SWGDE Linux Tech Notes

Disclaimer:
As a condition to the use of this document and the information contained therein, the SWGDE requests notification by e-mail before or contemporaneous to the introduction of this document, or any portion thereof, as a marked exhibit offered for or moved into evidence in any judicial, administrative, legislative or adjudicatory hearing or other proceeding (including discovery proceedings) in the United States or any Foreign country. Such notification shall include: 1) The formal name of the proceeding, including docket number or similar identifier; 2) the name and location of the body conducting the hearing or proceeding; 3) subsequent to the use of this document in a formal proceeding please notify SWGDE as to its use and outcome; 4) the name, mailing address (if available) and contact information of the party offering or moving the document into evidence. Notifications should be sent to secretary@swgde.org.

It is the reader's responsibility to ensure they have the most current version of this document. It is recommended that previous versions be archived.

Redistribution Policy:
SWGDE grants permission for redistribution and use of all publicly posted documents created by SWGDE, provided that the following conditions are met:

1. Redistribution of documents or parts of documents must retain the SWGDE cover page containing the disclaimer.
2. Neither the name of SWGDE nor the names of contributors may be used to endorse or promote products derived from its documents.
3. Any reference or quote from a SWGDE document must include the version number (or create date) of the document and mention if the document is in a draft status.

Requests for Modification:
SWGDE encourages stakeholder participation in the preparation of documents. Suggestions for modifications are welcome and must be forwarded to the Secretary in writing at secretary@swgde.org. The following information is required as a part of the response:

a) Submitter's name
b) Affiliation (agency/organization)
c) Address
d) Telephone number and email address
e) Document title and version number
f) Change from (note document section number)
g) Change to (provide suggested text where appropriate; comments not including suggested text will not be considered)
h) Basis for change

Intellectual Property:
Unauthorized use of the SWGDE logo or documents without written permission from SWGDE is a violation of our intellectual property rights.

Individuals may not misstate and/or over represent duties and responsibilities of SWGDE work. This includes claiming oneself as a contributing member without actively participating in SWGDE meetings; claiming oneself as an officer of SWGDE without serving as such; claiming sole authorship of a document; use the SWGDE logo on any material and/or curriculum vitae.

Any mention of specific products within SWGDE documents is for informational purposes only; it does not imply a recommendation or endorsement by SWGDE.

SWGDE Linux Tech Notes

Table of Contents

1. Purpose

The purpose of this document is to provide background information for the forensic examination of computers running Linux operating systems.

2. Scope

The intended audience is computer forensic examiners trained and experienced in the examination of Windows and/or Macintosh operating systems seeking direction in the analysis of Linux systems.

This document is targeted at novice to intermediate examiners seeking familiarization with basic examination of systems running Linux operating systems. It focuses on common workstation and simple server setups. It does not address the examination of enterprise-class Linux servers or production environments, though many of the topics discussed will apply in these contexts. While it discusses some considerations in acquiring data from Linux systems, these topics are generally outside the scope of this document.

3. Limitations

This document was prepared with the resources available at the time of publication. As with all information technology, Linux is a constantly evolving environment with frequent implementation of new features and innovations. The specific configuration of any particular installation will vary widely and may not comport with the standards cited here.

This document is not intended for use as a step-by-step guide for conducting a thorough forensic investigation, nor is it legal advice.

4. Overview of Linux

Linux is a free, open source, UNIX-like operating system. The community-developed Linux kernel provides basic low-level operating system functions. Numerous third parties bundle the kernel with the higher-level components needed to build a fully functional computing platform. These bundles are known as “distributions.”

The Linux ecosystem is a highly open platform, meaning there is often significant diversity and not a single, canonical implementation of higher-level functions, as with a Windows or Mac OS X system.

There are now nearly a thousand available distributions, each with important commonalities with and differences from the others. Most current Linux distributions fall into one of three main development branches: Debian, Slackware, or Red Hat. It is beyond the scope of this document to provide a detailed catalog of the differences and similarities among all major distributions; however, this document will provide an overview of considerations for the forensic examination of most Linux systems.

As open source, community-developed software, most Linux features are well documented both online and within the system manual (“man”) pages, see [1]. These manual pages are often an excellent source of background information on a system or command. They are stored on the target system, accessible via the “man” command on a running system, and generally published online as well.

5. Acquiring Data from Linux Computers

Media from computers running Linux operating systems, once powered down, can be forensically acquired and examined using the same tools and techniques as any other operating system. See *SWGDE Best Practices for Computer Forensics* [2] for more information.

Understanding some systems cannot be shut down for acquisition and being aware of the growing use of encryption, the responding examiner may need to acquire live images of mounted volumes. Both commercial and open source acquisition tools are available for Linux, including but not limited to the built-in dd command, command line versions of FTK Imager, dc3dd, LinEn, and Guymager.

Some file systems used by Linux (e.g. Ext2) can be damaged by failing to shut the computer down properly, therefore, if the decision is made to power down a system prior to acquisition, the suggested best practice is to initiate a graceful shutdown.

5.1 Collections from Running Systems

As with all modern operating systems, running Linux systems contain potentially important information, which is lost at shutdown. The examiner must consider the needs of the investigation to determine what, if any, volatile data to collect from a running system prior to shutdown (e.g. running processes, network connection status, mounted remote file systems, loaded kernel modules, logged-on users, contents of the /proc directory).

Live response to a Linux system, like any running computer, necessitates interaction with the system and risks alteration or corruption of important data. Using pre-scripted collection routines built on trusted binaries compiled for the subject system is a best practice.

Historically, Linux kernels provided access to physical and virtual memory via special device files, /dev/mem and /dev/kmem, and examiners could obtain the contents of memory via these devices. For security and other reasons, these devices are not available in modern kernels. Today, acquiring an image of system memory (RAM) often requires the use of kernel modules specifically compiled for this purpose. Specific techniques for doing so are rapidly evolving and under active research, see [3] for more information.

Examiners responding to running systems should attempt to identify the use of encryption by reviewing running processes, loaded kernel modules, mounted file systems, and device mapper configuration; and if appropriate, conduct live acquisition of the subject media. As with most operating systems, this acquisition may require administrative privileges.

6. File Systems & Disk Configurations

While many Linux systems both support and utilize common legacy file systems such as FAT and can be built on traditional disk partition schemes, there are number of disk configurations and file systems exclusive to the Linux ecosystem.

