Linux incident response - understanding stat

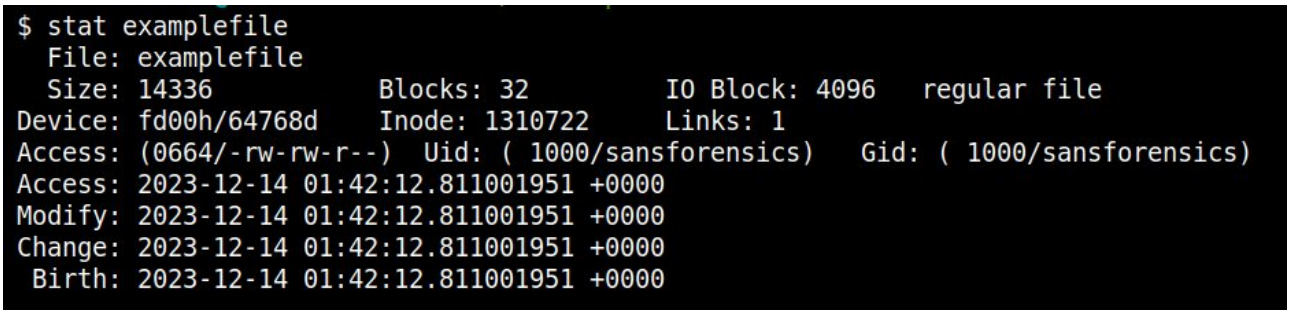
Taz Wake | Halkyn Consulting Ltd | Originally published 2023-12-14

Introduction

In Linux, we can use the stat command to display detailed information about a file or a file system. This tool is useful for system administrators, programmers, and digital forensic analysts as it provides comprehensive metadata about files.

It retrieves data directly from the inode of the file, which is a data structure on the file system that stores metadata about the file. This metadata includes attributes like the size of the file, the number of blocks allocated to it, access permissions (read, write, execute), UID (User ID) and GID (Group ID) of the file owner, the type of file (regular, directory, symlink, etc.), and timestamps (access, modify, and change times).

In this article, we will look at the specifics of the stat command's output explaining the significance of each field. To set the scene, here is an example of the command in use:



Stat command in use

Understanding the output fields

Size

The Size field indicates the total size of the file in bytes. It's a direct measure of the file's content length and does not include metadata stored in the file system.

Blocks

The Blocks field shows the number of filesystem blocks allocated to the file. Each block represents a fixed-size unit of data storage on the disk. The size of these blocks is determined by the file system configuration and can vary.

IO Block

IO Block specifies the size of the I/O block, in bytes, which is used by the file system for filesystem operations. Larger block sizes can improve the efficiency of reading and

writing data, especially for larger files, but may lead to wasted space (sometimes referred to as "*internal fragmentation*") for smaller files.

File Type

This field indicates the type of file, which can encompass several categories: regular files, directories, symbolic links, sockets, FIFOs (named pipes), character devices, and block devices. Regular files are the most common type, containing user data that can range from zero (empty files) to very large sizes. Directories are special files that store information about other files and directories, including their names and inode numbers. Symbolic links, or symlinks, are a type of file that points to another file or directory; they are essentially shortcuts or references to another file's path. Sockets are used for inter-process communication; they facilitate data exchange between different processes running on the same or different machines. FIFOs, also known as named pipes, are special files that serve as conduits for a first-in-first-out data stream, allowing communication between different processes. Character devices are files that allow processes to communicate with hardware in a character-by-character manner, such as keyboards or serial ports. Block devices, on the other hand, provide buffered access to hardware devices, and data is managed in blocks, such as with hard drives or disk partitions.

Device

The Device field shows the identifier of the device (in hexadecimal) on which the file resides. It is useful in identifying the physical or logical storage device.

Inode

Every file in a Unix/Linux system has an inode number, a unique identifier that stores metadata about the file, excluding its name or actual data. The Inode field displays this number, which is crucial for low-level file system operations.

Links

The Links field indicates the number of hard links to the file. A hard link is essentially an additional name for an existing file; multiple links to the same file share the same inode.

Access

Access shows the file's permission settings, indicating which users or groups can read, write, or execute the file. It is displayed in both symbolic (e.g., -rw-r--r--) and octal format (0644).

Uid and Gid

Uid (User ID) and Gid (Group ID) display the numerical ID of the file's owner and the group, respectively. These IDs are used for managing file permissions and access control.

Timestamps

The timestamps provided by stat include Access time (**atime**), Modification time (**mtime**), Change time (**ctime**) and file creation time (**btime**).

- atime is updated when the file's contents are read, although this depends on how the file system is mounted.
- mtime changes when the file's content is modified.
- ctime is altered when the file's metadata changes.
- btime is set when the file is created and *should* remain unchanged throughout the file's lifespan.

On 64-bit systems, these timestamps are accurate to the nanosecond. However, on 32-bit systems, the accuracy is typically limited to the second. This distinction is important in forensic analysis and system monitoring, as the precision of these timestamps can affect how file interactions are interpreted.

About Halkyn Consulting

Halkyn Consulting Ltd is a UK-based security and risk management consultancy providing 24/7 incident response, digital forensics, threat hunting, virtual security leadership (vCISO) and security compliance support to organisations worldwide. We deliver clear, practical, vendor-independent advice based on real-world threats — no hype, no upsell.

More free security resources are available at www.halkynconsulting.co.uk/security-resources.

Web: www.halkynconsulting.co.uk | **Email:** info@halkynconsulting.co.uk | **Tel:** +44 1244 940858