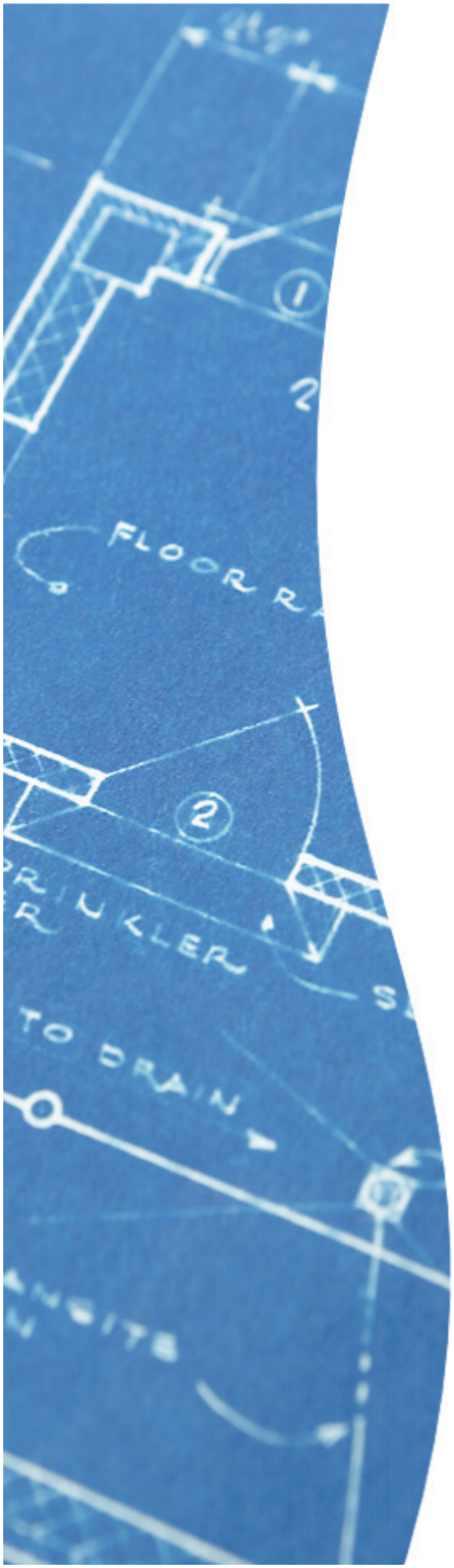

UNDERSTANDING THE SECURITY CAPABILITIES OF SOLARIS™ ZONES SOFTWARE

Glenn Brunette, Distinguished Engineer, Global Systems Engineering
Jeff Victor, Sr. Systems Engineer, Global Systems Engineering

Sun BluePrints™ Online

Part No 820-7017-10
Revision 1.0, 12/21/08

Table of Contents

Understanding the Security Capabilities of Solaris™ Zones Software

Part of the Solaris™ 10 Operating System (OS), Solaris Zones are widely discussed across all corners of the Web. Over time, Solaris Zones have grown in popularity, third-party support has increased, and the technology has been enhanced continually to support new and different kinds of features and configurations.

So why does the world need yet another article about Solaris Zones? Simple. Most publications and sites focus on the consolidation benefits of Solaris Zones. While server and service consolidation is a key use case for Solaris Zones, there is so much more to the technology. Other materials focus on system administration practices related to configuration, installation, management, and troubleshooting. This is incredibly useful information, but there is still an important gap. Namely, many people do not have a full appreciation of the security benefits enabled by Solaris Zones, and sparse root zone configurations more specifically.

This Sun BluePrints™ article brings greater attention to the key security benefits of Solaris Zones. Using practical concepts and examples, readers can gain a new appreciation for the unique security capabilities enabled by this technology.

Conventions

The following conventions are used in this article:

- The functionality, content, and examples described in this article are based upon the Solaris 10 OS 10/08 release, yet apply to the Solaris Express Community Edition (starting with Build 95). However, many of the features discussed apply to earlier versions of the Solaris 10 OS. Refer for the operating system release notes for guidance regarding specific features or functionality.
- This article focuses on the security configuration as realized by a sparse-root zone configuration. Some of the concepts discussed do not apply to whole-root zones configurations. Readers unfamiliar with the basic concepts and practices involving Solaris Zones can refer to “Introduction to Solaris Zones” in *System Administration Guide: Solaris Containers — Resource Management and Solaris Zones* located at `http://docs.sun.com/app/docs/doc/817-1592/zones.intro-1?a=view` for background information.
- Every command line prompt includes a host name indicating the zone in which the command is run. The name of the non-global zone is `web`, and the name of the global zone is `global`. Each prompt has a suffix indicating whether the command is to be run as an unprivileged user ($) or if it requires administrative privileges (#).

- Items in bold are commands that are run. Non-bold items are reserved for prompts and command output. If command output is abbreviated for brevity, it includes the `[...]` notation.

Zone Root File System

When a zone is configured, a path name for its unique root file system must be specified. If the directory pointed to by this path name does not exist, it is created when the zone is installed. The `zonecfg` command can be used to determine the zone path for a given configured zone.

```
global$ zonecfg -z web info zonepath
zonepath: /pool/zones/web
```

In this example, the `/pool/zones/web` directory is the root directory for the zone called `web`. To protect this directory from unauthorized tampering, the zone root directory is configured by default to be readable and writable only by the `root` user in the global zone. This is necessary in order to better isolate the zone from unprivileged users who might have access to the global zone. As a general practice, however, only those users who must perform entire system or zone management should be permitted access to the global zone.

Use the `ls -ld` command to verify that the default setting is in effect.

```
global$ ls -ld /pool/zones/web
drwx------   4 root     root           4 Aug  8 14:38 /pool/zones/web
```

If the default permissions are altered, the zone refuses to be boot. Consider an example where a zone root directory has altered permissions:

```
global# ls -ld /pool/zones/web
drwxr-xr-x   4 root     root           4 Aug  8 15:10 /pool/zones/web

global# zoneadm -z web boot
/pool/zones/web must not be group readable.
/pool/zones/web must not be group executable.
/pool/zones/web must not be world readable.
/pool/zones/web must not be world executable.
could not verify zonepath /pool/zones/web because of the above errors.
zoneadm: zone web failed to verify
```

Because both UFS and ZFS™ file systems support access control lists (ACLs) beyond the traditional UNIX® permissions, it is important to verify that no additional access has been granted using ACLs. This verification is important because ACLs can be added at any time by a user with administrative privileges in the global zone, and there are no ACL checks related to booting a zone. An ACL can be added to the zone root directory,

and the zone can be successfully booted. The following example shows how a zone can be booted even though there is a new ACL allowing the `gmb` user to list directories and read and execute files under the zone root directory.

```
global$ id
uid=100(gmb) gid=100(gmb)

global$ ls -l /pool/zones/web/root
/pool/zones/web/root: Permission denied

global# chmod A+user:gmb:read_data/list_directory/execute:allow
/pool/zones/web

global# ls -Vd /pool/zones/web
drwx------+  4 root     root           4 Aug  8 15:10 /pool/zones/web
          user:gmb:r-x-----------:-------:allow
            owner@:--------------:-------:deny
            owner@:rwxp---A-W-Co-:-------:allow
            group@:rwxp----------:-------:deny
            group@:--------------:-------:allow
         everyone@:rwxp---A-W-Co-:-------:deny
         everyone@:------a-R-c--s:-------:allow

global# zoneadm -z web boot

global$ ls -l /pool/zones/web/root
total 1002
lrwxrwxrwx   1 root     root           9 Aug  8 14:49 bin -> ./usr/bin
drwxr-xr-x   3 root     sys            3 Dec 22  2007 boot
drwxr-xr-x  46 root     sys           13 Aug  8 15:25 dev
drwxr-xr-x  82 root     sys          209 Aug  8 15:26 etc
[...]
```

It is important to ensure that only those who need to access zone root directories from the global zone are permitted to do so. Beyond access control, a file system quota can be set on the root directory of the zone to prevent denial of service attacks that focus on resource exhaustion. This can be useful in multizone environments to ensure a single zone does not exhaust the available file system space and negatively impact other zones running on the system. The following example sets a quota on the `/pool/zones/web` ZFS file system.