6.1 Logical Volume Management (LVM)

Some Linux installations use LVM, a logical volume manager for the Linux kernel that manages disk drives and similar mass-storage devices, providing an abstraction layer on top of traditional partitions and block devices. LVM provides a more flexible configuration of storage on block devices by virtualizing the partitions and allowing them to be split, combined, and or arrayed across independent physical disks or physical partitions. In order to parse on-disk structures properly, forensic tools must be LVM aware. Acquisition of an LVM volume must consider the logical configuration of the storage.

6.2 File System Implementations

Modern Linux installations use an abstraction layer, the Virtual File System, to provide a standard interface for the kernel and other applications to the implementation of specific file systems, such as Ext2, Ext3, Ext4, and NTFS. This enables access to data on any file system without regard to the specifics of the file system's implementation or the actual location of the data.

The directory hierarchy, as presented to the user, can potentially be comprised of multiple file systems stored in disparate locations, including local media, network locations, or in memory. When acquiring data from a Linux system, examiners must consider these possibilities, including the possibility relevant data may be on a file system not backed to a local disk.

Traditionally, the code used to implement file systems runs in kernel-space, a protected portion of the operating system typically reserved for lower-level system functions. Modern Linux kernels include support for user-space file systems, commonly implemented via the File System in Userspace (FUSE) kernel module, which allows non-privileged users to implement their own file systems without requiring additional privileges. From a forensic perspective, this increases the possible locations where any user may store data and the possible underlying file systems they may use.

6.3 Common File Systems

Modern Linux operating systems are typically installed on the Extended file system (Ext) family of file systems. Most current forensic tools can parse these systems and interpret their associated metadata. Ext allows security settings and ownership metadata to be applied to files and folders, as well as other metadata, including flags for read, write, and execute modes.

Ext2 replaced the first iteration of the Ext file system in 1993. Ext3, introduced in 2001, adds journaling and support for larger file and file system sizes to Ext2's feature set. Ext4, introduced in 2008, added support for file creation timestamps, again increased the maximum file size, increased the maximum number of files in a directory, and added numerous performance and reliability enhancements.

Some older versions of Linux may be installed on the UNIX File System (UFS), which is currently supported by few forensic tools (e.g. Autopsy). The Linux kernel also supports, either natively or through external modules, additional file systems other than the Ext variants, such as the Zettabyte File System (ZFS), FAT32, NTFS, and VFAT.

While not explicitly related to the file system, in Linux, files are hidden by prepending the filename with the period character, “.”. In addition, many files on Linux systems do not have file extensions, as the operating system does not use the file extension to determine how to handle the file.

6.4 Network file systems

Linux allows users to mount remote file systems in the same manner as local disks using multiple protocols, including SMB/CIFS, AFP, NFS, and SSH. When mounted, these remote file systems are mapped to local user-specified mount points. On running systems, mtab file or mount command, described below, should document mounted network file systems. Examiners responding to live systems should check for mounted network file systems to ensure they identify all locations that contain information of potential forensic relevance and do not inadvertently obtain data outside the scope of their search authority.

6.5 fstab - file systems table

The file systems table, located at /etc/fstab, is a text file, which contains information some programs use to determine which file systems are mounted by default and where they are mounted. The mount command reads this file at boot time to determine the overall file system structure. It is most often used to mount file systems in a desired way each time the system is booted in order to prevent loading conflicts. Not all file systems or devices listed in the fstab may be currently mounted. Examiners should review the mtab file, described below, to determine which file systems in the fstab file are currently mounted.

6.6 mtab - mounted file systems table

The mounted file systems table, usually located at /etc/mtab, lists all currently mounted file systems and their initialization options. The mount and unmount commands manage the contents of this file. While only available on a running system, the mtab will list manually mounted volumes, where the fstab does not. The mount command provides the same information as the mtab file.

6.7 Links

6.7.1 Symbolic Links

A symbolic link is a pointer to a specific path, either a file or directory. The operating system automatically traverses symbolic links transparently to the user. Symbolic links may span file systems, and link to shares on other systems.

6.7.2 Hard Links

A hard link is another directory entry for an existing file that points to the same data. Hard links are not updated after their creation. Hard links cannot be created for directories and cannot span file systems.

The operating system makes no distinction between the original filename and any subsequently created hard links to that file; they are merely multiple names for the same file.

7. Examining Linux Operating Systems

7.1 Directory Structure

While there are differences among distributions, most Linux distributions roughly conform to the File system Hierarchy Standard (FHS, detailed below) maintained by the Linux Foundation [4].

The root level of the system drive is commonly referred to simply as “root,” is represented by the “/” character, and will likely contain the following directories:

bin	Essential command binaries (system programs)
boot	Static files of the boot loader
dev	Device files (a file object representing hardware devices on the system)
etc	Host-specific system configuration
home	User home directories (optional)
lib	Essential shared libraries and kernel modules
media	Mount point for removable media; implementation varies by distribution
mnt	Mount point for mounting a file system temporarily; varies by distribution
opt	Add-on application software packages
root	Home directory for the root user (optional)
sbin	Essential system binaries, typically associated with administrative functions
proc	Pseudo file system used by running processes to store and access process, hardware, and system information; this is a virtual file system that is not backed to disk and only exists while the system is running
srv	Data for services provided by this system; not commonly implemented
tmp	Temporary files
usr	Secondary hierarchy; user commands, non-essential system binaries
var	Variable data (log files, mail spools, caches, lock files)

While Linux does not require swap space, most Linux installations utilize a swap partition for memory paging. The system administrator can configure the system to use a swap file in lieu of or in addition to this setting. Settings for auto-mounting swap files and partitions will be found in /etc/fstab.

The conventional default Linux user profile location, or home directory, is /home/<username>. User-created data and configuration information commonly resides within the user’s home directory. The “~/” is a relative path referring to the current user’s home directory.

The system administrator account is called the root account, or simply “root,” and its home directory is typically located in /root.

7.2 Basic System Configuration Information

Unlike Microsoft Windows, which uses a central registry, many Linux system configuration settings are stored in easy to access, text-based configuration files stored throughout the file system.

Most system-level configuration files reside in the /etc directory. Most user-level configuration files reside within the user's home directory, often within hidden directories or files (names beginning with the "." character).

