SUN BLUEPRINTS APPROVED

Limiting Service Privileges in the Solaris™ 10 Operating System

Glenn Brunette, Client Solutions

Sun BluePrints™ OnLine—May 2005

A Sun BluePrints Cookbook

http://www.sun.com/blueprints

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95045 U.S.A.
650 960-1300

Part No. 819-2680-10
Revision 1.0, 4/21/05
Edition: May 2005

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 ARE HELD TO BE LEGALLY INVALID.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REPONDRE A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.

Limiting Service Privileges in the Solaris™ 10 Operating System

This Sun BluePrints™ Cookbook describes how to use the Solaris™ 10 Service Management Facility (SMF) to start a service at boot time (or at any later time) with reduced privileges. This is accomplished by setting the user, group, and set of privileges used to start the service. While this type of information is generally discussed in the Solaris™ 10 Operating System (Solaris OS) product documentation and other forums, this article describes how to accomplish this in a practical context using a real service as an example.

About the Apache2 Service Example

The example used in this article is the Apache2 service that is available in Solaris 10. This Apache2 service will run inside a Solaris container. The details of how to configure and install a Solaris container are outside of the scope of this document. The container used in this example is based on the default Sun template. The container was changed only to define the container root directory (that is `zonepath`) and to define network interfaces. All of the commands used later in this article are run from a root shell within the container.

The Apache2 service is available in the SUNWapch2r and SUNWapch2u packages. It is identified by the SMF Fault Management Resource Identifier (“FMRI”) as: `svc:/network/http:apache2`. In this article, the abbreviated name of `apache2` will be used to control and refer to this service in any SMF commands.

Why Limit Service Privileges?

Answering this question requires an understanding of how services are started by the Service Management Facility. By default, services are started as the root user and root group. Just as important, services are also started with root's default set of privileges—all privileges, or all zone privileges when running in a Solaris container.

Services that are started in this manner have all (or all zone) privileges assigned to their effective ("E"), permitted ("P") and limit ("L") privilege sets. The basic set of privileges (available to all users by default) are assigned to the service's inherited ("I") privilege set. This means that, while the first process spawned by SMF for a given service will have essentially all privileges available to it, any children created by that process will (again, by default) run only with the basic set of privileges. For those not familiar with process privileges in Solaris 10, see the reference to Casper Dik's Weblog explanation in "Web Sites" on page 14. In addition, the Solaris documentation provides extensive information on Solaris privileges.

For many applications, this is just way too much rope. Many services simply do not need to be started with this much privilege, and doing so exposes system resources to potential risks that are easily prevented. By restricting how a service is started, you will better be able to contain its set of available privileges, and also possibly eliminate the need for the service to ever start as root.

Further, if a service is never started as root, or if it never has root's set of privileges available to it, then the window of exposure and criticality of impact is significantly reduced in the event that a flaw is found in that service. Essentially, running with reduced privileges is a recommended because it reduces the potential harm that a service can cause if it were to misbehave due to a bug, accident, or even a malicious exploit.

For example, if a service is not running with the `proc_fork` and `proc_exec` privileges, then it is unable to use those system calls. This is important in the event that an exploitable buffer overflow is found in that service, because an attacker would not easily be able to exploit that vulnerability to execute arbitrary code. Note that a service need not be coded to understand Solaris 10 process privileges in order for this approach to apply.

Default Apache2 Service Configuration

In the default configuration, the Apache2 service is disabled.

```
# svcs apache2
STATE          STIME    FMRI
disabled       13:34:10 svc:/network/http:apache2
```

Before the Apache2 service can be enabled, it needs to be configured. For the sake of simplicity, this article uses the default `httpd.conf-example` file that is shipped in the `/etc/apache2` directory. Depending on your organization's own policies and requirements, it might be necessary to customize this file before deploying the Apache2 service in your actual environment.