1. Set the quota.

```
global# zfs set quota=1g pool/zones/web
```

2. Verify that the quota is properly set.

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

Alternatively, the `zfs list` command can be used to determine the quota.

```
global# zfs list -o name,used,avail,quota,mountpoint pool/zones/web
NAME             USED  AVAIL  QUOTA  MOUNTPOINT
pool/zones/web  67.5M   956M     1G  /pool/zones/web
```

In this example, `/pool/zones/web` currently uses 67.5 MB of disk space, and a 1 GB disk space quota is set. Attempts to violate this quota result in an error. For example, attempting to create a 2 GB file when a 1 GB quota is in effect generates a disk quota exceeded error.

```
web# mkfile 2g /bigfile
/bigfile: initialized 902561792 of 2147483648 bytes: Disc quota exceeded

web# df -h /
Filesystem             size   used  avail capacity  Mounted on
/                        0K   1.0G     0K   100%    /
```

This example illustrates that the system does not permit users — even users with administrative access in the zone — to exceed the disk quota set for the zone's root file system. Just as important, it is not possible to change quotas or other settings from inside of the running zone because the ZFS pool and file system are not available.

```
web# zpool list
no pools available

web# zfs list
no datasets available
```

Process Containment

One of the key features of Solaris Zones is the ability to isolate processes from those running in other zones or even the global zone. This security capability creates a boundary beyond which a process running in a zone cannot influence. Consider the following process running in the global zone.

```
global$ sleep 6000 &
[1] 8244

global$ ps -fp 8244
     UID   PID  PPID   C    STIME TTY         TIME CMD
gbrunett  8244  8209   0 14:55:21 pts/2       0:00 sleep 6000
```

In this example, a `sleep(1)` command is run from, and can be seen in, the global zone as process ID `8244`. However, this process is not exposed to the zone named `web`.

```
web$ ps -fp 8244
     UID   PID  PPID   C    STIME TTY         TIME CMD
web$

web$ pgrep sleep
web$
```

Similarly, processes running in one zone are not visible to other zones. The global zone is the only exception. It can see the processes belonging to every zone. The `ps(1)` and `prstat(1M)` commands can be used to display the zone in which a process is running. Consider the following example, in which a different `cron(1M)` process is running in the `web` and `global` zones.

```
global$ ps -Zaef
    ZONE     UID    PID  PPID   C    STIME TTY         TIME CMD
[...]
  global     root   489     1   0   Sep 19 ?          0:00 /usr/sbin/cron
     web     root  5396     1   0   Sep 21 ?          0:00 /usr/sbin/cron
[...]
```

Certain process-related commands, such as `pkill(1)`, must be used with caution on systems running zones. When executed from the global zone, these commands act on all matching processes regardless of the zone in which they are running. Depending on the situation, this might not yield the desired result. Fortunately, command line options exist to limit the scope of command actions to the global zone or to specific non-global zones. For example, the following command applies only to the global zone (zone identifier `0`).

```
global$ pkill -z 0 sleep
```

Use the `zoneadm` command to determine the identifier for a given zone.

```
global$ zoneadm list -v
  ID NAME             STATUS     PATH                           BRAND    IP
  0 global           running    /                              native   shared
```

Operating System Privileges

The Solaris 10 OS effectively eliminates the concept of an all powerful superuser (`root` or UID `0`), instead using a fine-grained privilege model. Today, over 60 individual privileges are available in the Solaris 10 OS. These privileges can be granted to individual processes using a variety of methods, such as the Service Management Facility (SMF) and the Role-based Access Control (RBAC) facility.

The Solaris OS supports the notion of four distinct privilege sets available to every process:

- *Effective*, the set of privileges that are currently active (in effect) in the process
- *Permitted*, the complete set of privileges available to the process, active or inactive
- *Inherited*, the set of privileges that are inherited by a child process upon `exec(2)`
- *Limit*, the upper bound of all privileges that a process or its children can obtain

Just as with processes, a zone can have a privilege limit set that specifies the upper bound of privileges that the zone and its processes can obtain. Essentially, if a privilege is in the limit set of the zone, then it can be obtained by appropriately privileged processes running in that zone. Building upon this functionality, Solaris Zones offer even greater control by categorizing privileges into four groups: `default`, `required`, `prohibited`, and `optional`.

Default Privileges

Privileges listed in the `default` group are those available, by default, in the privilege limit set for the zone. Default privileges can be taken away if they are not required. If a given privilege is not in the limit set, it cannot be obtained by any process running in the zone regardless of other factors. Privileges in the `default` group include the `file_dac_read, proc_chroot`, and `sys_nfs` privileges.

By default, a zone has access to the `proc_chroot` privilege.

```
web$ ppriv -l zone | grep proc_chroot
sys_chroot
```

Use the `zonecfg` command to remove a default privilege from a zone's limit set.

```
global# zonecfg -z web set limitpriv="default,!proc_chroot"
global# zoneadm -z web reboot
```

The `proc_chroot` privilege is no longer available if the zone is rebooted.

```
web$ ppriv -l zone | grep proc_chroot
web$
```

By removing the `proc_chroot` privilege, no process running inside of the `web` zone, including those running as `root`, can change its root directory as shown in the following example.

```
web# ppriv -De chroot /mnt /usr/bin/ls
chroot[27457]: missing privilege "proc_chroot" (euid = 0, syscall = 61)
needed at secpolicy_chroot+0x20
chroot("/mnt"): Not owner
```

Removing privileges from a zone's limit set provides an additional layer of protection that cannot be circumvented by non-global zone administrators.

Required Privileges

Privileges in the `required` group are those that must be listed in the limit set for a zone. A zone fails to boot if these privileges are not made available to the zone. Privileges in the `required` group include the `proc_fork`, `proc_exec`, and `sys_mount` privileges. In the following example, the `sys_mount` privilege is not granted to the zone, and the zone fails to boot as a result.

```
global# zonecfg -z web set limitpriv="basic"

global# zoneadm -z web boot
required privilege "sys_mount" is missing from the zone's privilege set
zoneadm: zone web failed to verify
```

Note that for most practical purposes, a zone often requires more privileges than those listed in the `required` group in order to boot and operate correctly.

Prohibited Privileges

Privileges in the `prohibited` group are those that must not be listed in the limit set for a zone. These privileges can be used to circumvent the security model of a zone and therefore should not be available to processes running in the zone. Privileges in this group include the `dtrace_kernel`, `proc_zone`, and `sys_devices` privileges. Just as with `required` privileges, there exists a check at zone boot time related to prohibited privileges. A zone fails to boot if there is an attempt to grant a `prohibited` privilege to a zone.

```
global# zonecfg -z web set limitpriv="default,dtrace_kernel"

global# zoneadm -z web boot
privilege "dtrace_kernel" is not permitted within the zone's privilege
set
zoneadm: zone web failed to verify
```

Optional Privileges

Privileges listed in the `optional` group are those not granted to a zone by default but which can be added to a zone if necessary. These privileges generally are not required for normal operation but might be useful in some circumstances. Privileges in this category include the `dtrace_user`, `net_rawaccess`, and `sys_time` privileges. Optional privileges can be granted to a zone by using the `limitpriv` option to the `zonecfg(1M)` command as shown in the previous example.

An example use of an optional privilege is `sys_time`. When specified in a zone's limit set, this privilege allows the zone to set the clock for the entire system, including all zones running on that system. This can be useful when a Network Time Protocol (NTP) server is deployed within a zone.

The zone's limit set is comprised of the individual privileges assigned to the limit set for that zone. The list of privileges can be queried from within a zone using the `ppriv(1)` command:

```
web$ ppriv -l zone
contract_event
contract_observer
file_chown
file_chown_self
file_dac_execute
[...]
```

The actual list of privileges made available to a running zone varies as optional privileges are added or default privileges are removed from the zone's limit set.

```
global# zlogin web ppriv -l zone | wc -l
      34

global# zonecfg -z web set limitpriv="default,sys_time"

global# zoneadm -z web reboot

global# zlogin web ppriv -l zone | wc -l
      35
```

In this example, the `web` zone initially had 34 privileges in its limit set. The value increased to 35 privileges after the `sys_time` privilege was added. As a general rule, the default zone privilege group usually is sufficient for most zone and application configurations. Individual application requirements, such as the previous NTP example, might require the addition of specific privileges. It is important to understand the potential impact of adding or removing privileges before making zone configuration changes.

The list of privileges and their relationship to a zone could change in future updates or as new functionality is integrated. More information on privileges available to zones can be found in "Privileges in a Non-Global Zone" in *System Administration Guide: Solaris Containers — Resource Management and Solaris Zones* located at `http://docs.sun.com/app/docs/doc/817-1592/zones.intro-1?a=view`.

Operating System Kernel Modules

Regardless of its configuration, a zone cannot acquire all of the privileges available on the system (in the global zone) because prohibited privileges are not allowed within a running zone. As a result, there are commands that a zone cannot run. This is by design and is meant to help reinforce the security posture of the zone. For example, the following `ppriv` command attempts to load a kernel module. This operation requires `ALL` privileges in order to be run successfully.

```
web# ppriv -De modload -p /tmp/systrace
modload[21174]: missing privilege "ALL" (euid = 0, syscall = 152) needed
at modctl+0x52
Insufficient privileges to load a module
```

This is a very important point. To load kernel modules into a zone, `ALL` privileges are required, yet a zone can never attain `ALL` privileges. This restriction is needed because there is only one kernel on the system. Zones share the same kernel, and changes to the kernel in one zone impact all of the other zones running on the system. This restriction has the effect of eliminating the possibility of kernel-based root kits being introduced from within a zone. However, kernel modules can be loaded from the global zone and made available to other non-global zones.

