Building a Pentest Distribution

ISO build system, tool packaging, and reproducible live environment for nullsec

Lateralus Language

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

ABSTRACT nullsec is distributed as a bootable ISO image. This paper describes the ISO build system: how tool binaries are built reproducibly, how the live filesystem is assembled, how the boot process is configured for both BIOS and UEFI systems, and how the ISO is tested before release. We also describe the automated release pipeline that runs on every push to the main branch.

1. Build Architecture

The nullsec ISO build is a three-stage pipeline:

- **Stage 1 — Tool build**: compile all 30 nullsec tools from source using the Lateralus build system, producing statically-linked binaries with embedded SHA-256 hashes.
- **Stage 2 — Filesystem assembly**: copy tool binaries, configuration files, and the base system into a squashfs image (the live root filesystem).
- **Stage 3 — ISO assembly**: combine the squashfs image with the bootloader (GRUB for BIOS, systemd-boot for UEFI) into a hybrid ISO image bootable from both modes.

```
# Build command (reproducible)
ltl build-distro --config nullsec.toml --output nullsec-YYYYMMDD.iso
sha256sum nullsec-YYYYMMDD.iso  # deterministic hash
```

2. Tool Build: Static Linking and Reproducibility

All nullsec tools are statically linked — they embed all required libraries and have no runtime dependencies. This allows the tools to run on any Linux kernel without worrying about library versions.

```
# Build a single tool statically
ltl build --target x86_64-linux-musl --release nullsec::scan::port_scan
# musl libc for static linking; glibc cannot be fully statically linked
```

Reproducibility is achieved by: pinning all dependencies to specific versions in the Lateralus workspace manifest, using a hermetic build environment (Docker container with a fixed base image SHA-256), and stripping timestamps from the binary output. The resulting binary is bit-for-bit identical across builds.

3. Filesystem Layout

The nullsec live filesystem follows the FHS with nullsec-specific directories for tool data and engagement storage:

```
/usr/bin/              # nullsec tool binaries (and standard utilities)
/usr/lib/nullsec/      # tool support files (wordlists, signatures)
/opt/nullsec/          # compatibility shim for Kali tools
/etc/nullsec/          # tool configuration defaults
/home/null/            # default user home directory
/mnt/engagement/       # mount point for engagement storage volume
```

The engagement storage volume is a separate encrypted partition that is not part of the squashfs image. It is created at first boot and persists across reboots, allowing engagement data to survive session restarts.

4. Bootloader Configuration

The ISO is a hybrid image that boots in both BIOS (via GRUB) and UEFI (via systemd-boot) modes. The bootloader presents a minimal menu with two options: live boot (default) and forensic mode (boot without writing to attached storage).

```
# GRUB menu (grub.cfg)
menuentry "nullsec live" {
    linux /boot/vmlinuz boot=live quiet splash
    initrd /boot/initrd.img
}
menuentry "nullsec forensic mode" {
    linux /boot/vmlinuz boot=live forensic noauto
    initrd /boot/initrd.img
}
```

Forensic mode activates the noauto kernel parameter which prevents automounting of attached storage. All block devices are mounted read-only or not at all, preserving forensic integrity.

5. The Live Boot Process

At boot, the initrd unpacks the squashfs image and sets up overlayfs to allow writes to the live filesystem without modifying the squashfs. Changes are stored in RAM and lost on reboot unless explicitly saved to the engagement volume.

The overlay layers are:

```
Lower layer: squashfs (read-only, the base system)
Upper layer: tmpfs (read-write, in RAM)
Merged view: overlayfs (appears as one writable filesystem)
```

The engagement volume, if present, is mounted at /mnt/engagement after the overlay is set up. Tools that save output use this path by default.

6. Release Testing

Every release candidate is tested with an automated suite running in QEMU before the ISO is published:

```
# Automated release tests (CI pipeline)
1. Boot in QEMU (BIOS mode) — verify login prompt
2. Boot in QEMU (UEFI mode) — verify login prompt
3. Run each tool with --version — verify all 30 respond
4. Run port_scan against a test target — verify typed output
5. Run the login pipeline — verify audit log is written
6. Boot in forensic mode — verify no automount
```

The test suite runs in approximately 8 minutes on the CI server. Release gating: all 6 tests must pass and the ISO SHA-256 must match the build system's expected hash.

7. Automated Release Pipeline

The GitHub Actions workflow builds and publishes nullsec releases:

```
on:
  push:
    tags:
      - 'v*'
```

```
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Build ISO
        run: ltl build-distro --config nullsec.toml
      - name: Test in QEMU
        run: ./tests/run_qemu_suite.sh
      - name: Publish to GitHub Releases
        run: gh release create ${{ github.ref_name }} nullsec-*.iso
```

8. Verification by Users

Users who download the nullsec ISO should verify its SHA-256 against the published hash before booting:

```
# Download ISO and signature
wget https://releases.nullsec.dev/latest/nullsec.iso
wget https://releases.nullsec.dev/latest/SHA256SUMS
wget https://releases.nullsec.dev/latest/SHA256SUMS.sig

# Verify signature
gpg --verify SHA256SUMS.sig SHA256SUMS

# Verify ISO hash
sha256sum -c SHA256SUMS
```

The SHA256SUMS file is signed with the nullsec release key (fingerprint published on the nullsec website and in the repository). Verifying the signature ensures the file came from the nullsec team and not a compromised mirror.