New Security Features in Solaris 10 and DTrace

Chandan B.N, Sun Microsystems Inc.
17th Annual FIRST Conference; June 2005

A secure and robust operating system plays a key role in keeping a computing environment safe and secure. This paper illustrates how the new security features and improvements in the latest release of Solaris Operating Environment can help defend system integrity, enable secure computation with ease of deployment and manageability. There also an introduction to DTrace which is a powerful infrastructure to observer the behaviour of the system.

The most significant developments in Solaris 10 are improved hardening and minimization, application of principle of least privileges, introduction of zones and a new cryptographic framework. Apart from these there are a number of minor additions and enhancements that help in improving the OS security.

Solaris Privileges (Process Rights Management)

The traditional UNIX privilege model associates all privileges with the effective uid 0 or root. A privileged process if compromised can be used to gain full access to the system. It is also not possible to extend an ordinary user's capabilities with a restricted set of privileges.

Solaris 10 addresses these with the introduction of the *principle of least privileges* [Saltzer & Schroeder 1975] which says that a process should be given no more privilege than necessary to perform its job. Process Rights Management extends the Solaris process model with privilege sets. Each privilege set contains zero or more privileges. Each process has four privilege sets:

- *The Effective set* (E) is active privileges required to perform a privileged action; this is the set of privileges the kernel verifies its privilege checks against.
- *The Permitted set* (P) contains privileges a process is allowed to use. So a process is free to remove from or add to E any privilege, as long as it carries that privilege in P.
- *The Inheritable set* (I) allows a process to pass privileges on to child processes.
- *The Limit set* (L) is the upper limit of privilege a process and its off-springs can ever obtain. It takes effect only at exec-time.

Here is the list of privileges in Solaris 10:

```
“contract_event”  Request reliable delivery of events
“contract_observer” Observe contract events for other users
"cpc_cpu”         Access to per-CPU perf counters
"dtrace_kernel"   DTrace kernel tracing
"dtrace_proc"     DTrace process-level tracing
"dtrace_user"     DTrace user-level tracing
```

```
"file_chown"      Change file's owner/group IDs
"file_chown_self" Give away (chown) files
"file_dac_execute" Override file's execute perms
"file_dac_read"   Override file's read perms
"file_dac_search" Override dir's search perms
"file_dac_write"  Override (non-root) file's write perms
"file_link_any"   Create hard links to different uid files
"file_owner"      Non-owner can do misc owner ops
"file_setid"      Set uid/gid (non-root) to diff id
"ipc_dac_read"    Override read on IPC, Shared Mem perms
"ipc_dac_write"   Override write on IPC, Shared Mem perms
"ipc_owner"       Override set perms/owner on IPC
"net_icmpaccess"  Send/Receive ICMP packets
"net_privaddr"    Bind to privilege port (<1023+extras)
"net_rawaccess"   Raw access to IP
"proc_audit"      Generate audit records
"proc_chroot"     Change root (chroot)
"proc_clock_highres" Allow use of hi-res timers
"proc_exec"       Allow use of execve()
"proc_fork"       Allow use of fork*() calls
"proc_info"       Examine /proc of other processes
"proc_lock_memory" Lock pages in physical memory
"proc_owner"      See/modify other process states
"proc_priocntl"   Increase priority/sched class
"proc_session"    Signal/trace other session process
"proc_setid"      Set process UID
"proc_taskid"     Assign new task ID
"proc_zone"       Signal/trace processes in other zones
"sys_acct"        Manage accounting system (acct)
"sys_admin        System admin tasks (node/domain name)
"sys_audit"       Control audit system
"sys_config"      Manage swap
"sys_devices"     Override device restricts (exclusive)
"sys_ipc_config"  Increase IPC queue
"sys_linkdir"     Link/unlink directories
"sys_mount"       Filesystem admin (mount,quota)
"sys_net_config"  Config net interfaces,routes,stack
"sys_nfs"         Bind NFS ports and use syscalls
"sys_res_config"  Admin processor sets, res pools
"sys_resource"    Modify res limits (rlimit)
"sys_suser_compat" 3rd party modules use of suser
"sys_time"        Change system time

Note: "bold privileges" are basic non-root privileges given to
ordinary users by default.
```

Old applications which operated by old effective uid mechanisms would still continue to operate the old way. Backward compatibility with applications which are not aware of the new privileges is achieved by what we call *Privilege Awareness*. A process becomes privilege aware if it modifies its E or P set or when it requests to become privilege aware.