Analysis of a Linux system should begin with the identification of the Linux distribution and kernel version. This information is typically contained in one of the following locations:

/etc/issue

/etc/version

/etc/*-release

Other system information settings can be found in the following locations (the following are files generally in text format except as noted):

Hostname	/etc/hostname
IP Addresses	/etc/network/interfaces (Static - Debian variants) /etc/sysconfig/network-scripts/ifcfg-<interface name> (Static - Red Hat variants) /var/lib/dhclient/ (DHCP) /var/lib/dhcp/ (DHCP)
Time Zone	/etc/localtime
User Account List	/etc/passwd (/etc/passwd- contains the previous version of this file)
User Password Hashes	/etc/shadow (/etc/shadow- contains the previous version of this file)
User Group Lists	/etc/group (/etc/group- contains the previous version of this file)
Users with Admin Privileges	/etc/sudoers
Auto-Mounted File Systems	/etc/fstab
Auto Mounted Encrypted File Systems	/etc/crypttab

7.3 Users and Groups

7.3.1 User Account List

Most Linux distributions maintain the list of local user accounts in a world-readable, colon-delimited text file, /etc/passwd.

The fields include:

- Username – This value is unique.
- The user's encoded password field – The character "x" indicates shadow passwords are used on the system and passwords are stored in the "/etc/shadow" file, rather than the "/etc/passwd" file.
- Numeric user ID – This value is non-unique, used with the group field to identify which files belong to the user. User ID 0 is the root user and has unrestricted administrative privileges.
- Numeric primary group ID – A user may belong to one or more secondary groups, as specified in the /etc/group file.
- Full name of user
- The path to the user's home directory
- User's command shell – This program or file will be executed when the user logs in. For most users, this will be an interactive shell, such as "/bin/bash."

Note that the user ID, rather than the username, defines a unique user and their associated permissions. More than one username may share the same user ID.

7.3.2 Groups List

Groups is another way to manage user permissions to access files. Similar to the User ID (UID), groups have a Group ID (GID). Each user on a system is a member of at least one group (a primary group) and can be a member of supplementary groups to enable their ability to access files owned by another group. "/etc/group" is the file that defines the groups. There is one entry per line with the following fields, each separated by a colon:

- The group name (Group_Name) – This value is unique.
- Password
- The numeric group ID (GID)
- All the groups user's names separated by commas "user_list"
  (i.e. Group_name:password:GID:user_list)

7.3.3 Shadow file/Passwords

Most modern Linux systems store passwords in the /etc/shadow file. This is a colon delimited text file containing hashed passwords and account expiration information for all users. By default, only administrative users can read the /etc/shadow file, as opposed to the /etc/passwd file, which is world-readable by default, offering additional protection for the password hashes.

The fields include:

- Username
- Hashed and salted password, including an identifier for the hash algorithm used and the salt value – A blank entry (i.e. “::”) indicates a password is not required to log in. An asterisk or exclamation point (i.e. “:*:”, or “:!:”) indicates the account cannot log in using a password, but may be accessed using another means of authentication, if configured (e.g. su, or SSH key).
- Date of last password change (number of days since January 1, 1970)
- Number of days before password may be changed (0 indicates it may be changed at any time)
- Number of days after which password *must* be changed
- Number of days to warn user of an expiring password
- Number of days after password expiration that account becomes disabled
- Account expiration date (number of days since January 1, 1970)
- Reserved field for possible future use

7.4 Log Files

Most Linux systems are configured for robust logging of system, application, and user events, using the syslog facility. The syslog facility allows processes to send events for storage in log files in a central location locally, and potentially to remote stores, the latter configuration will be more commonly encountered in enterprise environments.

Many Linux daemons, services, and system-level functions will use the syslog facility for logging, rather than their own log files. Rsyslog and syslog-ng are the main syslogging tools used. For rsyslog,the /etc/rsyslog.conf and files in the /etc/rsyslog.d directory contain syslog-related configuration information. For syslog-ng, the directory /etc/syslog-ng contains the configuration information.

The majority of log files on Linux system are located in the /var/log directory. The table below lists some of the logs that may be of interest to the examiner.

Log	Description	Filename and location
System Log	System events, equivalent to Windows event log (device mounting, network configuration changes, security logs, etc.)	/var/log/syslog /var/log/messages
Authorization Log	Authentication-related events, including remote and local user logon/logoffs, sudo events and commands	/var/log/auth.log
Kernel Ring Buffer Log	Kernel message buffer; contains events from current boot, device errors	/var/log/dmesg
Kernel Log	Kernel debugging, info, and error messages	/var/log/kern.log
Installer Logs	Events generated during system installation	/var/log/installer /var/log/anaconda*
Package manager-related logs	Events related to installation of pre-built software packages	/var/log/rpm /var/log/apt /var/log/dpkg /var/log/yum.log
User Login History	wtmp tracks user logon logoff events in a binary format auth.log may contain similar information in a plain text format, depending on the system's configuration	/var/log/wtmp (Binary file) /var/log/auth.log

Most Linux systems rotate system-level log files using the logrotate facility and maintain a rolling time or size-based buffer of log files. Older log files are typically gzip compressed. Configuration files in /etc/logrotate.conf and /etc/logrotate.d/* specify the rotation intervals and retention period for archived logs.

7.5 Bootloaders

The bootloader is the first piece of software started (bootstrapped) by the BIOS or UEFI when a system is powered on or rebooted. The boot loader is not an operating system itself but is able to load and then transfer control of the computer over to the operating system. Boot loaders may enable the user to select among multiple different operating systems to which to boot the system. The boot loader may also allow the user to start the computer from alternative media (e.g. a USB thumb drive or external hard drive).

Examples of bootloaders found in current Linux distributions include GRUB2, GRUB, and LILO.

Reviewing the bootloader configuration files may provide details to the forensic examiner about alternate boot partitions or external media that is routinely present for the user to select. For most installations, bootloader configuration information is stored on the root of the primary hard disk, in the “boot” directory. This information will contain pointers to available kernels and other boot options. Users may also provide configuration options to the bootloader manually at boot time. System logs in the /var/log directory may document these options or one-time configuration changes.

7.6 Init Systems & Service Management

During startup, an initialization program, known as the init system, is the first process to start after the kernel is loaded. This process, typically named “init” and assigned process ID 1, manages system startup tasks and background processes, known as daemons. These daemons are equivalent to services in Windows. There are multiple competing implementations among various distributions. Current common init systems include System-VInit, systemd, and Upstart, each of which has different functionality, formats, and locations for storing their configuration information.

7.6.1 System-V Init

SysV-Init, Linux’s traditional init system, used numeric “runlevels” as a way to group daemons to be started and actions to be taken at certain pre-defined operating system states. While some newer init systems use other mechanisms, the concept of runlevels is often supported and referenced in some form for backwards compatibility with SysV-Init. While the meaning of specific runlevels is distribution specific, runlevel 0 (halt the system), 1 (single user mode without networking), and 6 (reboot the system) are common across distributions.

