be deployed by a network service provider to support multiple customers within a single infrastructure.

Through the use of appropriate Trusted Solaris software functionality, each customer appears to have its own virtual server, or community. This article describes some of the administrative procedures and configuration files that are required to set up fully contained communities. The configuration depends on some new functionality provided in the 4/01 update.

Using Labels for Containment

Containment is a critical requirement when hosting multiple clients in the same infrastructure. It must not be possible for the clients to interfere with each other or have any access to other's data. The mechanisms employed to implement this containment must provide high assurance and strength of protection. Trusted Solaris OE has been certified to meet the ITSEC F-B1 functionality requirements with an assurance level of E3. The current release of Trusted Solaris OE is also being evaluated using the Common Criteria protection profiles for Labeled Security(B1) and Role-Based Access Control, with an assurance level of EAL-4. The evaluation criteria are described in the Common Criteria Web site[2].

Labels are used to enforce a policy known as *Mandatory Access Control* (MAC). The policy is called *mandatory* because it is automatic and cannot be changed by normal users. This is in contrast to the *Discretionary Access Control* (DAC) provided in

standard operating environments, where the owner of data is responsible for access control. Labels are applied pervasively and automatically to all data objects and all information flows, including networking, file systems, windowing, and interprocess communication. Although many people tend to think of traditional labels like Top Secret, or Unclassified, labeling can be quite natural and straight-forward. In fact, in the network security provider environment, only the security administrator needs to know the names of the labels and their relationships.

Associating Labels with Customers

Label names and relationships are defined in a database known as `/etc/security/tsol/label_encodings`. Although the administrative tools deal with labels as textual elements, they are internally maintained in a binary representation. Labels consist of two components, a hierarchical classification and a non-hierarchical array of bits known as compartments.

In this article, we refer to a Service Provider database which is available for download from the Sun BluePrints Web site (`http://sun.com/blueprints/tools/tsolsp-scripts_license.html`). It provides for 100 independent client communities, arbitrarily named Customer1 to Customer100. The actual number of communities could be much larger. Although these names can easily be replaced with those of actual customers, it isn't required because the labels are not displayed by default. However, the installation of the `label_encodings` file is a prerequisite for most of the other steps described in this article; it is typically installed from a floppy disk after a CD-ROM installation of Trusted Solaris OE, or via Jumpstart™ software for a network installation.

In addition to the Customer labels, the database also includes the Public and Private labels that were discussed in the previous Sun BluePrints article. Finally, there are two special administrative labels which are implicitly included in every labels database. The following table summarizes the label names and functions.

TABLE 1 Summary of Service Provider Labels

Public Internet	This label is the minimum label which can be assigned to users
Private Intranet	This label can be assigned to normal users within a corporate network.
Customer1 (C1)	This label is used to isolate Customer1's data from the other customers. It has two compartments: • `HTTP FTP (Web Services)` represents the compartment in which the http server and anonymous ftp server are executed • `CGI` (`Web Scripts`) represents the compartment in which web scripts are executed

TABLE 1 Summary of Service Provider Labels

Customer2 (C2)	This label is used to isolate Customer2's data from the other customers. It has the same two compartments as Customer1: • `HTTP FTP (Web Services)` • `CGI (Web Scripts)`
Service Provider (SP)	This label is used as a maximum for all of the user labels.
Admin_Low	This is the lowest administrative label. Information at this label is read-only, except by authorized administrators. Users cannot execute processes at this label, so they cannot write to files or directories at this label.
Admin_High	This label is the highest administrative label. Information at this label may only be read by authorized users. It protects the audit trail and the labels database.

In the simplest case, a label *dominates* another if its classification is greater than or equal to the other and it contains all of the compartment bits of the other label. However, the relationship between labels can be *disjoint* when each label has at least one compartment bit not present in the other. Since there is no dominance relationship between disjoint labels, they are completely isolated from each other.

In the sample label database, the Private label and the Customer labels are all disjoint, but they all dominate the Public label and the `Admin_Low` label. The Service Provider label dominates everything except the `Admin_High` label, and can be used as a maximum when a network or a user is cleared to multiple levels. The complete dominance relationship is shown in FIGURE 1.