Configuring the Apache2 Service

This section describes the following steps to configure the Apache2 service:

- Step 1: Install the Apache2 Configuration File
- Step 2: Start the Apache2 Service
- Step 3: Check the Apache2 Service Status
- Step 4: Examine the Privileges With Which the Service Was Started
- Step 5: Shut Down the Apache2 Service
- Step 6: Configure the Apache2 Service with Reduced Privileges
- Step 7: Verify the Apache2 Service Privilege Configuration
- Step 8: Change Ownership of Log Files
- Step 9: Configure the PidFile and LockFile Location
- Step 10: Restart the Apache2 Service
- Step 11: Communicate with the Service

Step 1: Install the Apache2 Configuration File

To install the Apache2 configuration file (httpd.conf), run the following commands.

```
# cp /etc/apache2/httpd.conf-example /etc/apache2/httpd.conf
# ls -l /etc/apache2/httpd.conf
-rw-r--r--   1 root     root       37661 Feb 17 15:21 /etc/apache2/httpd.conf
```

Step 2: Start the Apache2 Service

After the httpd.conf file has been installed, start the Apache2 service.

```
# svcadm -v enable -s apache2
svc:/network/http:apache2 enabled.
```

Step 3: Check the Apache2 Service Status

After the service has been enabled, check its status.

```
# svcs apache2
STATE          STIME    FMRI
online         15:25:03 svc:/network/http:apache2
```

The service is online and ready to accept requests. Remember, however, that it is important to see how the Apache2 service was started.

```
# ps -aef | grep apache | grep -v grep
webservd 14836 14835   0 15:25:04 ?          0:00 /usr/apache2/bin/httpd -k start
webservd 14837 14835   0 15:25:04 ?          0:00 /usr/apache2/bin/httpd -k start
webservd 14839 14835   0 15:25:04 ?          0:00 /usr/apache2/bin/httpd -k start
    root 14835 10895   0 15:25:03 ?          0:00 /usr/apache2/bin/httpd -k start
webservd 14838 14835   0 15:25:04 ?          0:00 /usr/apache2/bin/httpd -k start
webservd 14840 14835   0 15:25:04 ?          0:00 /usr/apache2/bin/httpd -k start
# ptree 14835
10895 zsched
  14835 /usr/apache2/bin/httpd -k start
    14836 /usr/apache2/bin/httpd -k start
    14837 /usr/apache2/bin/httpd -k start
    14838 /usr/apache2/bin/httpd -k start
```

```
14839 /usr/apache2/bin/httpd -k start
14840 /usr/apache2/bin/httpd -k start
```

As is evident in the preceding ps(1) and ptree(1) output, the Apache2 service was originally started as root (see PID 14835 above) prior to switching to the webservd account. This confirms that SMF services are started by default as root.

Step 4: Examine the Privileges With Which the Service Was Started

Use the ppriv command to examine the set of privileges with which the service was started:

```
# ppriv -S 14835
14835:  /usr/apache2/bin/httpd -k start
flags = <none>
        E: zone
        I: basic
        P: zone
        L: zone
# ppriv -S 14836
14836:  /usr/apache2/bin/httpd -k start
flags = <none>
        E: basic
        I: basic
        P: basic
        L: zone
```

This is not shocking at all—in fact, this is precisely the default way in which the Apache web server has always been started on Solaris, Linux, and other general purpose operating systems. Because the Apache service often listens on a privileged port (such as TCP port 80 or 443), it must be started with root privilege. In the new world of Solaris 10 Process Rights Management, however, this is no longer required—other options are available and preferable.

Step 5: Shut Down the Apache2 Service

Shut down the Apache2 service.

```
# svcadm -v disable -s apache2
```

```
svc:/network/http:apache2 disabled.
```

Step 6: Configure the Apache2 Service with Reduced Privileges

Change the Apache2 service configuration so that it is started with a reduced privilege set (never as the root user) using the `svccfg` and `svcadm` commands.

```
# svccfg -s apache2
svc:/network/http:apache2> setprop start/user = astring: webservd
svc:/network/http:apache2> setprop start/group = astring: webservd
svc:/network/http:apache2> setprop start/privileges = astring:
basic,!proc_session,!proc_info,!file_link_any,net_privaddr
svc:/network/http:apache2> setprop start/limit_privileges = astring: :default
svc:/network/http:apache2> setprop start/use_profile = boolean: false
svc:/network/http:apache2> setprop start/supp_groups = astring: :default
svc:/network/http:apache2> setprop start/working_directory = astring: :default
svc:/network/http:apache2> setprop start/project = astring: :default
svc:/network/http:apache2> setprop start/resource_pool = astring: :default
svc:/network/http:apache2> end
# svcadm -v refresh apache2
Action refresh set for svc:/network/http:apache2.
```

