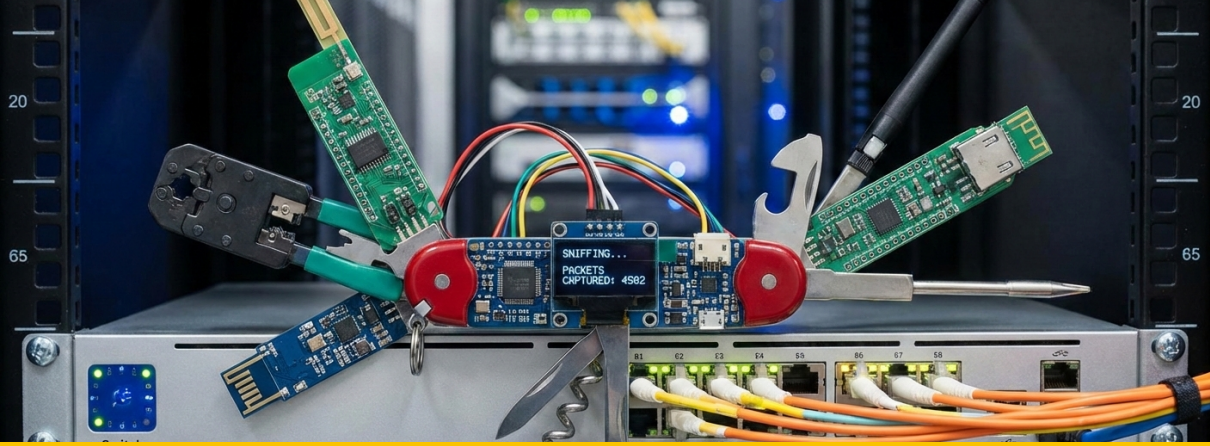


Swiss Army Knife Network Sniffer (NSAK)

Version 0.4

January 11, 2026

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

Introduction

Motivation

Subtitle

- **Cyberangriffe nehmen zu** Angriffsbarrieren sinken, Automatisierung steigt
- **Sicherheit erfordert Sichtbarkeit** Netzwerkangriffe sind oft nur auf Layer 2–4 erkennbar
- **Bestehende Frameworks sind komplex** Hoher Konfigurations- und Betriebsaufwand
- **Unser Ansatz (PoC)** Modularer, kontrollierter Network-Sniffer für Angriffsszenarien

Eher alles als Bilder (Franky)

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, reconessance or expoitation

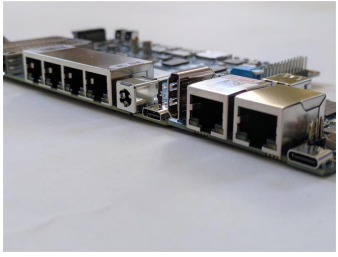
Concepts:

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



- RAM: 8 GB
- CPU: Quad-Core ARM
- Ports: 4× 1 GbE + 2z 10GB
- Wi-Fi: Optional

NanoPi R76S

- RAM: 16 GB
- CPU: RK3588S
- Ports: 2× 2.5 GbE
- Wi-Fi: Optional

Implementation

NSAK Framework Implementation

Components, technology stack and dependencies

Framework Components (Monorepo):

- Core: Framework implementation
- CLI: Uses the core to enable user interaction
- Resource Library: Contains devices, environments, scenarios and drills

System dependencies:

- Language and tooling: python3, uv
- OCI containers: podman, podman-compose
- Networking: iptables, nftables

Python dependencies:

- Linter and formatter: ruff
- Type checker: mypy
- Testing framework: pytest
- pre-commit: Package to enforce code quality tools for each commit
- pyyaml: Library for loading, validating and reading yaml files
- scapy: Library for red team operations
- click: Library for building CLIs

MITM Scenario Implementation

ARP-spoofing / Transparent TCP Proxy

Drills:

- Discover Hosts: Scans the target network
- ARP Spoof: Spoofs the MAC Addresses of discovered hosts
- Transparent TCP Proxy:
  - Creates a TCP client and server
  - Redirects traffic with IPTables / NFTables
  - Reads and modifies intercepted packages

Environment:

- Simulation with podman/docker compose
- Instructions for building a lab with two Raspberry PIs and a switch

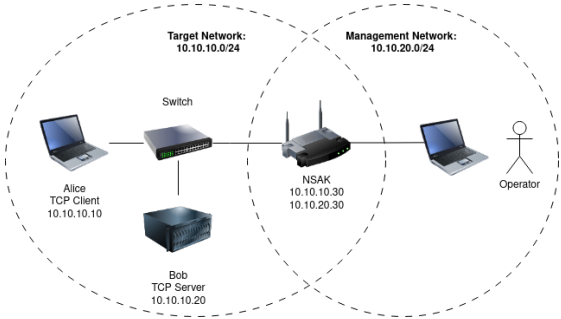


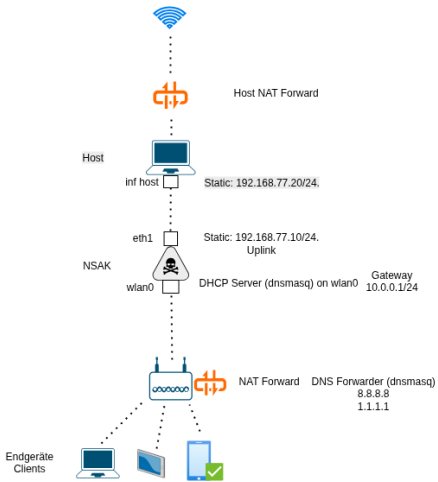
Figure: Simple TCP client - server environment

▸ Demo Video

(Lücku)

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

- REST API: Integration of frontends or systems
- GUI: Better UX and lower entry barrier
- Redesign cleanup management
- Implement configuration management
- Test coverage: Correctness and maintainability
- Security Concept: NSAK must be secure

Conclusion

We could show the feasibility of a PoC realization of a Network Swiss Army Knife (NSAK) and that the modular drills can be combined to effectively build scenarios, which can be executed on constraint hardware in simulated or real-world network environments.

The key difficulty is that the automation of scenarios requires a lot of assumptions about the target environment, which potentially renders them very brittle.

Access Point Demo: Can you spot the rogue AP?

Open questions?