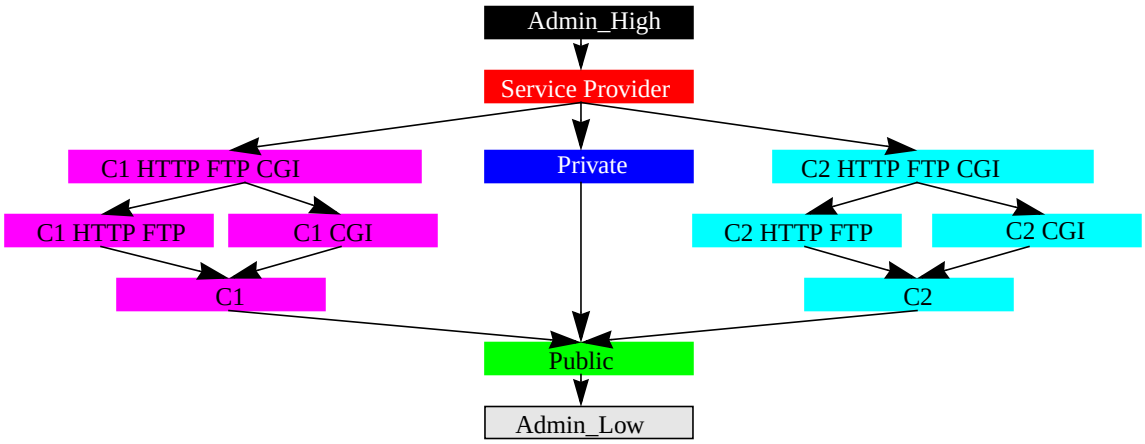

FIGURE 1 Label Dominance Relationships

Note that for each customer there are two disjoint labels, one for the web server and one for the CGI scripts. Each of these labels dominates the minimum customer label which is used to protect the customer's data.

These relationships are important because they provide the necessary containment. Each process runs with one of the labels shown in FIGURE 1. The label of the process prevents it from sending or receiving any information which would violate the MAC policy. For example, a process can only read a file if its label dominates that of the file, and can only write a file at its own label. The system is designed so that the policy is usually transparent and does not interfere with normal activity.

Labeled Processes

Generally a process' label is inherited from its parent process and remains constant. A few privileged launchers exist which can set a process label when executing a new process. These include the profiles shells (pfsh, pfksh, and pfcsh), the system shell (sysh), and the Internet daemon (inetd). Each of these processes interpret entries in the exec_attr database when executing programs.

All of the shells interpret the label keyword associated with a command when they execute a new process. inetd is able to launch servers at the same label as the requesting client. Such servers are said to be *polyinstantiated*, since a server instance can be launched corresponding to the label of each requesting client. The rules for determining the client's label are based on the contents of trusted networking databases that were discussed in the previous Sun BluePrints article[1]. In this article, we rely on inetd to polyinstantiate servers for ftp and telnet. Since this is the default behavior, no configuration changes are required.

The SMTP daemon, sendmail, has been modified in the Trusted Solaris OE to work in a polyinstantiated manner. It automatically forks off a process at the label of the requesting client. No special configuration is required, but a common configuration file is maintained by the service provider.

Creating a CGI Sandbox

Customers may want to develop and execute scripts on the server using the Common Gateway Interface (CGI) or Server Side Includes (SSI), so it is desirable to compartmentalize the web server and the CGI environment. Apache provides a small execution wrapper known as suEXEC, which allows scripts to be executed with a different user ID from the calling Web server. This concept can be extended to provide execution in a protected sandbox. By running the web server in the HTTP

FTP compartment, and the CGI scripts in the CGI compartment, the environments are isolated from each other except for a pipe to the web server used for standard I/O.

Neither the web server nor the CGI scripts are able to modify any files with the C1 label because they are not permitted to write down, but each can read these files unless they are protected by DAC. In addition, the web server and the CGI scripts cannot read any files created in the other compartment because neither dominates the other. The CGI environment is also prevented from connecting to any remote systems since its label is outside the accreditation range of all network interfaces connected to the Internet (see section, "Associating Labels with Network Interfaces," on page -7).

The suEXEC program is Open Source, so it can be modified to switch its label compartments prior to executing a CGI script. The modification consists of a few lines of C added to file suexec.c which are executed between fork(2) and exec(2). Since the MAC policy for files is enforced when a file or pipe is opened, it is possible to pass open file descriptors to a child process even if its process label is changed. The following code fragment turns off the HTTP FTP compartment and turns on the CGI compartment (error handling is omitted for clarity).

```
#include <tsol/label.h>
bclabel_t label;
int p;
getcmwplabel(&label);                          /* current label */
stobsl("- HTTP FTP", bcltosl(&label), 0, &p);  /* HTTP, FTP off */
stobsl("+ CGI", bcltosl(&label), 0, &p);       /* CGI on        */
setcmwplabel(&label, SETCL_SL);                /* set new label */
```

The system call getcmwplabel(2) returns the initial process label of suEXEC, which is the same as the web server. The function stobsl(3) translates a string to a binary label. In these calls it adds or removes the compartment bits corresponding to a specified string. The label translation requires the privilege proc_setsl. The resulting label is then applied to the current process by the system call setcmwplabel(2), which requires the privilege proc_setsl. Since no change is made to the classification component of the label, this single change works for any of the customer labels described above. A modified version of the file suexec.c which incorporates this logic is included in the configuration files for this article.

