Project 2

Swiss Army Knife Network Sniffer

Course of study	Bachelor of Science in Computer Science
Author	Frank Gauss (gausf1) and Lukas von Allmen (vonal3)
Advisor	Wenger Hansjürg

Version 1.0 of November 3, 2025

▶ Technik und Informatik

Abstract

One-paragraph summary of the entire study – typically no more than 250 words in length (and in many cases it is well shorter than that), the Abstract provides an overview of the study.

Contents

1 First Thesis Chapter

1.1 Introduction

What is the topic and why is it worth studying? – the first major section of text in the paper, the Introduction commonly describes the topic under investigation, summarizes or discusses relevant prior research (for related details, please see the Writing Literature Reviews section of this website), identifies unresolved issues that the current research will address, and provides an overview of the research that is to be described in greater detail in the sections to follow.

1.2 Network Environments

Each environment represents a practical setup in which the Network Swiss Army Knife (nsak) can be deployed for traffic analysis, performance testing, or security evaluation.

1.2.1 Category I: Basic / Local

E 1: Point-to-Point

Diagram: Laptop ↔ nsak ↔ Router

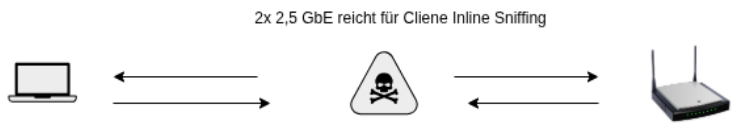

Figure 1.1

Description: Direct inline bridge between client and router. Used for basic LAN capturing, latency, and throughput testing.

E 2: Home Network

Diagram: Laptop, Smart Devices, Printer ↔ nsak (inline) ↔ Router

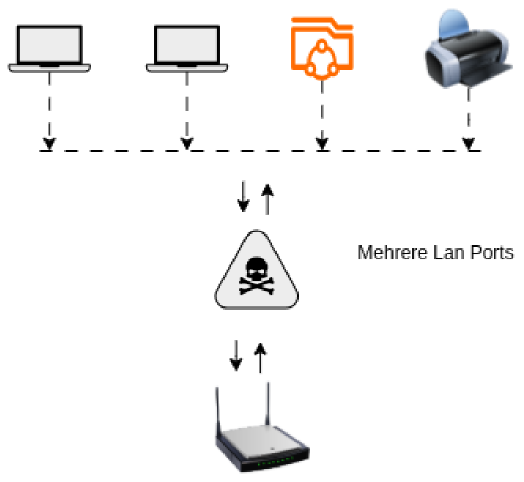

Figure 1.2

Description: Captures typical home traffic. The WLAN interface is required for Wi-Fi analysis. Useful for IoT discovery and local broadcast observation.

E 3: Business Network

Diagram: Devices ↔ Switch ↔ nsak (inline) ↔ Server / Router

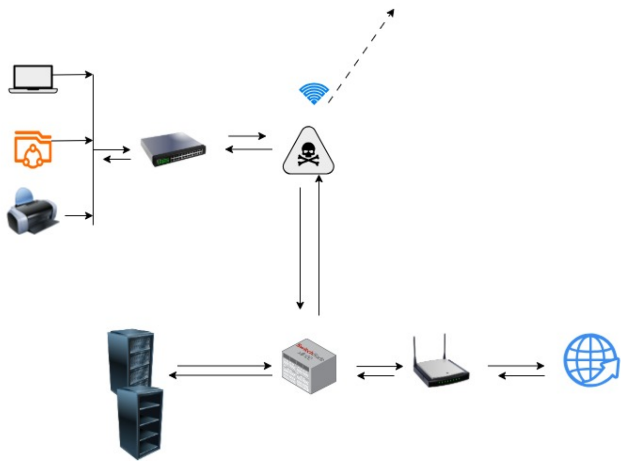

Figure 1.3

Description: Represents a small office LAN. nsak placed at uplink or server edge to monitor internal traffic, VLANs, and broadcast domains.