Several attributes can be used with the `start` method context. For example, you can define an alternate working directory, project, or even resource pool that will be used when the service is started. In the preceding example, some of those attributes have been included and assigned default values. You can find more information on these and other method context attributes in the `smf_method(5)` manual page at:

http://docs.sun.com/app/docs/doc/816-5175/6mbba7f3p?a=view

In the preceding example, the Apache2 service was configured to start as the `webservd` user and group and with the following privilege set: `proc_exec`, `proc_fork`, and `net_privaddr`. You can find some basic descriptions of these privileges using the `ppriv(1)` command as follows:

```
# ppriv -v -l proc_exec proc_fork net_privaddr
proc_exec
        Allows a process to call execve().
proc_fork
        Allows a process to call fork1()/forkall()/vfork()
```

```
net_privaddr
        Allows a process to bind to a privileged port number.
        The privilege port numbers are 1-1023 (the traditional
        UNIX privileged ports) as well as those ports marked as
        "udp/tcp_extra_priv_ports" with the exception of the ports
        reserved for use by NFS.
```

While the Apache2 service starts with the basic set of privileges, the `proc_session`, `proc_info`, and `file_link_any` privileges have been removed because the service just does not need these privileges in its current configuration. The `proc_exec` and `proc_fork` privileges cannot be removed because the `/lib/svc/method/http-apache2` command (assigned to the Apache2 `start` method) needs these privileges to run the `/usr/apache2/bin/apachectl` command that starts the Apache service.

Add the `net_privaddr` privilege, which is normally not given to users, so that the Apache2 process can bind to a privilege port (TCP port 80 in our example). If the Apache2 configuration listened only on a port greater than 1024 (and not on one specified by `tcp_extra_priv_ports`), then the `net_privaddr` privilege would not be needed.

Step 7: Verify the Apache2 Service Privilege Configuration

To verify that this configuration has been set, examine the service properties.

```
# svcprop -v -p start apache2
start/exec astring /lib/svc/method/http-apache2\ start
start/timeout_seconds count 60
start/type astring method
start/user astring webservd
start/group astring webservd
start/privileges astring
basic,!proc_session,!proc_info,!file_link_any,net_privaddr
start/limit_privileges astring :default
start/use_profile boolean false
start/supp_groups astring :default
start/working_directory astring :default
start/project astring :default
start/resource_pool astring :default
```

Before proceeding to start the Apache2 service, two additional steps must be completed because the Apache2 service is no longer running as root:

- Step 8: Change Ownership of Log Files
- Step 9: Configure the PidFile and LockFile Location

Step 8: Change Ownership of Log Files

Changing ownership of log files ensures that the `webservd` process has access to files under `/var/apache2/logs`. In its default configuration, the Apache2 service runs as root and therefore creates the log files under `/var/apache2/logs` as root. Ownership of these log files must be changed to `webservd`.

```
# cd /var/apache2/logs
# chown webservd:webservd access_log error_log
# ls -l
total 6
-rw-r--r--   1 webservd webservd     157 Feb 17 16:11 access_log
-rw-r--r--   1 webservd webservd    1244 Feb 17 16:12 error_log
```

This step is usually needed only when the Apache2 service had been previously started as root (as was done in our example). If the Apache2 service had not previously been started, then these files would not exist.

Step 9: Configure the PidFile and LockFile Location

Change the `httpd.conf` file to instruct the Apache2 service to attempt to create its `PidFile` and `LockFile` files in a non-default location. Ordinarily, when the Apache2 service is started as root, this is not a problem because the root user can write to any local directory (including the default of `/var/run`). However, because the Apache2 service will be started as `webservd`, different locations are required.

Change the default path for `PidFile` to `/var/apache2/run` and uncomment the existing `LockFile` variable in the `httpd.conf` file to use `/var/apache2/logs`. Edit the `/etc/apache2/httpd.conf` file, changing the values of the `LockFile` and `PidFile` variables as shown in the following example.

```
# diff httpd.conf-example httpd.conf
63c63
< #LockFile /var/apache2/logs/accept.lock
---
```

```
> LockFile /var/apache2/logs/accept.lock
87c87
< PidFile /var/run/apache2/httpd.pid
---
> PidFile /var/apache2/run/httpd.pid
```

Create the /var/apache2/run directory, because the Apache2 service will not create this directory on its own.

```
# mkdir -p -m 755 /var/apache2/run
# chown webservd:webservd /var/apache2/run
```

There is no need to create the /var/apache2/logs directory because it already exists.