The standard suEXEC program, which sets up the new user ID before executing the specified commands, requires the privilege proc_setid. These privileges are applied to the executable using the following commands:

```
# cd /usr/apache/bin
# setfpriv -s -a all -f proc_setid,sys_trans_label,proc_setsl suexec
```

The Apache documentation for `suEXEC` describes how to make it a `setuid` program by assigning it the permission bits 4711. Although the Trusted Solaris OE privilege policy does not require this setting, the Apache server refuses to communicate with `suEXEC` unless these permissions are set.

An alternative approach to modifying `suEXEC` is to write a privileged profile shell script wrapper around it. Such a wrapper would rely on the profile shell functionality to set the process label instead of `suEXEC`. However, the complexity of creating multiple profiles and the requirement to make the shell script wrapper a `setuid` program, make it a less attractive solution than modifying `suEXEC` directly.

Instantiating Servlets and JavaServer Pages™ Software

Another kind of web server application is a Java™ servlet container such as Apache's Tomcat server. A servlet container manages and invokes servlets on the behalf of users. The Tomcat servlet container described here is a combination of a web server plug-in and a Java container implementation that runs in a Java virtual achine or JVM™ software outside the web server. The web server plug-in and the Java servlet container JVM software communicate using TCP/IP sockets; therefore the servlet container can run on a backend machine, which is also running the Trusted Solaris OE. Unlike connections to the public Internet, the interfaces between these two systems are multilevel. That means that the respective interfaces can be used to communicate at any label within their accreditation ranges. We can use the Tomcat server's default port number, 8007, for all customers, since the port is polyinstantiated. Each Tomcat server will run in its own JVM software, and will only accept connections from a web server running with the same label.

Note – Adding backend servers is optional, but is consistent with the Trusted Solaris architecture since the compartmentalization provided by the MAC policy is enforced throughout the trusted network.

Service Provider Network Configuration

Each customer is assigned two IP addresses which are created using logical interfaces. This is sometimes referred to as virtual hosting. The customer's web server binds to one of these IP addresses, and the other is used for administrative activities like uploading new data, and remote login. Although the number of interfaces can be scaled up for larger configurations, we will limit the discussion to

two customers for clarity. A multi-homed host with two interfaces, `hme0` and `hme1` is used. The interface `hme0` is connected to the Internet and has two logical interfaces per customer. TABLE 2 summarizes the configuration for two customers, with a multilevel interface to backend Tomcat servers.

TABLE 2 Functionality of Network Interfaces

Interface	IP address	Hostname	Services
`hme0:0`	192.168.0.1	`c1accounts`	telnet, ftp
`hme0:1`	192.168.0.2	`c1webserver`	http, anonymous ftp
`hme0:2`	192.168.0.3	`c2accounts`	telnet, ftp
`hme0:3`	192.168.0.4	`c2webserver`	http, anonymous ftp
`hme1`	192.10.10.1	`tsolweb`	admin
(separate host)			
`hme0`	192.10.10.2	`tsolasp`	JSP

Each logical interface is defined in a corresponding hostname file. For example the file `/etc/hostname.hme0:0` contains `c1accounts`, which, in turn, is defined in `/etc/inet/hosts`. The entries for `hostname.hme1` and `hostname.hme0` on the second host are automatically created since they represent the default interfaces on each of the two Trusted Solaris machines.

Associating Labels with Network Interfaces

Each network interface, whether real or logical, has four labels associated with it. In addition to the default label discussed above, there is a default clearance. These attributes are applied to connections which are initiated remotely. The policy for transmission through a network interface is based on two additional labels which specify an upper and lower bound, called the *accreditation range*.

Outgoing packets must be routed through an interface with an accreditation range containing the label of the packet, or they are dropped. Incoming packets are dropped unless they are within the accreditation range of the interface on which they are received. This policy can be contrasted to a firewall, which can block packets targeted at a specific host or port. In this configuration, processes with the `CGI` compartment are blocked from any communication through the logical interfaces, whether inbound or outbound.