The /etc/inittab file defines runlevels and specifies the default run level. The /etc/init.d/ directory contains shell scripts to start or stop services or run tasks. The /etc/rcX.d (where X is the runlevel) directories contain symbolic links to scripts in the /etc/init.d directory that should be executed at that particular runlevel.

These links are named KXX<name> or SXX<name>, where XX is a number and “<name>” is the name of the service or task. "S" (Start) scripts are run when entering the runlevel. "K" (Kill) scripts are run when moving to another runlevel. The numeric value in the name indicates the order in which the scripts are run, with lower numbered scripts run first.

7.6.2 Systemd

Systemd uses the concepts of units and targets to manage services. A unit is systemd's representation of something it can manage, such as services, devices, sockets, mount points, and system states. Targets, themselves a kind of unit, group units together and are systemd's corollary to runlevels. Systemd uses several pre-defined special targets for moving the system into certain discrete states, including booting and shutdown, and for backwards compatibility. The "default" target specifies which target is used when the system boots.

Systemd's unit configuration files are named "<name>.<unit type>", for instance, "graphical.target" for the "graphical" target, the special target typically used for invoking the graphical user interface (GUI), or "ssh.service" for the SSH service.

System-wide unit and target definitions typically reside in the /lib/systemd/system or /usr/lib/systemd/system directories, depending on the distribution. These definitions may be overridden to provide system-specific configurations or enhancements; definitions in the /etc/systemd/system directory override the default system-wide definitions.

A newer addition to the systemd ecosystem, the mattdaemon (a.k.a. the Bourne Identity shell daemon), handles surreptitiously killing other processes and subsequently corrupts its own memory, eliminating all forensic evidence of said events.

Systemd may also be used to manage user level services; if enabled, this allows individual users to configure and run services in a similar fashion. System-wide user-level unit and target definitions typically reside in the /lib/systemd/user or /usr/lib/systemd/user directories, depending on the distribution, with overriding definitions in the /etc/systemd/user directory. Each user's own systemd definitions typically reside in their ~/.config/systemd/user directory.

7.6.3 Upstart

Upstart uses the concepts of jobs and events to manage services. A job is a process or task that Upstart can manage, such as a daemon. Upstart job configuration files typically reside in the /etc/init directory.

An event is a notification from Upstart to all jobs and other events that something has occurred on the system that may trigger a job to change state, such as the system booting, shutting down, or transitioning between runlevels. In their configuration files, jobs specify the actions they will take in response to certain events, such as starting or stopping when a transition occurs to or from a specific runlevel. For backwards compatibility, Upstart also supports SysV-Init scripts and will run the respective SysV-Init scripts for a particular runlevel in addition to the Upstart jobs for that runlevel.

Upstart may also be used to manage user-level jobs; if enabled, this allows individual users to create and configure jobs in a similar fashion. Per-user job configuration files typically reside in the ~/.init or ~/config/upstart directories, depending on the Upstart version. A system may also have system-wide jobs configured to run in a user-level upstart instance; for instance, these jobs may be used to manage GUI-related services specific to a single user's login session. Configuration files for these jobs typically reside in the /usr/share/upstart/sessions directory.

7.7 Common Services

Listed below are notable daemons commonly in use on Linux systems that may have forensic significance. For information on log files from these services, see **Section 7.4** Log Files.

7.7.1 Mail Transfer Agents & Mail Servers

Most Linux systems have an internal mail transfer agent (MTA) or mail server for routing email to and from local users, and potentially to and from the Internet. Many system processes deliver informational and error messages to users via this local mail facility; local user mailboxes may contain forensically relevant information about events that have occurred on the system.

MTA	Process Names	Configuration Files
Sendmail	sendmail	/etc/mail
Exim	exim4	/etc/exim4
Postfix	master, qmgr, local, smtpd, and others	/etc/postfix

7.7.2 Remote Access

Linux systems may use several protocols for remote access, including SSH, telnet on legacy systems, and VNC. See **Section 7.10 Remote Access and Administration** for detailed information on these protocols and services.

7.7.3 Internet Services Daemons (inetd and xinetd)

Historical Linux and UNIX implementations used “super-server”, the internet services daemon, or inetd, to manage and launch internet services, including telnet, IMAP and POP3 server, FTP servers, and web servers. inetd maintained a configuration file, typically at /etc/inetd.conf, specifying all the services inetd provided.

Over time, other implementations have replaced inetd. The extended internet services daemon, xinetd, provides similar functionality in a more secure fashion xinetd's configurations file, which specifies the services it provides, are typically located at /etc/xinetd.conf (general configuration) and /etc/xinetd.d (per-service configuration).

These daemons are not an essential part of a modern Linux system; they may not be present at all and these services may be provided via freestanding daemons or other means, such as systemd units.

7.7.4 File Sharing

Linux has implementations of many file sharing protocols, including FTP, UNIX’s Network File System (NFS), Windows’ SMB/CIFS, and Apple’s AppleTalk protocols.

NFS - NFS is a remote procedure call (RPC)-based distributed file system protocol commonly used by Linux and UNIX-based systems. Directories or filesystems available via an NFS server are known as exports; the /etc/exports configuration file on the server lists the exported directories and associated configuration options. NFS clients then mount the appropriate export using the `mount` command. Processes associated with an NFS server may include nfsd, rpc.nfsd, rpcbind, rpc.statd, rpc.idmapd, rpc.lockd, rpc.mountd, or similarly named processed.

Server Message Block (SMB) & Common Internet File System (CIFS) - Samba provides SMB/CIFS file sharing services and allows Linux systems to participate in Windows and Active Directory environments. Samba's processes include "smbd," which provides SMB/CIFS server, "nmbd," which provides NetBIOS and Windows Internet Name Server services, and "winbindd," which enables user and group resolution from Windows servers. Samba configuration files typically reside in the /etc/samba directory.

AppleTalk - The AppleTalk Filing Protocol (AFP) daemon, afpd, provides a Linux implementation of an AFP server. afpd's configuration files reside in the /etc/netatalk directory and the process runs as "afpd."

FTP & SFTP - A variety of traditional FTP implementations exist for Linux. Configuration files, forensic artifacts, and process names will vary depending on the specific application used. SFTP, a secure version of FTP using the SSH protocol, often runs as a sub-system of the SSH server; SFTP configuration options often reside within the SSH server's configuration file in this case. For additional information on SSH, see **Section 7.10.2 Secure Shell (SSH)**.

