PETER BAER GALVIN

# Pete's all things Sun: swaddling applications in a security blanket

Peter Baer Galvin is the CTO of IT Architecture for Corporate Technologies, a premier systems integrator and VAR (www.cptech.com). Before that, Peter was the systems manager for Brown University's Computer Science Department. He has written articles and columns for many publications and is co-author of the *Operating Systems Concepts* and *Applied Operating Systems Concepts* textbooks. As a consultant and trainer, Peter teaches tutorials and gives talks on security and system administration worldwide. Peter blogs at http://www.galvin.info and twitters as "PeterGalvin."

*pbg@cptech.com*

THE IMMUTABLE SERVICE CONTAINERS (ISC) project seeks to increase systemic security within Solaris. An ISC is a potentially perfect locale in which to run applications where increased security is desired. Within the ISC ecosystem is the ability to clone ISCs and reset them to a known good state, and the potential for automatic actions in case of a security incident. ISC is not currently an integrated or support project, but it is an important security step for Solaris and therefore worth discussing even at this early stage.

## Immutable Service Containers

ISC currently consists of a plan and documents, as well as the "OpenSolaris Immutable Service Container construction kit" [1], a set of tools for building ISCs within OpenSolaris. The project's goal is to go beyond OpenSolaris, creating, for example, an ISC from a VirtualBox virtual machine. Fundamentally, ISCs are a set of tools, steps, and techniques that can be used to more securely run applications in highly managed environments and can be set up and used wherever the operating system or virtualization tools provide the features required. According to their definition, "ISCs provide a security-reinforced container into which a service or set of services is deployed."

The design goals for ISCs include limiting exposure by reducing services and using resource controls to run those that are required, limiting change by making service and critical operating environment configuration read-only, limiting rights based on the least privilege model, and increasing integrity by isolating the service for monitoring and enforcement [2]. There are several benefits to deploying infrastructure based on ISCs, including a more secure starting point for deployment and management, automation of application deployment, built-in best practice security aspects, decreased chance of break-ins, decreased chance of damage from a break-in, and more likely standardization of security within the infrastructure.

The reasoning behind the ISC project is that, even when the "right" steps to increase security are known, they are infrequently followed and even less frequently checked and updated. An ISC has all of those right steps already integrated, easing the effort needed to secure an application's environment. Certainly the world would be a better place

if all applications were run in ISCs, but the first steps are designing, testing, and documenting them. My hope for this column is that it brings attention to ISCs and helps to encourage their propagation.

ISC is a core building block of some security initiatives at Sun and security trends in general. It can be part of an adaptive security architecture [3] which responds to threats quickly while minimizing potential damage. Further, ISCs can be part of an autonomic security layer that can be self-cleansing, self-updating, and can automatically roll back and quarantine its components, even performing a self-assessment or self-destruction if needed.

The ISC Architecture consists of an ISC "dock" and the ISC itself. The dock provides security enforcement, monitoring, resource controls, and other management functions. It communicates via SSH to one or more ISCs. The ISC consists of the security-hardened container (be it a Solaris container, a virtual machine, or other similar structure), plus security functions.

## Hands-on

Even though the ISC project states that it is more of a proof-of-concept than a production-ready service, I thought it would be interesting to take the current implementation for a test drive. Currently, ISCs can be implemented on OpenSolaris via Solaris Zones (a.k.a. "containers"). Other options will be possible in the future (as discussed in the next section). Implementation and configuration of ISCs currently take a few steps, from downloading and running the scripts through editing firewall configuration files. The steps are outlined in the project wiki [4], but I include them here and add some explanation as well as some testing results. If you want to save some steps, you can download a virtual machine image of OpenSolaris already configured with an ISC.

The ISC description so far is certainly high on promise. Unless components or implementations are released officially and supported by Sun, it will be difficult to judge how effectively ISC meets its goals. The current pre-release should give a good indication of how close ISCs are to production usability, how easy they will be to operate, and how close they are to delivering on their promise.

For the purposes of this column I tested the OpenSolaris V 1.0 preview of ISC. At this point ISC is not even an OpenSolaris package. Rather, it's available as a Mercurial (source code management) repository. I started from a fresh copy of OpenSolaris 2009.06. For a shortcut, a prebuilt OpenSolaris containing ISC and an ISC container can be downloaded in OVF format and run as a VM guest inside of an OVF-format supporting virtual machine manager (such as VirtualBox).