1.2.2 Category II: Mobile / Wireless

E 4: Access Point Mode

Diagram: Wi-Fi Device ↔ AP-nsak ↔ Internet

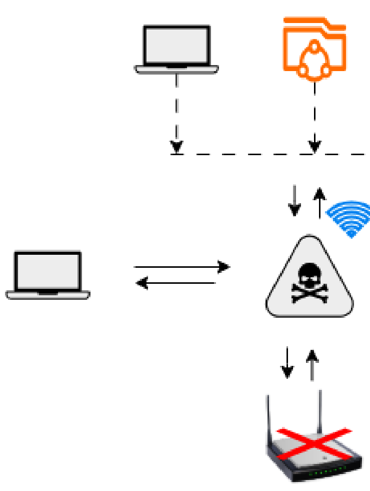

Figure 1.4

Description: nsak acts as Wi-Fi AP, providing connectivity and packet capture. Captures management and data frames for WLAN analysis.

E 5: Mobile Hotspot

Diagram: Smartphone ↔ nsak ↔ LTE/5G
Description: Mobile tethering or hotspot scenario. Focus on NAT behavior, encryption overhead, and power constraints.

1.2.3 Category III: Secure / Advanced

E 6: Data Center / Server Rack

Diagram: Servers ↔ Switch ↔ Firewall ↔ nsak
Description: High-performance setup for 2.5G–10G traffic capture. Focus on throughput, buffering, and VLAN-tagged traffic.

E 7: VPN Gateway

Diagram: Router ↔ nsak (VPN Endpoint) ↔ Remote Peer
Description: nsak configured as WireGuard gateway. Measures encrypted vs. unencrypted traffic, tunnel stability, and CPU load.

1.2.4 Category IV: IoT / Special Purpose

E 8: VLAN-Segmented Enterprise Network

Diagram: Switch + Router + Multiple VLANs ↔ nsak
Description: Used to verify VLAN isolation and detect inter-segment leaks or misconfigurations.

E 9: IoT Sensor Network

Diagram: IoT Devices ↔ Local Gateway (Broadcast) ↔ nsak
Description: Passive capture of local IoT or broadcast-based communication. Identifies timing, protocol use, and unsecured traffic.

1.2.5 Category V: Virtual / Simulation

E 10: Attack Simulation Network

Diagram: Virtual Attacker ↔ Target VM ↔ nsak **Description:** Virtual lab for testing detection systems, malware traffic, and replay scenarios.

E 11: Remote Management Environment

Diagram: Admin ↔ VPN/SSH ↔ nsak (headless) **Description:** Remote-controlled nsak for automated capture and monitoring in unattended operation.

E 12: Dual-Sniffer Setup

Diagram: nsak-A ∥ nsak-B (same link) **Description:** Two synchronized devices capture the same traffic path. Used for timestamp comparison and hardware validation.

1.3 Software Modules

We define a software module as a sequence of actions with a predefined goal. Actions can be but are not limited to the execution of scripts, cli tools or steps in a program, but also human interaction. Such an action may appear in multiple software modules, with different configurations and in interplay with other actions.

We generally differentiate between two software module types:

- Active Software Module: Does actively intervene with other devices or alter data. Can be an active attack, but also testing for behavior of another device, depending on the scenario.
- Passive Software Module: Does not actively intervene in communication and does not alter data. Can be a passive attack, but also monitoring or analysis, depending on the scenario.

To group similar software modules together, we defined the following categories:

1. Network Traffic Collection
2. Network Mapping and Port Mapping
3.annotateMiddle (MITM)
4. Exploits
5. Brute Force
6. Denial of Service (DoS)
7. Physical (TBD?)

1.3.1 Category 1: Network Traffic Collection

In this category are software modules to collect network traffic with the help of network sniffers like WireShark/TShark or tcpdump.

Module 1.1: Live Network Traffic Monitoring

The idea of this module is to allow an operator to use WireShark and connect a remote interface to it, which will then show all packets sent and received on nsack. The operator can then add filters and analyze the traffic in real time. This module requires a live connection from a remote device to nsak, in a real world attack scenario the attacker must be nearby for WI-FI access or could be identified via the cellular connection.

- Type: Passive
- Goal: Monitor live network traffic

Action sequence:

1. nsak: Enable remote access via Wi-Fi or cellular connection.
2. nsak: Enable layer 2 bridge between interfaces.
3. nsak: Start tshark, listening on all interfaces, no filters.
4. Remote Device: Start WireShark and connect to a remote interface.
5. Remote Device: Analyze live network traffic, apply filters

Module 1.2: Network Traffic Collection (Local)

This module requires physical access to nsak to extract the collected network traffic, which might be challenging or dangerous in a real-world attack scenario. The amount of data which can be extracted this way depends heavily on the amount of network traffic, how specific the filters are set, and the disk space available.