7.7.5 Web Servers

Apache and nginx are two web servers commonly found on Linux systems. Apache typically runs as several processes named "apache2," with configuration files in the /etc/apache2 or /usr/local/apache2/conf directories. Nginx typically runs as several processes named "nginx," with configuration files in the /etc/nginx, /usr/local/nginx/conf, or /usr/local/etc/nginx. The location of content being served varies, but will be specified in the web server's configuration file.

7.7.6 Database Servers

Some Linux systems will have database servers installed, either to maintain information about system functions or as a part of applications installed on the system. Some of the more common database servers examiners may encounter include MySQL, MariaDB, and PostgreSQL. MariaDB is a community-developed fork of MySQL and, at this time, functions identically for forensic purposes.

Database	Process Name(s)	Configuration Files
MySQL & MariaDB	mysqld	/etc/my.cnf
	mysqld_safe	/etc/mysql/my.cnf
PostgreSQL	postgres	/usr/local/pgsql/data
	postmaster (older versions)	/var/lib/pgsql/data
		/etc/postgresql
		(typical locations; no default)

7.7.7 Task Management

Linux systems use the at and cron daemons, atd and crond, respectively, to manage and run scheduled tasks. See **Section 7.8 Scheduled Tasks** for additional information on these services.

7.7.8 Printing

The Common UNIX Printing System (CUPS) is the de-facto standard printing system for Linux systems. See **Section 7.9 Common UNIX Printing System** for additional information.

7.8 Scheduled Tasks

Linux provides three primary task scheduling facilities, the ***cron*** and ***at*** daemons, and ***anacron***.

7.8.1 cron Daemon

The cron daemon handles recurring tasks to be run at regular intervals, defined in cron tables (crontabs). The /etc/cron.allow and /etc/cron.deny files, when they exist, may restrict use of the cron facility to specific users.

System-level crontabs contain jobs not associated with a particular user and reside in one of the following locations:

/etc/crontab

/etc/cron.d/*

/etc/cron.<time interval>/*

User-level crontabs contain jobs associated with specific users and typically reside within the /var/spool/cron/ directory. The “crontab” command is used to edit user-level crontabs. Crontabs specify months, days of the month, days of the week, hours, and minutes a command should be run.

In most Linux installations, output from cron jobs is sent to a syslog facility or the owning user’s local mailbox, unless otherwise specified.

7.8.2 anacron

Systems that are not continuously powered, such as desktops or laptops, on may use the *anacron* facility for task scheduling. *Anacron* jobs may be configured to run on a daily, weekly, monthly, or annual basis, and will execute as close to the specified schedule as the system’s uptime permits.

The file /etc/anacrontab contains the definitions of *anacron* jobs, specifying the period, delay, a job identifier, and the command to be run.

Files in /var/spool/anacron contain the last execution date for the job with the identifier specified in the file name. Anacron uses the dates in these files to determine when to run a job.

7.8.3 *at* Daemon

The *at* daemon runs tasks to be executed once at a specified time. The /etc/at.allow and /etc/at.deny files, when they exist, may restrict use of the *at* facility to specific users. *at* jobs are created using the at command and are stored as shell scripts in the /var/spool/cron/atjobs directory. The final eight characters of the job file's name are a UNIX epoch timestamp in hexadecimal specifying the job's execution time.

The *at* command auto-generates the first portion of the job file with commands to restore the environment variables that existed in the user's shell when they created the job. The user-provided commands are at the end of the file.

Also part of the *at* job facility, jobs created with the "batch" command execute when the system load drops below a specified level, rather than at a specific time.

7.9 Common UNIX Printing System

The Common UNIX Printing System (CUPS) is the standards-based, open source printing system developed by Apple Inc. for OS X and other UNIX-like operating systems, see [5]. Many Linux installations make use of CUPS. CUPS uses the Internet Printing Protocol (IPP) to support printing to local and network printers. The CUPS application runs as a service (daemon) providing central print scheduling process that dispatches print jobs, processes administrative commands, provides printer status information to local and remote programs, and informs users as needed.

CUPS writes job files to a spool directory, typically /var/spool/cups. Two types of files will be found in the spool directory: control files starting with the letter "c" ("c00001", "c99999", "c100000", etc.) and data files starting with the letter "d" ("d00001-001", "d99999-001", "d100000-001", etc.) Control files are Internet Printing Protocol (IPP) messages based on the original IPP Print-Job or Create-Job messages while data files are the original print files that were submitted for printing. There is one control file for every job known to the system and 0 or more data files for each job. Data files can be formatted as text, PDF, postscript, and other image file types.

Control files are normally cleaned out after the 500th job is submitted; data files are removed immediately after a job has *successfully* printed. Data files submitted and not successfully printed may remain for extended time periods. Deleted data files can often be carved from unallocated space.

Information about printers attached to the system can be found in /var/cache/cups.

7.10 Remote Access and Administration

7.10.1 Telnet

A legacy protocol, Telnet allows user access to the command line interface of a remote system over a virtual terminal connection. Telnet is fundamentally insecure as it does not provide encryption of traffic and does not offer secure authentication protocols. For this reason, Telnet, if installed, is typically disabled on modern distributions.

Telnet is commonly managed by the internet services daemon (inetd) or one of its replacements, such as xinetd or systemd. Telnet settings will be found in the configuration for these services (e.g. etc/xinetd.d/telnet).

7.10.2 Secure Shell (SSH)

The SSH protocol provides a means for creating secure, encrypted connections between hosts that can be used to obtain remote console and GUI access, transfer files (typically using the scp and sftp commands), and tunnel generic TCP traffic. SSH allows for secure access and administration of remote systems by providing robust authentication protocols and encrypted connections. Authentication can be accomplished by several means, including passwords and asymmetric key pairs.

Most Linux systems utilize OpenSSH, an open source SSH client and server. Configuration settings and usage artifacts can be found in the locations detailed in the table below.

File	Description
/etc/ssh/ssh_config	System-wide SSH client configuration
/etc/ssh/sshd_config	SSH server configuration
~/.ssh/known_hosts	Tracks the public keys of SSH servers to which a user has connected; hostnames in this file may be hashed depending on the client configuration
~/.ssh/authorized_keys	Contains the public keys for key pairs this particular user may use to login
~/.ssh/config	User-level SSH client configuration

7.10.3 Virtual Network Computing (VNC)

Virtual Network Computing is a platform-independent desktop sharing protocol that enables users to interact with graphical interfaces on remote computers. A variety of implementations exists across the Linux ecosystem. Configuration files and forensic artifacts vary depending on the application. System-wide configuration files typically reside in /etc/ or subdirectory; user-specific settings and artifacts typically reside in a hidden directory within the user's home directory.

