ORACLE
SOLARIS

An Oracle White Paper
May 2010

How to Eliminate Web Page Hijacking Using Oracle® Solaris 10 Security

ORACLE

Introduction

This white paper instructs Oracle® Solaris system administrators and security professionals in the process of securing common Web servers. By the end of the paper, an example configuration is created that allows Web content to be maintained securely by content owners, while the Web server itself runs with a minimized set of privileges in its own secured container.

Administrators are guided step-by-step through the process and at the end of the paper should be able to perform the following.

- Create a basic Oracle Solaris Container for hosting applications
- Configure the Apache2 Web Server to run in an Oracle Solaris Container
- Use User and Process Rights Management to reduce application privileges
- Use the Oracle Solaris Service Manager to reduce security risk of a container
- Share data securely between two containers

This white paper is not exhaustive and does not cover all optional features of these technologies. However, the reference section provided at the end of the document provides pointers to where administrators can learn more.

Oracle Solaris Security: Overview

The Oracle Solaris 10 operating system contains a number of breakthrough technologies for security enhancements and this paper deals with three of them in particular.

- Oracle Solaris User and Process Rights Management
- Oracle Solaris Service Manager
- Oracle Solaris Containers

These three tools can work together to allow system administrators to secure and consolidate multiple functions or applications together on a system, without the need to change or modify existing application code.

This paper combines existing material for a unique solution to a common problem facing enterprises today — Web page hijacking. Malicious modification or *hijacking* of Web pages typically occurs when a vulnerability in a Web server application is exploited by hackers. Such vulnerabilities often allow the hacker to upload new Web pages, gain super-user shell access to a system or otherwise modify the pages that are being serviced by the Web server process. This paper shows how this issue can be easily solved without the need for costly additional software or specially modified applications.

Figure 1 is a diagram of the example configuration built in the course of this paper using Oracle Solaris 10 security features. It features a simple system with two network interfaces. One interface (`bge1`) is connected to a company's intranet/LAN and the other (`bge0`) is connected to the public Internet through a firewall or other means. The system is running Oracle Solaris 10 and is configured with two containers. One container, the data container, has write access to the HTML files and is connected only to the intranet/LAN. The other container, the Web container, is running the Web server process itself with a reduced set of privileges. The Web container has read-only access to the HTML files served by the data container.

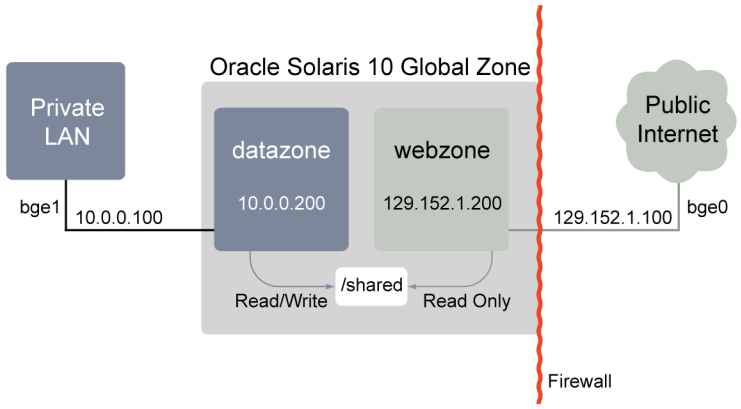

Figure 1— An Example Configuration to Prevent Web Page Hijacking

Architectural flexibility includes the ability to move a workload from one computer to another when necessary. Oracle Solaris 10 containers include this capability. This document describes several scenarios which can be used to move a container from one computer to another.

To create a secure Web server you need to use several newer Oracle Solaris security features that are reviewed in the following sections.

Oracle Solaris User and Process Rights Management

Oracle Solaris User Rights Management and Process Rights Management offer fine-grained privileges in the kernel and user access space of Oracle Solaris. The practical benefit of these technologies is the elimination of the need for applications or users to have unlimited access to the system in order to perform their duties. The kernel itself in Oracle Solaris 10 checks only for Process Rights Management attributes, not `root` or super-user access. This paper utilizes Process Rights Management to run the Apache2 Web Server from a non-super-user account and with just one special privilege (`net_privaddr`) to dramatically reduce or eliminate the risk normally associated with Web servers on Unix systems.

