RH354 - Security Extras

Travis Michette

Version 1.0

Table of Contents

Session Recording

1. What is Session Recording

Session recording provides the ability to record and play back user terminal sessions. Session recording depends on the **tlog** package and can be configured to capture sessions on a per user or a per group requirement based on configurations made to the SSSD service.

Session recording is used for auditing user sessions on systems that are sensitive to the level of security or they can be used in the event of a system breach or hack. Reviewing the recorded sessions would be part of the forensic system analysis.

Recorded sessions can be viewed from the terminal or the web console (Cockpit) using **tlog-play**

Session Recording Limitations

There are several limitations to session recording. The most notable sessions are listed below.

- **tlog** only records shell sessions. Gnome 3 graphical sessions are not recorded or supported.
- If **tlog** is configured to log to ***journal/syslog*** directory, recording/viewing the results can generate circular logging and flood the output.

Listing 1. Session Recording - Loop Work-around

```
# journalctl -f | grep -v 'tlog-rec-session'
```

1.1. Required Packages for Session Recording

There are a few required packages for session recording. Some of these packages are already installed on most systems, but it can be a good idea to install them yourself to ensure all packages needed by session recording have been installed and configured.

Required Packages

- tlog
- SSSD
- cockput-session-recording
- *systemd-journal-remote - (Not specifically required but good to have - enables exporting of recorded sessions)*

2. Implementing Session Recording

Installing and Configuring Session Recording

1. Install required packages

Listing 2. Installing Required Pacakges

```
[root@serverb ~]# yum install -y tlog cockpit-session-recording systemd-journal-remote
```

2. Start and Enable Cockpit Services

Listing 3. Starting and Enabling Cockpit

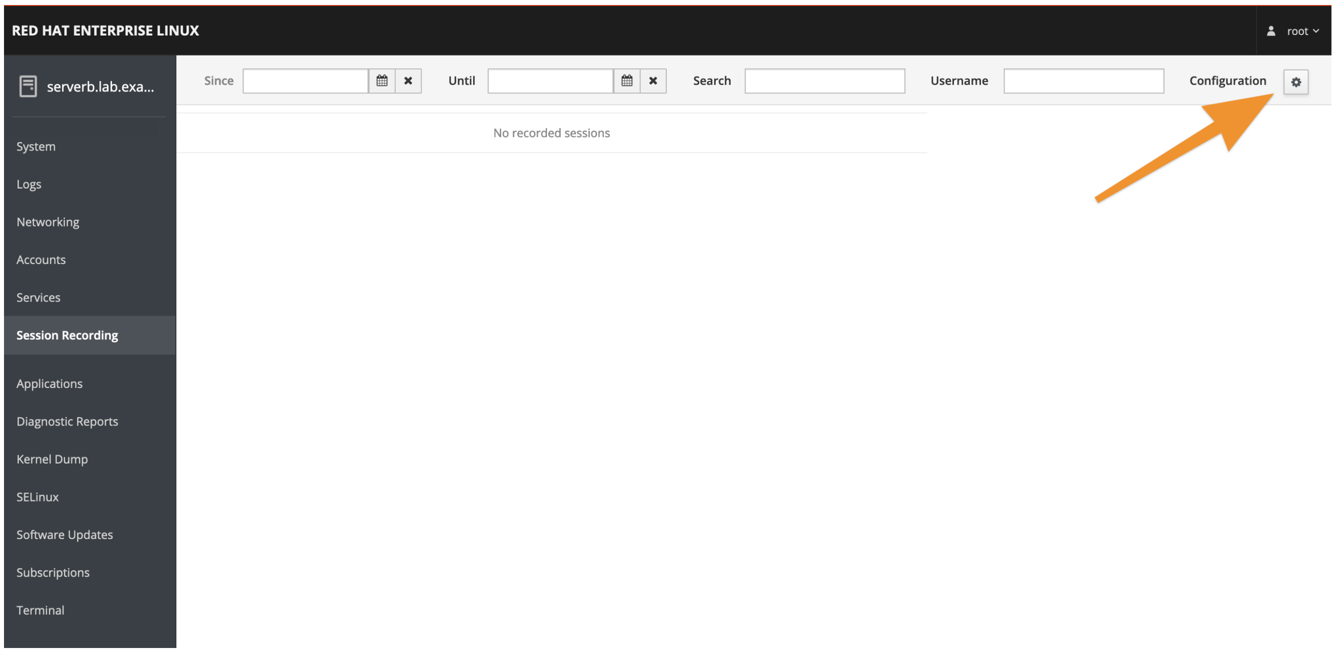
```
[root@serverb ~]# systemctl enable cockpit.socket --now
Created symlink /etc/systemd/system/sockets.target.wants/cockpit.socket → /usr/lib/systemd/system/cockpit.socke
```