Each of the interfaces with connections to the Internet will communicate using its specified default label. An additional interface for the internal use of the Service Provider will accept all labels because it will be communicating on a protected wire. Packets within the Service Provider's network will be explicitly labeled using the `tsol` protocol which transmits label attributes within each packet. The interface label assignments are summarized in TABLE 3.

TABLE 3 Default Labels and Accreditation Ranges for Network Interfaces

Interface	Default Label and Clearance	Minimum Label	Maximum Label
`hme:0`	C1	Admin_Low	C1
`hme0:1`	C1 HTTP FTP	Admin_Low	C1 HTTP FTP
`hme0:2`	C2	Admin_Low	C2
`hme0:3`	C2 HTTP FTP	Admin_Low	C2 HTTP FTP
`hme1`	Private Intranet	Admin_Low	Service Provider
(separate host)			
`hme0`	Private Intranet	Admin_Low	Service Provider

Note – In order to use dynamic routing on an interface, the minimum label must be `Admin_Low`. Routing protocols broadcasts, multicasts, and unicasts are always sent at the `Admin_Low` label.

These values are maintained in the trusted network interface database, `tnidb`. Like most of the databases which are unique to Trusted Solaris OE, it is stored in the directory `/etc/security/tsol`. Except for the `label_encoding` file discussed above, all of the databases which assign labels can be maintained by authorized administrators using GUI tools, command line utilities, or direct editing. The Solaris™ Management Consol software has been extended in the Trusted Solaris OE,

and provides full support for applying labels to network interfaces, hosts, commands, users, and roles. For example, the GUI to assign a label to a network interface is shown in FIGURE 2.

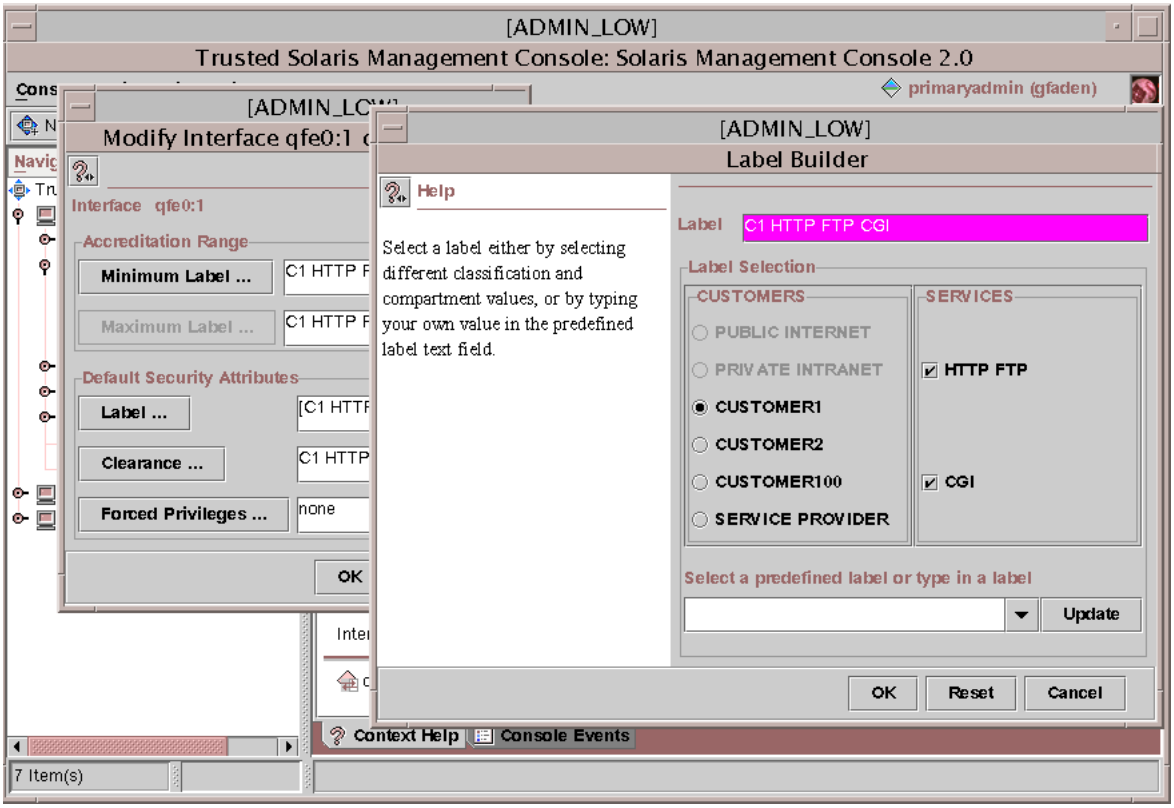

FIGURE 2 Solaris Management Console Interface Manager Label GUI

Associating Services With Interfaces

FIGURE 3 demonstrates how the data paths between the various services are compartmentalized.