Oracle Solaris Service Manager Profiles

Oracle Solaris Service Manager is a new feature introduced in Oracle Solaris 10 that starts long-running processes (also referred to as services), monitors their status and automatically restarts services as needed. The Service Manager works with the Oracle Solaris Fault Manager to isolate and report hardware and software errors and provide graceful shutdown of services, hardware components and dependant processes. It is part of the Oracle Solaris Predictive Self-Healing functionality and is designed to aid in system administration and diagnosability.

This paper uses the Service Manager's capability to specify run-time attributes with a service, such as the privileges and userid a service runs as, to put constraints on the execution of the Apache2 Web server. This paper also uses the Service Manager's profile capability to limit what network services are running in the Web and data containers.

Oracle Solaris Containers

Oracle Solaris Containers are a new virtualization and security isolation technology in Oracle Solaris 10 that allows customers to securely host multiple applications on the same system. Containers make use of `zones(5)`, `privileges(5)` and resource management technologies to create a secure, isolated, virtual environment. This paper uses Oracle Solaris Containers to create an isolated environment for the Apache2 Web Server to run in and a separate isolated environment from which Web pages are maintained. By doing this, administration of the Web server and maintenance of the Web pages are isolated from each other. Oracle Solaris Containers also allow for audit file entries to be stored *outside* the container in the Global Zone, which prevents attackers from erasing the audit trail should they successfully break into a container.

Build the Secured Web Server Environment

To build a secured system that offers Web services, you need to perform the following steps.

- Create the data container
- Create the Web Server container
- Reduce network exposure for the containers using the Service Manager
- Reduce the privileges associated with the Apache2 service
- Verify the configuration is working

The following four sections describe each of these steps in detail, with examples. For simplicity, assume that all commands are run as the `root` user or another role that has appropriate authorization. Creation of such a role is outside the scope of this paper.

Create the Data Container

The data container in this example has the following characteristics.

- Write access to the *HTML* and *CGI-BIN* directory (located at */shared*)
- Read-only access to the Apache2 log files and *PID* file (located at */shared/logs* and */shared/run*)
- Root directory mounted from */zones/datazone*
- Accessible only from the private intranet/LAN interface (`bge1`)

This paper utilizes a unique capability of Oracle Solaris Containers to share common directories using different mount point names and different write permissions on these mount points. To clarify, Figure 2 shows how the common */shared* directory is mounted in the Web and data containers and what write policy is used.

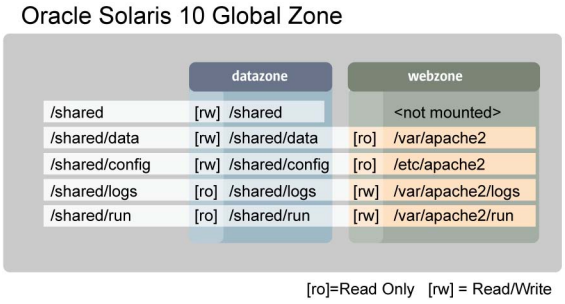

Figure 2— Shared Directory Mount Points

To create a data container with these characteristics, perform the following steps.

1. From the Global zone, create the */shared* documents folder and populate it with Apache2 sample data files and configuration files. Note that this directory tree is mounted by both the data container and the Web container, each using a different set of write permissions.

```
# mkdir /shared
# mkdir /shared/data
# mkdir /shared/config
# mkdir /shared/logs
# mkdir /shared/run
# chown -R webservd:webservd /shared/run
# chown -R webservd:webservd /shared/logs
# mkdir /shared/data/run
# cp -R /etc/apache2/* /shared/config
# cp -R /var/apache2/* /shared/data
# mkdir /zones
```

2. Create the data container; specify its root directory in */zones*.

```
# zonecfg -z datazone
datazone: No such zone configured Use 'create' to begin configuring a new
zone.
zonecfg:datazone> create
zonecfg:datazone> set zonepath=/zones/datazone
zonecfg:datazone> set autoboot=true
```

3. Mount the */shared* directory, which contains the Apache2 configuration data, content to be served, *CGI-BIN* scripts and more, with read-write permissions at the */shared* mount point of the data container.

```
zonecfg:datazone> add fs
zonecfg:datazone:fs> set dir=/shared
zonecfg:datazone:fs> set special=/shared
zonecfg:datazone:fs> set options=[rw,nodevices,noexec,nosuid]
zonecfg:datazone:fs> set type=lofs
zonecfg:datazone:fs> end
```