Following example illustrates how NFS lockd (1M) daemon has just the privilege to bind to only certain ports. If the daemon is exploited by a vulnerability to fork a child, then the child would not even have the basic privileges that may be available to other ordinary users!

```
# ppriv -v 'pgrep lockd'
182:    /usr/lib/nfs/lockd
flags = PRIV_AWARE
        E: sys_nfs
        I: none
        P: sys_nfs
        L: none

# ppriv -l -v sys_nfs
sys_nfs
        Allows a process to perform Sun private NFS specific system
calls. Allows a process to bind to ports reserved by NFS: ports 2049
(nfs) and port 4045 (lockd).
```

Links:
Casper Dik's weblog entry http://blogs.sun.com/roller/page/casper/20040722

Zones (Containers)

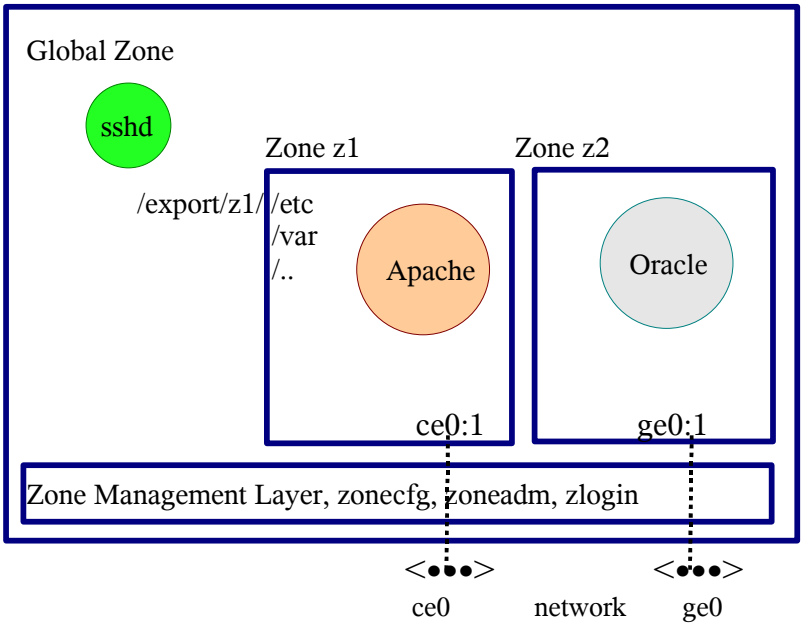
Zones provide a new secure isolation primitive for Solaris, which is flexible, scalable and lightweight. These are virtualized OS services which look like different Solaris instances. Together with the existing *Solaris Resource Management* (SRM) framework, Solaris Zones forms the basis of containment technology in Solaris.

At the highest level, zones are lightweight "sandboxes" within an operating system instance, in which one or more applications may be installed and run without affecting or interacting with the rest of the system. Users or applications in one zone cannot see or access contents in another zone. Each zone can have an IP address associated with it. Who ever logs into that address would feel that they are in a separate machine. Zones are great for isolation of network services, sharing resources on a large server, or creating development environments. Each zone has it's own root password, it's own /etc/ and /var/ files, and it's own OS files if installed in that way.

Zones can be used for quarantining potentially risky software or isolating multiple dis-trusting parties. They help in containing potential damage by a breach. The default system is called the *Global Zone* can observe all activities inside each zone, and not be seen by software in each non-global zone. It can change the contents or processes in each non-global zones. A global zone can contain intrusion detection systems (IDS) that is undetectable and tamper-protected from zones. Non-global Zones run with less privileges.

Zones differ from other hardware partitioning or virtual machine technologies like User Mode Linux or IBM LPARs, by having a single underlying operating system kernel. Zones are comparable to *Jails* in FreeBSD and *VServer* in Linux. Zones are designed for

more commercial workloads and are well integrated with rest of the Operating System.



