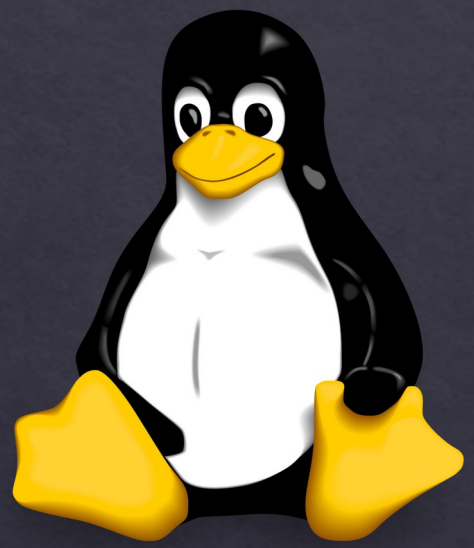


Linux Essentials for Cybersecurity

Text Editors

Text Editors

- They are software applications used for writing and editing plain text files.
- Essential for programming, configuration files, and scripting in Linux.
- Types:
  - Graphical Editors: gedit, VS Code, Sublime Text
  - Command-line Editors: Vim, Emacs, nano

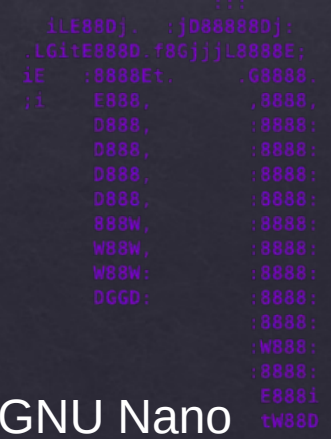
Common Text Editors

Vim

GNU Emacs

Neovim

gedits



GNU Nano

VS Code


Just use Nano
Real
hackers use Vim

Using Vim

- Vi (or Vim – Vi IMproved) has powerful features for editing large files efficiently.
- Available by default on almost all Linux distributions.

Using Vim - Modes

- **Normal Mode** – Default mode for navigation and commands
- **Insert Mode** – For explicitly inserting and modifying text
- **Visual Mode** – Enables selection of text
- **Command Mode** – For saving, exiting, and searching
- Switching modes:
  - Press i to enter Insert Mode.
  - Press Esc to return to Normal Mode.

Using Vim - Text Manipulation

- u – undo last change
- yy – copy a line
- dd – delete a line
- p – paste copied/deleted text
- /text – search for "text"
- n – repeat last search

Using Vim - Find and Replace

- :s/old/new/ – replace first occurrence of "old" with "new" in the current line.
- :s/old/new/g – replace all occurrences in the current line.
- :%s/old/new/g – replace all occurrences in the entire file.
- :%s/old/new/gc – ask for confirmation before replacing each instance.


When you finally exit vim
You know, I'm something of a scientist myself

Using Vim - Saving and Exiting

- :w – save the file.
- :q – quit (only if no unsaved changes).
- :wq or ZZ – save and quit.
- :q! – quit without saving.


RTFM
Read The
F**king
Manual

Task

- You are a cybersecurity analyst assessing a Kali Linux system for potential security risks. Your goal is to gather intelligence about the system and its users to identify any potential security risks.
- Use Linux command-line tools to collect system and user-related information and document your findings.

Task - Resources

- **man** pages
- uname
- uptime
- whoami
- ip -a

- /etc/sudoers
- /etc/passwd
- /etc/group
- lsb_release
- cat

Task - Deliverable

- Document the collected information in a report.
- Highlight any unusual findings (e.g., unknown users with sudo access).

Process Management


THE GATES OF SHELL

Processes

- A process is an instance of a running program.
- Types:
  - **Foreground Process:** Requires user interaction and runs in the terminal.
  - **Background Process:** Runs without user interaction (e.g., system services).