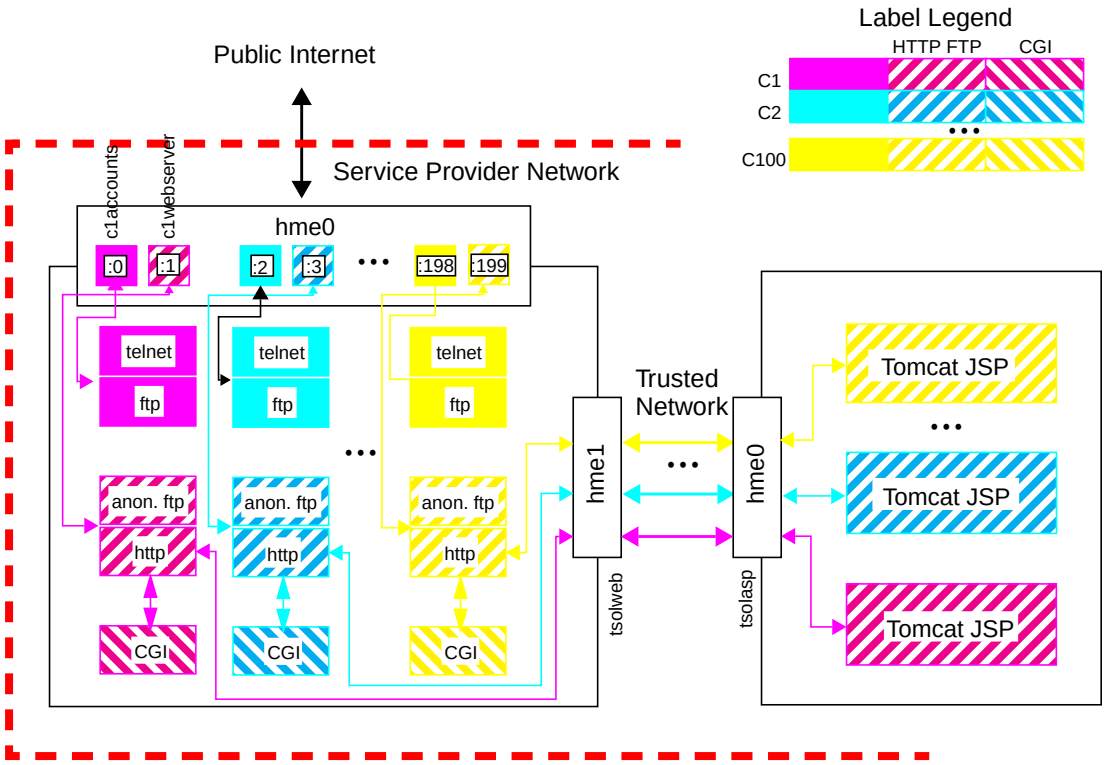

FIGURE 3 Virtualized Services with the Service Provider Network

From a client's perspective, each of these interfaces might as well be a separate machine because MAC rules prevent unprivileged services bound to the different network interfaces from communicating with each other. Furthermore, access to these interfaces can be restricted to authorized individuals through authentication policies and by limiting the services bound to these interfaces[3].

Associating Labels with Incoming Packets

Packets delivered to a Trusted Solaris OE are labeled upon receipt. The value of the label depends on several factors, including the attributes associated with the sender, and the network interface from which the packet was received. In this configuration,

the Trusted Solaris OE automatically labels each incoming request using the default label of the interface specified in the packet's target IP address. Network packets which originate from within a Trusted Solaris software system are normally sent with the label of the process sending the data.

Hosts that do not transmit explicit labels, and are referred to as *unlabeled*. Packets received from unlabeled hosts are implicitly labeled by matching their source address against a kernel cache of trusted networking templates. These templates provide the rules for interpreting the security attributes of remote hosts. The association of an IP address to a template is based on administrative assignments maintained in the `tnrhdb` database.

