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 December 30, 2025

Technik und Informatik

Abstract

We describe NSAK as an embedded, modular, open source, scenario-based network sniffing and security framework. We provide an overview of the system design, emphasizing the functionality of a Swiss Army Knife in a networking context. The attack scenario includes drills that configure or target a specific task to observe or open an attack vector in the system. NSAK is based on a core back-end part that manages specific environments, scenarios, and drills. The whole concept is based on containerization, where each container builds or prepares a scenario that can be triggered in a network environment. In the analogy of the Swiss Army Knife, the right knife with the proper drills for the necessary task can be selected. Modularity comes into play when a drill is selected across multiple scenarios.

Keywords Network intrusion detection, Network Monitoring, Red/Blue Team

Mit Regeln von Abstract gegenchecken und querlesen

Rules to apply

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.

Inhalt Abstract – Hintergrundinformationen wie Ausgangslage, Relevanz, Forschungskontext in ein bis zwei Sätzen zusammenfassen. – Fragestellung und Ziel explizit formulieren. – Die wichtigsten Eckpunkte zum methodischen Vorgehen angeben, bei empirischen Studien auch Angaben zu den Daten wie etwa die Charakteristika der Stichprobe. – Im Hauptteil des Abstracts die relevanten Ergebnisse und deren Bedeutung mit wichtigen Kennzahlen aufführen (ca. zwei Drittel des Abstracts). – Mit wichtigen Schlussfolgerungen oder Anwendungsmöglichkeiten das Abstract abrunden. – Das Abstract enthält keine Quellenverweise

Contents

Abstract iii

1 First Part Thesis: Evaluation 1
1.1 Current State of Research . . . 2

2 Evaluation 3
2.1 Hardware Selection . . . 3
2.1.1 Hardware Requirements . . . 3
2.1.2 Evaluated Boards . . . 4
2.1.3 Decision . . . 5
2.1.4 Hardware Specification . . . 5
2.2 Software Selection . . . 7
2.2.1 Framework Technology Stack . . . 7
2.2.2 System Dependencies . . . 7

3 Architecture and Design 9
3.0.1 Framework Concepts . . . 9
3.1 Use-Cases . . . 10
3.2 Component-diagram . . . 16

4 Second Part Thesis: Implementation 17
4.1 Implementation . . . 17
4.2 Network Environments . . . 17
4.2.1 Category I: Basic / Local . . . 17
4.2.2 Category II: Mobile / Wireless . . . 19
4.2.3 Category III: Secure / Advanced . . . 20
4.2.4 Category IV: IoT / Special Purpose . . . 20
4.2.5 Category V: Virtual / Simulation . . . 20
4.3 Software Modules . . . 21
4.3.1 Category 1: Network Traffic Collection . . . 22
4.3.2 Category 2: Network Mapping and Port Mapping . . . 23
4.3.3 Category 3: ManintheMiddle (MITM) . . . 23
4.3.4 Category 4: Exploits . . . 24
4.3.5 Category 5: Brute force . . . 25
4.3.6 Category 6: Denial of Service (DoS) . . . 25
4.4 Scenarios . . . 25
4.5 Conclusion . . . 25

Bibliography 29

List of Figures 31

List of Tables 33

Listings 35

.1 First Appendix Chapter 36

.1.1 Project 2 Proposal 36

1 First Part Thesis: Evaluation

Braucht es mehr quellen? sind alle Punkte erfüllt

Introduction

The combination of red team activities and blue team observation techniques is widely adopted within the cybersecurity community. While the red team focuses on emulating adversarial behavior, the primary objective of the blue team is to detect such activities through non-invasive monitoring and analysis of system behavior [1].

As the economic and operational costs of launching cyberattacks continue to decrease due to AI automation [2], the need for continuous surveillance and Zero Trust Architecture is rising. One approach to reduce operational security costs is to adopt multiple modular frameworks that can be easily extended, configured, and executed continuously in a controlled manner [3].

The threat emulation frameworks introduced by Zilberman et al. evaluate multiple attack phases, including lateral movement, persistence, and attack execution. The Network Swiss Army Knife focuses on containerized, orchestrated scenarios that execute specific attack drills in a controlled environment. Future extensions will focus on enriching the assessment layer by systematically capturing and evaluating defensive responses of multiple scenarios.