4. Mount the */shared/run* directory, which contains the Apache2 PID data, with read-only permissions at the */shared/run* mount of the data container.

```
zonecfg:datazone> add fs
zonecfg:datazone:fs> set dir=/shared/run
zonecfg:datazone:fs> set special=/shared/run
zonecfg:datazone:fs> set options=[ro,nodevices,noexec,nosuid]
zonecfg:datazone:fs> set type=lofs
zonecfg:datazone:fs> end
```

5. Mount the */shared/logs* directory, which contains the Apache2 log data, with read-only permissions at the */shared/logs* mount point of the data container. In this way, Web page content owners can analyze, but not remove or modify, Web page log files.

```
zonecfg:datazone> add fs
zonecfg:datazone:fs> set dir=/shared/logs
zonecfg:datazone:fs> set special=/shared/logs
zonecfg:datazone:fs> set options=[ro,nodevices,noexec,nosuid]
zonecfg:datazone:fs> set type=lofs
zonecfg:datazone:fs> end
```

6. Create the virtual network interface for the container on the private/LAN interface.

```
zonecfg:datazone> add net
zonecfg:datazone:net> set address=10.0.0.200
zonecfg:datazone:net> set physical=bge1
zonecfg:datazone:net> end
```

7. Set the containers name in the comment field. Optional—useful if you want to check the configuration parameters later with `zoneadm -z webzone info` from the Global Zone.

```
zonecfg:datazone> add attr
zonecfg:datazone:attr> set name=comment
zonecfg:datazone:attr> set type=string
zonecfg:datazone:attr> set value="Data Container"
zonecfg:datazone:attr> end
```

8. Verify and commit the container.

```
zonecfg:datazone> verify
zonecfg:datazone> commit
zonecfg:datazone> exit
```

9. Install, boot and verify the container.

```
# zoneadm -z datazone install
Preparing to install zone <datazone>.
Creating list of files to copy from the global zone.
[Some output was omitted here for brevity]
# zoneadm -z datazone boot
# zoneadm list -c -v
ID NAME               STATUS         PATH
0 global              running        /
1 datazone            running        /zones/datazone
```

10. Login on the console device of the container and configure its root password information, hostname and name service data.

```
# zlogin -C datazone
SunOS Release 5.10 Version Generic_118822-22 64-bit
Copyright 1983-2005 Sun Microsystems, Inc. All rights reserved.
Use is subject to license terms.
Hostname: datazone
[Some output was omitted here for brevity]
datazone console login:
(Press ~. to disconnect from the console session)
(logout)
```

Create the Web Server Container

Creating a container for the Web server is almost identical to the process for creating the data container. The only differences are as follows.

- The container is bound to the external interface (`bge0`)
- The name of the container is *webzone*
- The Web server configuration data is mounted read-only
- The Web server is allowed to write to its log files via the */shared/logs* directory
- The Web data files are mounted read-only as well

To create the Web container, perform the following steps.

1. Create a zone; specify its root directory.

```
# zonecfg -z webzone
webzone: No such zone configured Use 'create' to begin configuring a new
zone.
zonecfg:webzone> create
zonecfg:webzone> set zonepath=/zones/webzone
zonecfg:webzone> set autoboot=true
```

2. Mount the */shared/config* directory, which contains the Apache2 configuration data, with read-only permissions at the */etc/apache2* mount point of the Web container.

```
zonecfg:webzone> add fs
zonecfg:webzone:fs> set dir=/etc/apache2
zonecfg:webzone:fs> set special=/shared/config
zonecfg:webzone:fs> set options=[ro,nodevices,nosuid,noexec]
zonecfg:webzone:fs> set type=lofs
zonecfg:webzone:fs> end
```

3. Mount the */shared/data* directory, which contains the Apache2 HTML files, CGI-BIN scripts and other data to be served, with read-only permissions at the */var/apache2* mount point of the Web container. Please note that this mount point is at the */var* directory and not at the */etc* directory as in the previous step.

```
zonecfg:webzone> add fs
zonecfg:webzone:fs> set dir=/var/apache2
zonecfg:webzone:fs> set special=/shared/data
zonecfg:webzone:fs> set options=[ro,nodevices,nosuid,noexec]
zonecfg:webzone:fs> set type=lofs
zonecfg:webzone:fs> end
```

