AlMustafa University

Jamiat AlMustafa

Cybersecurity Technical Engineering

Lecture 2

Linux Filesystem Navigation and Manipulation

Instructor: Dr. Husam Alkinani

Ph.D. in Computer Science and Engineering

Second Year - Cybersecurity Technical Engineering

Last Updated: October 1, 2025

Linux Filesystem Navigation

Filesystem

Fundamental Filesystem Concepts

The filesystem is the backbone of any operating system, providing structured data organization and access control mechanisms. It serves as the foundation for cybersecurity operations, digital forensics, and system administration.

The filesystem continually evolves to keep up with increasing storage needs (managing how data is stored and retrieved) and to ensure data security, making it an essential part of the digital infrastructure in every computer.

For cybersecurity professionals, understanding the filesystem is crucial for:

- Digital forensics and evidence collection
- System monitoring and log analysis
- Security configuration management
- Malware analysis and detection

Core Filesystem Functions

Essential Filesystem Operations

Storage Management: Defines how data is stored and organized into files and folders with security attributes.
Access Control: Determines how to access stored files and implement permission-based security.
Naming Convention: Specifies file and folder naming rules, including security-relevant extensions.
Hierarchical Structure: Creates organized tree structures for efficient data access and security boundaries.

Key Components of a Filesystem

Critical Filesystem Components for Security

Blocks: Fundamental data storage units that can be analyzed for forensic recovery and security assessment.
Inodes: Metadata structures containing crucial forensic information - file attributes, timestamps, permissions, and data location pointers.
Path Systems: Absolute and relative addressing mechanisms that define access routes and security boundaries within the filesystem.
Files and Directories: Basic organizational units that form the foundation of access control and security policy implementation.

Types of Filesystems

Different operating systems use various filesystem types, each with specific security characteristics and limitations. Figure 1 provides a comprehensive comparison of the most common filesystem types used across different platforms.

Filesystem Types and Security Features

Windows Filesystems:

FAT32: 4GB files, Basic security, Limited forensics

NTFS: 16TB files, ACLs & Encryption, Excellent forensics

Linux Filesystems:

EXT4: 16TB files, Unix permissions & Journaling, Good forensics

macOS Filesystems:

APFS: 8EB files, Built-in encryption & Snapshots, Moderate forensics

Figure 1: Filesystem Types and Security Features

Security Implications by Filesystem:

- **FAT32**: Limited security features, vulnerable to data recovery attacks, commonly found on USB devices in cybersecurity investigations
- **NTFS**: Advanced security with Access Control Lists (ACLs), BitLocker encryption support, extensive forensic capabilities
- **EXT4**: Strong Unix-style permissions, journaling for integrity, widely used in Linux security appliances

- ▷ **APFS**: Built-in encryption, copy-on-write for data integrity, snapshot capabilities for incident response

As shown in Figure 1, each filesystem has distinct security advantages and limitations. The choice of filesystem significantly impacts digital forensics capabilities and security implementation.

Filesystem Hierarchy

Files in the operating system are organized in a tree structure that starts with a root directory (”/”). From the root, folders and files branch out. Figure 3 illustrates the Linux filesystem hierarchy structure.

In Unix and Linux systems, the file structure is arranged in a hierarchy that starts from ”/” as the root of all files and folders, as shown in Figure 3.

Filesystem Operations

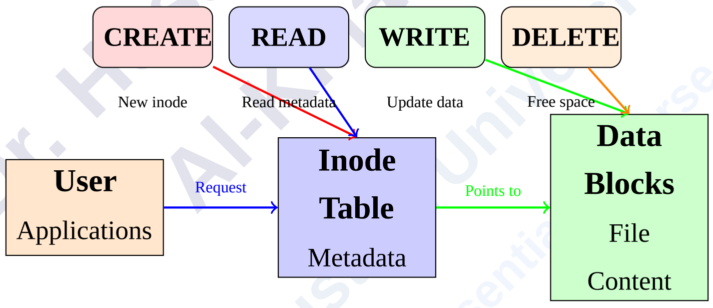

Figure 2: Filesystem Operations: Clear Component Flow

- ▷ **Creating Files**: A name and storage space on the disk are allocated, and the inodes are updated.
- ▷ **Reading Files**: The OS parses the file path, accesses the appropriate inode, and then loads the blocks containing the data.
- ▷ **Modifying Files**: When a file is modified, the system writes the new data into the blocks and updates the inodes.
- ▷ **Deleting Files**: The inodes associated with the file are removed, and the blocks allocated to it are freed.

Filesystem Challenges

- ▷ **Fragmentation**: Over time, files are scattered across multiple, non-contiguous blocks, affecting performance and slowing down file reads.

- **Cross-System Compatibility**: Different filesystems are not always compatible with all operating systems, such as NTFS and EXT4, which require additional tools to access data across systems.