First, Mercurial must be installed, and the Mercurial repository containing ISC downloaded:

```
opensolaris$ pfexec pkg install SUNWmercurial
. . .
opensolaris$ hg clone https://kenai.com/hg/isc-source isc
. . .
```

Next, the configuration script is run to modify the system and create an ISC:

```
opensolaris$ pfexec isc/bin/iscadm.ksh
Setting netmask of isc0 to 255.255.255.0
Installing SMF method: /lib/svc/method/svc-isc-enc-swap
Installing SMF manifest: /var/svc/manifest/site/isc-enc-swap.xml
Installing SMF method: /lib/svc/method/svc-isc-enc-scratch
```

```
Installing SMF manifest: /var/svc/manifest/site/isc-enc-scratch.xml
isc1: No such zone configured
Use 'create' to begin configuring a new zone.
A ZFS file system has been created for this zone.
  Publisher: Using opensolaris.org (http://pkg.opensolaris.org/release/).
      Image: Preparing at /export/isc/isc1/zone/root.
      Cache: Using /var/pkg/download.
Sanity Check: Looking for 'entire' incorporation.
  Installing: Core System (output follows)
DOWNLOAD                PKGS         FILES     XFER (MB)
Completed               20/20        3021/3021 42.55/42.55

PHASE                   ACTIONS
Install Phase           5747/5747
   Installing: Additional Packages (output follows)
DOWNLOAD                PKGS         FILES     XFER (MB)
Completed               37/37        5598/5598 32.52/32.52

PHASE                   ACTIONS
Install Phase           7329/7329

            Note: Man pages can be obtained by installing SUNWman
  Postinstall: Copying SMF seed repository ... done.
  Postinstall: Applying workarounds.
         Done: Installation completed in 373.243 seconds.

    Next Steps: Boot the zone, then log into the zone console
                (zlogin -C) to complete the configuration process
 Global zone version: entire@0.5.11,5.11-0.111:20090514T145840Z
Non-Global zone version: entire@0.5.11,5.11-0.111:20090514T145840Z
            Evaluation: Packages in isc1 are in sync with global zone.
Attach complete.
 Global zone version: entire@0.5.11,5.11-0.111:20090514T145840Z
Non-Global zone version: entire@0.5.11,5.11-0.111:20090514T145840Z
            Evaluation: Packages in isc1 are in sync with global zone.
Attach complete.
```

Installation transforms the OpenSolaris deployment from a general-use system to a much more secured environment. Even the boot name changes. The /etc/motd is changed to display a message about unauthorized use. And the GUI login is disabled in favor of command-line interactions. Clearly this should not be done on a desktop deployment of OpenSolaris—it's all about creating secure server environments in which to run services. A Solaris container called "isc1" is preinstalled, with a default password of "iscroot" that needs to be changed. Note that this is not a security hole, because there is no way to connect to the container from outside the system until services are enabled and the global zone is configured to allow communication to the ISC.

A boot environment cache update and reboot gets the system ready for use:

```
opensolaris$ pfexec bootadm update-archive
opensolaris$ pfexec shutdown -g 0 -i 0 -y
```

Once an ISC container is built, it has many interesting aspects. For example, the configuration is hardened, auditing enabled, and the stack set to non-executable. Also, as well as the usual default container file systems, there is a new /scratch one provided. This is a non-persistent encrypted file system that applications can use to securely store log files, temporary files, and other contents. Because ZFS does not yet implement encryption, there is

some indirection involved in the implementation of the encrypted scratch space. Essentially, a ZFS zvol (volume) is the core, and then a LOFI (loop-back file system) is created with encryption enabled (using an ephemeral key that will be lost when the system is shut down) and a zpool on top of that, with the end result of exporting a file system that is encrypted:

```
root@isc1:~# df -kh
Filesystem             size   used  avail capacity  Mounted on
. . .
/scratch               63M    19K    63M     1%    /scratch
. . .
pbg@opensolaris:~$ zfs list
NAME                           USED  AVAIL  REFER  MOUNTPOINT
. . .
rpool/export/scratch           300M  3.31G    19K  /export/scratch
rpool/export/scratch/global    100M  3.31G    19K  /export/scratch/global
rpool/export/scratch/global/   100M  3.41G  1.19M  -
    scratch_file
rpool/export/scratch/isc1      100M  3.31G    19K  /export/scratch/isc1
rpool/export/scratch/isc1/     100M  3.41G  1.19M  -
     scratch_file
scratch-global                71.5K  62.9M    19K  /scratch-global
scratch-isc1                  71.5K  62.9M    19K  scratch-isc1
. . .
pbg@opensolaris:~$ zpool status -v scratch-isc1
      pool: scratch-isc1
     state: ONLINE
     scrub: none requested
    config:

            NAME          STATE     READ     WRITE     CKSUM
            scratch-isc1  ONLINE       0         0         0
              /dev/lofi/3 ONLINE       0         0         0
pbg@opensolaris:~$ lofiadm
Block Device      File                                Options
/dev/lofi/1       /devices/pseudo/zfs@0:1c            Encrypted
/dev/lofi/2       /devices/pseudo/zfs@0:2c,raw        Encrypted
/dev/lofi/3       /devices/pseudo/zfs@0:3c,raw        Encrypted
/dev/lofi/4       /devices/pseudo/zfs@0:5c,raw        Encrypted
```