7.11 Encryption

There are two primary encryption configurations frequently observed with modern Linux installations, block device level encryption and file system level encryption. As is common with Linux, there are multiple implementations examiners may encounter; this paper covers several of the more common Linux-specific technologies. Examiners may also encounter cross-platform encryption tools, such as TrueCrypt, in use on Linux systems.

7.11.1 Block Device Encryption

With block device encryption, the entire block device is encrypted and must be decrypted and mounted to access the file system on it. The /etc/crypttab file lists encrypted block devices (or logical volumes) that will be mounted when the system boots. Typically, the boot loader or initial ramdisk handles boot-time authentication.

One common method of encrypting block devices involves using the encryption subsystem for the kernel's device mapper, a built-in facility for mapping physical block devices to virtual block devices. In this scenario, the physical block device is encrypted and the device mapper presents an unencrypted virtual device to the operating system and users. Examiners may identify this configuration on running systems by inspecting the device mapper configuration using the dmsetup command. When encountering an encrypted device or volume that has not been unencrypted, examiners should ensure they acquire the partition or volume's header, as this contains information needed to decrypt the partition, including the master encryption key, in some configurations.

Device-level encryption may be used with LVM, either by encrypting the underlying block devices, the logical volumes, or both. A Linux installation may use any number or combination of encrypted and unencrypted block devices or logical volumes, which may use different encryption keys.

Of note to examiners, some full disk encryption configurations encrypt the swap and /tmp file systems using separate, random, ephemeral keys generated each time the system boots. These areas may contain information of forensic significance but will be inaccessible after the system shuts down, even if the examiner possesses the keys for the root file system. If this is the case, the lines in the /etc/crypttab file for these file systems will reference a random number generator, like /dev/urandom, or another entropy source.

7.11.2 Stacked file system encryption

File system encryption allows users to encrypt individual directories within a file system, such as a user's home directory. In common implementations, there is a specific directory that contains the encrypted file system. The encryption utility handles decryption and mounts the unencrypted version of the file system to a user-specified mount point.

Depending on the implementation, examiners may be able to glean useful metadata, such as file sizes, timestamps, and directory structures for the encrypted data, by examining these encrypted storage locations, even if the data cannot be decrypted. The output of the mount command or contents of the /etc/mtab file may allow examiners to identify mounted encrypted directories on running systems.

7.11.3 Linux Unified Key Setup (LUKS)

LUKS is a common disk encryption scheme used on Linux systems. It provides management functions and metadata needed to enable multi-user support and password changes. LUKS supports encryption of multiple container types, including entire devices, partitions, logical volumes, and files to be mounted as loopback devices.

A LUKS-encrypted container is encrypted using a random master key generated when the header is created. The LUKS header has eight key slots, each of which can store a version of the master key encrypted with multiple iterations of a salted version of the user's passphrase. A passphrase can be either a literal passphrase or a file, or portion of file, used as a key. Examiners can recognize a LUKS-encrypted container by its unique header of the ASCII string "LUKS" followed by the hex values `0xba 0xbe`.

In some situations, the LUKS master key may be recoverable when the container is mounted; the LUKS documentation, mailing list, and source code repository contain some suggestions and utilities for doing so. Consequently, examiners should avoid powering off or rebooting systems that may be LUKS-encrypted without evaluating this possibility.

7.12 Command Line Interface and Associated Artifacts

7.12.1 Command Shells

A shell is a command-line interpreter that provides a user interface for the OS on a Linux system. The user can enter commands as text that a command line interpreter executes. Users usually interact with a shell through a terminal emulator or console.

The following are some of the commonly seen command shells in the Linux environment:

- BASH – Bourne Again Shell (This shell is usually the default shell for most Linux distributions.)
- Csh – C Shell
- Ksh – Korn shell
- Zsh – Z shell

The following table lists some common locations containing artifacts of Command Shell related user data.

Location / file	Description
~/.bash_history **~/.sh_history**	History of commands executed by the user in the bash shell History of commands executed by the user in the Korn shell
~/.aliases **~/.bash_aliases**	persistent aliases are sometimes defined in separate login scripts; this script's location varies and must be executed by one or more of the login scripts listed below. These are two paths frequently used for this purpose.
/etc/profile	script file executed when a user logs in to a bash login shell; this file executes before all other login scripts if it exists
/etc/environment	system-wide environment variable assignments
~/.bash_profile **~/.bash_login** **~/.profile**	bash executes only the first of these scripts files it finds when a user logs in. These files often call ~/.bashrc and are used to store user preferences for the command shell and basic shell environment variables (e.g. command history length, keyboard settings)
~/.bash_logout	script file executed when a user logs out of a bash login shell
~/.bashrc	script file executed when a user creates a non-login bash shell (e.g. a terminal window in a graphical user interface). Often used to store user settings specific to the bash shell, may contain persistent aliases and variables for specific programs

7.12.2 Environment Variables

Environment and shell variables provide a location shared between multiple processes or applications within a session to store configuration information and information about the environment. These variables may contain information such as the paths to search for executable files when a command is executed, the current and previous working directories, desktop environment, user's home directory and more.

Some common, forensically relevant variables are listed in the table below.

Variable Name	Description
PATH	Contains colon-delimited list of directories to search for executable files; when a user executes a command, the directories in the PATH variable are searched for executable files matching the command name, in the order specified.
PWD	Current working directory
OLDPWD	Previous working directory
SHELL	Path to the running shell
HOME	Path to the user's home directory
DESKTOP_ENVIRONMENT or DE	In a GUI, specifies the current desktop environment
Begins with "XDG_"	GUI-related configuration information

7.12.3 Basic Commands

Linux Command	Description	Windows Equivalent
Cd	change directory	cd
Pwd	print working directory	chdir
mkdir	make new directory	mkdir
Ls	list contents of a directory	dir
Cp	copy	copy
mv	move (or rename) a file from one location to another	move / ren
rm	remove (delete) a file	del
cat	display contents of a file or concatenate contents of multiple files	type
Dd	convert and copy files	
head	displays the first part of a file	
tail	prints the last part of a file	
less	similar to more but allows backwards and forwards movement	more
more	peruse contents of a file one screen at a time	more