4. Mount the */shared/logs* directory, which contains the Apache2 log and PID data, with read-write permissions at the */var/apache2/logs* mount point of the Web container.

```
zonecfg:webzone> add fs
zonecfg:webzone:fs> set dir=/var/apache2/logs
zonecfg:webzone:fs> set special=/shared/logs
zonecfg:webzone:fs> set options=[rw,nodevices,nosuid,noexec]
zonecfg:webzone:fs> set type=lofs
```

```
zonecfg:webzone:fs> end
```

5. Mount the */shared/run* directory, which contains the Apache2 PID data, with read-write permissions at the */var/apache2/run* mount point of the Web container.

```
zonecfg:webzone> add fs
zonecfg:webzone:fs> set dir=/var/apache2/run
zonecfg:webzone:fs> set special=/shared/run
zonecfg:webzone:fs> set options=[rw,nodevices,nosuid,noexec]
zonecfg:webzone:fs> set type=lofs
zonecfg:webzone:fs> end
```

6. Create the virtual network interface for the container on the public interface.

```
zonecfg:webzone> add net
zonecfg:webzone:net> set address=129.152.1.200
zonecfg:webzone:net> set physical=bge0
zonecfg:webzone:net> end
```

7. Set the containers name in the comment field. [Optional—useful if you want to check the configuration parameters later with `zoneadm -z webzone info` from the Global Zone]

```
zonecfg:webzone> add attr
zonecfg:webzone:attr> set name=comment
zonecfg:webzone:attr> set type=string
zonecfg:webzone:attr> set value="Web Container"
zonecfg:webzone:attr> end
```

8. Verify and commit the container.

```
zonecfg:webzone> verify
zonecfg:webzone> commit
zonecfg:webzone> exit
```

9. Install, boot and verify the container.

```
# zoneadm -z webzone install
Preparing to install zone <webzone>.
Creating list of files to copy from the global zone.
```

```
[Some output was omitted here for brevity]
# zoneadm -z webzone boot
# zoneadm list -c -v
ID NAME               STATUS          PATH
0 global              running         /
1 datazone            running         /zones/datazone
2 webzone             running         /zones/webzone
```

10. Log in on the console device of the container and configure its root password information, hostname and name service data.

```
# zlogin -C webzone
SunOS Release 5.10 Version Generic_118822-22 64-bit
Copyright 1983-2005 Sun Microsystems, Inc. All rights reserved.
Use is subject to license terms.
Hostname: webzone
[Some output was omitted here for brevity]
webzone console login:
(Press ~. to disconnect from the consoles session)
(logout)
```

Reduce Network Exposure

To reduce the network services exposed to possible attack, use the profile capability of the Oracle Solaris Service Manager. In this example you change the default services profile by loading the *Generic Limited Networking* profile. This profile minimizes the set of network services in each container. However, the *Generic Limited Networking* profile is not the only method you can use to secure your system. There may be additional services that you wish to stop or disable. You may use the Services Manager `svcadm(1M)` command to disable services on a per-container basis, or you may utilize the Oracle Solaris Security Toolkit.

Use of the Oracle Solaris Security Toolkit is not covered in this paper, please see the reference material.

1. Enable the *Generic Limited Networking* profile for the Web container to be the default profile and apply it to the running copy of the container.

```
# zlogin webzone
[Some output was omitted here for brevity]
webzone console login: root
Password: <enter password here>
# cd /var/svc/profile
```

```
# rm generic.xml
# ln -s generic_limited_net.xml generic.xml
# svccfg apply /var/svc/profile/generic_limited_net.xml
# exit
```

2. Apply the same changes to the data container.

```
# zlogin datazone
[Some output was omitted here for brevity]
datazone console login: root
Password: <enter password here>
# cd /var/svc/profile
# rm generic.xml
# ln -s generic_limited_net.xml generic.xml
# svccfg apply /var/svc/profile/generic_limited_net.xml
# exit
```

Now that the default services profile has been changed, each of the containers runs with a reduced set of network services. Each container can have its own unique services profile, so system administrators can selectively enable services such as FTP for the data container while allowing only `ssh(1)` access for the Web container.