- Type: Passive
- Goal: Collect network traffic

Action sequence:

1. nsak: Enable layer 2 bridge between interfaces.
2. nsak: Start tshark, listening on all interfaces, set filters, write to a local file.

Module 1.3: Network Traffic Collection (Remote)

This module requires a live connection from a remote device to nsak, in a real world attack scenario the attacker must be nearby for WI-FI access or could be identified via the cellular connection.

- Type: Passive
- Goal: Collect http network traffic

Action sequence:

1. nsak: Enable remote access via Wi-Fi or cellular connection.
2. nsak: Enable layer 2 bridge between interfaces.
3. nsak: Start tshark, listening on all interfaces, set filters, write to a remote file.

1.3.2 Category 2: Network Mapping and Port Mapping

This category is concerned about mapping out the environment nsak is currently located in.

Module 2.1: Network Discovery

- Type: Active
- Goal: Discover Network Participants

Action sequence:

1. nsak: Run arp-scan, on interface in a specific subnet

Module 2.2: Host Service/Version Detection

- Type: Active
- Goal: Detect which OS and/or services are participating in the network

Action sequence:

1. nsak: Run nmap, against a whole subnet or a specific host

1.3.3 Category 3: ManintheMiddle (MITM)

Modules in this category will try to convince other devices or services in the network that nsak is a legitimate participant.

Module 3.1 SSL/TLS MITM

This module tries to intercept SSL/TLS handshakes and convince Alice and Bob that nsak is the respective counterpart.

- Type: Active
- Goal: Intercept encrypted traffic

Action sequence:

1. nsak: Execute SSL/TLS MITM attack (TBD)

Module 3.2 IP Spoofing

- Type: Active
- Goal: Traffic interception

Action sequence:

1. nsak: Set nsaks IP to the one of another device

Module 3.3: WLAN SSID Spoofing

- Type: Active
- Goal: Traffic interception

Action sequence:

1. nsak: Set nsaks WLAN SSID to the one of an actual WI-FI access point.

1.3.4 Category 4: Exploits

This category contains modules which execute known exploits.

Module 4.1: Join Network via WPS

- Type: Active
- Goal: Wireless Network Access (intrusion)

Action sequence:

1. nsak: TBD

1.3.5 Category 5: Brute force

With this module we try to gain access to a device or network with brute force.

Module 5.1: SSH Login

- Type: Active
- Goal: Server Remote Access

Action sequence:

1. nsak: Load dictionary for the current locale or better data set if available
2. nsak: Brute Force SSH authentication

Module 5.2: WLAN Login

- Type: Active
- Goal: Wireless Network Access (intrusion)

Action sequence:

1. nsak: Load dictionary for the current locale or better data set if available

2. nsak: Brute Force WI-FI authentication

1.3.6 Category 6: Denial of Service (DoS)

Module 6.1: Classic Denial of Service

- Type: Active
- Goal: DoS

Action sequence:

1. nsak: Send as many requests as possible until the service is not available anymore

1.4 Scenarios

A scenario is described as a combination of an environment, with a defined location of the sniffer and one or multiple software modules.

2 Hardware Selection

2.1 Requirements

The following requirements were defined for the hardware platform used in this project:

- At least two native Ethernet interfaces for inline packet sniffing
- Support for 2.5 GbE or higher
- Onboard Wi-Fi with access point (AP) and monitor mode support
- Low power consumption suitable for 24/7 operation
- Compact form factor for laboratory and prototype setups
- Strong community and software support
- Affordable cost (below 150 CHF)

2.2 Evaluated Boards

Several boards were considered as potential variants. Their main specifications relevant to the project are listed in Table 2.1.