This proof of concept comprises the design and implementation of a modular, isolated open-source security framework that focuses on extensibility and the controlled execution of attack-based scenarios.

The objective of this work is to investigate whether such a framework can provide a flexible, extendable, and safe foundation for modular security testing in a network environment.

2.2 Einleitung Die Einleitung führt einerseits zum Thema hin (Ausgangslage), und informiert andererseits darüber, warum (Fragestellung/Problem) und wozu (Ziel/Zweck) es die Arbeit gibt sowie ggf. wie sie zustande gekommen ist (methodisches Vorgehen). Die Einleitung kann je nach Umfang und Thema mit oder ohne Unterkapitel verfasst werden. Sie umfasst ca. 5–10 Einen Überblick über die Kapitel der Arbeit gibt es höchstens bei sehr langen Arbeiten (beispielsweise Masterarbeit). Die Ausgangslage beschreiben – Relevanz: Warum ist dieses Thema überhaupt bedeutsam? Schon hier gilt es, nicht einfach etwas zu behaupten, sondern Fakten und Aussagen mit Fachliteratur zu belegen. – Aktualität: Gibt es einen aktuellen Bezug? – Zusammenhang: Wie lässt sich das Thema einordnen? In welchem fachlichen Kontext steht die Arbeit? – Forschungsstand: Was ist schon erforscht? Gibt es bereits Untersuchungen? (Ist-Zustand und Zusammenhang mit dem eigenen Thema.) Je nach Art und Umfang der Arbeit gibt es zum Wissensstand ein separates Kapitel (siehe 2.3). – Wenn vorhanden externe Auftraggeber, Auftraggeberinnen: Wer sind die beteiligten Partnerinnen oder Stakeholder? – Den Kurs bzw. das Modul, in dem die Arbeit entsteht, erwähnt man auf dem Titelblatt (siehe 4.1). 8 Das Problem und das Ziel darstellen – Zweck der Arbeit: Welche Aufgabe, Herausforderung, welches Problem soll gelöst werden? Warum sollte man die Arbeit lesen? – Fragestellung: Auf welche konkrete Hauptfrage (evtl. mit konkretisierenden Unterfragen) soll im Schlusskapitel eine fundierte Antwort gegeben werden? Ist die Fragestellung genügend eingegrenzt? Gibt es Hypothesen? – Abgrenzung: Wo liegen die Grenzen der Untersuchung (zeitlich, geografisch, thematisch, methodisch, in der Auswahl der Hilfsmittel usw.)? Was wird nicht untersucht? Was kann die Arbeit nicht leisten? – Ziel: Was soll die Untersuchung genau bewirken? Was ist die Absicht hinter der Arbeit? – Erwartung: Was möchte die Arbeit leisten (Nutzen der Untersuchung, Soll-Zustand)? Für welche konkrete Zielgruppe sind die Ergebnisse der Arbeit von Nutzen? Was ist zu erwarten? Was ist nicht zu erwarten? Das methodische Vorgehen andeuten Wie man methodisch vorgeht, wird ausführlich im Methodenkapitel beschrieben (siehe 2.4). Oft erwähnt man aber in der Einleitung bereits in ein bis zwei Sätzen, mit welcher Methode man arbeitet (Literaturarbeit, Umfrage, Entwickeln eines Prototyps, Variantenstudium usw.).

1.1 Current State of Research

Der «Stand der Forschung» beantwortet folgende Fragen:

– Was wurde zum Thema der Arbeit bereits erforscht (Forschungsstand)? – Auf welche Forschungsarbeiten stützt sich die Arbeit ab? – Welche Theorien oder Konzepte sind für die Beantwortung der Fragestellung relevant? – Welche Begriffe müssen definiert werden? – Welche Normen spielen für die Untersuchung eine Rolle

2 Evaluation

2.1 Hardware Selection

2.1.1 Hardware 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)

2.1.2 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

2.1.3 Decision

Based on the defined requirements and the evaluation of alternatives, the **Banana Pi R4** and the**NanoPI R76S** are the most suitable hardware platforms 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 cannot 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. Raspberry 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.