Hosts may be explicitly assigned a template or implicitly via network wildcard assignments. If the template which most closely matches the source IP address specifies a default label value, then that label is applied to the packet. Otherwise, the default label assignment for the interface is applied (as described in the preceding *Associating Labels With Network Interfaces* section). If no default value is specified, the packet is dropped. The value 0.0.0.0 is the most general wildcard. Assigning the `InterfaceDefault` template to this wildcard, specifies that the incoming labels for remote hosts will be determined by attributes of the interface. The IP addresses associated with the directly connected interfaces all use the `tsol` template. The template assignments are maintained in the remote host database, `tnrhdb`, and are shown in the following example.

```
0.0.0.0             InterfaceDefault
192.168.0.0         tsol
192.10.10.0         tsol
```

These templates are defined in the remote host template database, `tnrhtp`. The `tsol` template is included by default, and should be assigned to all local IP addresses. The following is an example of the `InterfaceDefault` entry.

```
InterfaceDefault:host_type=unlabeled;\
def_label=empty;def_cl=empty;\
min_sl=Admin_Low;max_sl=Admin_High;\
ip_label=none;ripso_label=empty;ripso_error=empty;doi=0;
```

Customer Web Site Configuration

The Service Provider grants a certain amount of configuration control to authorized customer administrators. Most of the customers activities are beyond the scope of this article, but we will discuss how the customer configures the web server and updates the web content.

The Service Provider creates one or more administrative accounts for each customer. No such administrative accounts need to run as `root`, nor are there any files owned by `root` within their label range. In fact, even if they were given the `root` password it would do them no good since they are neither authorized to assume the `root` role, nor to write down to the `Admin_Low` label. For example, an administrative account for Customer1, created using the SMC Users tool, would have an `/etc/user_attr` entry like the following:

```
c1admin::min_label=C1;clearance=C1 HTTP FTP CGI;\
profiles=C1 Rights;
```

The customer's web configuration files would be owned by `c1admin`, and maintained at the `C1` label; therefore, no privileges would be required for most operations, such as maintaining the web server configuration file, or adding Web users via `htpasswd`. The `C1` Rights profile would be used to start and stop the web server since that will typically require the use of privilege. It should also include the authorization `solaris.login.remote` for remote access to `telnet` and `ftp`. The authorization is listed in the `/etc/security/prof_attr` entry as follows:

```
C1 Rights:::Rights for Customer 1 Administrator:\
auths=solaris.login.remote;help=index.html
```

In addition, an `/etc/security/exec_attr` entry that includes the `apachectl` shell script is as follows:

```
C1 Rights:tsol:cmd:::/usr/apache/bin/apachectl:uid=nobody;\
gid=nobody;clearance=C1 HTTP FTP CGI;label=C1 HTTP FTP;\
privs=proc_owner,net_privaddr
```

Access to Other Compartments

Since `c1admin` is cleared for all `C1` compartments, he or she can operate at the `HTTP FTP` or `CGI` compartments. For example, if `c1admin` could `telnet` to `c1webserver`, the connection would be made at the `C1 HTTP FTP` label. However, in a hardened environment, remote login could be blocked for that interface by a firewall rule.

The SunScreen™ 3.2 Firewall has been optimized for Trusted Solaris 8 OE, and its ability to block access to specific ports on specific interfaces complements the label-based polices discussed in this article. By default, most trusted services listen for connections at any label from any interface. Therefore, firewall rules should be applied to restrict these services.

A more flexible way to provide access to other compartments is by use of explicit profile entries. For example, a wrapper to switch to the `CGI` compartment could be done as follows:

```
#!/bin/sh
# Start a new profile shell
# For clarity, display the current label value in the prompt
PS1=‘plabel|cut -f2‘" " export PS1
/usr/bin/pfsh
```

If this wrapper is stored as `/usr/local/bin/gotoCGI`, then the following `exec_attr` entry provides the `c1admin` access to the `CGI` compartment from the `C1` label.

```
C1 Rights:tsol:cmd:::/usr/local/bin/gotoCGI;label=C1 CGI;\
clearance=C1 CGI
```

Access to the `CGI` compartment is only available from the `C1` label because the clearance specified in such a profile entry must dominate both the current process label and the new label. For example, the `c1admin` could `telnet` to `c1accounts` and successfully use the `gotoCGI` wrapper, but the wrapper would fail with `c1webserver` since the default label, `C1 HTTP FTP` is not dominated by `C1 CGI`.

Specifying Pathnames for Web Configuration Files

Multilevel Directories (MLDs) are often used in the Trusted Solaris OE to allow processes running at different labels to use the same pathnames when reading and writing data. The real pathname to a file in an MLD is dependent on the label of the process which opens it. By default, Apache looks for its server configuration file in `/etc/apache/http.conf`. The directory `/etc/apache` could be converted into an

MLD to polyinstantiate a unique instance for each customer. Although this approach is automatic, it is not convenient in this case since each web server runs at a higher label than its configuration file.