Note – The Solaris OS supports a modular kernel that can dynamically load and unload kernel modules. Consequently, it is possible for a non-global zone to cause a kernel module to be loaded in the global zone by exercising kernel functionality that requires the new module to be loaded. The global zone loads the kernel module if the module exists in its search path (by default in `/platform/platform-name/kernel`, `/kernel`, or `/usr/kernel`).

The unloading of kernel modules is restricted within a zone for the same reasons. The following output shows that the `sys_config` privilege is required to unload a kernel module. However, the `sys_config` privilege is included in the list of prohibited zone privileges. As a result, it is not possible to unload a kernel module from within a zone.

```
web$ modinfo | grep dtrace
  4 ffffffff8360000  1b570 155   1  dtrace (Dynamic Tracing)

web# ppriv -De modunload -i 4
modunload[21178]: missing privilege "sys_config" (euid = 0, syscall =
152) needed at modctl+0x52
Insufficient privileges to unload a module
```

Operating System Devices

Once a zone is configured and running, it looks and behaves in many ways just like the global zone. One important difference is that zones are treated as virtual application environments. By default, zones are not permitted to access the physical device paths on the system directly. Instead, zones are granted virtualized access to many of the default devices. For example, raw disk devices are not exposed within a zone by default.

```
web# format
Searching for disks...done
No disks found!

web$ ls /dev/dsk
web$
```

Instead, disk access usually is made available through file systems that are mounted into the zone from the global zone. In the example below, `/mydir` in the global zone is mounted into the `web` zone using a read-only loopback mount.

```
global# zonecfg -z web info fs dir=/mydir
fs:
    dir: /mydir
    special: /mydir
    raw not specified
    type: lofs
    options: [ro,nodevices,nosetuid]
```

Using the ZFS file system, datasets and pools can be delegated to a zone using the `zonecfg(1M)` command.

```
global# zonecfg -z web
zonecfg:web> add dataset
zonecfg:web:dataset> set name=test/mydata
zonecfg:web:dataset> end
zonecfg:web> exit

global# zoneadm -z web reboot

global# zlogin web zfs list
NAME          USED  AVAIL  REFER  MOUNTPOINT
test         4.40G  74.4G    27K  /test
test/mydata    18K  74.4G    18K  /test/mydata
```

The `zonecfg add device` command can be used in cases where a raw device needs to be added to a zone. It is important to understand the failure modes of such an addition before adding a raw device to a zone. For example, it is possible for a malicious user with write access to craft a file system on a raw disk device that could cause the Solaris OS kernel to panic. In this case, a better approach is to use file systems or ZFS data sets and limit the exposure of raw devices whenever possible.

The risks highlighted by the introduction of devices into a zone are not limited to just disks. For example, a network device can be added directly to a zone that is configured to use a shared IP networking configuration using the following commands.

```
global# zonecfg -z web
zonecfg:web> set limitpriv=default,net_rawaccess
zonecfg:web:device> add device
zonecfg:web:device> set match=/dev/nge0
zonecfg:web:device> end
zonecfg:web> exit
```

The net effect of this action is that `/dev/nge0` is exposed to the zone, and the zone is granted the `net_rawaccess` privilege. As a result, zone administrators have raw access to the network layer for that device, and could use the `snoop(1M)` command in the zone. While this can be useful, it has a potentially dangerous side effect. Administrators can see all network traffic flowing over that device, including traffic to and from other zones on the system. This risk goes well beyond monitoring unauthorized network traffic because the `net_rawaccess` privilege also allows administrators to modify or inject network packets.

Similarly, some devices are not exposed to zones as they are global in nature or could be used to subvert the global zone or other zones running on the system. For example, the devices associated with the virtual address space of the kernel and its symbol table are global in nature and are unavailable to a zone.

```
web$ ls -l /dev/kmem /dev/ksyms
/dev/kmem: No such file or directory
/dev/ksyms: No such file or directory
```

Consequently, applications that rely on these devices cannot function in a zone.

```
web# mdb -k
mdb: failed to open /dev/ksyms: No such file or directory
```

As an added security protection, administrative users cannot add devices from within the zone. This is enforced by the Solaris OS kernel as creating new devices requires the `sys_devices` privilege — a privilege that is prohibited from being used within a zone.

```
web# ppriv -De mknod /dev/kmem c 13 1
mknod[21187]: missing privilege "sys_devices" (euid = 0, syscall = 126)
needed at mknod+0x60
mknod: Not owner
```

Similarly, devices cannot be removed from a zone.

```
web# rm /dev/zero
rm: /dev/zero not removed: Operation not supported
```

Devices that are exposed by default to the zone are specifically selected and evaluated to ensure that their use does not compromise the system or other running zones. If needed, devices can be added to a zone from the global zone by using the `zonecfg(1M)` command. A zone reboot is required for such changes to take effect. Exercise caution when deciding to modify the default device configuration of a zone, as improper changes can leave the zone in a weakened security state.

Note that devices can be introduced over remote file systems or removable media. To guard against this problem, mounted file systems are automatically configured to use the `nodevices` option to prevent the introduction of devices from outside the zone.

```
web# mount -F nfs nfs_server:/tmp/dev /mnt

web$ ls -l /mnt
total 0
crw-r--r--   1 root     root      13,  1 Aug 11 10:38 kmem

web$ mount -p | grep tmp/dev
nfs_server:/tmp/dev - /mnt nfs - no rw,nodevices,xattr,zone=web
```

Networking

Originally, Solaris Zones supported what is now called shared IP networking. In the shared IP model, each non-global zone is assigned its own IP address(es). However, the rest of the TCP/IP infrastructure is managed by the global zone and shared across all zones on the system. Each shared IP zone can have a process listening on a given TCP or User Datagram Protocol (UDP) port because each shared IP zone has distinct IP addresses, allowing traffic to be dispatched to the correct zone.

The Solaris 10 OS 8/07 release introduced the exclusive IP model. In this model, zones have separate IP instances, and separation reaches down to the data link layer. The global administrator assigns one or more data link names, which can be a network interface card (NIC) or a virtual LAN (VLAN) on a NIC, to an exclusive IP zone. The zone administrator can configure IP on those data links with the same flexibility and options as in the global zone.

The `zonecfg` command can be used to determine whether a zone is using shared or exclusive IP. The command reports either `shared` or `exclusive` depending on the configuration of the zone. While a number of factors drive the decision toward either shared or exclusive IP configurations, it is important to understand a few of the security tradeoffs of these models.

```
global$ zonecfg -z web info ip-type
ip-type: shared
```

Shared IP

Networking access is virtualized in a shared IP zone configuration. Network traffic flowing to, or from, other zones and the global zone can share the same physical network interface. The actual raw network device, such as `/dev/nge0`, is not exposed within the zone. Instead, each zone has a virtual network interface defined.

```
web$ ifconfig nge0:1
nge0:1: flags=201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu
1500 index 2
inet 192.168.1.253 netmask ffffff00 broadcast 192.168.1.255
```

This is useful from a security context. A non-global zone administrative user is not able to set the device into a promiscuous mode or use a network sniffer by default. This approach protects against a common method used by attackers. Of course, the `snoop(1M)` command can be used by authorized users within the global zone.

```
web# snoop
snoop: No network interface devices found

web# snoop -d nge0
snoop: cannot open “nge0”: DLPI link does not exist
```

As noted previously, a global zone administrator can add this device to a zone configured for shared IP — but not without introducing substantial risk. The zone can see network traffic destined for other zones using the same physical network interface.

Key characteristics of shared IP zones include:

- The shared IP approach does not permit a zone administrator to change the IP address, network mask, or the status (up or down) of the network interface.

  ```
  web# ifconfig nge0:1 inet 192.168.3.1
  ifconfig: SIOCGLIFNETMASK: nge0:1: no such interface

  web# ifconfig nge0:1 down
  ifconfig: setifflags: SIOCGLIFFLAGS: nge0:1: no such interface
  ```

- Shared IP zones cannot spoof source IP addresses, Media Access Control (MAC) addresses, or Address Resolution Protocol (ARP) replies, and they cannot send Bridge Protocol Data Unit (PDU) or Open Shortest Path First (OSPF) packets. This functionality is implemented by the global zone outside of the context of the

shared IP zone. However, it is possible for an administrative user to spoof Routing Information Protocol (RIP) packets and Internet Control Message Protocol (ICMP) redirects from within a shared IP zone.