Linux Filesystem Hierarchy for Cybersecurity

```
/                          # Root directory - High privilege
|-- bin/                   # Essential commands (targets)
|-- boot/                  # Boot files (system critical)
|-- dev/                   # Device files (hardware control)
|-- etc/                   # Config files (security policies)
|   |-- passwd            # User accounts (authentication)
|   |-- shadow            # Password hashes (critical)
|   |-- hosts             # Network hosts (security)
|   +-- ssh/              # SSH config (remote access)
|-- home/                  # User directories (data boundary)
|   |-- user1/            # User space (privilege separation)
|   +-- user2/            # User space (isolation)
|-- lib/                   # Libraries (dependency security)
|-- media/                 # Removable media (threat vector)
|-- mnt/                   # Mount points (boundaries)
|-- opt/                   # Optional software (third-party)
|-- proc/                  # Process info (runtime monitor)
|-- root/                  # Root home (highest privilege)
|-- run/                   # Runtime data (temp context)
|-- sbin/                  # System binaries (privileged ops)
|-- tmp/                   # Temporary files (cleanup)
|-- usr/                   # User applications (software)
|   |-- bin/              # User commands
|   |-- lib/              # User libraries
|   +-- share/            # Shared data
+-- var/                   # Variable data (logs/monitor)
    |-- log/              # System logs (security)
    |-- cache/            # Application cache
    +-- tmp/              # Temporary storage
```

Figure 3: Linux Filesystem Hierarchy with Security Annotations

Permissions and Security

Linux Permission System

The filesystem implements a robust security model through discretionary access controls, managing who can read, modify, or execute files using the fundamental rwx (Read, Write, Execute) permission system applied to User, Group, and Others.

The Linux permission system forms the foundation of filesystem security, providing granular control over data access and system operations.

Recent Filesystem Enhancements

- **Encryption**: Maintaining data confidentiality through encryption techniques at the filesystem level.
- **Snapshots**: Creating recoverable copies of files at a specific point in time.
- **Failure Recovery**: Modern systems like ZFS offer automatic error correction and data recovery features.

Filesystem Components in Linux

Critical Linux Directories for Cybersecurity

- **/ (root)**: The highest level containing all system folders and files - critical for system integrity
- **/home**: User directories with personal files and settings - target for privilege escalation
- **/etc**: System configuration files and security settings - contains password hashes, network configs
- **/var**: Logs and variable files - crucial for security monitoring and incident response
- **/tmp**: Temporary files - often monitored for suspicious activity and malware
- **/bin**: Essential system commands and executables - potential target for replacement attacks
- **/usr**: User programs and applications - contains most security tools and utilities
- **/opt**: Optional software packages - third-party security tools installation
- **/boot**: Boot loader files and kernel - critical for system security and integrity
- **/proc**: Virtual filesystem with system information - useful for system analysis

Commands for Navigating the Filesystem

Basic Navigation Commands

Directory Navigation for Security Analysis

```
# Current location identification
pwd                          # Print working directory

# Directory listing and analysis
ls                           # List files and directories
ls -la                       # Detailed listing with permissions and hidden
    files
ls -lh                       # Human-readable file sizes
ls -lt                       # Sort by modification time (useful for
    forensics)
ls -lS                       # Sort by file size

# Navigation commands
cd /home/kali                # Move to specific directory
cd ..                        # Go to parent directory
cd /                         # Navigate to root filesystem
cd ~                         # Move to home directory
cd -                         # Return to previous directory

# Directory structure visualization
tree                         # Display directory tree structure
tree -a                      # Include hidden files in tree
tree -L 2                    # Limit tree depth to 2 levels
```

Security Tip: Use `ls -la` regularly to identify hidden files that may contain malware or unauthorized configurations.

Creating Directories and Files

Secure Directory Creation

Directory Management for Security Projects

```
# Essential directory commands
mkdir directory_name       # Create directory
mkdir -p path/to/dir      # Create nested directories
mkdir -m 700 private_dir   # Create private directory

# Security project structure
mkdir -p ~/cybersec_project/{logs,scripts,reports}
touch filename.txt         # Create file
echo "content" > file.txt  # Create file with content
```

Security Best Practices:

▷ Always set appropriate permissions when creating security-related directories
▷ Use descriptive naming conventions for incident response materials
▷ Store sensitive files in protected directories with limited access

File Creation and Content Management

Basic File Operations

```
# Essential file commands
touch newfile.txt          # Create empty file
echo "content" > file.txt   # Create file with content
echo "more" >> file.txt     # Append to file
```

Viewing and Editing Files for Security Analysis

File Content Analysis Commands

File Analysis for Cybersecurity

```
# Basic file viewing
cat filename.txt           # Show entire file content
head -20 /var/log/syslog   # Show first 20 lines
tail -f /var/log/syslog    # Follow log file in real-time

# File analysis tools
file suspicious_file       # Determine file type
strings binary_file        # Extract readable strings
```