Reduce Privileges of the Apache2 Service

Next, modify the Apache2 configuration file to use the new directories. This allows the Apache2 server to have write access for logging and PID information. Also, use the Service Manager to modify the privileges that the Apache2 Web Server receives. Here is an outline of the steps required.

- Run the Apache2 Web Server as userid `webservd` rather than `root`
- Grant just the three needed privileges to the Apache2 Web Server: `proc_exec`, `proc_fork` and `net_privaddr`
- Change ownerships of the needed log files for Apache2
- Change the Apache2 configuration file to utilize the new Web container directories
- Verify the Apache2 Server is running with fewer privileges

It's worth noting that these extra steps are taken to further harden and reduce the risk of intrusion with the Apache2 Web Service only within the Web container. The services used in the data container already run with reduced privileges set as their default behavior in Oracle Solaris 10.

1. From the Global Zone, change the Apache2 *httpd.conf* file to use the new directories by editing two lines. Remember that the rest of the *httpd.conf* file settings for directories are generally fine as the Web container mounts */shared/config* as */etc/apache2* and */shared/data* as */var/apache2*, which are the default directories that Apache2 looks for.

```
# vi /shared/config/httpd.conf
[Some output was omitted here for brevity]
LockFile /var/apache2/logs/accept.lock
[Some output was omitted here for brevity]
PidFile /var/apache2/run/httpd.pid
[Some output was omitted here for brevity]
```

2. Log in to the Web container to begin the modifications.

```
# zlogin webzone
[Some output was omitted here for brevity]
webzone console login: root
webzone console login: <password here>
webzone >
```

3. Modify the userid and group that the Apache2 service uses.

```
# svccfg -s apache2
svc:/network/http:apache2> setprop start/user = astring: webservd
svc:/network/http:apache2> setprop start/group = astring: webservd
```

4. Reduce the privileges the Apache2 service uses to just those necessary, complete the definition of the service and refresh the service.

```
svc:/network/http:apache2> setprop start/privileges = astring:
basic,!proc_session,!proc_info,!file_link_any,net_privaddr
svc:/network/http:apache2> setprop start/limit_privileges = astring: :default
svc:/network/http:apache2> setprop start/use_profile = boolean: false
svc:/network/http:apache2> setprop start/supp_groups = astring: :default
svc:/network/http:apache2> setprop start/working_directory = astring:
:default
svc:/network/http:apache2> setprop start/project = astring: :default
svc:/network/http:apache2> setprop start/resource_pool = astring: :default
svc:/network/http:apache2> end
# svcadm -v refresh apache2
```

5. Verify that the Apache2 service is running with fewer privileges by restarting it, verifying that there are no `root` processes and using the `ppriv(1)` command to verify the reduced set of privileges used by the service. Note the output of the `ppriv` command shows the Apache2 Web Server running with just three privileges.

```
# svcadm -v enable -s apache2
svc:/network/http:apache2 enabled.
# svcs apache2
STATE STIME FMRI
online 12:02:21 svc:/network/http:apache2
# ps -aef | grep httpd | grep -v grep
webservd 5568 5559 0 12:02:22 ? 0:00 /usr/apache2/bin/httpd -k start
webservd 5567 5559 0 12:02:22 ? 0:00 /usr/apache2/bin/httpd -k start
webservd 5561 5559 0 12:02:22 ? 0:00 /usr/apache2/bin/httpd -k start
webservd 5562 5559 0 12:02:22 ? 0:00 /usr/apache2/bin/httpd -k start
webservd 5563 5559 0 12:02:22 ? 0:00 /usr/apache2/bin/httpd -k start
webservd 5559 23382 0 12:02:21 ? 0:00 /usr/apache2/bin/httpd -k start
# ppriv -S 5559 #This is the starting process
5559: /usr/apache2/bin/httpd -k start
flags = <none>
E: net_privaddr,proc_exec,proc_fork
I: net_privaddr,proc_exec,proc_fork
P: net_privaddr,proc_exec,proc_fork
L: zone
```

Verify the Configuration

At this point, the Apache2 Web Server is running inside of its own Web container, with reduced exposure on the network and with reduced privileges. It is also serving HTML files to which it has read only access. If the Web server is attacked or compromised, the HTML data files to which it is providing access cannot be damaged because of the security constraints placed by Process Rights Management and Oracle Solaris Containers.

