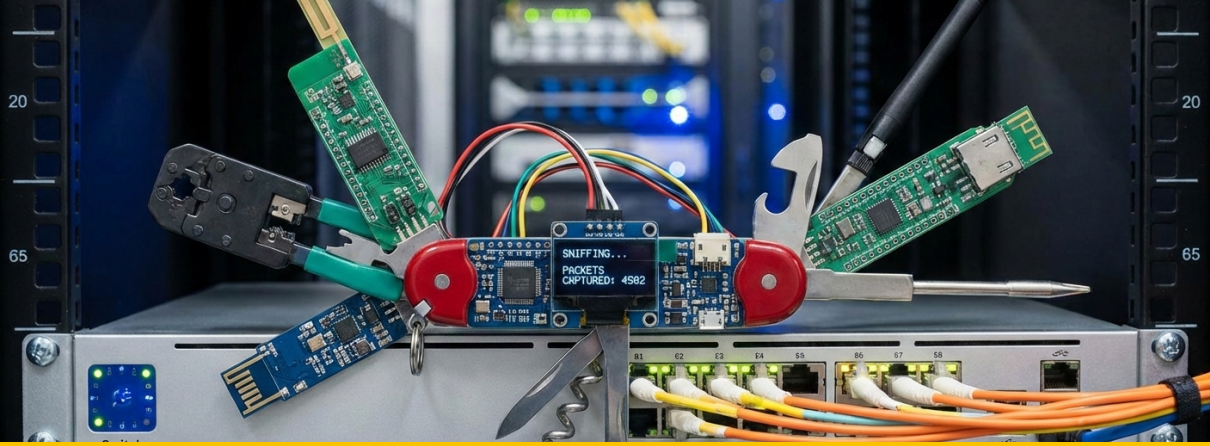


Swiss Army Knife Network Sniffer (NSAK)

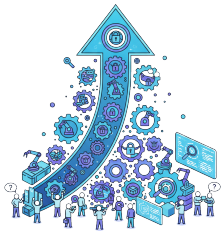
Version 0.4

January 14, 2026

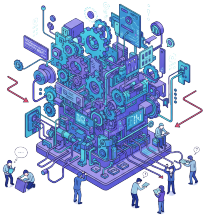
Lukas von Allmen (vonal3) and Frank Gauss (gausf1) | Bern University of Applied Sciences

Introduction

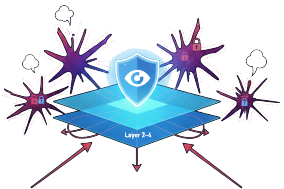
Motivation



Automation & AI

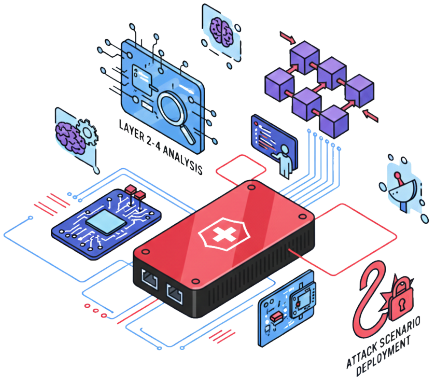


Complex frameworks with configuration overhead



Layer 2 - 4 Network Testing

Network Swiss Army Knife



- Modularity
- Configuration Friendly
- Portable and Isolated
- Containerized

Network Swiss Army Knife (NSAK)

Design and Architecture

NSAK Design and Architecture Overview

Resources and Concepts

Resources:

Devices	Physical or virtual machine, which can be provisioned as a NSAK device
Environments	Network topology incl. infrastructure components, servers, clients and services
Scenarios	Designed to be run in environements, consisting of a sequence of drills
Drills	Reusable building blocks for configuration, reconnaissance or exploitation

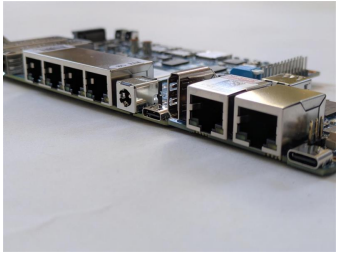
Concepts:

Operator	A single IT-specialist, a red, blue, or purple team
Operation	Deployment of NSAK in a real network or lab

Hardware Evaluation

Hardware Selection

Banana Pi R4 Pro



- RAM: 8 GB
- CPU: Quad-Core ARM
- Ports: 4×1 2×10 GbE
- Wi-Fi: Optional with a module

NanoPi R76S

- RAM: 16 GB
- CPU: Octa-Core ARM
- Ports: 2×2.5 GbE
- Wi-Fi: Optional with a module

Implementation

NSAK Framework Implementation

Components, stack, dependencies

Framework (Monorepo)

- **Core**: framework logic
- **CLI**: user interaction
- **Resource Library**: devices, environments, scenarios, drills

System Dependencies

- Python Toolchain: `python3`, `uv`
- OCI Containers: `podman`, `podman-compose`
- Net Control: `iptables/nftables`

Python Dependencies

- Quality: `ruff`, `mypy`, `pre-commit`
- Testing: `pytest`
- Pre-commit: rule enforcement
- Config: `pyyaml`
- Networking: `scapy`
- CLI: `click`

MITM Scenario Implementation

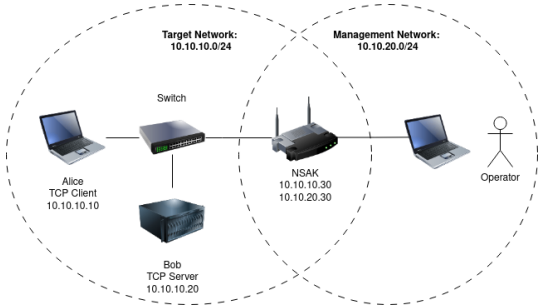
ARP spoofing & transparent TCP proxy

Drills

1. Discover hosts *scan target network*
2. ARP spoof *discover hosts, poison ARP tables*
3. Transparent TCP proxy
   - Client+server sockets for interception
   - Redirect traffic via `iptables`/`nftables`
   - Read & modify forwarded packets

Environment

- Simulation: `podman-compose`
- Physical lab: 2×Raspberry Pi & switch

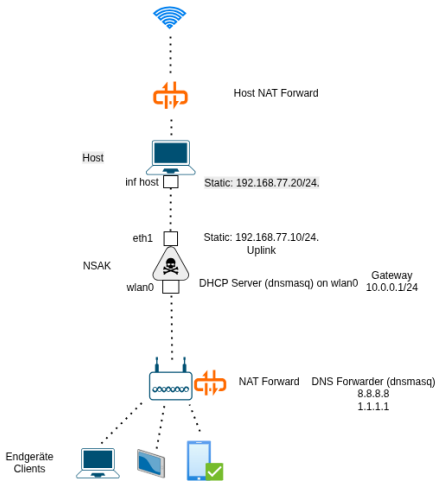


Simple TCP Client–Server Lab

Demo video

Rogue AP – Drill Order

Controlled Wi-Fi Scenario



Drills

- **hostapd**
  Start Rogue AP
- **network setup**
  IP, interfaces, routing
- **dnsmasq**
  DHCP + DNS
- **forwarding**
  NAT / IP forwarding
- **tshark**
  Traffic capture
- **future drills**
  Captive portal, deauth

Evaluation and Discussion

Future Work and Conclusion

Key Insights and Takeaways

Future work

- Interfaces
  - REST API for frontend and system integration
  - GUI for improved UX and accessibility
- Framework maturity
  - Configuration and cleanup management
  - Increased test coverage
- Security
  - Holistic security concept
  - Hardened execution and isolation model

Conclusion

- Feasibility of a **Network Swiss Army Knife** was demonstrated on real world or simulated networks
- Modular drills enable flexible and realistic scenarios
- Automation is challenging **due to environmental assumptions**

Access Point Demo: Can you spot the rogue AP?

Open questions?