It is easier to create unique pathnames for each customer and use MAC to restrict each customer's files to be those processes which dominate the customer's minimum label, e.g., `C1`. Similarly, the `DocumentRoot` directory should not be an MLD, either. Instead, the Service Provider should create these directories with the customer's label and ownership. The disjoint labels assigned to directories and files provides strong isolation without the complexity and overhead of using `chroot`(1M) to create disjoint directories

The path to the configuration file can be explicitly passed to the web server as a command line option, with each customer having a unique pathname for their configuration file which is maintained in a regular directory at a unique label. In our previous article, we discussed how the script `/etc/init.d/apache` could be modified to provide for labeled execution. Here we make a few more modifications to provide a unique pathname for each customer's server configuration file:

```
#!/bin/sysh
#
#ident "@(#)apache.sh 1.3 99/11/10 SMI"
#
# extract customer name and profile name from script name
CUSTOMER=‘basename $0|cut -f2 -d .‘
setprof ${CUSTOMER}
CONF_DIR=/home/${CUSTOMER}admin
if [ ! -d ${CONF_DIR} ]; then
    exit 0
fi
# Allow webservers to create entries PID entries in /var/run
setfacl -m group:webserver:rwx /var/run
setfacl -m mask:rws /var/run
```

The `/usr/apache/bin/apachectl` file is similarly modified to find the `$CONF_DIR` using the process label:

```
# Extract customer name from process label
CUSTOMER=‘plabel -s|tr -s [:upper:] [:lower:]|\n
cut -d [ -f2|cut -d ’ ’ -f1‘

CONF_DIR=/home/${CUSTOMER}admin
HTTPD="/usr/apache/bin/httpd -f ${CONF_DIR}/http.conf"
```

For each customer, a hard link to the `init.d` script is created in `/etc/rc3.d` based on the customers name, for example:

```
# ln /etc/init.d/apache /etc/rc3.d/S50apache.c1
```

Specifying the Security Attributes for the Web Server

The Service Provider is responsible for system security, which includes the maintenance of rights profiles. A rights profile, with a name corresponding to the link's suffix, must be created in which the label, clearance, uid, gid, and privileges (if any) are specified. The contents and format of this profile were discussed in the previous article. In this context, the `exec_attr` entry to start the Apache web server would look as follows:

```
c1:tsol:cmd:::/usr/apache/bin/apachectl:uid=nobody;\
gid=webserver;clearance=C1 HTTP FTP CGI;label=C1 HTTP FTP;\
privs=proc_owner,net_privaddr
#
c1:tsol:cmd:::/usr/bin/setfacl:clearance=C1 HTTP FTP CGI;label=C1
HTTP FTP
```

When using `suexec`, it is necessary for all the web servers to run with the same user ID that was specified when `suexec` was compiled. The default value is set to `nobody` in the Solaris OE distribution. The web server is not privileged to change its user ID, so the Service Provider must specify this value in the rights profile. Furthermore, the web server is not privileged to write to files owned by root, so a common group, `webserver`, is specified. A group ACL is set on the MLD `/var/run` in which the web server records its process ID (this ACL is applied at boot time since `/var/run` is mounted on `swap`). The logging directory `/var/apache/logs` uses this group ID as well.

Customer Web Server Configuration

It is the responsibility of the customer to properly configure the web server, but there are no settings that `c1admin` can choose that will allow access to any other customers, since they are all compartmentalized. Even if the web server is misdirected to the wrong IP address, for example, it won't affect any other customers because the ports on every interface are polyinstantiated. Furthermore, this containment prevents a customer from creating local hyperlinks to another customer's files.

Setting Up Anonymous FTP

Since `ftpd` is managed by `inetd`, the server is automatically started with the same label as the requesting client. For anonymous `ftp`, the setup procedure described in the man page `ftpd`(1M) is essentially the same in Solaris OE and Trusted Solaris OE. The only additional considerations are that the procedure should be done at the `Admin_Low` label, and the directory `~ftp/pub` should be an MLD. The home directory for `ftp` should not be an MLD because everything but the contents of the `pub` directory is the same for all customers, and protected at the `Admin_Low` label.