Once an ISC is created, applications can be installed and enabled within it. From the ISC wiki comes the example of installing and enabling Apache:

```
opensolaris$ pfexec zlogin isc1 pkg install SUNWapch22
opensolaris$ pfexec zlogin isc1 svcadm enable apache22
```

Because the default is for no communication to be allowed to the ISC, the firewall rules much be changed to allow communication. This is done by editing /etc/ipf/ipf.conf and, if the IP address of the ISC guest is 192.168.0.1, adding a line such as:

```
pass in quick on e1000g0 proto tcp from any to 192.168.0.1 port = 80 keep state
```

Because by default the ISC's network is not accessible from outside the system, a new NAT rule has to be put in place to route traffic that reaches the system on port 80 into the ISC housing the Web server. Edit /etc/ipf/ipnat.conf and add a line such as:

```
rdr e1000g0 0.0.0.0/0 port 80 -> 192.168.0.1 port 80
```

For those commands to take effect, the firewall must be told to reload its configuration files via:

```
opensolaris$ pfexec ipf -Fa -f /etc/ipf/ipf.conf
opensolaris$ pfexec ipnat -FC -f /etc/ipf/ipnat.conf
```

From a separate system, pointing a Web browser to the IP address of the host containing the ISC should allow connection to the secure Web server within the container.

The ISC infrastructure includes a new command-line script to manage and modify ISCs. For example, to create a new ISC, say ISC 2, the command line would be:

```
opensolaris$ pfexec isc/bin/iscadm.ksh -c -i -n 2
```

Currently, the iscadm script performs no other major actions. The plan is for it to control the creation of snapshots, deletion of ISC environments, verification that an ISC environment has not been modified, and so on. Such changes would be a welcome addition to the ISC functionality.

## The Future

ISC is an active project with several steps likely in the future. Glenn Brunette, a Sun Distinguished Engineer, is leading the charge on this project and actively working on designing, implementing, and automating the creation of ISCs. Next steps potentially include the following areas:

- Updates to take advantage of new OpenSolaris functionality as it is integrated (such as ZFS encryption, Validated Execution, Always-On Auditing, and other projects).
- New reference configurations that can utilize VirtualBox as the containment model in place of OpenSolaris zones, allowing for the use of alternative guest operating systems beyond OpenSolaris. (For now, the project will likely continue with OpenSolaris as the host OS, due to the security feature set it provides.)
- New operational configurations that implement the autonomic security use cases [6].

The project is also giving consideration to more advanced configuration tools that allow a user to create virtual ISC networks (using Crossbow) [7]. Another area of interest is migration and validation tools to help people move their applications into ISCs and ensure that the security configuration is implemented properly. And on the practical front, there are plans to have ISCs available at some point on Amazon EC2 as an extension of Sun's security-enhanced OpenSolaris AMI efforts.

The project is also actively seeking input from Sun customers, which in turn creates RFEs (requests for enhancement) that get put into the development queue. If security is of interest at your site, downloading and using the current toolset and giving feedback as to what works, what doesn't, and what features you would like to see should be on your to-do list.

For more information on the ISC project, including discussion forums, publications, podcasts, and presentations, visit the project's home at http://kenai.com/projects/isc/pages/Home.

## Conclusion

ISC has lofty goals, and the current non-production implementation meets quite a few of them. Assuming the project does move into production with a complete feature set, ISCs will be a huge leap forward for cloud computing and datacenter application deployment. Easy to use, high-potency security is the nirvana data managers seek but frequently don't find. ISCs could be one of those rare exceptions.

### REFERENCES


[1] http://kenai.com/projects/isc/pages/OpenSolaris.

[2] http://kenai.com/projects/isc/pages/Architecture.

[3] Sun Adaptive Security Architecture Blueprint 820-6825.

[4] http://kenai.com/projects/isc/pages/OpenSolaris#Service_Installation_and_Confi.

[5] http://kenai.com/projects/isc/downloads.

[6] http://kenai.com/projects/isc/pages/Autonomic.

[7] http://kenai.com/projects/isc/pages/Networking.