Linux Command	Description	Windows Equivalent
grep **sed** **awk**	Grep searches the named input for lines containing a match to a given pattern. Sed is a stream editor used to perform basic text transformations on an input stream. Awk will perform pattern matching, text file manipulation, and processing on file and stream inputs	find findstr
vi **nano** **pico** **vim**	basic text editors	edit
diff	compare two files	fc
Df	reports file system disk usage (the -h command prints sizes in human readable format e.g. 1K, 256M, 5G)	dir displays space available at end of directory listing
Du	disk file usage (the -h command prints sizes in human readable format e.g. 1K, 256M, 5G)	
man	displays the manual for commands	command /?
mount	attached the specified file system to the specified directory	
Ps	list running processes	tasklist
sudo	allows a user to execute commands with root privileges without the user account having root privileges.	
Su	switch user	
clear	clear the screen	cls
Date	display date and time	date time
echo	display (echo) text	echo
Env	runs a program in a modified environment; takes environment variables to set as arguments, in addition to the target command	
printenv	prints the values of environment variables	set
export	sets an environment variable	set
Set	set or unset values of shell options and positional parameters	set

7.12.3.1 su and sudo

The su command allows a user to run a shell as a different user, after authenticating. Commands run in an su session are logged in the shell history file for the impersonated user.

The sudo command allows a user to execute commands as a different user. In most cases, sudo is used to execute commands as a superuser. In the default configuration, the /etc/sudoers file defines the specific commands users or groups may execute and the users they may impersonate. Typically, the auth log facility logs all commands executed via the sudo facility.

7.12.4 Shell Scripts

A shell script is a text-based program containing a sequence of commands to be interpreted and run by the Linux shell. It combines into a "script" a series of commands that would otherwise have to be entered by typing each command separately on the command line. The advantage of such a script is that the commands are automated and do not require user intervention to trigger each phase of the process. Shell scripts can be written in a variety of scripting languages and can be used for a variety of purposes.

Generally, shell scripts do not have a file extension unless they are intended to be loaded into a running shell using the source mechanism; However if a file extension is given, generally the .sh extension is used.

7.12.5 Aliased Commands

The alias command allows a user to create a unique command shortcut that can be used to execute any command or group of commands with arguments. An alias can be created using the same name as the target command or with any user generated name. Aliases are frequently simple names and can even be a single character. Of note to the digital investigative analyst, the **alias** (not the actual command aliased) is recorded in the shell's history file. The examiner should be aware that due to aliasing, the entries in the history file may not accurately represent the commands executed.

Alias commands can be temporary or permanent for a user's command shell. Temporary aliases expire when the terminal shell exits. User or system-level login scripts restore permanent aliases for new shell sessions. Examiners should review these scripts to identify possible permanent aliases.

As an example, "*ls*" is a command commonly used to list the names of the files and directories in the current directory. Two common arguments to the "ls" command are "*-a*" to show all files including hidden files and the "*-l*" argument to display detailed information about each file (aka: long listing format). A user could create an alias "l" defined as "ls -la", so every time the user executed the "*l*" command, the alias would list all files and directories (including hidden files) in the current directory using a long listing format.

7.13 Graphical User Interfaces and Associated Artifacts

There is no one desktop environment for Linux; there are many Graphical User Interfaces (GUIs) from which a user may choose. A user may install multiple desktop environments on a Linux system and switch between them at login; conversely, most Linux distributions can be installed without a GUI.

Many Linux distributions come with a default desktop environment. Some of the more common desktop environments include:

- Gnome - A common desktop system used by many popular Linux distros.
- Unity - Ubuntu's default configuration of the Gnome desktop environment
- KDE - Another formerly popular desktop environment similar to Windows.
- Xfce - A graphically lightweight desktop environment

7.13.1 Basic Components

GUIs on Linux systems are comprised of several components.

A windowing system, typically the X Window System (X11), forms the GUI's foundation is the only component essential to a GUI. The windowing system provides primitive GUI functions, including support for drawing windows and interacting with input devices. Users can run a single graphical application, such as a terminal or web browser, under X11 without any additional components.

Windows managers manage, organize, and provide users an interface to interact with multiple windows.

File managers are applications that provide a graphical interface for browsing the filesystem. Some file managers also provide support for mounting and traversing external and network file systems.

7.13.2 Desktop Environments

Desktop environments bundle a window manager, file manager, and other applications that provide a complete desktop experience, including basic graphical applications, such as preferences panes, text editors, a notification management system, and widgets. While desktop environments provide a default set of these components, users may replace any of the default components with their own choices. Common desktop environments include GNOME, KDE, Xfce, and LXDE.

Forensic artifacts in a Linux desktop environment are created by the individual application providing the functionality in question. For instance, the file manager application may generate artifacts documenting recently used files. Because a Linux desktop environment is a conglomeration of discrete applications filling specific functions multiple options exist for each of these applications, forensic artifacts available to an examiner will vary based on the specific applications used in the user's desktop environment.

7.13.2 Application Artifacts

The diversity of Linux distributions and the variety of applications available precludes the creation of a comprehensive guide for locating and analyzing forensically-relevant application artifacts. As each version of each distribution may have different "default" included applications it would not be useful attempt to identify each here. While application developers have freedom to store the configuration files and application data nearly anywhere on the system volume, the following conventions are typically followed:

Type of Data	Location
User Specific Application Data	~/.<application name>
Default (Distribution Included) Application Binaries	/usr/bin/<application name>
User Installed Applications	/usr/local/bin
System-level Application Logs	/var/log/<application name>

Below is a high-level survey of applications commonly included in desktop environments for popular distributions:

7.13.2.1 Nautilus

Similar to Windows Explorer, Nautilus is a default graphical file manager (browser) for the Gnome desktop environment and allows a user to configure their desktop, browse pictures, access network resources, and more all from one integrated interface. A listing of recently used files can be found at ~/.recently-used.xbel.

7.13.2.1.1 Thumbnails

Unlike Windows Explorer which stores thumbnails in a database within each directory, many Linux file managers store PNG thumbnails of viewed graphics in one central location for each user, though this location may differ by installation. As an example, Nautilus, the default file manager for the GNOME desktop environment, saves these thumbnails in a hidden directory (.thumbnails) within the user's home folder. These thumbnails are named with a system generated filename which is the the MD5 hash of the Uniform Resource Identifier (filename with full path).