3. Configure **SSSD** for User/Group Session Recording

Listing 4. Creating a SSSD Configuration File

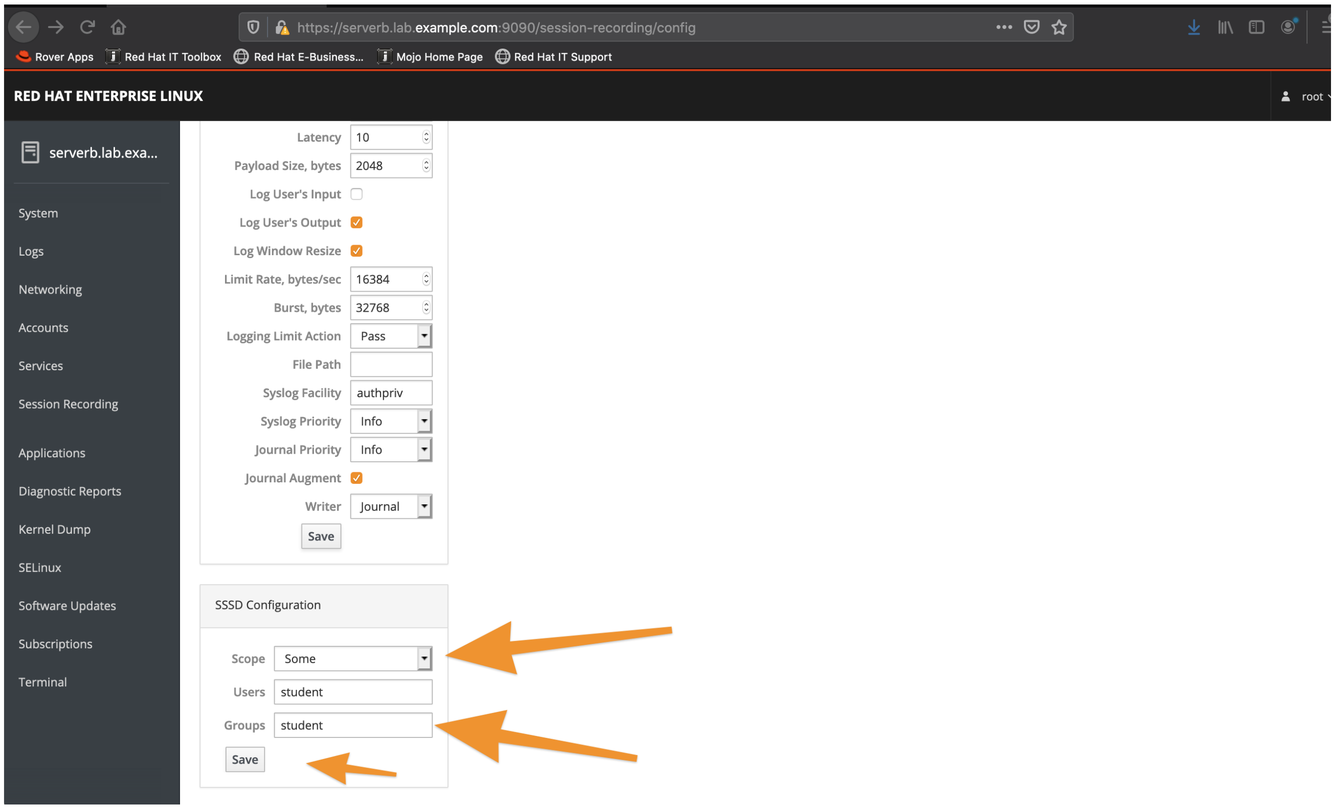
```
[root@serverb ~]# vim /etc/sssd/conf.d/sssd-session-recording.conf

[session_recording]
scope = some
users = student
groups = student
```