Step 10: Restart the Apache2 Service

To enact this new configuration, restart the Apache2 service.

```
# svcadm -v enable -s apache2
svc:/network/http:apache2 enabled.
# svcs apache2
STATE          STIME    FMRI
online         12:02:21 svc:/network/http:apache2
```

These results are encouraging. Verify that the new user, group, and privilege settings are now in effect by using the ps(1) and ppriv(1) commands.

```
# ps -aef | grep httpd | grep -v grep
webservd  5568  5559   0 12:02:22 ?          0:00 /usr/apache2/bin/httpd -k start
webservd  5567  5559   0 12:02:22 ?          0:00 /usr/apache2/bin/httpd -k start
webservd  5561  5559   0 12:02:22 ?          0:00 /usr/apache2/bin/httpd -k start
webservd  5562  5559   0 12:02:22 ?          0:00 /usr/apache2/bin/httpd -k start
webservd  5563  5559   0 12:02:22 ?          0:00 /usr/apache2/bin/httpd -k start
webservd  5559 23382   0 12:02:21 ?          0:00 /usr/apache2/bin/httpd -k start
```

As is evident from this output, there is no master process (like PID 14835 in the original example above) that is running as root. As expected, all of the service instances are running as the webservd. Similarly, note from the ppriv(1) output that the defined privileges have been enforced.

```
# ppriv -S 5559
5559:   /usr/apache2/bin/httpd -k start
flags = <none>
```

```
          E: net_privaddr,proc_exec,proc_fork
          I: net_privaddr,proc_exec,proc_fork
          P: net_privaddr,proc_exec,proc_fork
          L: zone
```

Step 11: Communicate with the Service

Conduct a quick sanity check to verify that you are able to connect to the Apache2 service. Although this example shows testing with the `telnet(1)` command, you could just as easily connect to the service using a web browser. The goal of this test is simply to ensure that you can connect to the service and that the service is able to respond to a basic query (in this case, the `HEAD` command).

```
# telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
HEAD / HTTP/1.0
HTTP/1.1 200 OK
Date: Fri, 18 Feb 2005 17:06:24 GMT
Server: Apache/2.0.52 (Unix) DAV/2
Content-Location: index.html.en
Vary: negotiate,accept-language,accept-charset
TCN: choice
Last-Modified: Fri, 04 May 2001 00:01:18 GMT
ETag: "44181-5b0-40446f80;44197-961-8562af00"
Accept-Ranges: bytes
Content-Length: 1456
Connection: close
Content-Type: text/html; charset=ISO-8859-1
Content-Language: en
Expires: Fri, 18 Feb 2005 17:06:24 GMT
Connection to localhost closed by foreign host.
```

Complete Apache2 Service Manifest

The following code listing shows the complete updated Apache2 service manifest. The manifest contains all of the SMF parameters associated with the Apache2 service. It is provided only for completeness for those who might be interested. The service manifest was generated using the `svccfg(1M)` command with its `export` option. This format can also be easily saved, modified, validated, and even imported back into the Service Management Facility. For more information, see the `svccfg(1M)` manual page at:

http://docs.sun.com/app/docs/doc/816-5166/6mbb1kqhs?a=view

```
# svccfg export http
<?xml version='1.0'?>
<!DOCTYPE service_bundle SYSTEM '/usr/share/lib/xml/dtd/service_bundle.dtd.1'>
<service_bundle type='manifest' name='export'>
  <service name='network/http' type='service' version='0'>
    <instance name='apache2' enabled='true'>
      <dependency name='loopback' grouping='require_all' restart_on='error' type='service'>
        <service_fmri value='svc:/network/loopback:default'/>
      </dependency>
      <dependency name='physical' grouping='optional_all' restart_on='error' type='service'>
        <service_fmri value='svc:/network/physical:default'/>
      </dependency>
      <exec_method name='start' type='method' exec='/lib/svc/method/http-apache2 start'
        timeout_seconds='60'>
        <method_context>
           <method_credential user='webservd' group='webservd'
             privileges='basic,!proc_session,!proc_info,!file_link_any,net_privaddr'/>
        </method_context>
      </exec_method>
      <exec_method name='stop' type='method' exec='/lib/svc/method/http-apache2 stop'
        timeout_seconds='60'>
        <method_context/>
      </exec_method>
      <exec_method name='refresh' type='method' exec='/lib/svc/method/http-apache2 refresh'
        timeout_seconds='60'>
        <method_context/>
      </exec_method>
      <property_group name='httpd' type='application'>
        <stability value='Evolving'/>
        <propval name='ssl' type='boolean' value='false'/>
      </property_group>
      <property_group name='startd' type='framework'>
        <propval name='ignore_error' type='astring' value='core,signal'/>
      </property_group>
     </instance>
     <stability value='Evolving'/>
     <template>
       <common_name>
         <loctext xml:lang='C'>Apache 2 HTTP server</loctext>
       </common_name>
       <documentation>
         <manpage title='apache2' section='1M'/>
         <doc_link name='apache.org' uri='http://httpd.apache.org'/>
       </documentation>
     </template>
   </service>
</service_bundle>
```