A global zone containing two zones

In the following example, a zone is created that by default will share most of the OS with the global zone.

```
# zonecfg -z z1
z1: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:z1> create
zonecfg:z1> set autoboot=true
zonecfg:z1> set zonepath=/export/z1
zonecfg:z1> add net
zonecfg:z1:net> set address=192.34.56.78
zonecfg:z1:net> set physical=ce0
zonecfg:z1:net> end
zonecfg:z1> verify
zonecfg:z1> exit
# mkdir /export/z1
# chmod 700 /export/z1
# zoneadm -z z1 install
Preparing to install zone <z1>.
Creating list of files to copy from the global zone.
Copying <2574> files to the zone.
[...]
```

A single server may run many zones some. Here we list all the zones and login to one of them,

```
# zoneadm list -cv
  ID NAME             STATUS         PATH
```

```
   0 global           running        /
   1 testzone1        running        /export/testzone1
   2 workzone1        running        /export/workzone1
   3 z2               running        /export/z2
   4 z1               running        /export/z1
#
# zlogin workzone1
[Connected to zone 'workzone1' pts/2]
Last login: Tue Apr 25 09:39:57 on pts/2
Sun Microsystems Inc.   SunOS 5.10      Generic January 2005
Welcome to Sol10_Generic on sfe2900
#
```

Resource control is possible for CPU and Memory in a variety of ways. The *Fair Share Scheduler* can divide CPU resources between busy zones depending on ratios. The following demonstrates the result of attempting to give workzone1 60%, z2 30% and z1 10% of the CPUs:

```
# prstat -Z
PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
2008 root     4000K 1168K cpu513  28    0   0:02:11 3.7% cpuhog.pl/1
2018 root     4000K 1168K cpu1    32    0   0:02:11 3.7% cpuhog.pl/1
[...]
ZONEID    NPROC  SIZE   RSS MEMORY      TIME  CPU ZONE
     2       51  182M   93M   0.5%   0:37:27  59% workzone1
     4       51  182M   92M   0.5%   0:16:25  30% z2
     3       51  183M   93M   0.5%   0:16:30  10% z1
     0       61  359M  194M   1.1%   0:00:11 0.1% global
     1       34  116M   72M   0.4%   0:00:12 0.0% testzone1
Total: 248 processes, 659 lwps, load averages: 51.19, 40.28, 20.52
```

Links:

Solaris Zones: "Operating System Support for Consolidating Commercial Workloads" (USENIX LISA '04) Daniel Price and Andrew Tucker – Sun Microsystems, Inc.
http://www.sun.com/bigadmin/content/zones/zones_lisa.pdf

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

Solaris Cryptographic Framework

The *Solaris Cryptographic Framework* (SCF) is a user and kernel space set of Application Programming Interfaces (APIs) and Service Provider Intefaces (SPIs) for providing software and hardware cryptographic algorithms to application and other kernel systems. Userland part of the framework implements the RSA PKCS#11 interface. It can advantage of cryptographic hardware accelerators.

Various components like IKE, Kerberos, IPsec, and SASL now utilize a single cryptographic API which is modular and extensible. The **cryptoadm**(1M) command can be used list all providers, install or uninstall software providers, and enable or disable hardware providers.

SCF provides two new commands **digest**(1) and **encrypt**(1):

```
$ digest -v -a md5 /usr/bin/ls
md5 (/usr/bin/ls) = b46d86445cb33dff0c3029730aab3a1f

$ encrypt -l
Algorithm       Keysize:  Min   Max (bits)
----------------------------------------
aes                       128   128
arcfour                     8   128
des                        64    64
3des                      192   192
```

Links:
http://www.sun.com/bigadmin/xperts/sessions/12_crypt
http://www.sun.com/bigadmin/features/articles/crypt_framework.html

Solaris Hardening and Minimization

The goal of Solaris minimization and hardening is to defend system from unauthorized access and to provide high assurance of system integrity.