7.13.2.2 KDE Artifacts

KDE is one of the more common Linux desktop environments. Its component applications and frameworks generate certain artifacts which may be of forensic relevance. User-level KDE artifacts typically reside within the ~/.kde directory.

For instance, the KActivities framework tracks certain user activities in applications that use it. Of forensic significance, KDE's two primary file managers, Dolphin and Konqueror, report a user's directory browsing activity via this framework. The framework tracks these activities in a SQLite database located at ~/.kde/share/apps/actvitymanager/resources/database which contains the directory accessed, application used, and start and end timestamps for the activity. Many of KDE's applications also report file activity via this framework.

7.13.2.3 Firefox

The Firefox web browser profile container stores browsing history, bookmarks, and other configuration settings in the same format as in installations on Windows and OS X. The typical installation will place these files at ~/.mozilla/firefox/

7.13.2.4 Chromium / Google Chrome

The Chromium and Google Chrome web browsers store browsing history, bookmarks, web logins, downloads, search terms and archived artifacts in an SQLite format. The typical installation will place these files at ~/.config/google-chrome/ or ~/.config/chromium.

7.13.2.5 Thunderbird

Thunderbird is a free email client by Mozilla which supports the MBOX mail format. Profile folders are located at ~/.thunderbird/<profile name>. However, the profile folders for the Debian and Ubuntu builds are stored at the following location: ~/.mozilla-thunderbird/<profile name>.

7.13.2.6 Empathy

Empathy is a messaging program which supports text, voice, video chat, and file transfers over the following protocols: Google Talk (Jabber/XMPP), MSN, IRC, Salut, AIM, Facebook, Yahoo!, Gadu Gadu, Groupwise, ICQ and QQ. Empathy is capable of private and group chat, as well as the ability to log chat conversations, which can be turned on or off by the user. Account settings are located at ~/.mission-control/ accounts/accounts.cfg (except passwords which are stored in gnome-keyring). New chat conversation logs (since Empathy 2.31.4) are located at ~/.local/share/ TpLogger/logs. Previous to Empathy version 2.31.4, chat conversation logs are located at ~/.local/share/Empathy/logs/. Empathy configuration is stored in DConf and ~/.config/Empathy/. Avatars are cached in ~/.cache/telepathy/avatars/.

7.13.2.7 LibreOffice

LibreOffice is a free and open source office suite which includes a word processor, spreadsheet application, presentation engine, drawing and flowcharting application, database application and mathematics editor. User files are stored in the default document folder of the system.

Other potentially useful artifacts are stored in the following locations:

- AutoCorrect - This folder stores AutoCorrect texts at ~/user/autocorr
- AutoText - This folder stores AutoText at ~/user/autotext
- Backups - Automatic backup copies of documents are stored at ~/user/backup
- Temporary files - LibreOffice places its temporary files at ~/user/temp

7.13.2.8 Evolution Mail

Evolution Mail is a personal information management application providing integrated mail, calendar and address book functionality. The user's data files are located at

~/.local/share/evolution. Various configuration and state files can be found at ~/.config/evolution. Disposable data caches are located at ~/.cache/evolution. Additional configuration files are located at ~/.gconf/apps/evolution.

7.13.2.9 Pidgin

Pidgin is an open-source messaging application capable of adding the following protocols: AIM (Oscar and TOC protocols), ICQ, MSN Messenger, Yahoo, IRC, Jabber, Gadu-Gadu, and Zephyr. User data and configuration files are stored at ~/.purple.

7.13.2.10 GIMP

GIMP (GNU Image Manipulation Program) is an advanced picture editor which can be used to edit, enhance, and retouch pictures and scans using fine-control settings similar to Adobe PhotoShop. GIMP can also be used to create drawings and images. GIMP supports many of the popular image formats. Images saved using GIMP and its configuration files are stored at ~/.gimp-*(version)*.

7.13.2.11 VNC

The Virtual Network Computing (VNC) application is a graphical remote desktop application that uses the Remote Frame Buffer (RFB) protocol. VNC allows the user to control the mouse and keyboard input, as well as see the events on the GUI, on a remote computer from their own computer.

7.13.2.12 Totem

Totem is the free audio/video player integrated into the GNOME desktop environment and Nautilus file manager. A user can clear recent history for the Totem media player and many other GNOME based applications by removing the following file ~/.local/share/recently-used.xbel.

7.14 Software Package Management

Most Linux distributions use packages—bundles of pre-built software with metadata, including information about the software and its dependencies—to distribute software to end-users. This saves users from having to build the software themselves. The specific package format is distribution-specific. In general, Debian-derivative distributions use the dpkg format and Red Hat-derivative distributions us the Red Hat Package Manager (RPM) format.

Distributions also include package management tools, such as dpkg and apt in Debian-derivatives and rpm and yum in Red Hat-derivatives, to provide an interface for users to install, update, and remove packages, and resolve dependencies. These package management tools may also allow users to retrieve and install packages over the Internet from distribution or third-party managed repositories.

These package management tools may generate forensically relevant artifacts, such as logs of package installation and records of which files are associated with a particular package. For dpkg-based systems, log files at /var/log/apt and /var/log/dpkg.log may contain forensically relevant information; For RPM-based systems, the /var/lib/yum/history and /var/lib/rpm directories may contain similar artifacts.

8. References

[1] The Linux Documentation Project. [Online]. http://www.tldp.org/

[2] Scientific Working Group on Digital Evidence, "SWGDE Best Practices for Computer Forensics". [Online]. https://www.swgde.org/documents/Current%20Documents

[3] J. Stuettgen and M. Cohen, "Robust Linux Memory Acquisition with Minimal Target Impact," *Digital Investigation*, vol. 11, pp. S112–S119, May 2014. [Online]. http://www.dfrws.org/2014eu/proceedings/DFRWS-EU-2014-14.pdf

[4] "Filesystem Hierarchy Standard," The Linux Foundation, LSB Workgroup, Version 3.0, June 3, 2015. [Online]. http://refspecs.linuxfoundation.org/fhs.shtml

[5] The Common UNIX Printing System (CUPS). [Online]. http://www.cups.org/

History

Revision	Issue Date	Section	History
1.0	09/17/2015	All	Initial draft created. Voted by SWGDE for release as a Draft for Public Comment.
1.0	09/29/2015	All	Formatting and technical edit performed for release as a Draft for Public Comment.
1.0	01/14/2016	All	Minor edits made throughout. Voted by SWGDE for release as an Approved Document.
1.0	02/08/2016	All	Formatting and technical edit performed for release as an Approved Document.