Process Attributes

- **Process ID (PID)**: Unique identifier assigned by the OS.
- **Parent Process ID (PPID)**: ID of the process that created it.
- **User ID (UID) & Group ID (GID)**: Defines ownership and permissions.
- **Priority (Nice Value)**: Determines scheduling priority.
- **Process State**: Running, Sleeping, Stopped, or Zombie.
- **Open File Descriptors**: Files that the process is using.

ps (Process Status)

- Displays a snapshot of the current processes
- Common options:
  - $ ps aux # show all processes.
  - $ ps -ef # show all processes in full format.
  - $ ps -u <username> # show processes of a specific user.
  - $ ps aux | grep <process-name> # search for a running process by name

pgrep (Process Grep)

- Used to search for processes by name.
- Common options:
  - $ pgrep <process_name> → Returns PIDs of matching processes.
  - $ pgrep -u <username> <process_name> → Search for processes owned by a specific user.
  - $ pgrep -l <process_name> → Show process name and PID.

Process States

SIGNAL	DESCRIPTION
R (Running)	Actively executing or ready to run.
S (Sleeping)	Waiting for an event (e.g., I/O).
D (Uninterruptible Sleep)	Waiting for a hardware event (e.g., disk I/O).
T (Stopped)	Suspended, e.g., by Ctrl + Z or SIGSTOP.
Z (Zombie)	Process completed but not cleaned up by parent.

Signals

- A signal in Unix-like operating systems is a software interrupt used to notify a process of an event.

- When a signal is sent to a process, the process can:
  - Handle it using a signal handler function.
  - Ignore it (if allowed).
  - Perform the default action, which is predefined for each signal (e.g., terminate, stop, or continue execution).

How Windows ask a process to terminate

How Linux ask a process to terminate

Signals

SIGNAL	DESCRIPTION
SIGTERM / TERM (15)	Gracefully terminate the process
SIGKILL / KILL (9)	Forcefully terminate the process
SIGSTOP / STOP (19)	Suspend process execution
SIGCONT / CONT (18)	Resume a stopped process
SIGINT / INT (2)	Interrupt process (same as Ctrl + C)

Killing Processes - kill

- Basic Process Termination
  - $ kill <PID> → Sends SIGTERM (default, graceful termination).
  - $ kill -9 <PID> → Sends SIGKILL (force kill).
  - $ kill -l → Lists all available signals.

Killing Processes - pkill

pkill (Process Kill by Name)

- $ pkill <process_name> → Kill all processes matching a name.
- $ pkill -u <username> <process_name> → Kill processes of a specific user
- $ killall <process_name> → Kills all processes matching a name.

Process Control Shortcuts

Shortcut	Signal Sent	Effect
Ctrl + C	SIGINT (2)	Interrupts and terminates the foreground process.
Ctrl + Z	SIGTSTP (20)	Suspends (stops) the foreground process, allowing it to be resumed later with fg or bg.
Ctrl + D	(EOF)	Signals end-of-input in the terminal (useful for closing interactive shells like cat or python).

Process Control Shortcuts

Shortcut	Signal Sent	Effect
Ctrl + S	(XOFF)	Pauses output to the terminal (useful if too much text is scrolling).
Ctrl + Q	(XON)	Resumes output stopped by Ctrl + S.
Ctrl + T	SIGINFO (on macOS/BSD)	Displays process information (such as status and CPU usage).

Jobs

- Job - a task that the shell manages, which consists of one or more processes
- Jobs are identified by Job IDs (JID) and can be in the foreground, background, or suspended states.
- Jobs are shell-level entities, unlike processes which are OS-level entities.
- Allow users to run multiple processes in parallel.
  - Users can suspend, resume, and terminate jobs as needed

Foreground jobs & processes

- actively running and directly interacts with the terminal.
- The user must wait for it to complete before running another command.
- Example:
  - $ nano somefile.txt
  - $ ping google.com > out_file.txt

Background jobs & processes

- runs in the background, allowing the user to continue using the terminal.
- Start a background job using “&”
- Move a job to the background with bg
- Example:
  - $ ping google.com > out_file.txt &

Job Control Commands

- jobs → Lists active jobs with their status.
- Ctrl + Z → Suspends a foreground job.
- $ kill %<JID> → Sends a termination signal to a specific job.
- $ fg %<JID> → Resumes a job in the foreground.
- $ g %<JID> → Resumes a suspended job in the background.
- $ kill %<JID> → Sends SIGTERM (default) to terminate a job.

Example - Job

- $ top # start “top” process foreground
- Ctrl + Z # suspend the process
- $ bg # move the process to the background
- $ jobs # list jobs
- $ fg %1 # bring the “top” process back to the foreground
- $ kill %1 # terminate the job

Thank you