No recorded sessions

Figure 1. Web Console Session Recording Settings



Latency	10
Payload Size, bytes	2048
Log User's Input	☐
Log User's Output	✓
Log Window Resize	✓
Limit Rate, bytes/sec	16384
Burst, bytes	32768
Logging Limit Action	Pass
File Path	
Syslog Facility	authpriv
Syslog Priority	Info
Journal Priority	Info
Journal Augment	✓
Writer	Journal

Scope	Some
Users	student
Groups	student

Figure 2. Session Recording Configuration

Session Recording SSSD Scope Settings

There are three settings for Session Recording

- **all** - Record all sessions
- **none** - Record no sessions
- **some** - Record sessions of specified users/groups

SSSD is preferred method of Session Recording Configuration

It is important to note while it is possible to manually configure session recording the preferred method it to perform the configuration via the CLI or the RHEL 8 web console.

Session Recording Warning

It should be noted, once Session Recording has been activated, a message will be displayed notifying the user that the session is being recorded.

Listing 5. Student Login of Recorded Session

```
[student@workstation ~]$ ssh student@serverb
Warning: Permanently added 'serverb,172.25.250.11' (ECDSA) to the list of known hosts.
Web console: https://serverb.lab.example.com:9090/ or https://172.25.250.11:9090/

Last login: Sat May 16 07:41:00 2020

ATTENTION! Your session is being recorded!
```

Session Recording Blog - Brian Smith

Getting Started with Session Recording: https://www.redhat.com/en/blog/getting-started-session-recording-red-hat-enterprise-linux-8-beta

Configuring Terminal Session Recording - Red Hat GovIO Workshop

Configuring Terminal Session Recording: https://www.redhat.com/en/blog/getting-started-session-recording-red-hat-enterprise-linux-8-beta

Session Recording - RHEL 8

Recording Sessions: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/recording_sessions/index

3. Exporting Recorded Sessions

As noted above, in order to export recorded sessions, it is necessary to have the **systemd-journal-remote** package installed on the system.

1. Install **systemd-journal-remote** package if not already installed.
2. Create a directory for the exported session

Listing 6. Directory Creation

```
[root@serverb ~]# mkdir /tmp/sessions
```

3. Export the session to the newly created directory

Listing 7. Use **journalctl** *to export the journal*

```
[root@serverb ~]# journalctl -o export | /usr/lib/systemd/systemd-journal-remote -o /tmp/sessions/student.journal -
Finishing after writing 13914 entries
```

Managing Exported Sessions

It is possible to copy the exported file to **/var/log/journal** or you can create a new directory **/var/log/journal/remote** for exported files from multiple remote hosts.

4. Replaying Recorded Sessions

Recorded sessions can be played back either from a file or from the **systemd** journal. The ***tlog-play*** tool loads parameters from **/etc/tlog/tlog-play.conf** and is able to playback terminal input/output recorded using the **tlog-rec** tool.

Playback can take place both from the RHEL 8 Web Console or the regular CLI using the **tlog-play** command.

4.1. Replaying from the CLI

Playback can take place both from a file or from the **systemd** journal.

Listing 8. **tlog-play** *- Playing back from a File*

```
tlog-play --reader=file --file-path=tlog.log
```

Listing 9. **tlog-play** *- Playing back from a SystemD Journal File*

```
tlog-play -r journal -M TLOG-REC=<your-unique-host-id>
```

