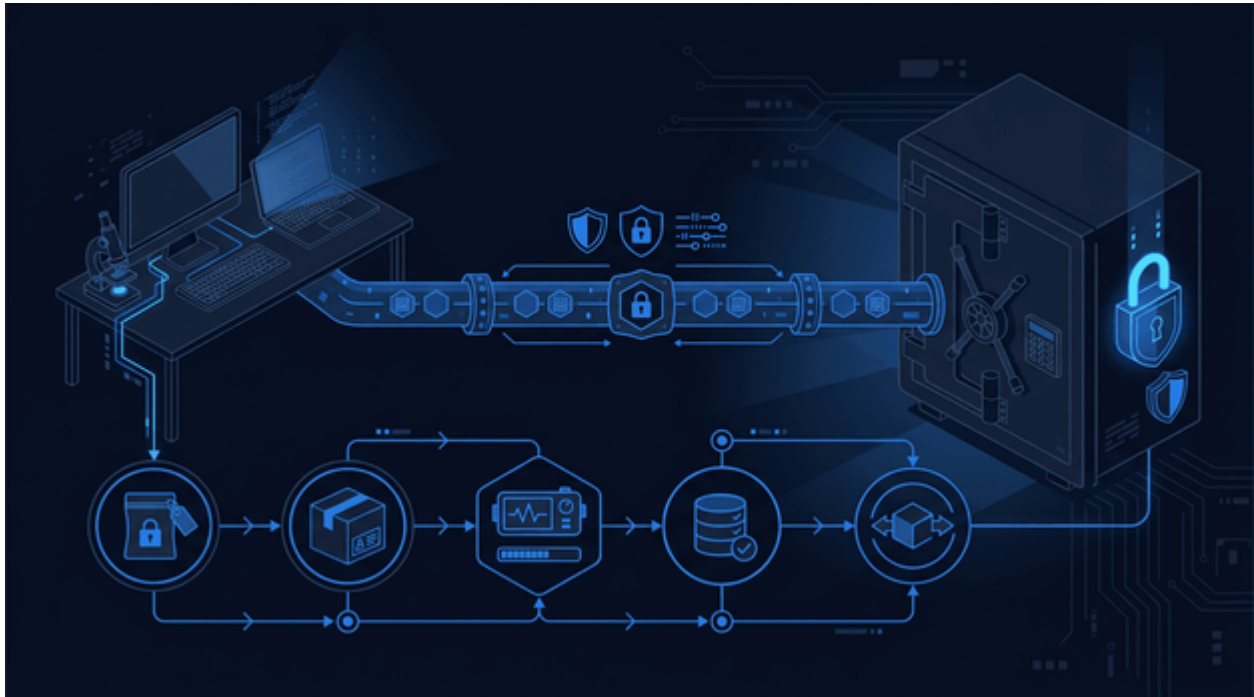


Building a Forensic Evidence Pipeline: Workstation to Evidence Locker

Tue, Jul 7, 9:00 AM EDT · <https://scottslab.io/posts/forensic-evidence-pipeline-workstation-to-locker>

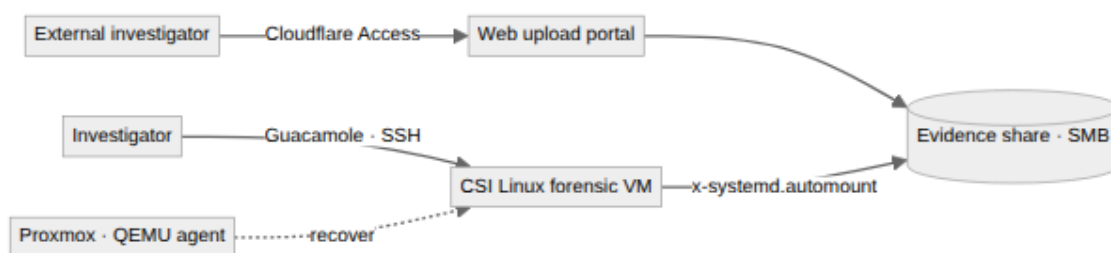


TL;DR

The evidence share came back empty after every reboot, which turned out to be a boot race rather than anything to do with forensics. `x-systemd.automount` with `nofail` and an explicit `vers=3.0` fixed it and cut boot time roughly in half. The rest of the work was getting reliably back onto the box: an `sshd` drop-in, an IP collision with an identity container, and a Cloudflare Access portal so outside investigators can hand over files without touching the network.