Secure File Editing

Secure Text Editing Commands

```
# Text editors for security configuration
nano /etc/hosts            # Edit hosts file for network security
vim /etc/ssh/sshd_config   # Configure SSH security settings

# Backup before editing critical files
cp /etc/passwd /etc/passwd.backup
nano /etc/passwd

# View file with line numbers (useful for configuration files)
nl /etc/ssh/sshd_config
cat -n /etc/security/limits.conf
```

Security Considerations:

- Always backup configuration files before editing
- Use `tail -f` to monitor logs in real-time during security incidents
- Verify file integrity after modifications using checksums
- Use appropriate text editors for different file types and security levels

Advanced File Searching and Security Analysis

Security-Focused Search Commands

Cybersecurity Search and Analysis

```
# Essential security searches
grep "Failed password" /var/log/auth.log     # Find failed login attempts
grep -i "error" /var/log/syslog             # Search for errors
find / -perm -4000 2>/dev/null              # Find SUID files
netstat -tuln | grep ":22"                  # Check SSH connections
```

Secure File Deletion

Secure File Management

```
# Essential file operations
rm filename.txt            # Delete file
rm -rf directory/          # Remove directory and contents
shred -n 3 sensitive_file.txt  # Securely delete sensitive data
```

Security Best Practices:

- Use `grep` with regular expressions for advanced pattern matching in log analysis
- Combine search commands with pipes for complex security investigations
- Always use secure deletion methods for sensitive data
- Regular cleanup of temporary files to prevent data leakage

Copying and Moving Files

- **cp**: Copies files and directories.
  - `cp source_file destination_directory`: Copies a file.
  - `cp -r source_directory destination_directory`: Recursively copies a directory and its contents.
- **mv**: Moves files and directories.
  - `mv oldname.txt newname.txt`: Renames a file.
  - `mv myfolder /home/user/documents/`: Moves a directory.

Basic Piping for Security Analysis

Command Piping in Cybersecurity

Piping is a fundamental technique in Linux for combining simple commands to create powerful security analysis workflows. The pipe operator enables real-time data processing essential for incident response and threat hunting.

Core Piping Concepts

Piping Fundamentals

Pipe Operator (|): Connects command output to input, creating data processing chains.
Data Flow: Information flows left-to-right through the pipeline with each command processing and transforming data.
Real-time Processing: Essential for live log analysis, system monitoring, and incident response operations.
Command Chaining: Multiple simple commands combine to perform complex security analysis tasks.

Cybersecurity Piping Examples

Basic Piping Examples

```
# Simple piping examples
cat /var/log/auth.log | grep "Failed password"
# Find failed login attempts

ps aux | grep ssh
# Find SSH processes

ls -la | more
# View file listing page by page
```

Basic Piping Uses:

- Combine simple commands for better results
- Filter and search through files
- View long outputs in manageable pieces

Basic Commands for System and Information Management

- **uname**: Displays basic system information.
  - `uname -a`: Shows all basic system information.
  - `uname -r`: Shows the kernel version.
- **df (Disk Free)**: Displays information about available and used disk space.
  - `df -h`: Shows the disk space in human-readable format (MB, GB).
- **du (Disk Usage)**: Displays disk usage by files and directories.
  - `du -h /path/to/directory`: Displays the disk usage of a directory in human-readable format.
- **lsblk (List Block Devices)**: Lists the structure of disks connected to your system, including partitions and space.
  - `lsblk`: Displays block devices.
- **fdisk**: Displays partition information on the hard disk.
  - `fdisk -l`: Lists partition tables.

Managing Processes and System Information

- **ps (Process Status)**: Displays a list of running processes.
  - `ps`: Shows running processes.
  - `ps aux`: Shows more detailed process information.
  - a: Shows processes of all users.
  - u: Shows processes in detailed format (including the user running the process).
  - x: Shows processes not attached to a terminal.
- **top**: Displays real-time information about running processes, including CPU and memory usage.
  - `top`: Shows running processes in real-time. Press q to exit.
- **kill**: Terminates processes using the process ID (PID).
  - `kill [PID]`: Terminates a specific process.
  - `killall [process_name]`: Terminates all processes with the same name.
  - `killall firefox`: Closes all Firefox windows.

Homework

1. Explain the role of inodes in the Linux filesystem. What information do they store, and how are they crucial for file management?

2. Describe the differences between the FAT32, NTFS, EXT4, and APFS filesystems. Include a discussion of their primary use cases and limitations.

3. Using the Linux command line, demonstrate how to create a new directory structure with multiple nested folders and set specific permissions for the new directories.
4. What is the purpose of the grep command in Linux? Provide an example of how you would use it to search for a pattern within multiple text files.
5. Explain how piping works in Linux. Give an example where you combine the output of the cat command with grep to filter file content.