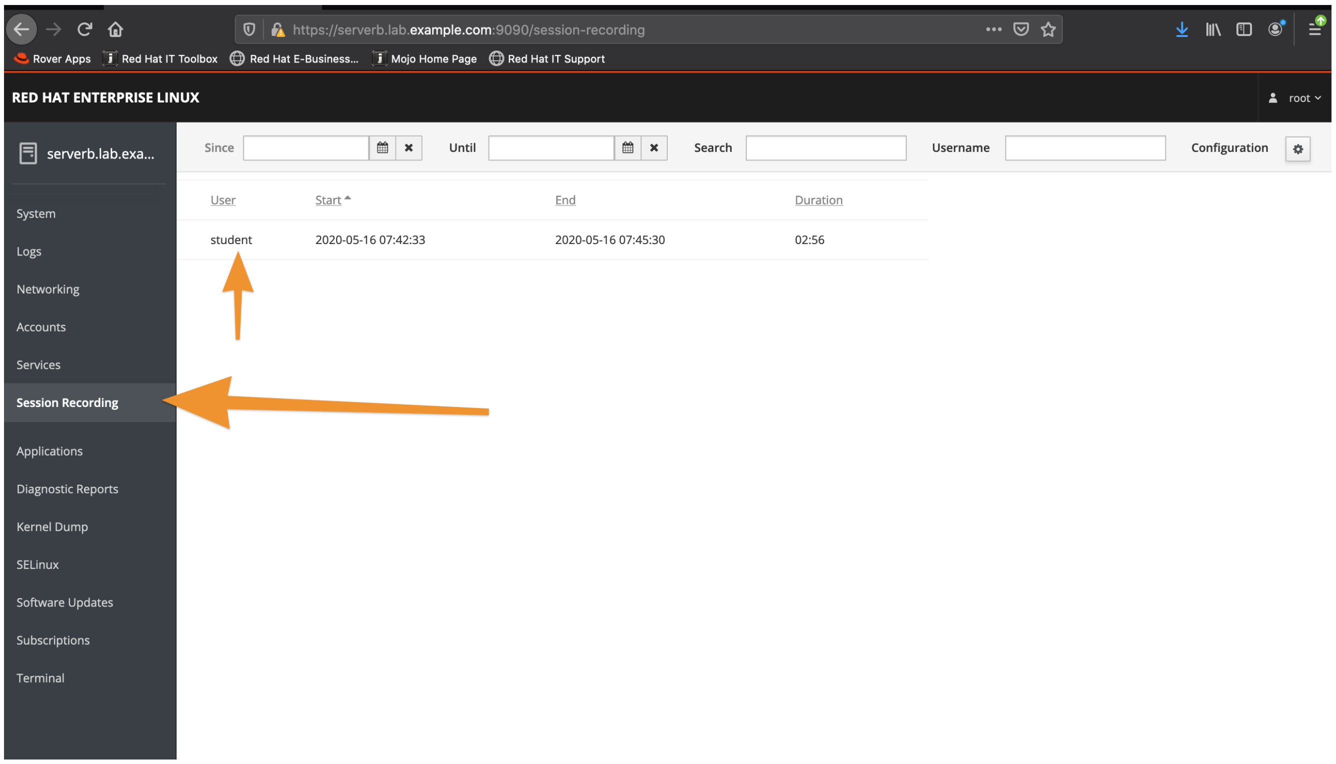
Replaying from the Journal

In practice however, playback from Journal is usually done with a single match against the TLOG_REC Journal field. The TLOG_REC field contains a copy of the rec field from the logged JSON data, which is a host-unique ID of the recording.

4.2. Replaying from the Web Console

It is possible to replay from the RHEL 8 Web Console.