Table 2.1: Comparison of Board Variants

Board	SoC / CPU	RAM / Storage	Ethernet Ports	Power (typ.)	Wireless (on-board)
Banana Pi R3 Mini	MT7986A, Quad-core ARM Cortex-A53 @ 1.3 GHz	2 GB DDR4, 8 GB eMMC, microSD	2 × 2.5 GbE	5–7 W	MT7976C, Wi-Fi 6 (AP/Client/Monitor)
Banana Pi R3	MT7986A, Quad-core ARM Cortex-A53 @ 1.3 GHz	2–4 GB DDR4, eMMC, microSD	1 × 1 GbE, 2 × 2.5 GbE, 4 × 1 GbE	7–10 W	MT7976C, Wi-Fi 6
Banana Pi R4	MT7988A, Quad-core ARM Cortex-A73	4 GB DDR4, NVMe option	4 × 2.5 GbE, 2 × 10 GbE (SFP+)	10–15 W	None (M.2 Wi-Fi module required)
Banana Pi R5	MT7988B, Quad-core ARM Cortex-A73	4 GB DDR4, NVMe option	2 × 10 GbE, 2 × 2.5 GbE	12–18 W	None (M.2 Wi-Fi module required)
Raspberry Pi 4	BCM2711, Quad-core ARM Cortex-A72 @ 1.5 GHz	2–8 GB LPDDR4, microSD	1 × 1 GbE (second via USB dongle)	6–8 W	Wi-Fi 5 (AP/-Client only)
Raspberry Pi 5	BCM2712, Quad-core ARM Cortex-A76 @ 2.4 GHz	4–8 GB LPDDR4X, microSD	1 × 1 GbE (second via PCIe card)	8–12 W	Wi-Fi 5 (AP/-Client only)
NanoPi R76S	Rockchip RK3588S, Octa-core (4× Cortex-A76 @ 2.4 GHz + 4× Cortex-A55 @ 1.8 GHz)	16 GB LPDDR4X / LPDDR5, NVMe (option via M.2)	3 × 2.5 GbE (RJ45)	10–15 W	None (M.2 Wi-Fi 6E module recommended)

Table 2.2: Requirements Fulfillment by Candidate Boards

Requirement	R3 Mini	R3	R4	R5	RPi 4	RPi 5	NanoPi R76S
≥ 2 native Ethernet interfaces	✓	✓	✓	✓	✗	✗	✓
RAM > 4GB	✗	✗	✓	✓	✗	✓	✓
≥ 2.5 GbE support	✓ (2×)	✓ (2×)	✓ ✓ (4×)	✓ (2×)	✗	✗	✓
Onboard Wi-Fi with AP & Monitor mode	✓	✓	✗	✗	✗	✗	✗
Low power consumption (<10 W)	✓	✓/▲	✗	✗	✓	▲	✓
Compact form factor	✓	✗	✗	✗	✓	✓	✓
Strong community & software support	✓	✓	▲	▲	✓ (general)	✓ (general)	✓
Suitable for inline packet sniffing	✓	✓ (overkill)	▲ (overkill)	▲ (expensive)	✗	✗	✓

Legend: ✓ = Requirement fulfilled, ✗ = Requirement not fulfilled, ▲ = Partially fulfilled / limited

2.3 Decision

Based on the defined requirements and the evaluation of alternatives, the **Banana Pi R4** and the**NanoPI R76S** are most suitable hardware platform for this prototype implementation.

The Banana Pi R4 offers two native 2.5 GbE interfaces for inline sniffing the board is compact, affordable, and supported by a strong community. In Addition, the two 10 GbE SFP+ ports provide flexibility for extensions as fiber-based packet capturing. A drawback of the R4 is the weaker CPU and a larger size compared to the NanoPI R76S

The NanoPi R76S is more compact and provides up to 16GB of RAM, which is advantageous for memory intensive processing and buffering tasks. While it lacks built-in Wi-fi, it can be expanded via the M.2 Wi-Fi 6E module. It can not host both a Wi-Fi card and NVMe SSD simultaneously. Consequently, data storage must be provided via microSD card or external USB SSD

Alternative boards such as the Banana Pi R3 Mini, R3 are limited overall performance. Rasberry PI 4 or 5 offer higher single core performance but were ultimately discarded because they provide only a single native Ethernet interface, requiring external adapters that reduce performance for inline sniffing scenarios.