There are a number of features that provide secure deployment, like secure network install, minimal initial install, profile-based install, validated execution or ELF signing (**elfsign(1)**) and file integrity protection (bart (1M)).

Containment of security violations is achieved through minimal process privileges and service containment within a zone. Solaris provides a variety of access control mechanisms like RBAC, file ACLs and packet filtering.

Solaris has Auditing capabilities with centralized logging. Auditing can also be done per zone. Solaris also has automated patch installation, for example it can be configured that only security patches can be applied automatically per schedule.

BART (Integrity)

The *Basic Audit Reporting Tool* (bart (1M)) provides a database of message digest and inode details. This can be helpful for many reasons, such as intrusion detection and forensics.

An example of BART detecting a change is,

```
# find /etc |bart create -I > etc1.bart
# vi /etc/passwd
  --- change /etc/passwd ---
# find /etc |bart create -I > etc2.bart
# bart compare etc1.bart etc2.bart
/etc/passwd:
  size  control:638  test:627
```

```
  mtime  control:427449ab  test:4252d3f9
  contents  control:e68173ba26f038728412237532057366
test:d25c811fbb8efe068f16e9129e861a41
```

Links:
http://www.sun.com/blueprints/0405/819-2260.pdf

IP Filter

IP Filter is a packet filtering firewall now integrated into Solaris. It has a simple command line interface and a lightweight look and feel, but is quite powerful.

The following is a summary of IP Filter usage,

```
# ipf -Fa -f /etc/ipf/ipf.conf          # load rules
# ipfstat -ionh                         # list rules
# ipnat -CF -f /etc/ipf/ipnat.conf      # load NAT
# ipnat -l                              # list NAT
# vi /etc/ipf/pfil.ap                   # activate interface
# ls /usr/share/ipfilter/examples       # examples,
BASIC.NAT   example.10  example.2   example.6   example.sr  ip_rules
pool.conf
BASIC_1.FW  example.11  example.3   example.7   firewall    mkfilters
server
BASIC_2.FW  example.12  example.4   example.8   ftp-proxy   nat-setup
tcpstate
example.1   example.13  example.5   example.9   ftppxy      nat.eg
```

This is a simple rule set that blocks all inbound TCP traffic except for SSH, and allows all outbound traffic,

```
# cat /etc/ipf/ipf.conf
pass in quick on hme0 proto tcp from any to any port = 22 keep state
block return-rst in log on hme0 proto tcp from any to any
pass out on hme0 proto tcp from any to any keep state

# ipfstat -ionh
0 @1 pass out on hme0 proto tcp from any to any keep state
2 @1 pass in quick on hme0 proto tcp from any to any port = ssh keep
state
9 @2 block return-rst in log on hme0 proto tcp from any to any
```

Links:
IP Filter Homepage: http://coombs.anu.edu.au/~avalon
IP Filter Examples: http://coombs.anu.edu.au/ipfilter/examples.html
IP Filter Docs: http://docs.sun.com/app/docs/doc/816-4554

DTrace

One of the popular new features in Solaris 10 is DTrace - the dynamic tracing facility. While it is not a tool that would come under security category, it provides a powerful infrastructure to permit administrators, developers, and service personnel to concisely answer arbitrary questions about the behavior of the operating system and user programs. It can be quite helpful in locating the root cause of aberrant system behavior.

DTrace instrumentation can be done by the new D programming language. Here a *probe* is a point of instrumentation. There are several thousands of these on the system. When DTrace is not in use it has zero probe effect on the system. Using D language one can specify a set of *actions* to be taken when a probe fires. Using a *predicate*, one can allow actions to be taken only when certain conditions are met. Actions are programmable.