2.1.4 Hardware Specification

Each environment represents a practical setup in which the NSAK device can be deployed. For traffic analysis, performance testing or security evaluation.

Category I — Inline:

Diagram: Laptop $\leftrightarrow$ NSAK device $\leftrightarrow$ Router

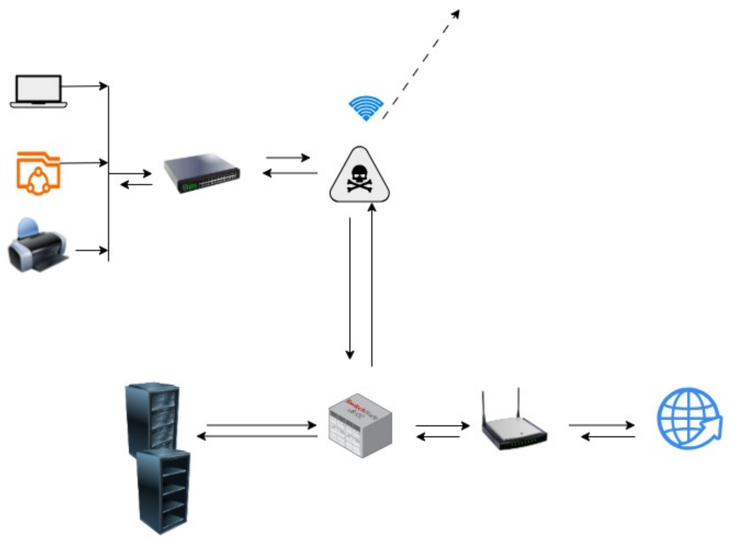

Figure 2.1

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

Category II — Wireless:

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

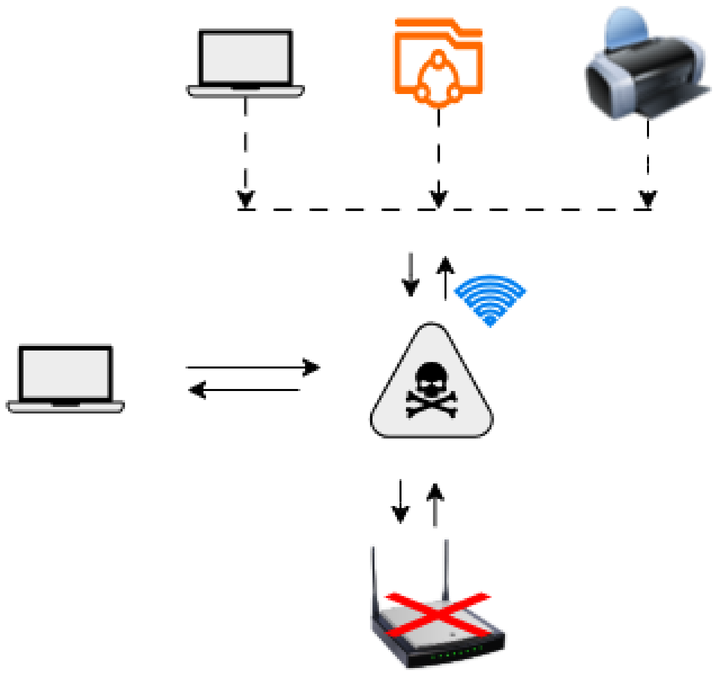

Figure 2.2

Description: The NSAK device is inline and lets traffic pass but intercepts as Rouge AP and capture data

2.2 Software Selection

2.2.1 Framework Technology Stack

Core / CLI

Describe why we choose this technology, maybe we find references which underline the ease of use and advantages for modularity which are comming with python

Programming language: Python Dependency manager: uv Virtual environment manager: uv Package build tool: uv Linter: ruff Formatter: ruff Type checker: mypy Testing framework: pytest

Dependencies: click: Library for building CLIs pyyaml: Library for loading, validating and reading yaml files scapy: Library for red team operations pre-commit: Package to enforce code quality tools for each commit

2.2.2 System Dependencies

Write this section nicer and explain what the advantages are of such a design is, especially in relation to modularity

As we leverage the abstraction of OCI containers to run scenarios in an encapsulated environment, we have only a minimal set of system dependencies. All system dependencies that are required for running a drill or a scenario are installed into the scenario image, during the build process.