3 Architecture and Design

Table 3.1: Use Cases Specification (nsak)

NR	Details
UC-02	**Use-Case:** Select Scenarios **Description:** From the chosen environment, the user selects which scenarios (and related drills) shall be executed and orders them by mission priority / noise level. **Actor:** User **Trigger:** User wants to define a mission plan. **Preconditions:** Environment selected (UC-0001); scenario list available. **Main Scenario:** 1. User requests list of scenarios. 2. System displays available scenarios with metadata (impact, noise). 3. User selects one or more scenarios and sets execution order. 4. System validates selection and stores it. **Alternative Scenarios:** No scenarios available → inform user. **Error Scenarios:** Conflicting resources between scenarios **Result:** Selected scenarios recorded. **Postconditions:** Mission plan saved; ready for preparation (UC-03).

NR	Details
UC-03	**Use-Case:** Prepare Scenario **Description:** Load, validate and prepare the selected scenarios: collect drills, resolve dependencies, build container. **Actor:** User `nsak prepare <scenario>`. **Preconditions:** Environment + scenario selection available (UC-01/02). **Main Scenario:** 1. System fetches referenced drills and their `drill.yaml`. 2. System validates configuration (includes UC-04.3 Verify). 3. System aggregates APT/Pip dependencies and resolves conflicts. 4. System builds container image / prepares runtime. 5. System marks scenario as *prepared* and logs results. **Alternative Scenarios:** CI pipeline performs the same steps. **Error Scenarios:** Validation or dependency failure → preparation aborted with diagnostics. **Result:** Prepared scenario image / runtime available. **Postconditions:** Scenario state = *prepared*.
UC-04	**Use-Case:** Use Commands (CLI) **Description:** Unified CLI to manage scenarios (list, prepare, start, stop, verify, status, help, version). **Actor:** User **Trigger:** User runs `nsak <subcommand>`. **Preconditions:** nsak installed **Main Scenario:** 1. CLI parses args and dispatches to subsystem. 2. Command executes and writes logs. 3. CLI prints structured output and exit code. **Alternative Scenarios:** Machine-readable output via `--json`. **Error Scenarios:** Invalid args → help; subsystem error → non-zero exit. **Result:** Requested action performed or error reported. **Postconditions:** State updated accordingly.

NR	Details
UC-04.1	**Use-Case:** List Scenarios **Description:** Show available scenarios for an environment with key metadata. **Actor:** User **Trigger:** `nsak list [--env <env>]`. **Preconditions:** Environment known or provided. **Main Scenario:** 1. System reads scenario manifests. 2. System prints table/list (name, desc, noise). **Alternative Scenarios:** ? **Error Scenarios:** ? **Result:** Scenarios visible to the user. **Postconditions:** User can select scenarios (UC-02).
UC-04.2	**Use-Case:** Show Help **Description:** Provide manual information and examples for CLI/subcommands. **Actor:** User **Trigger:** `nsak help` **Preconditions:** CLI available. **Main Scenario:** 1. System displays cmd in ordered form. **Alternative Scenarios:** - **Error Scenarios:** - **Result:** List of all cmds **Postconditions:** —
UC-04.3	**Use-Case:** Verify Scenario / Drill **Description:** Validate `scenario.yaml`/`drill.yaml` structure and declared dependencies (syntax + semantics). **Actor:** User **Trigger:** `nsak verify <scenario|drill>`. **Preconditions:** Files present and accessible. **Main Scenario:** 1. System parses YAML and required keys. 2. System checks referenced drills and dependencies. 3. System reports validation result and diagnostics. **Alternative Scenarios:** Verification in CI on PRs. **Error Scenarios:** Missing files / invalid YAML → error with line/col. **Result:** OK or detailed error. **Postconditions:** Decision basis for preparation.

NR	Details
UC-04.4 needs to be discussed	**Use-Case:** Show Nsak Status **Description:** Present runtime status: prepared/running/stopped scenarios, container ids, last errors. **Actor:** User **Trigger:** `nsak status [<scenario>]` **Preconditions:** State file prepared ??????? **Main Scenario:** 1. System reads state file and inspects container runtime. 2. System prints summary and optional details. **Alternative Scenarios:** `--json` for monitoring. **Error Scenarios:** Corrupt state → warning + fallback inspection. **Result:** User sees current state. **Postconditions:** —
UC-04.5	**Use-Case:** Show Version **Description:** Display nsak CLI/image version, build meta (commit, build time) and core component versions. **Actor:** User **Trigger:** `nsak version`. **Preconditions:** CLI installed. **Main Scenario:** 1. System prints CLI version, git commit/tag. 2. System prints base image + key tools versions. **Alternative Scenarios:** - **Error Scenarios:** Return a report. **Result:** Version info available for troubleshooting/reporting. **Postconditions:** —