Every time the forensic workstation rebooted, the evidence folder came back empty. Not an error, not a prompt, just an empty directory where a mounted SMB share used to be. The mount was firing at boot before the network was up, failing silently, and staying failed until I noticed and remounted it by hand. Fine on a lazy Tuesday. Not fine at the start of an investigation, when the first thing an investigator sees is a workstation that can't reach the evidence. That's the moment people start copying case files to the local disk and breaking chain of custody to save five minutes.

That's the whole point of an evidence pipeline: it has to be invisible. Open the folder, the case files are there. Close the laptop, come back tomorrow, they're still there. No ceremony, no "did you try remounting the share." Getting to that meant fixing four things that had nothing to do with forensics and everything to do with the workstation being reachable and reliable.



The mount that wouldn't stay

The empty-folder problem was a boot race, and the fix is a mount option I now reach for by default. Instead of a hard CIFS mount at boot, `x-systemd.automount` registers a lightweight automount point immediately and defers the real mount until something actually touches the directory:

```
//<EVIDENCE_HOST>/evidence /mnt/evidence cifs
credentials=/root/.smbcreds,uid=1000,gid=1000,vers=3.0,x-
systemd.automount,x-systemd.mount-timeout=30,nofail 0 0
```

`nofail` keeps a missing share from stalling the boot, `mount-timeout=30` gives up instead of hanging, and (the part I missed the first time) `NetworkManager-wait-online.service` actually has to be enabled, or nothing waits for the network at all. The explicit `vers=3.0` cleared a separate silent failure where the server refused version negotiation. Net effect: the share mounts on first access every time, and boot dropped from about ninety seconds to forty-five, because it's no longer sitting through a mount timeout that was always going to fail.

Getting back in

Before I could fix any of that, I had to get onto the box, and CSI Linux ships with password SSH disabled. Rather than edit the shipped `sshd_config`, a drop-in does it without touching the original file:

```
cat > /etc/ssh/sshd_config.d/99-local.conf << 'EOF'
PasswordAuthentication yes
PermitRootLogin prohibit-password
EOF
systemctl restart sshd
```

Then the workstation started answering on the wrong address: .131 instead of the .112 I expected. Something else had claimed .112: an identity-provider container with a static config living entirely outside DHCP. Two systems, one address, intermittent failures that are miserable to chase. I moved the container to .117 and pinned the workstation to .112 with a DHCP reservation on the firewall, so the binding lives in one place instead of scattered across VM configs. The lesson I keep relearning: before you assign an IP, check every source that can hand one out (leases, reservations, and static container configs), not just the DHCP table.

With SSH back and the address stable, the rest of the management chain fell into place: the QEMU guest agent reporting to Proxmox, and Guacamole serving the full desktop in a browser for the tools that need one: Autopsy, Wireshark, Maltego. Three independent ways back onto the workstation, which matters when the thing you're recovering is the machine you do recovery from. One caveat worth stating plainly: you have to enable the QEMU agent from inside the VM first. You can't remotely bootstrap an agent that isn't running.

Evidence in, findings out

External investigators don't get network access; they get a web upload portal behind Cloudflare Access. Email-based identity verification, no VPN, no inbound firewall ports opened: the tunnel dials out. Files they upload land in the same evidence share the workstation automounts, so there's one path in and one source of truth, and the workstation itself never has to be exposed to make that work.

What I'd add next

The pipeline is reliable now, but it isn't yet self-proving. The next piece is chain of custody: SHA-256 every file on upload and log access with timestamps, because evidence that can't demonstrate its own integrity is just data. After that, a Wazuh rule to alert on mount failures, so I hear about a broken share from the SIEM instead of from an investigator, and an Ansible playbook to rebuild the whole workstation from scratch in minutes instead of hours.

Automount was the real unlock, though. It's a patient mount option: it waits until you actually need the files, then connects quietly when you're not looking. That's exactly the behavior I want from forensic infrastructure. Do the job, don't make a fuss about it.

Building your own DFIR lab? The persistent mount problem bites everyone eventually. Hope this saves you a few hours of boot-timeout debugging.