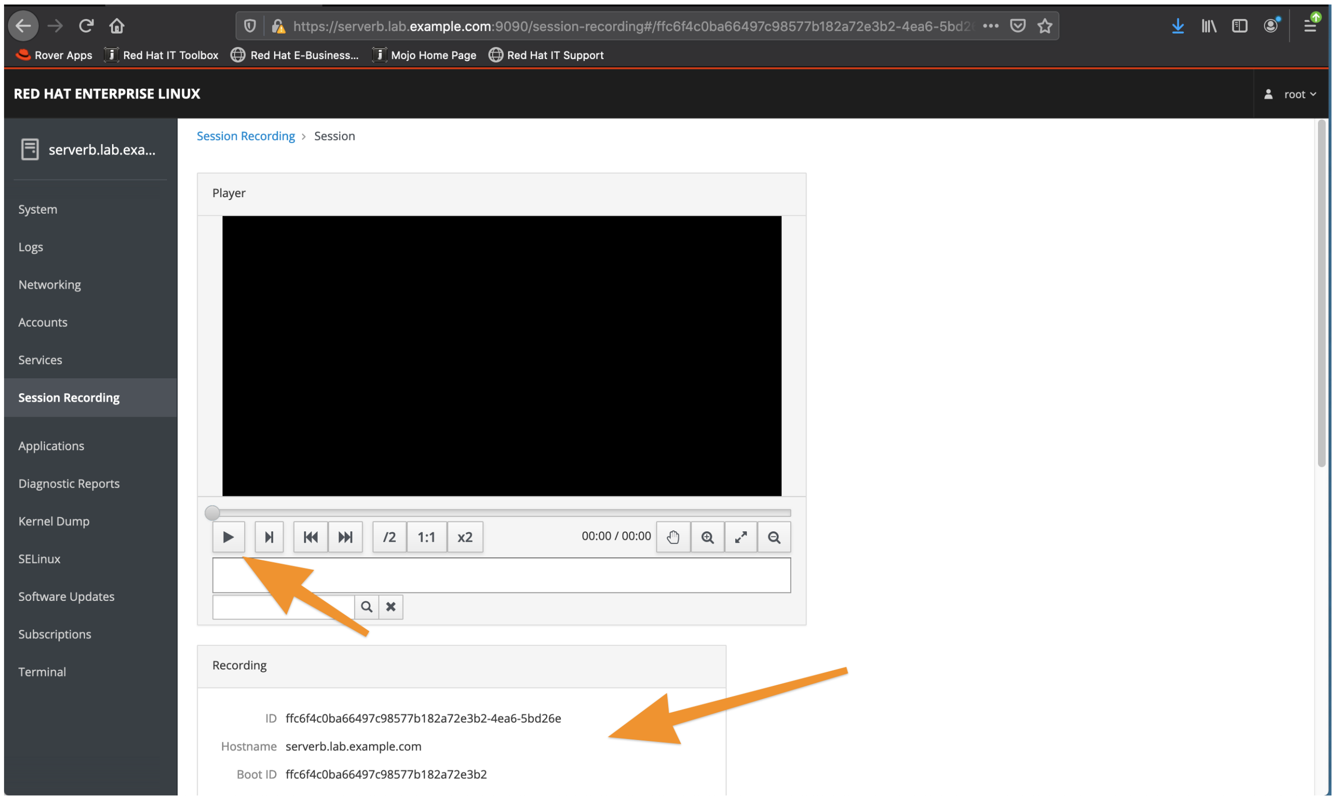
1. Click on Session Recording and Select the Session