NR	Details
UC-05	**Use-Case:** Simulate Scenario **Description:** Run a prepared scenario, monitor runtime and record outputs. **Actor:** User **Trigger:** `nsak simulate <scenario>`. **Preconditions:** Scenario prepared. **Main Scenario:** 1. System starts containers and services. 2. User monitors progress. 3. System stores logs and results. **Alternative Scenarios:** Dry-run with test data. **Error Scenarios:** Container startup failure. **Result:** Simulation finished. **Postconditions:** Results stored
UC-06	**Use-Case:** Start Scenario **Description:** Start a prepared scenario manually or automatically (SystemD). **Actor:** User, SystemD **Trigger:** `nsak start`. **Preconditions:** Scenario prepared. **Main Scenario:** 1. System reads configuration. 2. Launches containers and logs IDs. 3. Marks scenario as running. **Alternative Scenarios:** Auto-start at boot. **Error Scenarios:** Start failure or missing dependency. **Result:** Scenario running. **Postconditions:** State = running.
UC-07	**Use-Case:** Stop Scenario **Description:** Stop a running scenario and persist results. **Actor:** User, SystemD **Trigger:** `nsak stop`. **Preconditions:** Scenario running. **Main Scenario:** 1. System sends stop signal. 2. Saves results and logs. 3. Marks scenario as stopped. **Alternative Scenarios:** Forced stop on timeout. **Error Scenarios:** Process hang. **Result:** Scenario stopped safely. **Postconditions:** Logs persisted, state updated.

NR	Details
UC-0008	**Use-Case:** Extract Data **Description:** Export logs, pcap files and reports from completed simulations. **Actor:** User **Trigger:** `nsak extract`. **Preconditions:** Simulation/ mission finished. **Main Scenario:** 1. System locates and verifies artifacts. 2. Exports data to chosen directory or archive. 3. Reports export result. **Alternative Scenarios:** Filtered export by time or type. **Error Scenarios:** Missing artifacts **Result:** Data exported. **Postconditions:** Export Files created.
UC-0009	**Use-Case:** Remote Access SSH **Description:** Allow secure SSH access to the nsak host or container for diagnostics. **Actor:** Admin / Red Team **Trigger:** SSH login attempt. **Preconditions:** Authorized key configured; network reachable. **Main Scenario:** 1. User connects via SSH. 2. System verifies public key. 3. Access granted to restricted shell. **Alternative Scenarios:** Jump host or port forwarding enabled. **Error Scenarios:** Authentication failure / blocked key. **Result:** Secure shell established or denied. **Postconditions:** Access logged; keys remain intact.

4 Second Thesis Chapter

4.1 Implementation

4.1.1 Architecture

Declaration of Authorship

I hereby declare that I have written this thesis independently and have not used any sources or aids other than those acknowledged.

All statements taken from other writings, either literally or in essence, have been marked as such.

I hereby agree that the present work may be reviewed in electronic form using appropriate software.

November 3, 2025

Frank Gauss (gausf1) and Lukas von Allmen (vonal3)

Bibliography

List of Figures

List of Tables

Listings

Glossary

This document is incomplete. The external file associated with the glossary 'main' (which should be called `documentation.gls`) hasn't been created.

Check the contents of the file `documentation.glo`. If it's empty, that means you haven't indexed any of your entries in this glossary (using commands like `\gls` or `\glsadd`) so this list can't be generated. If the file isn't empty, the document build process hasn't been completed.

If you don't want this glossary, add `nomain` to your package option list when you load `glossaries-extra.sty`. For example:

```
\usepackage[nomain]{glossaries-extra}
```

Try one of the following:

- Add `automake` to your package option list when you load `glossaries-extra.sty`. For example:

  ```
  \usepackage[automake]{glossaries-extra}
  ```

- Run the external (Lua) application:

  ```
  makeglossaries-lite.lua "documentation"
  ```

- Run the external (Perl) application:

  ```
  makeglossaries "documentation"
  ```

Then rerun LaTeX on this document.

This message will be removed once the problem has been fixed.

.1 First Appendix Chapter

.1.1 Project 2 Proposal