Version control: git Network tooling: iptables (we should switch to nf_tables) OCI container manager: podman OCI container orchestrator: podman-compose Programming languages: python3, python3-pip, uv Utilities: curl, sudo

3 Architecture and Design

3.0.1 Framework Concepts

This section describes the high-level concepts which can be managed with the NSAK framework.

I did not think of a specific order, add a short excerpt in the overview list, add a diagram which shows the relation of the concepts

Overview:

- Devices
- Environments
- Drills
- Scenarios

Devices

Environments

Drills

Scenarios

3.1 Use-Cases

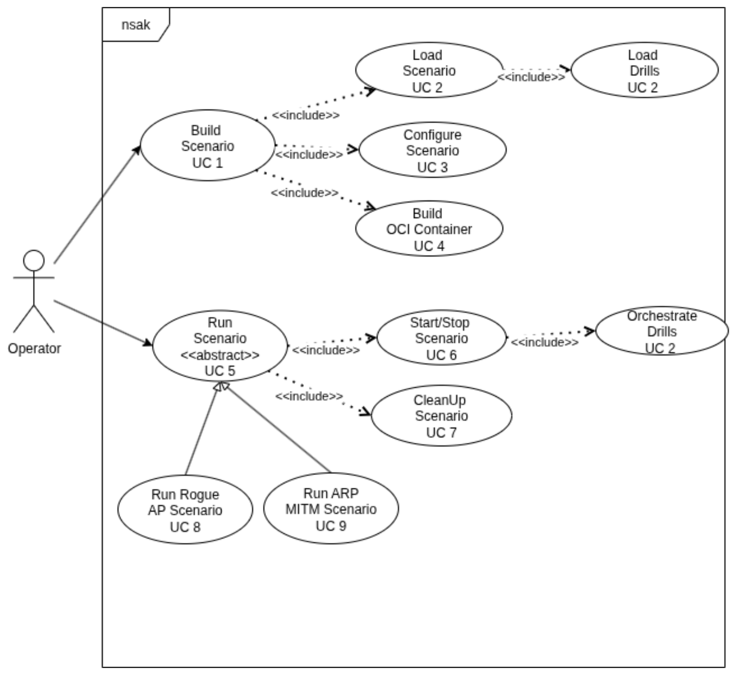

Figure 3.1

drüber lesen und mit UC final ableichen

Figure 3.1 illustrates the use case structure of the proposed NSAK framework.

Table 3.1: Use Cases Specification (NSAK)

NR & Details	
UC-01	**Use-Case:** Build Scenario **Description:** Builds a scenario container based on a selected scenario configuration. **Actor:** Operator **Trigger:** Operator initiates scenario build via the command-line interface. **Preconditions:** NSAK initialized; scenarios available. **Main Scenario:** 1. Operator selects a scenario to build using the command-line interface. 2. System validates the selected scenario. 3. System executes the included use cases: ▶ Configure Scenario (UC-2) ▶ Load Scenario (UC-3) ▶ Load Drills (UC-4) ▶ Build OCI Container (UC-5) **Alternative Scenarios:** No scenarios available → inform operator. **Error Scenarios:** Conflicting scenario configuration detected → build aborted. **Result:** Scenario container successfully built. **Postconditions:** Scenario container stored and ready to run.
UC-02	**Use-Case:** Configure Scenario **Description:** Defines scenario-specific build parameters such as network interfaces and execution options. **Actor:** System **Trigger:** Scenario selected for build (UC-01). **Preconditions:** Scenario selection available. **Main Scenario:** 1. System applies scenario-specific configuration parameters. **Result:** Scenario configuration created. **Postconditions:** Scenario configuration available for loading.