Conclusion

For sites that are more security conscious, assigning the web server configuration files (and perhaps content) to another user would prevent an attacker from being able to modify those files in the event that (a) a flaw were found with the service, and (b) the attacker were granted access to the filesystem. These files could be assigned to a web server administrator account, thereby separating the administrative credentials from those used to operate the service. Describing these tasks is outside the scope of this article.

While the example in this article focused on the Apache2 service, this approach can apply to just about any service. By understanding how to limit the privileges with which services are started, your organization will be better protected against flaws in, or malicious use of, those services.

References and Related Sources

Publications

- Sun Microsystems, Inc. “svcadm(1) Manual Page”, Sun Solaris 10 OS Product Documentation

  http://docs.sun.com/app/docs/doc/816-5166/6mbb1kqhr?a=view
- Sun Microsystems, Inc. “svccfg(1) Manual Page”, Sun Solaris 10 OS Product Documentation

  http://docs.sun.com/app/docs/doc/816-5166/6mbb1kqhs?a=view
- Sun Microsystems, Inc. “svcs(1) Manual Page”, Sun Solaris 10 OS Product Documentation

  http://docs.sun.com/app/docs/doc/816-5165/6mbb0m9th?a=view
- Sun Microsystems, Inc. “svcprop(1) Manual Page”, Sun Solaris 10 OS Product Documentation

  http://docs.sun.com/app/docs/doc/816-5165/6mbb0m9tg?a=view
- Sun Microsystems, Inc. “ppriv(1) Manual Page”, Sun Solaris 10 OS Product Documentation

  http://docs.sun.com/app/docs/doc/816-5165/6mbb0m9p2?a=view
- Sun Microsystems, Inc. “ps(1B) Manual Page”, Sun Solaris 10 OS Product Documentation

  http://docs.sun.com/app/docs/doc/816-5165/6mbb0m9pk?a=view
- Sun Microsystems, Inc. “smf_method(5) Manual Page”, Sun Solaris 10 OS Product Documentation

  http://docs.sun.com/app/docs/doc/816-5175/6mbba7f3p?a=view
- Sun Microsystems, Inc. “ptree(1) Manual Page”, Sun Solaris 10 OS Product Documentation

  http://docs.sun.com/app/docs/doc/816-5165/6mbb0m9pp?a=view

Web Sites

- Glenn Brunette’s Solaris 10 OS Security Weblog

  http://blogs.sun.com/gbrunett?catname=Solaris%2010%20Security
- Caspar Dik’s Weblog (entry on Solaris 10 Privileges)

  http://blogs.sun.com/casper/20040722#solaris_privileges
- Sun's BigAdmin Predictive Self Healing Portal

  http://www.sun.com/bigadmin/content/selfheal/
- Sun's BigAdmin Solaris Zones Portal

  http://www.sun.com/bigadmin/content/zones/

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 the following people for their inspiration, technical feedback, and overall support in the development of this article: Keith Black, Stephen Hahn, Radha Krishnan, Menno Lageman, Serge Nadon, Scott Rotondo, Collin Sampson, and Susan Sano.

Ordering Sun Documents

The SunDocs^{SM} program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals through this program.

Accessing Sun Documentation Online

The `docs.sun.com` web site enables you to access Sun technical documentation online. You can browse the `docs.sun.com` archive or search for a specific book title or subject. The URL is `http://docs.sun.com/`

To reference Sun BluePrints OnLine articles, visit the Sun BluePrints OnLine Web site at: `http://www.sun.com/blueprints/online.html`