To verify configuration, connect to the Web server's IP address from your desktop session with a Web browser. You should see the Apache2 Documentation page.

For command-line verification, you can also use the `telnet 129.152.1.200 80` command to connect to the Web server port and enter `HEAD /HTTP/1.0`, which returns the default Apache2 Web page.

For further verification, connect from a system on the private/LAN network and modify an HTML page. Notice that your Web server has immediate access to that modified Web page. Remember that Web page authors modify the content in the */shared/data* directory while logged into the data container.

The Web server container sees these changes automatically because it mounts the exact same directory as */var/apache2*.

Additional Enhancements

As with any complex system, there are a variety of areas for enhancement in a sample configuration such as this. Additional areas include the following topics.

- Configuring users and roles for both data management and for eliminating the need for using the `root` role
- Using Resource Management to dynamically control the CPU, memory and network resources assigned to containers
- Securing the Global Zone or container to treat it more as a privileged console
- Adding Secure Sockets Layer to the Apache2 service for encrypted communications
- Performance tuning of the Apache2 Web service
- Configuring the IP Filter firewall for the Global zone to reduce network exposure
- Utilizing the BART file integrity checking tool to monitor for additional unwanted data or system file modifications
- Utilizing the Oracle Solaris Security Toolkit to reduce network exposure and risk

See the *For More Information* section for details on how to implement these enhancements.

Conclusion

This paper explored combining various technologies to address the common issue of Web server security and Web page defacement. Because of the advances in Oracle Solaris 10 security, system administrators have new possibilities open to them to solve problems that previously would have taken many more systems, complex add-on products, changes in networking topology or other such compromises. Explore additional Sun documentation and articles for more ideas on how to use Oracle Solaris 10 to creatively solve your business and security issues.

For More Information

TABLE 1. WEB RESOURCES

WHITE PAPERS	
Consolidating Servers and Applications with Solaris Containers, Joost Pronk van Hoogeveen, Oracle White Paper, May 2010	http://wikis.sun.com/display/BluePrints/Main
Limiting Service Privileges in the Solaris 10 Operating System, Glenn Brunette	www.sun.com/blueprints/0505/819-2680.pdf
Web Consolidation on the SunFire T1000 Server Using Solaris Containers, Kevin Kelly	www.sun.com/blueprints/1205/819-5149.pdf
Enforcing the Two-Person Rule Via Role Based Access Control in Solaris 10, Glenn Brunette	www.sun.com/blueprints/0805/819-3164.pdf
Integrating BART and the Fingerprint Database in the Solaris 10 Operating System, Glenn Brunette	www.sun.com/blueprints/0405/819-2260.pdf
Solaris 10 Operating System: Unparalleled Security	www.sun.com/software/whitepapers/solaris10/s10security.pdf
Solaris Security Toolkit v4.2	www.sun.com/download/products.xml?id=42e6becd
MANUALS	
Oracle Solaris Manuals	docs.sun.com/
System Administration Guide: Solaris Containers Resource Management and Solaris Zones	docs.sun.com/app/docs/doc/817-1592
System Administration Guide: Security Services	docs.sun.com/app/docs/doc/816-4557
Solaris Administration Guide: Basic Administration	docs.sun.com/app/docs/doc/817-1985
SUN EMPLOYEE LOGS	
Glenn Brunette	blogs.sun.com/gbrunett
Alec Muffat	www.crypticide.com/dropsafe

Casper Dik	blogs.sun.com/casper
Wyllys Ingersoll	blogs.sun.com/wyllys
EDUCATION RESOURCES	
Solaris OS Security Administration: Course Overview	www.sun.com/training/catalog/operating_systems/security_admin.html
Solaris 10—Ten Moves Ahead of the Competition: Course Overview	www.sun.com/training/catalog/courses/WS-245.xml

Oracle is committed to developing practices and products that help protect the environment

How to Eliminate Web Page Hijacking Using Oracle Solaris 10 Security
May 2010
Author: Mark Thacker

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

SOFTWARE. HARDWARE. COMPLETE.