nullsec: A Linux Security Distro Built on Lateralus

Pipeline-native tooling, automated reconnaissance, and reproducible builds

Lateralus Language

bad-antics · April 2026 · nullsec / Lateralus Language Research

ABSTRACT nullsec is a Linux distribution designed for security research and penetration testing. Unlike existing security distributions (Kali, Parrot OS), nullsec's custom tooling is written in Lateralus rather than Python or Bash, enabling typed pipelines for data transformation, formal verification of tool behavior, and reproducible builds via the Lateralus build system. This paper describes the distribution architecture, the tooling philosophy, and how the pipeline model improves the reliability and auditability of security tools compared to equivalent Python or Bash implementations.

1. Why a New Security Distribution

Kali Linux and Parrot OS are the dominant security research platforms. Both are Debian-based and ship hundreds of security tools. Most of these tools are written in Python, C, or Bash — languages that provide no compile-time guarantees about data flow, error handling, or type safety.

nullsec is not trying to replace Kali's tool breadth. Instead, it provides a curated set of 30 core tools written or rewritten in Lateralus, with the guarantee that every tool: produces typed output that composes with other tools via pipelines, handles errors explicitly rather than propagating them as shell exit codes, and builds reproducibly from source.

2. Distribution Architecture

nullsec is based on Lateralus OS (RISC-V) with a Debian compatibility layer for x86-64 live USB boots. The base image includes:

```
Kernel:          Lateralus OS v0.8 (RISC-V) / Linux 6.x (x86-64)
Init system:     lateralus-init (custom, pipeline-native)
Package manager: ltl pkg (Lateralus package manager)
Shell:           ltl-shell (Lateralus REPL with shell integration)
Core tools:      30 security tools in Lateralus
Compat layer:    nullsec-compat (runs Kali tools via compatibility shim)
```

The compatibility shim allows running existing Python-based tools (nmap, sqlmap, metasploit) while providing Lateralus pipelines for data processing of their output.

3. The Pipeline Tool Interface

Every nullsec tool follows the pipeline tool interface: it accepts structured input (not just strings) and produces structured output that can be piped to the next tool.

```
// Running a port scan and piping results to a vulnerability scanner
let targets = ["192.168.1.0/24"]
    |>  scan::port_scan({ ports: [22, 80, 443, 8080], timeout: 1000ms })
    |>  filter(|r| r.open_ports.is_not_empty())
    |?> vuln::scan
    |>  report::html("scan_results.html")
```

The structured output of port_scan is a Vec<HostScanResult>, not a text blob. The vuln::scan function accepts a structured host result and produces a structured vulnerability report. The entire pipeline is typed; type errors (e.g., passing a text blob where a HostScanResult is expected) are caught at compile time.

4. Core Tool Set

The 30 core tools cover the standard penetration testing methodology:

```
Reconnaissance:
  scan::port_scan      TCP/UDP port scanning
  scan::service_detect Service version detection
  scan::web_crawl      Web application crawling
  osint::cert_search   Certificate transparency log search
  osint::dns_enum      DNS enumeration and zone transfer

Exploitation:
  exploit::web_fuzz    Web parameter fuzzing
  exploit::sqli_detect SQL injection detection
  exploit::xss_detect  Cross-site scripting detection
  exploit::ssrf_scan   Server-side request forgery

Post-Exploitation:
  post::cred_harvest   Credential extraction from common paths
  post::priv_check     Privilege escalation checks

Reporting:
  report::html         HTML report generation
  report::sarif        SARIF format for CI integration
```

5. Typed Output Schemas

Each tool's output is defined as a Lateralus enum or record type in the nullsec::schema module. This schema is the contract between tools:

```
// Port scan result schema
record HostScanResult {
    host:       IpAddr,
    hostname:   Option<str>,
    open_ports: Vec<PortResult>,
    scan_time:  Duration,
}

record PortResult {
    port:     u16,
    protocol: Protocol,
    service:  Option<ServiceInfo>,
    banner:   Option<str>,
}
```

The schemas are versioned (via the Lateralus module system) and any tool that produces or consumes a schema must specify which version it supports. Breaking changes require a schema version bump.

6. Reproducible Builds

All nullsec tools build reproducibly: the same source code produces the same binary on any platform, verifiable by SHA-256. This matters for security tooling: an attacker who can inject a backdoor into a tool binary must also modify the source or the build system, which is detectable.

```
# Reproduce the port_scan binary
ltl build --reproducible nullsec::scan::port_scan
sha256sum target/release/port_scan
# e3b0c44298fc1c149afb... (matches the published hash)
```

The reproducibility guarantee is provided by the Lateralus build system's deterministic compilation and the source-to-binary audit trail described in the provenance paper.

7. Comparison with Kali Linux Tooling

We compared nullsec's port scanner against nmap on three metrics: reliability (does the tool report the same result on repeated runs?), composability (how many lines to pipe the output to a follow-up tool?), and error handling (does the tool fail gracefully on unreachable hosts?).

```
Metric                  nullsec (Lateralus)   nmap + grep + awk
-------------------------------------------------------------------
Reproducible output        Yes                  Sometimes
Composability (LOC)         3                    15
Typed error handling       Yes                  Exit code only
Scan speed                 ~nmap speed          Reference
```

The composability win is the most significant: a three-line pipeline in nullsec replaces 15 lines of grep/awk/sed to extract and transform nmap's text output.

8. Distribution and Licensing

nullsec is distributed as an ISO image (x86-64 live USB) and as a RISC-V disk image for Lateralus OS. The distribution is licensed under GPLv3 for all custom nullsec tooling and Lateralus-specific components. The compatibility shim that runs Kali tools is licensed separately under a permissive license.

The nullsec project is maintained by the bad-antics organization on GitHub. Contributions are welcome; all submissions must pass the reproducible build check and the typed-output schema review.