- Shared IP zones have no control over their network routing table. All routing configuration and management is performed in the global zone. Zones can be assigned specific default routes by using the `defrouter` network option to the `zonecfg(1M)` command. Added in the Solaris 10 OS 10/08 release, this configuration option automates the creation of default routes for the zone within the global zone's routing table. Regardless of whether this option is used, the global zone routing table is used to make routing decisions for the zone.
- Solaris IP Filter is configured from within the global zone and enforces a policy for all shared IP zones based upon their IP address. In this configuration, an attacker gaining administrative access in the zone cannot see, change, or disable the packet filtering policy or logs.
- Solaris Zones are permitted to communicate with one another only using network semantics. Even this is only possible if the zones exist on the same network, or are able to reach one another through an external router.
- When two non-global zones configured with shared IP exist on the same logical network, they can communicate over a loopback interface in the Solaris OS for performance reasons (rather than passing packets out to the NIC). Starting with the Solaris 10 OS 8/07 release, Solaris IP Filter can filter communication occurring over the loopback interface by setting the following parameter at the top of the */etc/ipf/ifp.conf* file. Be careful when enabling this functionality. All loopback network traffic, including `lo0`, is subject to the Solaris IP Filter policy defined on the system.

```
set intercept_loopback true;
```

- Internet Protocol Security (IPSec) is configured from the global zone. The IPSec policy configuration file, */etc/inet/ipsecinit.conf*, exists only in the global zone. The file can include entries that apply to non-global zones (specified by IP address), as well as entries that apply to the global zone. Essentially, the IPSec policy is defined in the global zone, and users with administrative privileges in the zone cannot alter that policy. Note that a port-only policy, if specified, is applied to all zones running on the system.
- Internet Key Exchange (IKE) is configured and run from the global zone. The IKE policy configuration file, */etc/inet/ike/config*, exists only in the global zone. Manual key exchange is possible, but it must be configured and managed from the global zone.

See "Networking in Shared-IP Non-Global Zones" in *System Administration Guide: Solaris Containers — Resource Management and Solaris Zones* located at `http://docs.sun.com/app/docs/doc/817-1592/zones.intro-1?a=view` for more information on shared IP networking.

Exclusive IP

Exclusive IP configurations provide enhanced network isolation. Network traffic is bound to one or more specific network interfaces that are not shared by other zones, nor by the global zone. As a result of this exclusivity, a zone can manage its own network configuration, such as IP address and routes — something that is not possible with a shared IP configuration. Because the actual raw network device is made available within the zone, a user with administrative privileges can reassign the IP address, set the status of the network interface, or use a network sniffing program.

Key characteristics of exclusive IP zones include:

- Unlike shared IP configurations, routing can be configured and managed within each specific zone. Routing configurations defined in one zone do not impact routing decisions made in other zones.
- Solaris IP Filter can be configured and used within specific zones. While this approach offers greater flexibility, including the ability not to have IP Filter running for specific zones, it allows a zone administrator to see, change, or disable the packet filtering policy or logs of the zone.
- Unlike shared IP configurations, zones using exclusive IP cannot communicate with each other over the loopback interface regardless of their network configuration. For two or more zones to communicate, they must be able to reach each other through an external device, such as a hub, switch, or router.
- IPSec can be configured and used within an exclusive IP zone, just as it can be used from within the global zone. This has the same tradeoff as Solaris IP Filter — a user with administrative privileges can see, change, or disable the IPSec configuration.
- Internet Key Exchange is configured and run only from the global zone. Manual key exchange is possible and can be configured individually on a per-zone basis.

See "Networking in Exclusive-IP Non-Global Zones" in *System Administration Guide: Solaris Containers — Resource Management and Solaris Zones* located at `http://docs.sun.com/app/docs/doc/817-1592/zones.intro-1?a=view` for more information on exclusive IP networking.

Operating System Files

By default, Solaris Zones are created using a template that specifies a sparse-root configuration. In this configuration, the vast majority of operating system files are accessed as read-only, loopback mounts from the global zone. This is a significant security benefit as the vast majority of operating system files are made read-only in such a way that they cannot be changed from within the zone. This notion of immutability can be used to protect the system from common forms of user space root kits that replace key operating system files with versions that are compromised in some way. Four directory trees are mounted using read-only loopback mounts from the global zone: the `/lib`, `/platform`, `/sbin` and `/usr` directories. This configuration can be verified by looking at how the directories are mounted within the zone.

```
web$ mount -p | awk '($4 == "lofs")' | grep "ro,"
/lib - /lib lofs - no ro,nodevices,nosub
/platform - /platform lofs - no ro,nodevices,nosub
/sbin - /sbin lofs - no ro,nodevices,nosub
/usr - /usr lofs - no ro,nodevices,nosub
```

The following examples attempt to change files under these read-only directories. The files selected are common targets of user space root kits and other malicious software.

```
web# rm /usr/bin/su
rm: /usr/bin/su: override protection 555 (yes/no)? yes
rm: /usr/bin/su not removed: Read-only file system

web# rm /usr/lib/ssh/sshd
rm: /usr/lib/ssh/sshd: override protection 555 (yes/no)? yes
rm: /usr/lib/ssh/sshd not removed: Read-only file system

web# mkdir /usr/ccs/bin/...
mkdir: Failed to make directory "/usr/ccs/bin/..."; Read-only file
system
```

It is possible to mount additional files and directories as read-only loopback file systems within the zone. In an earlier example, the `/mydata` directory was mounted in this way from the global zone. Using this same technique, global zone administrators can create immutable files and directories within the zones as needed. For example, the */etc/inet/ntp.conf* file can be mounted as a read-only loopback mount from the global zone to ensure a consistent NTP configuration between zones.

```
web$ mount -p | grep 'ntp.conf'
/etc/inet/ntp.conf - /etc/inet/ntp.conf lofs - no ro,nodevices,nosetuid

web$ cat /etc/inet/ntp.conf
server 192.168.1.200
server 192.168.2.200
[...]
```

Doing so prevents users in the zone from modifying or removing this file.

```
web# echo "junk" >> /etc/inet/ntp.conf
/etc/inet/ntp.conf: cannot create

web# rm /etc/inet/ntp.conf
rm: /etc/inet/ntp.conf: override protection 644 (yes/no)? yes
rm: /etc/inet/ntp.conf not removed: Device busy
```

Another benefit of this approach is that the file cannot be overlaid by a mount from within a non-global zone. While this approach protects this specific file from accidental or malicious change, a malicious administrative user can choose to run the service with an alternate configuration file as in the following example.

```
web# /usr/lib/inet/xntpd -c /tmp/my_ntp.conf
```

This approach can be used to create application-specific immutable files and directories that protect application binaries, libraries, configuration files, application data, and other content from being modified or removed from within the zone. It also can be used to prevent users from seeing content in the zone by overlaying an existing file or directory with one from the global zone.

```
global# zlogin web ls -l /mydir/mysubdir
total 110
-r--r--r--   1 root     sys         1092 Jul 22 16:04 README
dr-xr-xr-x   4 root     bin            7 Sep 19 11:35 install
[...]

global# mkdir /pool/zones/web/zone-mysubdir

global# zonecfg -z web
zonecfg:web> add fs
zonecfg:web:fs> set dir=/mydir/mysubdir
zonecfg:web:fs> set special=/pool/zones/web/zone-mysubdir
zonecfg:web:fs> set type=lofs
zonecfg:web:fs> add options ro,nodevices,nosetuid
zonecfg:web:fs> end
zonecfg:web> exit

global# zoneadm -z web reboot

global# zlogin web ls -l /mydir/mysubdir
total 0
```

While the previous examples focused on the use of read-only loopback mounts, read-write mounts also are possible. In fact, a number of useful loopback mount options can be used when adding file systems to a zone's configuration (Table 1).

Table 1. Loopback mount options

Option	Description
`nodevices`	Disallows the opening of any device special files
`noexec`	Disallows the execution of programs in the file system, and disallows `mmap(2)` with `PROT_EXEC` for files within the file system
`nosetuid`	Disallows `setuid` or `setgid` execution
`ro`	Marks the file system as read-only when accessed within the zone
`rw`	Marks the file system as read-write when accessed within the zone

See "Security Restrictions and File System Behavior" in *System Administration Guide: Solaris Containers — Resource Management and Solaris Zones* located at `http://docs.sun.com/app/docs/doc/817-1592/zones.intro-1?a=view` for more information about the security issues and restrictions related to file system behavior.

Operating System Security Configuration

Whether in response to application or operational technical requirements or organizational security policies, one of the most important tasks taken after zone installation is security hardening and configuration tuning. In this particular area, Solaris Zones look and respond no differently than the global zone or other Solaris systems. In fact, each and every Solaris Zone can have its own security configuration a variety of areas (Table 2).

Table 2. Security configurations

Area	Examples
Service Configuration	Disabled, enabled, local-only, specific settings
Naming Services Configuration	Files, NIS, NIS+, LDAP
Authentication Mechanisms	Password, Kerberos, third-party
Account and Password Policies	Password algorithm, aging, history, account lockout
Role-Based Access Control Configuration	Authorizations, rights profiles, roles

Each of these areas can be adjusted independently, based on the requirements of the zone and the applications running within it. The process by which these changes are made is identical to what is used in the global zone. For example, the `netservices` command is used to enable the Solaris Secure by Default configuration.

```
web# netservices limited
restarting syslogd
restarting sendmail
restarting wbem
```

Similarly, the `roles` command is used to assign the user `joe` to the `postgres` role.

```
web$ roles joe
No roles

web# usermod -R postgres joe

web$ roles joe
postgres
```

These are just two examples illustrating the type of security-related configuration changes that can be made within a zone. For more information about Solaris 10 OS security hardening and related security controls, see the following documents:

- The Center for Internet Security, “Solaris 10 Security Benchmark v4.0”
- “An Overview of Solaris 10 Operating System Security Controls” (Companion piece to the CIS “Solaris 10 Security Benchmark v4.0”)

Resource Management

Denial-of-service (DoS) attacks are a security risk that do not necessarily require unauthorized intrusion. Such attacks attempt to overwhelm the capacity of a service or device to perform its specific function. For example, a malicious but unprivileged user can run a program that allocates as much memory as possible, in an attempt to use all of the system's physical or virtual memory, thereby preventing legitimate processes from using memory. In such a condition, legitimate processes can fail, causing the denial of service.