User	Start	End	Duration
student	2020-05-16 07:42:33	2020-05-16 07:45:30	02:56

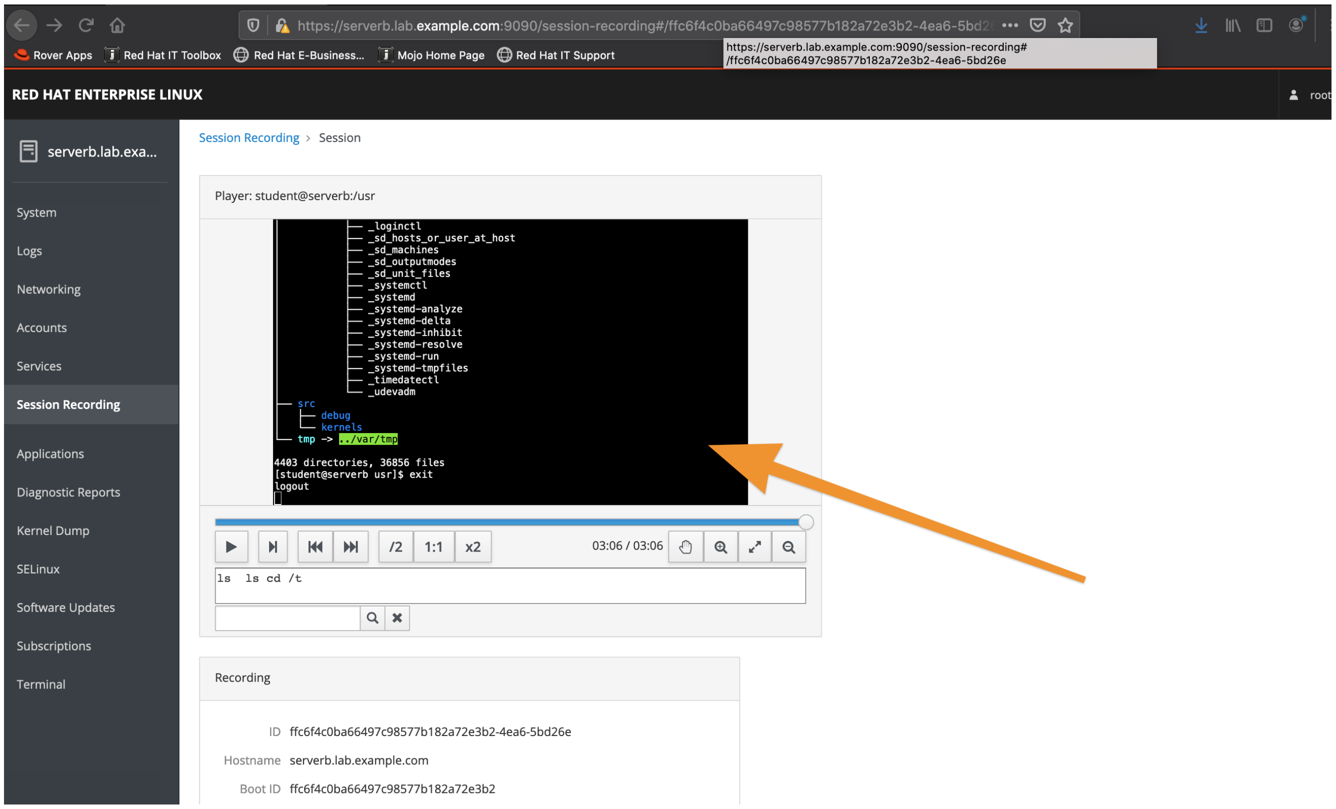
Figure 3. Session Recordings

2. Click the "Play" button to watch the session



ID ffc6f4c0ba66497c98577b182a72e3b2-4ea6-5bd26e
Hostname serverb.lab.example.com
Boot ID ffc6f4c0ba66497c98577b182a72e3b2

Figure 4. Session Recording - Reviewing



```
│           ├── _loginctl
│           ├── _sd_hosts_or_user_at_host
│           ├── _sd_machines
│           ├── _sd_outputmodes
│           ├── _sd_unit_files
│           ├── _systemctl
│           ├── _systemd
│           ├── _systemd-analyze
│           ├── _systemd-delta
│           ├── _systemd-inhibit
│           ├── _systemd-resolve
│           ├── _systemd-run
│           ├── _systemd-tmpfiles
│           ├── _timedatectl
│           └── _udevadm
├── src
│   ├── debug
│   └── kernels
└── tmp -> ../var/tmp

4403 directories, 36856 files
[student@serverb usr]$ exit
logout
```

```
ls  ls cd /t
```

ID	ffc6f4c0ba66497c98577b182a72e3b2-4ea6-5bd26e
Hostname	serverb.lab.example.com
Boot ID	ffc6f4c0ba66497c98577b182a72e3b2

Figure 5. Session Recording - Reviewing 2

Cryptographic Policies

5. Managing Cryptographic Policies

Along with the introduction of RHEL 8 as a new operating system, Red Hat introduced system-wide crypto policies. This ensures that cryptographic policies are applied consistently to running services and also that they are kept up-to-date as part of software and system updates and upgrades.

The default crypto policy is conservative and disables many of the legacy protocols like TLS 1.1 and earlier. It is possible to select and create either stricter policies for more stringent security or configure a more lax policy to support compatibility with older systems and applications.

RHEL 8 Provided Crypto Policies

- Legacy
- Default
- Future
- FIPS

RHEL 8 Crypto Policies

System-wide Crypto Policies in RHEL 8: https://access.redhat.com/articles/3666211

Strong crypto defaults in RHEL 8 and deprecation of weak crypto algorithms: https://access.redhat.com/articles/3642912

Cryptographic Policy Management

Managing Cryptographic Policies: http://redhatgov.io/workshops/rhel_8/exercise1.5/

It is possible to change the Crypto Policies and customize the policies for your system and environment. The easiest thing is to switch through the existing crypto policies as they are fully supported and defined by Red Hat and the Red Hat Security Team.