DTrace can arbitrarily and dynamically instrument both the kernel and running applications such that data and control flow can be followed across boundaries. This example measures the time spent in read(2) system call by various processes.

```
#!/usr/sbin/dtrace -sq

syscall::read:entry
{
    self->t = timestamp;
}

syscall::read:return    /* --- this is a probe specifier --- */
/self->t != 0/          /* --- this is a predicate --- */
{
    /* --- these are actions --- */
    printf("%d/%d spent %d nsecs in read(2)\n",
        pid, tid, timestamp - self->t);

    self->t = 0;
}


# ./read.d
Xorg - 626/1 spent 2576 nsecs in read(2)
gnome-settings-d - 755/1 spent 3495 nsecs in read(2)
xscreensaver - 750/1 spent 4420 nsecs in read(2)
gnome-session - 737/1 spent 4201 nsecs in read(2)
clock-applet - 815/1 spent 4270 nsecs in read(2)
gnome-netstatus- - 821/1 spent 4719 nsecs in read(2)
Xorg - 626/1 spent 3964 nsecs in read(2)
...
```

Here is another example that can trace all programs creating files in /tmp that can help in detecting insecure /tmp file operations.

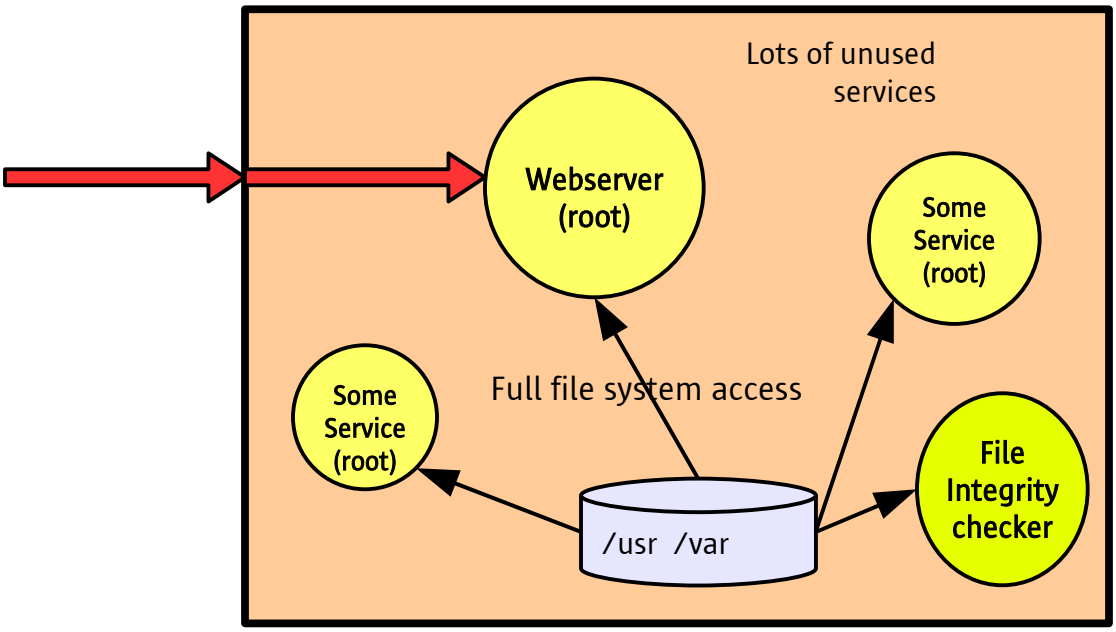
```
#!/usr/sbin/dtrace -qs

dtrace:::BEGIN
{
        printf("CMD\tFILE\n");
}

syscall::open:entry, syscall::open64:entry,syscall::creat:entry,
syscall::creat64:entry
{
        self->traceme = 1
        self->path = copyinstr(arg0);
}

syscall::open:return, syscall::open64:return,syscall::creat:return,
syscall::creat64:return
{
        self->traceme = 0;
        self->path = "";
}

fbt:tmpfs:tmp_create:entry
/self->traceme == 1 && self->path != ""/
{
        printf("%10s\t%s\n", execname, self->path);
}



# ./tmpfiles.d
CMD     FILE
     touch      /tmp/atest-file
```

Links
http://www.sun.com/bigadmin/content/dtrace/

Conclusion

The following figure illustrates how the application of new security features can make a great difference in enhancing the security of a simple web-server. In the first figure the web-server running as root has full access to underlying file system. In the second one it is running with reduced privileges inside a zone protected by a packet filtering firewall in a global zone which is minimized and hardened along with file system integrity checker (bart).

Today's webserver deployment



Solaris 10 protected: Zones, LP and BART