Historically, the Solaris OS has implemented a number of resource controls, methods that a system administrator can use to limit the amount of a resource that a particular process or group of processes can use. The zones feature set also includes a wide variety of resource controls. By default, however, these resource controls are disabled, and must be enabled to protect against denial-of-service attacks. The classes of resource controls include memory, CPU, and miscellaneous controls.

Memory Controls

Solaris Zones support a variety of memory controls.

Physical and Virtual Memory Capping

A malicious or poorly written process can attempt to deny other processes the ability to properly allocate memory. There are several ways in which this can be accomplished. One simple method is to repeatedly allocate memory until all of the virtual memory (RAM and swap space) is consumed. Further allocation attempts by any process fail. In fact, few applications can continue to operate normally under these conditions. Most systems are configured with more swap space than needed because the alternative is too horrible to consider.

To protect against this attack, each zone can be assigned a swap cap — a control that limits the amount of virtual memory (physical RAM plus swap disk usage). The `zonecfg` command can be used to limit a zone to 4 GB of virtual memory.

```
global# zonecfg -z web
 zonecfg:web> add capped-memory
 zonecfg:web:capped-memory> set swap=4g
 zonecfg:web:capped-memory> end
 zonecfg:web> exit

global# zoneadm -z web reboot
```

Once the zone reboots, the processes running in that zone can use only 4 GB of virtual memory, in aggregate. This limit can be queried using the `prctl` command.

```
global# prctl -n zone.max-swap -i zone web
zone: 6: web
NAME    PRIVILEGE       VALUE    FLAG   ACTION            RECIPIENT
zone.max-swap
        privileged       4.0GB      -   deny              -
        system          16.0EB    max   deny              -
```

The first attempt by one of the zone's processes to use more virtual memory fails with the same error code as if memory was exhausted on the system.

```
web# ls
bash: fork: Not enough space
```

In other words, the behavior of applications in that zone is the same as it would be if the applications were running on a non-zoned system that had simply exhausted its virtual memory. Note that some applications do not handle this condition gracefully. Use caution to avoid this situation under normal operating conditions. This is similar to sizing swap space for a non-zoned system.

This limit can also be changed while the zone is running. The `prctl` command can be used to change the zone's virtual memory limit (swap cap) to 6 GB without requiring a reboot.

```
global# prctl -n zone.max-swap -v 6g -t privileged -r -e deny -i zone web
```

Other zones, including the global zone, are not directly affected if one zone reaches its virtual memory cap. However, zones can be affected if most or all of the RAM is consumed as a side effect. Whether there is a shortage of virtual memory or not, the over-consumption of RAM can cause excessive paging, which can affect all zones on the system. To protect against over-consumption of RAM by one zone, there is a memory cap for physical memory. To add a memory cap to the zone configured previously, use the `zonecfg` command.

```
global# zonecfg -z web
zonecfg:web> select capped-memory
zonecfg:web:capped-memory> set physical=2g
zonecfg:web:capped-memory> end
zonecfg:web> exit

global# zoneadm -z web reboot
```

Use the `rcapstat` command to verify that a physical memory cap is in place.

```
global# rcapstat -z
    id zone          nproc    vm   rss   cap    at avgat    pg avgpg
     6 web              24  123M   96M    2G  140M    0K   98M    0K
     6 web               -  123M   96M    2G    0K    0K    0K    0K
```

The physical memory cap can be modified while the zone is running by using the `rcapadm` command.

```
global# rcapadm -z web -m 3g
```

As with other resource controls, rebooting the zone results in the zone's resource limits reverting back to the values set using `zonecfg(1M)` during the configuration process.

If a process running in a memory capped zone attempts to exceed its limit, application behavior is consistent with application behavior in a non-zoned system with insufficient memory — the operating system begins forcing memory pages out to swap space. Other than the performance penalty of paging, this action should be transparent to the application.

To attain that consistency, the Solaris OS begins to force memory pages associated with the zone out to one or more swap disks if the zone's processes reach its physical memory cap. Because the application should be allowed to continue while paging occurs, the paging activity is performed asynchronously. As a result, the zone's processes can temporarily use more than the physical memory cap.

Note – Be careful when using this cap. Excessive paging can impact overall system performance, especially if other zones are causing paging or are using the same disk or storage I/O connection.

Shared Memory

Some applications use shared memory so that multiple processes can access common data. A typical example is database software, which uses shared memory to store table indexes. Database performance can be severely impacted if that data is paged out to disk. When applications use shared memory pages through Intimate Shared Memory (ISM) or Dynamic ISM (DISM), those memory pages are locked into memory and cannot be paged out by the kernel.

This functionality can be used to craft a denial-of-service attack. Although a RAM cap prevents this condition, under normal operations a zone might need 30 GB of RAM in a 32 GB system, but might only need to lock down 2 GB. Allowing its processes to lock all 30 GB could cause other zones to cease functioning normally. To prevent a zone from allocating so much shared memory that other workloads are adversely impacted, a resource cap for shared memory also exists. Use the `zonecfg` command to set a shared memory cap of 2 GB for the zone.

```
global# zonecfg -z web
zonecfg:web> set max-shm-memory=2g
zonecfg:web> exit

global# zoneadm -z web reboot
```

After the zone reboots, this value can be queried using the `prctl` command.

```
global# prctl -n zone.max-shm-memory -t privileged -i zone web
zone: 6: web
NAME    PRIVILEGE       VALUE    FLAG   ACTION             RECIPIENT
zone.max-shm-memory
        privileged       2.0GB      -   deny                   -
```

Just as with the previous examples, this resource constraint also can be changed dynamically for a running zone.

```
global# prctl -n zone.max-shm-memory -v 4g -t privileged -r -e deny \
   -i zone web
```

Locked Memory

In addition to shared memory, a program can lock down other memory pages. The Solaris OS provides this functionality for well-behaved applications in order to improve performance. This ability can be abused using a method similar to the one described for shared memory. To limit the amount of memory that a zone can lock down, use the following `zonecfg` command.

```
global# zonecfg -z web
 zonecfg:web> select capped-memory
 zonecfg:web:capped-memory> set locked=2g
 zonecfg:web:capped-memory> end
 zonecfg:web> exit

global# zoneadm -z web reboot
```

After the zone is rebooted, this value can be queried using the `prtcl` command.

```
global# prctl -n zone.max-locked-memory -t privileged -i zone web
zone: 6: web
NAME    PRIVILEGE       VALUE    FLAG   ACTION          RECIPIENT
zone.max-locked-memory
        privileged      20.0MB      -   deny                -
```

Similarly, this resource constraint can be changed dynamically for a running zone.

```
global# prctl -n zone.max-locked-memory -v 4g -t privileged -r -e deny \
   -i zone web
```

CPU Controls

One of the simplest and most common denial-of-service attacks is over-consumption of CPU cycles. This situation can arise when an attacker causes arbitrary code to be run on the target computer. By consuming as many CPU cycles as possible, other legitimate workloads might not get sufficient time on the system's CPU(s) and fail to meet response time goals. In the worst case, CPU-starved applications are unable to function correctly and can fail.

In this context, the ultimate goal is to ensure that legitimate workloads are scheduled to run often enough, with sufficient time slices, to meet their needs. The Solaris Zones feature set includes several mechanisms to thwart this type of attack and achieve that goal. The following sections provide information to aid the choice of an appropriate method.

See *System Administration Guide: Solaris Containers — Resource Management and Solaris Zones* located at `http://docs.sun.com/app/docs/doc/817-1592/zones.intro-1?a=view` for more information.

Note – For the purpose of resource management, the Solaris OS considers a CPU to be a CPU core on all systems (SPARC® and x86/x64) except for Sun servers with chip multithreading technology (CMT) (systems incorporating UltraSPARC® T1, T2, and T2 Plus processors). On CMT systems, the Solaris OS considers a strand (or hardware thread) to be a CPU. For example, a Sun Fire™ T5440 server has four CPU sockets, with eight cores per CPU chip and eight strands per core, for a total of 256 CPUs.

Fair Share Scheduler

A solid CPU control mechanism is the Fair Share Scheduler (FSS). The FSS can control the allocation of available CPU resources among workloads, based on their importance. This importance is expressed by the number of `cpu-shares` assigned to each workload. The scheduler compares the number of shares assigned to a particular workload to the aggregate number of shares assigned to all workloads. If one workload has 100 shares, and the total number of shares assigned to all workloads is 500, the scheduler ensures that the workload receives at least one-fifth of the available CPU resources.