Crypto Policies and Failed Connections

It should be noted if attempting to connect to older, legacy applications, it might be necessary to change the crypto policy to Legacy in order to establish connections using older insecure protocols.

Viewing and Changing the Crypto Policies

1. Show the current crypto policy

   Listing 10. Displaying the Current Crypto Policy

   ```
   [root@serverb ~]# update-crypto-policies --show
   DEFAULT
   ```

2. Using and Verifying the Crypto Policy

Listing 11. Using OpenSSL Client to Connect and Verify Crypto Policy

```
[root@serverb ~]# openssl s_client --connect tls-v1-1.badssl.com:1011
CONNECTED(00000004)
139741458040640:error:1425F102:SSL routines:ssl_choose_client_version:unsupported protocol:ssl/statem/statem_lib.c:1907:
---
no peer certificate available
---
No client certificate CA names sent
---
SSL handshake has read 70 bytes and written 334 bytes
Verification: OK
---
New, (NONE), Cipher is (NONE)
Secure Renegotiation IS NOT supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
Early data was not sent
Verify return code: 0 (ok)
---


[root@serverb ~]# timeout 3 openssl s_client --connect tls-v1-1.badssl.com:1011 | grep Protocol
140056880396096:error:1425F102:SSL routines:ssl_choose_client_version:unsupported protocol:ssl/statem/statem_lib.c:1907:
```

3. Switch to Legacy Policy to Allow TLS 1.1 Protocol

Listing 12. Changing Security Policy

```
[root@serverb ~]# sudo update-crypto-policies --set LEGACY
Setting system policy to LEGACY
Note: System-wide crypto policies are applied on application start-up.
It is recommended to restart the system for the change of policies
to fully take place
```

4. Testing with TLS and Legacy Crypto Policy

Listing 13. Testing with Legacy Crypto Policy

```
[root@serverb ~]# timeout 3 openssl s_client --connect tls-v1-1.badssl.com:1011 | grep Protocol
depth=2 C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert Global Root CA
verify return:1
depth=1 C = US, O = DigiCert Inc, CN = DigiCert SHA2 Secure Server CA
verify return:1
depth=0 C = US, ST = California, L = Walnut Creek, O = Lucas Garron Torres, CN = *.badssl.com
verify return:1
    Protocol  : TLSv1.1
```

Crypto-Policies Man Page

Listing 14. RHEL 8.2 Crypto Policies

```
[root@rhel82-demo ~]# man crypto-policies
CRYPTO-POLICIES(7)                                          CRYPTO-POLICIES(7)

NAME
       crypto-policies - system-wide crypto policies overview

DESCRIPTION
       The security of cryptographic components of the operating system does not remain
       constant over time. Algorithms, such as cryptographic hashing and encryption,
       typically have a lifetime, after which they are considered either too risky to
       use or plain insecure. That means, we need to phase out such algorithms from the
       default settings or completely disable them if they could cause an irreparable
       problem.

       While in the past the algorithms were not disabled in a consistent way and
       different applications applied different policies, the system-wide
       crypto-policies followed by the crypto core components allow consistently
       deprecating and disabling algorithms system-wide.

       The individual policy levels (DEFAULT, LEGACY, FUTURE, and FIPS) are included in
       the crypto-policies(7) package. In the future, there will be also a mechanism for
       easy creation and deployment of policies defined by the system administrator or a
       third party vendor.


       ... output omitted ...


       /etc/crypto-policies/config
           The active crypto-policies level set on the system.

       /etc/crypto-policies/local.d
           Additional configuration shipped by other packages or created by the system
           administrator. The contents of the <back-end>-file.config is appended to the
           configuration from the policy back end as shipped in the crypto-policies
           package.

SEE ALSO
       update-crypto-policies(8), fips-mode-setup(8)

AUTHOR
       Written by Tomáš Mráz.

crypto-policies                 12/16/2019                  CRYPTO-POLICIES(7)
```

SystemD Overview

6. Understanding SystemD Unit Files and Creating a Service

Starting in RHEL 7, **SystemD** replaced the older style Linux SystemV (sysvinit daemon). This change continued through with RHEL 8 and more systemd tools replaced the traditional legacy tools. This small section will provide references and an overview of how **systemd** can be used to replace the older *init* scripts that were placed in ***/etc/init.d/*** or some of the other directories.