Resource Constraints

In order to provide predictable quality of service levels to customers, the amount of CPU and virtual memory available to each customer must be specified. The Solaris™ Resource Manager software release 1.2 can be used to provide this functionality. The command `srmuser`(1SRM) can be used to specify which set of resource limits are assigned to each web server. Each customer's admin account serves as limit nodes (*lnodes*) for this purpose. The script `/usr/apache/bin/apachectl` can be further modified to setup the resource limits before invoking the web server:

```
HTTPD="/usr/srm/bin/srmuser -l ${CUSTOMER}admin \n /usr/apache/
bin/httpd -f ${CONF_DIR}/http.conf"
```

The `srmuser` command requires the privilege `sys_devices` to associate a process with an lnode. Therefore, the process attributes for the command `apachectl` in the `c1` and `c1 Rights` profiles require the following additional privilege:

```
c1:tsol:cmd:::/usr/apache/bin/apachectl:uid=nobody;\
gid=webserver;clearance=C1 HTTP FTP CGI;label=C1 HTTP FTP;\
privs=proc_owner,net_privaddr,sys_devices
```

Since the Web server runs as uid `nobody`, but with the limits of `c1admin`, the account should be specified as a subgroup of `nobody` as follows:

```
# /usr/srm/sbin/limadm set sgroup=nobody c1admin
```

For more information on resource limits, refer to the section *Resource Management of Multiple Virtual Web Servers* in the Sun BluePrints article, *Solaris™ Resource Manager* (April 1999)[6].

Summary

Trusted Solaris OE provides a unique solution to Service Providers who want to expand their customer services while minimizing the cost of hardware, administration, and total cost of ownership. Lightly loaded servers can be consolidated without exposing private customer data. It should not be surprising that the Trusted Solaris OE can provide the necessary containment. Although its history predates that of the Web, its features have always been designed to provide this kind of compartmented operation. While other technologies such as jails[4] in Free BSD, and Trusted Linux[6], offer partial solutions, the strength of Trusted Solaris OE lies in the consistency of its security policy. The ability to communicate at multiple labels while compartmentalizing communication with external hosts provides both flexibility and containment. Clients on remote hosts are subject to the MAC policy even though they are unaware of its existence.

Since Trusted Solaris OE provides complete binary compatibility with the Solaris OE, it is not necessary to rewrite or recompile applications to take advantage of these features. Although additional configuration files must be maintained, almost the entire configuration can be set up with graphical tools. As a convenience to the reader, the actual configuration files described in this article are available for download from the Sun BluePrints Web site. Several scripts are also provided to automate the setup of customer accounts and web servers.

Samples of the following databases discussed in this article are available from the *Scripts and Tools* page on the Sun BluePrints Web site (`http://sun.com/blueprints/tools/tsolsp-scripts-license.html`).

TABLE 4 Configuration Files and Scripts Available for Download

`label_encodings`	Definition of classifications and compartments
`mkcustomer`	Configures a customer account
`mkwebserver`	Configures a customer web server
`apache`	Init script to start Apache at multiple labels
`http.conf`	Sample web server configuration
`apachectl`	Apache control script called by apache (above)
`suexec.c`	Source code changes to `suEXEC`
`Makefile`	Makefile changes to compile `suEXEC`

For a more complete description of Trusted Solaris OE concepts take a look at the *Administrative Overview* section in the *Trusted Solaris Answerbook* at: `http://docs.sun.com` [7].

Acknowlegements

Gary Winiger first suggested that associating default labels with network interfaces, rather than IP adresses, was a better model for untrusted networks.

References

1. Glenn Faden, Sun BluePrints OnLine, *Maintaining Network Separation with Trusted Solaris™ 8 Operating Environment*, (March 2001)

   `http://www.sun.com/blueprints/0301/MainNet.pdf`

2. CommonCriteria.org, *Common Criteria Protection Profiles*,

   `http://www.commoncriteria.org/protection_profiles/pp.html`

3. Alex Noordegraaf, Keith Watson, Sun BluePrints OnLine, *Solaris™ Operating Environment Network Settings for Security*, (December 1999)

   `http://www.sun.com/blueprints/1299/network.pdf`

4. FreeBSD.org, *`jail`*(2) *man pages*,

   `http://www.FreeBSD.org/cgi/man.cgi`

5. C. Dalton and T.S. Choo, *An Operating System Approach to E-Services*, CACM 44(2), February 2001, p58-64,

   `http://www.hpl.hp.com/research/papers/trustedlinux.html`

6. Richard Mcdougall, Sun BluePrints OnLine, *Solaris™ Resource Manager*, (April 1999)

   `http://www.sun.com/blueprints/0499/solaris1.pdf`

7. Sun Microsystems, *Trusted Solaris™ 8 Answer Book*, November 2000,

   `http://docs.sun.com/ab2/coll.175.4/`

Author's Bio: Glenn Faden

Glenn Faden has worked as an architect and technical contributor in the Trusted Solaris OE 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 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 Solaris OE and Trusted Solaris OE, and the new Solaris Management Console tools: User Account Manager, Administrative Role Manager, and Rights Manager, which support mixed Solaris OE and Trusted Solaris environments.