The last point is very important. A key benefit of this CPU control mechanism is that it does not waste idle CPU cycles. Any workload can use idle CPU cycles, as long as each workload receives its guaranteed minimum. For example, use the `zonecfg` command to allocate 100 `cpu-shares` to a zone.

```
global# zonecfg -z web
zonecfg:web> set cpu-shares=100
zonecfg:web> exit

global# zoneadm -z web reboot
```

After the zone is rebooted, this value can be queried using the `prctl` command.

```
global# prctl -n zone.cpu-shares -t privileged -i zone web
zone: 6: web
NAME    PRIVILEGE       VALUE    FLAG   ACTION            RECIPIENT
zone.cpu-shares
        privileged        100       -   none                  -
```

Similarly, this resource constraint can be changed dynamically for a running zone.

```
global# prctl -n zone.cpu-shares -r -v 200 -i zone web
```

When CPU share assignments are changed, or when zones that are using the FSS are added or removed, the proportion of CPU time allowed for the running zones changes based upon the new total number of shares.

CPU Capping

In specific situations, a workload should not be allowed to use more than a fixed amount of CPU time, measured as *CPU equivalents*. A zone can be assigned a CPU cap, which limits the amount of computation that the zone's processes can perform over a period of time. This cap is specified in hundredths of a CPU. For example, use the following `zonecfg` command to assign a CPU cap equivalent to four-and-one-third CPUs.

```
global# zonecfg -z web
zonecfg:web> add capped-cpu
zonecfg:web:capped-cpu> set ncpus=4.33
zonecfg:web:capped-cpu> end
zonecfg:web> exit

global# zoneadm -z web reboot
```

After the zone is rebooted, this value can be queried using the `prctl` command.

```
global# prctl -n zone.cpu-cap -i zone web
zone: 6: web
NAME    PRIVILEGE       VALUE    FLAG   ACTION            RECIPIENT
zone.cpu-cap
        system           4.33     inf   deny                  -
```

Similarly, this resource constraint can be changed dynamically for a running zone.

```
global# prctl -n zone.cpu-cap -t privileged -v 300 -i zone web
```

When using the `prctl(1)` command, the CPU resource constraint is defined as an integer value measuring hundredths of a CPU. In the previous sample command, 300 equates to 300 hundredths of one CPU, or more simply 3.00 CPUs.

Private Pool

There are situations in which a workload should be given exclusive access to a fixed set of processors, or to a set of processors that can grow and shrink dynamically. When using such a *private pool*, a zone's processes cannot use processors not assigned to this pool. Conversely, processes in other zones or the global zone cannot use this zone's processors. For example, use the following `zonecfg` command to dedicate two processors to a zone.

```
global# zonecfg -z web
zonecfg:web> add dedicated-cpu
zonecfg:web:dedicated-cpu> set ncpus=2
zonecfg:web:dedicated-cpu> end
zonecfg:web> exit

global# zoneadm -z web reboot
```

Global zone administrators can view the current configuration of pools by using the `poolstat(1M)` command. A private pool has a name beginning with `SUNWtmp_` followed by the name of the zone.

```
global# poolstat
                           pset
 id pool                 size used load
  1 SUNWtmp_web             2 0.00 0.00
  0 pool_default            2 0.00 0.24
```

Note that the pool is associated with a processor set, which contains the CPUs. The size of a pool can be changed by changing the number of CPUs in the processor set. This requires learning the name of the processor set.

```
global# poolstat -r pset
 id pool              type rid rset              min  max size used load
  1 SUNWtmp_web       pset   1 SUNWtmp_web       2    2    2   0.00 0.00
  0 pool_default      pset  -1 pset_default      1  66K    1   0.00 0.24
```

The column labeled `rset` lists the name of the processor set. The quantity of CPUs in the processor set can be changed with the following `poolcfg` command.

```
global# poolcfg -dc 'transfer 1 from pset SUNWtmp_web to pset \
pset_default'
```

The compute needs of some workloads are more dynamic than others. The Solaris OS can shift CPUs from one pool to another as the needs of workloads change (while still preventing abuse) by assigning a range of values. This Solaris OS feature must be manually enabled using the `svcadm` command.

```
global# svcadm enable pools/dynamic
```

The following commands allocate between two and four processors to the zone.

```
global# zonecfg -z web
zonecfg:web> add dedicated-cpu
zonecfg:web:dedicated-cpu> set ncpus=2-4
zonecfg:web:dedicated-cpu> set importance=5
zonecfg:web:dedicated-cpu> end
zonecfg:web> exit

global# zoneadm -z web reboot
```

The `importance` parameter tells the operating system the relative importance of this pool, compared to other pools. For more information about the ability to load balance by automatically shifting CPUs from one pool to another, see *System Administration Guide: Solaris Containers — Resource Management and Solaris Zones* located at `http://docs.sun.com/app/docs/doc/817-1592/zones.intro-1?a=view`

Shared Pool

Similar to a private pool, a *shared pool* is a set of processors that is shared by multiple zones. The most common use of private pools and shared pools is to reduce software license costs. Another use is preventing an interrupt handler from running on a CPU that should only be running processes of a particular workload.

The creation and use of shared pools is beyond the scope of this document. See *System Administration Guide: Solaris Containers — Resource Management and Solaris Zones* located at `http://docs.sun.com/app/docs/doc/817-1592/zones.intro-1?a=view` for more information about Solaris 10 OS shared pools and related resource controls.

Miscellaneous Controls

One well-known method to over-consume system resources is a fork-bomb. This method does not necessarily consume a great deal of memory or CPU resources, but rather seeks to use up all of the process slots in the kernel's process table. In the Solaris OS, a running process starts with just one thread of execution, also called a Light Weight Process (LWP). Many programs generate new threads, becoming multithreaded processes. By default, Solaris systems with a 64-bit kernel can run over 85,000 LWPs simultaneously. A booted zone that is not yet running any applications has approximately 100 to 150 LWPs. To prevent a zone from using too many LWPs, a limit can be set on their use. The following command sets a limit of 300 LWPs for a zone.

```
global# zonecfg -z web
zonecfg:web> set max-lwps=300
zonecfg:web> exit

global# zoneadm -z web reboot
```

This parameter can be used, but should not be set so low that it impacts normal application operation. An accurate baseline for the number of LWPs for a given zone should be determined in order to set this valuable at an appropriate level. The number of LWPs used by a zone can be monitored using the following `prstat` command.

```
global# prstat -LZ
[...]
ZONEID    NLWP  SWAP   RSS MEMORY      TIME  CPU ZONE
     0     248  468M  521M   8.6%   0:14:18 0.0% global
    37     108   76M   61M   1.0%   0:00:00 0.0% web


Total: 122 processes, 356 lwps, load averages: 0.00, 0.00, 0.01
[...]
```

In this example, the `web` zone currently has 108 LWPs. This value changes as processes are created or exit. It should be inspected over a period of time in order to establish a more reliable baseline, and updated when the software, requirements, or workload change.

Using the `max-lwps` resource control successfully usually requires the use of a CPU control, such as the FSS or pools to ensure that there is enough CPU power in the global zone for the platform administrator to fix any problems that might arise.

File Integrity Checks

A Solaris Zone is just like any other operating environment when it comes to file integrity checking. A user running within the zone can use a tool like the Basic Audit and Reporting Tool (BART) in the Solaris OS to create or compare point-in-time snapshots of a given file hierarchy. By preserving this operational consistency with other operating systems, zones also inherit their primary weaknesses, including:

- The file integrity checking tool is stored and run from within the “system” being monitored.
- The configuration rules and results of the file integrity checks are stored on the “system” being monitored.

Both of these issues are problematic, as a malicious administrator or attacker could simply replace the binary associated with the tool, alter the configuration or rules, or even prevent the checks from running in the first place. Fortunately, when using zones, there is another way.

BART or other compatible file system integrity checking tools can be run from the global zone and peer into specific running zones to evaluate their files. This is a useful technique as the scans can be scheduled within the context of the global zone. Similarly, the configuration and result files can be stored in the global zone, protecting

them from users and processes running from within the zone being evaluated. Using BART in this way helps to improve overall integrity of the processing and results because they are being created outside the scope of the zone.

```
# zonecfg -z web info zonepath
zonepath: /pool/apps/apache/zone1

# bart create -R /pool/zones/web/root
! Version 1.0
! Sunday, September 21, 2008 (14:24:38)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/bin L 9 120777 - 48d3c620 0 0 ./usr/bin
/etc/.java/.systemPrefs/.system.lock F 0 100644
owner@:execute:deny,owner@:read_data/write_data/append_data/write_xattr
/write_attributes/write_acl/write_owner:allow,group@:write_data/append_
data/execute:deny,group@:read_data:allow,everyone@:write_data/append_da
ta/write_xattr/execute/write_attributes/write_acl/write_owner:deny,ever
yone@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
47e95fa4 0 2 d41d8cd98f00b204e9800998ecf8427e
[...]
```

Caution – Accessing files in a non-global zone from the global zone potentially is unsafe. If caution is not taken, processes could inadvertently access, manipulate, or delete files outside of the targeted non-global zone. The use of BART, as described previously, is safe in that it performs read-only operations that do not follow symbolic links. These two restrictions are necessary to prevent malicious users in the zone from potentially damaging files that exist in the global zone or other zones.

