🔐 Linux Command Resolution & File Discovery for Security Operations

---

🎯 Objective

Demonstrate how Linux command-resolution and file-discovery utilities (`which`, `whereis`, and `find`) can be applied to **system security validation, incident response triage, and environment integrity checks**.

🧩 Scenario

In a Linux environment supporting development and production workloads, security analysts and system administrators must:

- Verify that executables resolve to **trusted locations**
- Identify whether required tools are missing or shadowed
- Locate configuration files, logs, and recently modified artifacts
- Rapidly scope systems during troubleshooting or incident response

This project reframes common Linux utilities as **defensive security tools**, not just administrative commands.

🛠️ Tools & Technologies

- Linux (Bash shell)
- `which`
- `whereis`
- `find`
- File metadata (size, modification time, type)
- Output redirection for documentation and review

🔍 Methodology

1️⃣ Executable Resolution Verification (`which`)

- Validated how commands resolve through the system `PATH`
- Confirmed that commonly used binaries execute from expected directories
- Identified behavior when commands are missing or deprecated

Security relevance:
Prevents PATH hijacking, reduces risk of executing unauthorized binaries, and supports incident investigation when command behavior is suspicious.

2️⃣ Binary, Configuration, and Documentation Mapping (`whereis`)

- Located executable binaries, configuration files, and manual pages
- Distinguished between runtime components, configuration scope, and documentation availability
- Assessed whether expected system components were present

Security relevance:
Supports system baselining, hardening verification, and root-cause analysis when tools behave unexpectedly.

3️⃣ Evidence-Driven File Discovery (`find`)

Applied structured file triage techniques:

- **Name & extension filtering** to identify likely evidence (logs, configs, notes)
- **Size-based filtering** to flag unusually large files
- **Time-based filtering** to prioritize recent activity
- **Targeted content review** for rapid situational awareness

Security relevance:
These techniques mirror host-based triage during incident response and forensic scoping.

🧠 Key Findings

- Command resolution depends strictly on `PATH` priority, not intent

- Missing binaries may fail silently, depending on shell behavior
- Combining file metadata (name, size, time) drastically reduces investigation noise
- Redirecting output to files supports documentation and auditability

---

⚠️ Security Impact

This project supports real-world defensive tasks such as:

- SOC alert triage
- Host-based incident response
- Configuration drift detection
- Verification of secure system baselines
- Supporting forensic workflows when combined with hashing and access controls

---

🧪 Limitations & Improvements

Limitations

- Command discovery does not verify binary integrity
- File discovery alone does not indicate malicious intent
- Direct file access may alter metadata in live systems

Improvements

- Integrate hashing (`sha256sum`) for integrity validation
- Use read-only or forensic mounts during investigations
- Correlate findings with SIEM or EDR telemetry
- Automate checks with scheduled scripts for continuous monitoring