SystemD References

Linus Torvalds and others on Linux's systemd: https://www.zdnet.com/article/linus-torvalds-and-others-on-linuxs-systemd/

SysVinit Vs systemd Cheatsheet: https://www.2daygeek.com/sysvinit-vs-systemd-cheatsheet-systemctl-command-usage/

1. Create the Init Script

Listing 15. Init Script to Run at Startup

```
[root@servera ~]# vim /usr/bin/ups_test_service.sh
#!/usr/bin/bash

DATE=`date '+%Y-%m-%d %H:%M:%S'`
echo "This is a sample service started at ${DATE} for the UPS RH354 course." | systemd-cat -p info

while :
do
echo "Looping...";
sleep 30;
done
```

2. Make script executable

*Listing 16. Running **chmod** on script*

```
[root@servera ~]# chmod +x /usr/bin/ups_test_service.sh
```

3. Edit SystemD Service (Unit File)

Listing 17. Editing the .service File

```
[root@servera ~]# vim /etc/systemd/system/ups_test_service.service

[Unit]
Description=UPS example systemd service.

[Service]
Type=simple
ExecStart=/bin/bash /usr/bin/ups_test_service.sh

[Install]
WantedBy=multi-user.target
```

4. Fix Permissions on **.service** File

Listing 18. Running **chmod** *on .service File*

```
[root@servera ~]# chmod 644 /etc/systemd/system/ups_test_service.service
```

SystemD Services

Once a script has been created and made executable and a service file has properly been created and placed in **/etc/systemd/system/** directory it is possible to use the **systemctl** command to interact with the service file and make it active on boot.

Default SystemD Directories

It is important to note that there are several default SystemD directories. When defining your own files, the proper location is to place them in **/etc/systemd**. There is more information available in other Red Hat courses, specifically the RH442 Performance Tuning course.

Default Location: **/lib/systemd/**

1. Starting and Enabling a Custom Service

Listing 19. Controlling a Custom Service with **systemctl**

```
[root@servera ~]# systemctl enable ups_test_service.service --now
Created symlink /etc/systemd/system/multi-user.target.wants/ups_test_service.service → /etc/systemd/system/ups_test_service.service.
```

2. Checking Status of a Custom Service

Listing 20. Using **systemctl** *to Check Service Status*

```
[root@servera ~]# systemctl status ups_test_service.service
● ups_test_service.service - UPS example systemd service.
   Loaded: loaded (/etc/systemd/system/ups_test_service.service; enab>
   Active: active (running) since Wed 2020-05-20 18:08:22 EDT; 19s ago
 Main PID: 2136 (bash)
    Tasks: 2 (limit: 23896)
   Memory: 940.0K
   CGroup: /system.slice/ups_test_service.service
           ├─2136 /bin/bash /usr/bin/ups_test_service.sh
           └─2140 sleep 30

May 20 18:08:22 servera.lab.example.com systemd[1]: Started UPS examp>
May 20 18:08:22 servera.lab.example.com bash[2136]: Looping..
```

3. Checking **/var/log/messages** for Custom Service

Listing 21. Seaching Log file for Custom Service

```
[root@servera ~]# grep -i ups /var/log/messages
May 20 18:08:22 jegui journal[2139]: This is a sample service started at 2020-05-20 18:08:22 for the UPS RH354 course.
```

SystemD References

Creating a Service at Boot: https://www.linode.com/docs/quick-answers/linux/start-service-at-boot/

Overview of SystemD for RHEL7: https://access.redhat.com/articles/754933

Converting traditional sysV init scripts to Red Hat Enterprise Linux 7 systemd unit files: https://www.redhat.com/en/blog/converting-traditional-sysv-init-scripts-red-hat-enterprise-linux-7-systemd-unit-files

Creating and Modifying SystemD Unit Files: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system_administrators_guide/sect-managing_services_with_systemd-unit_files

Creating a Linux service with systemd: https://medium.com/@benmorel/creating-a-linux-service-with-systemd-611b5c8b91d6

How to create systemd service unit in Linux: https://linuxconfig.org/how-to-create-systemd-service-unit-in-linux