The following example helps to explain these restrictions. In this example, two files within the zone are inspected, */etc/myfile1* and */etc/myfile2*. The */etc/myfile1* file is a regular text file, and the */etc/myfile2* file is a symbolic link. BART properly recognizes the symbolic link and does not attempt to follow it. However, a problem could arise if a process performing this evaluation and running in the global zone attempts to follow that symbolic link. The process exits the zone's root file system space and evaluates the */etc/passwd* file in the global zone. This can create a source of confusion and could lead to questionable results for these kinds of read-only operations.

```
# bart create -R /pool/zones/web/root -I /etc/myfile1 /etc/myfile2
! Version 1.0
! Sunday, September 21, 2008 (14:35:30)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/etc/myfile1 F 9 100644
owner@:execute:deny,owner@:read_data/write_data/append_data/write_xattr
/write_attributes/write_acl/write_owner:allow,group@:write_data/append_
data/execute:deny,group@:read_data:allow,everyone@:write_data/append_da
ta/write_xattr/execute/write_attributes/write_acl/write_owner:deny,ever
yone@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
48d69359 0 0 4f7c53173cc5334259c16d415ee959f1
/etc/myfile2 L 46 120777 - 48d69352 0 0
../../../../../../../../../../../../etc/passwd
```

For write operations, the results could be disastrous. Consider an example where an attacker creates the following */etc/motd* file in the zone:

```
web# ln -s ../../../../../../../../../etc/nologin /etc/motd
```

Normally, this would not be a cause for alarm. In this example, however, an unsuspecting administrator in the global zone could change the contents of the message of the day file by using the following `touch` and `echo` commands.

```
global# touch /pool/zones/web/etc/motd
global# echo "This is just a test" > /pool/zones/web/etc/motd
```

In standard configurations this would not be a problem. However, if this operation is executed from the global zone, this seemingly benign operation can lead to a denial-of-service attack, where normal users are no longer able to log in to the global zone as a result of the */etc/nologin* file created by following the */etc/motd* link.

```
global# ls -l /etc/nologin
-rw-r--r--   1 root     root          20 Sep 21 14:47 /etc/nologin

global# cat /etc/nologin
This is just a test
```

An added benefit of using sparse-root non-global zones is that their inherited package directories do not need to be validated by BART, as they are mounted from the global zone as read-only loopback mounts that cannot be changed from within the zone. This means that large numbers of the files and directories made available to the zone do not

need to be scanned using BART. This represents a significant savings in terms of time and I/O load on the system. The same is true for other file systems that are mounted into a non-global zone using this strategy, allowing the administrator to focus solely on content that is writable within the zone.

Security Auditing

For many environments it is critical to preserve a record of what activities occur on a system. This information is useful for detecting potential security breaches by revealing suspicious or abnormal patterns of system usage. Solaris OS auditing features can be used to provide a record of system activities and administrative actions. Auditing is disabled by default and must be enabled using the `bsmconv` command.

```
global# /etc/security/bsmconv
This script is used to enable Solaris Auditing and device allocation.
Shall we continue with the conversion now? [y/n] y
bsmconv: INFO: checking startup file.
bsmconv: INFO: turning on audit module.
bsmconv: INFO: initializing device allocation.

Solaris Auditing and device allocation is ready.
If there were any errors, please fix them now.
Configure Solaris Auditing and device allocation by editing files
located in /etc/security.
Reboot this system now to come up with auditing and device allocation
enabled.
```

Auditing in the Solaris OS supports two modes of operating with respect to zones.

- *Global zone-based auditing*
  In this mode the Solaris OS audit daemon (`auditd`) is run only from the global zone. There is a single queue for audit events, and audit trails are stored only in the global zone. This is inherently good for security as the audit trails cannot be modified or destroyed even in the event of an attacker gaining administrative access within a zone. In addition, the audit trail of all zones is stored sequentially in the audit trail, making it easier to analyze events happening across multiple zones. By default, global zone-based auditing is configured. The following `auditconfig` command can be used to set this behavior explicitly.

```
global# auditconfig -setpolicy -perzone
```

- *Per-zone-based auditing*
  In this mode a separate audit daemon is run within each zone. There is a unique audit queue for each zone, and each zone's audit trail is stored locally within the zone. This approach is useful when auditing is necessary for one or more (but not all) of the zones, or when a zone administrator must be able to access the audit trail. It can also

help when the zones use different naming services that might specify different auditing configurations, such as `audit_user`. The security downside of this approach is that administrative users in the zone can disable the auditing service as well as modify or destroy zone-specific audit records. Use the following `auditconfig` command for per-zone-based auditing.

```
global# auditconfig -setpolicy +perzone
```

The remainder of this section uses a global zone-based auditing configuration. Several different auditing policy parameters can be used based upon an organization's specific requirements. These policy settings can be configured using the `auditconfig(1M)` command as shown in the following example.

```
global# auditconfig -setpolicy +argv,zonename
global# auditconfig -getpolicy
audit policies = argv,cnt,zonename
```

Note – Policy settings defined using the `auditconfig(1M)` command must be included in the */etc/security/audit_startup* file if they are to be preserved across reboots.

The previous example configures Solaris OS auditing to record the name of the zone associated with each audit record and the command line arguments associated with `execv(2)` audit records. These settings are used in the examples that follow. With this policy configuration complete, the actual audit configuration should be determined. Because the audit events and classes that can be tracked are no different from those that could be used in the Solaris OS global zone, they are not described in detail here. See the `audit_control(4)` and `audit_user(4)` man pages for information about the audit configuration files.

The following audit configuration file is used in the example that follows.

```
global# cat /etc/security/audit_control
dir: /var/audit
flags: lo,ad,ex
minfree:20
naflags: lo,ex
```

The audit configuration must be present not only in the global zone but also in all of the zones to be audited. To simplify administration and ensure that all of the audit configuration files in the global and non-global zones are identical, the */etc/security* directory is mounted as a read-only loopback mount from the global zone to all of the non-global zones. This is accomplished using the following commands for each zone.

```
global# zonecfg -z web
zonecfg:web> add fs
zonecfg:web:fs> set dir=/etc/security
zonecfg:web:fs> set special=/etc/security
zonecfg:web:fs> set type=lofs
zonecfg:web:fs> add options ro,nodevices,nosetuid
zonecfg:web:fs> end
zonecfg:web> end

global# zoneadm -z web reboot
```

There are a few side effects of using this approach, as opposed to mounting each audit configuration file directly. Password history functionality does not work because it causes writes to the */etc/security/passhistory* file. Furthermore, this approach is not recommended when Solaris Trusted Extensions is enabled. Other sensitive files stored under this directory tree should not be exposed to non-global zones. In these scenarios, the individual files needed can be mounted rather than mounting the entire directory.

Once the zones are rebooted to install the loopback mount, each zone contains an audit configuration that is identical to that in the global zone. Next, the auditing service is instructed to reread the updated */etc/security/audit_control* configuration file. This ensures that audit classes are identified in the `flags` and `naflags` parameters are audited on the system. This is accomplished using the `audit(1M)` command.

```
global# audit -s
```

For this example, it is useful to start with a fresh audit trail. Run the `audit` command again using an option that closes the existing audit trail and opens a new one.

```
global# audit -n
```

At this point, Solaris OS auditing is enabled, configured, and running. The next step is to log in remotely to a Solaris Zone on the system and execute a few commands in order to demonstrate the kind of information captured in the audit trail.

```
web$ cat /etc/passwd
root:x:0:0:Super-User:/root:/sbin/sh
[...]

web$ mkdir /myhome
web$
```

Back in the global zone, the `auditreduce(1M)` and `praudit(1M)` commands can be used to select and display specific audit trail records. In the following example, the `auditreduce` command selects records from the `web` zone's active (not terminated) audit trail that include events in the `ex` class.

```
global# auditreduce -c ex -z web /var/audit/*not* | praudit -s
[...]
header,159,2,AUE_EXECVE,,gateway,2008-08-29 13:02:59.101 -04:00
path,/usr/bin/cat
attribute,100555,root,bin,0,276,0
exec_args,2,cat,/etc/passwd
subject,2000,2000,2000,2000,2000,11997,2414533645,11766 5632 192.168.2.1
return,success,0
zone,web

header,168,2,AUE_EXECVE,,gateway,2008-08-29 13:03:11.877 -04:00
path,/usr/bin/mkdir
attribute,100555,root,bin,0,388,0
exec_args,2,mkdir,/myhome
subject,2000,2000,2000,2000,2000,11998,2414533645,11766 5632 192.168.2.1
return,success,0
zone,web
[...]
```

The `ex` class includes the following audit events.

```
global$ bsmrecord -c ex

exec
  system call exec                See exec(2)
  event ID    7                   AUE_EXEC
  class       ps,ex               (0x40100000)
      header
      path
      [attribute]                 omitted on error
      [exec_arg]                  output if argv policy is set
      [exec_env]                  output if arge policy is set
      subject
      [use_of_privilege]
      return

execve
  system call execve              See execve(2)
  event ID    23                  AUE_EXECVE
  class       ps,ex               (0x40100000)
      header
      path
      [attribute]                 omitted on error
      [exec_arg]                  output if argv policy is set
      [exec_env]                  output if arge policy is set
      subject
      [use_of_privilege]
      return
```

A review of these records shows that two commands (`/usr/bin/cat` and `/usr/bin/mkdir`) were run by UID and GID `2000` (real and effective) in the zone `web`. This user was logged into the system from IP address `192.168.2.1`. Both of the programs executed were owned by `root`, had group `bin`, and had permissions `555`. The commands were able to be executed (`success` return value). For more information about the fields available in an audit record, see the `audit.log(4)` man page.

In addition to using the binary audit trail, Solaris OS auditing can be configured to send audit records to the `syslog` system log. To configure Solaris OS auditing to use `syslog`, add the following line to the */etc/security/audit_control* file. Note that the `lo` and `ex` audit flags are specified. This parameter can be adjusted based on the audit records that need to be captured and transmitted through `syslog`.

```
plugin: name=audit_syslog.so; p_flags=lo,ex
```

The output format differs from that produced by `praudit`. The following two records show what is transmitted to the system log for the audit trail produced.

```
[...]
Aug 29 13:02:59 gateway audit: [ID 702911 audit.notice] execve(2) ok
session 2414533645 by 2000 as 2000:2000 in web from 192.168.2.1 proc_uid
bin obj /usr/bin/cat

Aug 29 13:03:11 gateway audit: [ID 702911 audit.notice] execve(2) ok
session 2414533645 by 2000 as 2000:2000 in web from 192.168.2.1 proc_uid
bin obj /usr/bin/mkdir
[...]
```

Solaris Trusted Extensions

Solaris Trusted Extensions extends default Solaris OS security features by enforcing a mandatory access control policy. Sensitivity labels are automatically applied to all sources of data (including networks, file systems, and graphical interface windows), as well as consumers of data (user and processes). Access to all data is restricted based on the relationship between the label of the data (object) and the consumer (subject).

Solaris Trusted Extensions implements its label strategy using zones. Each non-global zone has a single and unique label associated with it, and everything within a given non-global zone, including its processes, devices, and data, is assigned the label of that zone. The global zone is the only exception as it is assigned both the `ADMIN_HIGH` and `ADMIN_LOW` labels. Zones are assigned their respective labels when they are booted. Once running, this label cannot be changed dynamically.

When enabled, Solaris Trusted Extensions enforces a number of additional constraints upon inter-zone communication:

- Communication between zones using networking or door services is permitted in accordance with the mandatory access control policy. Doors must be created in the global zone — only the global zone has sufficient rights to create the door. Further, the global zone must specifically permit a given zone to use a door, thereby ensuring strict global zone control over such communication.
- Communication between processes using FIFOs, such as named pipes, is permitted in accordance with a *write up* policy. Processes operating at a lower sensitivity level can write only to processes operating at an equal or higher sensitively level. Writes from higher sensitivity levels to lower levels are not permitted.
- In the Solaris 10 OS, the kernel prevents UNIX domain sockets from being connected across zone boundaries, even if the rendezvous file is shared (for example, `lofs`). Since UNIX domain sockets cannot be used for cross-zone communication, Solaris Trusted Extensions must utilize the `all-zones` networking option along with multilevel ports for services that are privileged to listen for connections from other zones.
- File system mounts across non-global zones are restricted such that a file system can be writable in only one of the non-global zones. However, these file systems can be read by any number of zones that have a dominating label.

For more information about the architecture and functionality enabled by Solaris Trusted Extensions, see "Solaris Trusted Extensions: Architectural Overview" located at `http://opensolaris.org/os/community/security/projects/tx/TrustedExtensionsArch.pdf`.

Summary

This article presented a detailed analysis of the security capabilities and features of Solaris Zones as they exist today. The capabilities fostered by this functionality provide a strong foundation upon which organizations can build virtualization strategies and deploy network services. The virtualization and consolidation capabilities of Solaris Zones provide a security model that provides unique capabilities, including sparse-root functionality, limited privileges, kernel and user space root kit protections, and more.

Solaris Zones are a key element of the Solaris Common Criteria Certification evaluation for the Solaris OS and Solaris Trusted Extensions. Zones are included in the security target that achieved Evaluation Assurance Level 4+ (EAL4+) certification against the Controlled Access Protection Profile (CAPP), the Role-based Access Control Protection Profile (RBACPP), and the Labeled Security Protection Profile (LSPP). For more information, see Solaris Common Criteria Certification on Sun's Web site at `http://www.sun.com/software/security/securitycert/`.

About the Authors

Glenn Brunette is a Global Systems Engineering Director and Chief Security Architect at Sun, where he leads a global team focused on advanced information security architectures. For his achievements and contributions to information security, Glenn was named a Sun Distinguished Engineer, an honor granted to less than 100 people in the company. His team was awarded a 2008 Sun Innovation Award for their project, Sun Systemic Security.

For over 15 years, Glenn has architected, developed, and delivered security solutions for a wide range of customers and industries. Currently, Glenn works in the Chief Architect's Office where he defines Sun's global security strategy and architecture, and works to improve the security of products and services used by Sun's customers. Glenn is the founder of Sun's Systemic Security approach, an OpenSolaris™ Security Community Leader, the co-founder of the Solaris Security Toolkit software, and a frequent author, contributor, and speaker at both Sun and industry events. Externally, Glenn has served in leadership positions at the National Cyber Security Partnership, the Enterprise Grid Alliance, as well as at the Center for Internet Security.

Jeff Victor is a Sr. Systems Engineer at Sun Microsystems, Inc. He uses his 20+ years of UNIX experience to drive the adoption of Sun's virtualization technologies through customer education, lecturing at industry conferences, and writing Sun BluePrints articles, other published documents, and his blog. Jeff is an OpenSolaris Zones Community Leader and original developer of the OpenSolaris Zones FAQ.

Acknowledgments

The authors would like to thank the following people for their inspiration, technical feedback, and overall support in the development of this article: John Banghart, Glenn Faden, John Howard, Jerry Jelinek, Dan McDonald, Erik Nordmark, Denny Olson, Scott Rotondo, Christoph Schuba, Sharon Veach, and Gary Winiger.

References

The Center for Internet Security, "Solaris 10 Security Benchmark v4.0"
`http://www.sun.com/security/docs/CIS_Solaris_10_Benchmark_v4.pdf`

"An Overview of Solaris 10 Operating System Security Controls"
(Companion piece to the CIS "Solaris 10 Security Benchmark v4.0")
`http://www.sun.com/security/docs/s10-cis-appendix-v1.1.pdf`

"Eliminating Web Page Hijacking Using Solaris 10 Security"
`http://www.sun.com/software/solaris/howtoguides/s10securityhowto.pdf`

"Introduction to Solaris Zones," *System Administration Guide: Solaris Containers — Resource Management and Solaris Zones*
`http://docs.sun.com/app/docs/doc/817-1592/zones.intro-1?a=view`

"Networking in Exclusive-IP Non-Global Zones," *System Administration Guide: Solaris Containers — Resource Management and Solaris Zones*
http://docs.sun.com/app/docs/doc/817-1592/gepxo?a=view

"Networking in Shared-IP Non-Global Zones," *System Administration Guide: Solaris Containers — Resource Management and Solaris Zones*
http://docs.sun.com/app/docs/doc/817-1592/z.admin.ov-9?a=view

OpenSolaris Zones and Containers FAQ
http://opensolaris.org/os/community/zones/faq/

"Privileges in a Non-Global Zone," *System Administration Guide: Solaris Containers — Resource Management and Solaris Zones*
http://docs.sun.com/app/docs/doc/817-1592/z.admin.ov-18?a=view

Schuba, Christoph. "Solaris Zones Security," *Sun BluePrints Online*, December 2008
https://wikis.sun.com/display/BluePrints/Solaris+Zones+Security

"Security Restrictions and File System Behavior," *System Administration Guide: Solaris Containers — Resource Management and Solaris Zones*
http://docs.sun.com/app/docs/doc/817-1592/z.admin.ov-7

Solaris Common Criteria Certification
http://www.sun.com/software/security/securitycert/

"Solaris Trusted Extensions: Architectural Overview"
http://opensolaris.org/os/community/security/projects/tx/TrustedExtensionsArch.pdf

System Administration Guide: Solaris Containers — Resource Management and Solaris Zones
http://docs.sun.com/app/docs/doc/817-1592

Ordering Sun Documents

The SunDocs℠ 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.

Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 USA **Phone** 1-650-960-1300 or 1-800-555-9SUN (9786) **Web** sun.com

Printed in USA 12/08