NR & **Details**	
UC-03	**Use-Case:** Load Scenario **Description:** Loads and validate the selected scenarios **Actor:** System **Trigger:** Scenario configuration available (UC-02). **Preconditions:** Scenario configuration created. **Main Scenario:** 1. System retrieves the scenario definition files (scenario.yaml, scenario.py, README.md). 2. System validates the scenario structure and resolves declared dependencies. **Error Scenarios:** Validation or dependency failure - preparation aborted with Error Log. **Result:** Scenario is successfully loaded. **Postconditions:** Scenario representation available for drill loading.
UC-04	**Use-Case:** Load Drills **Description:** Loads the attack drills required by the selected scenario. **Actor:** System **Trigger:** Scenario loaded (UC-03). **Preconditions:** Scenario representation available. **Main Scenario:** 1. System resolves drill references defined in the scenario configuration. 2. System instantiates drill objects and loads associated metadata. **Error Scenarios:** Invalid drill definition, drill not found, or ambiguous drill reference. **Result:** Required drill objects loaded. **Postconditions:** Drills available for container build.

NR & **Details**	
UC-05	**Use-Case:** Build OCI Container **Description:** Builds an OCI-compliant container image for the loaded scenario. **Actor:** System **Trigger:** Scenario and drills loaded (UC-03, UC-04). **Preconditions:** Scenario representation and drill objects available. **Main Scenario:** 1. System generates the container build context. 2. System builds the scenario container image with required privileges and network configuration. **Error Scenarios:** Container build failure - build aborted with error message. **Result:** OCI-compliant scenario container image built. **Postconditions:** Scenario container image stored and ready for execution.
UC-06	**Use-Case:** Run Scenario **Description:** Executes a previously built scenario container. Specific scenarios such as Rogue AP or ARP MITM represent specialized configurations of this use case. **Actor:** Operator **Trigger:** Operator initiates scenario execution via the command-line interface. **Preconditions:** Scenario container image available (UC-05). **Main Scenario:** 1. System starts the scenario container with the required execution parameters. 2. System executes the included use cases: ▶ Execute Scenario (UC-07) ▶ Clean Up Scenario (UC-09) **Result:** Scenario container execution started. **Postconditions:** Scenario execution context active.

NR & Details	
UC-07	**Use-Case:** Execute Scenario **Description:** Orchestrates the execution of a previously built scenario container and coordinates the execution of the associated attack drills. **Actor:** System **Trigger:** Run Scenario (UC-6) **Preconditions:** Scenario Image available and started **Main Scenario:** 1. System Scenario Manager executes for the selected scenario 2. System Drill Manager execute drill UC-8 include use-case **Error Scenarios:** Scenario not found or scenario container not available. **Result:** Scenario execution initiated and drill execution orchestrated. **Postconditions:** Scenario container is running and drills are being executed.
UC-08	**Use-Case:** Execute Drills **Description:** Executes the attack drills defined in the scenario configuration within the running scenario container. **Actor:** System **Preconditions:** Scenario execution context initialized. **Main Scenario:** 1. System Drill Manager retrieves the list of configured drills. 2. System Drill Manager executes the drills according to the defined order and parameters. **Error Scenarios:** Drill execution failure or missing drill definition. **Result:** Configured attack drills executed.

NR & Details	
UC-09	**Use-Case:** Clean Up Scenario **Description:** Terminates the running scenario container and restores the system to a defined baseline state. **Actor:** System **Trigger:** Stop Scenario (UC-06) **Preconditions:** Scenario container is running. **Main Scenario:** 1. System stops the running scenario container. 2. System invokes the included use case Clean Up Drills (UC-10). **Error Scenarios:** Scenario container cannot be terminated. **Result:** Scenario execution terminated. **Postconditions:** Scenario container stopped and removed.
UC-10	**Use-Case:** Clean Up Drills **Description:** Cleans up artifacts and state changes introduced by executed attack drills. **Actor:** System **Preconditions:** Drill execution completed or aborted. **Main Scenario:** 1. System Drill Manager terminates active drill processes. 2. System Drill Manager removes temporary artifacts and resets modified parameters. **Error Scenarios:** Incomplete cleanup due to failed drill termination. **Result:** Drill-related artifacts removed and state reset.

3.2 Component-diagram

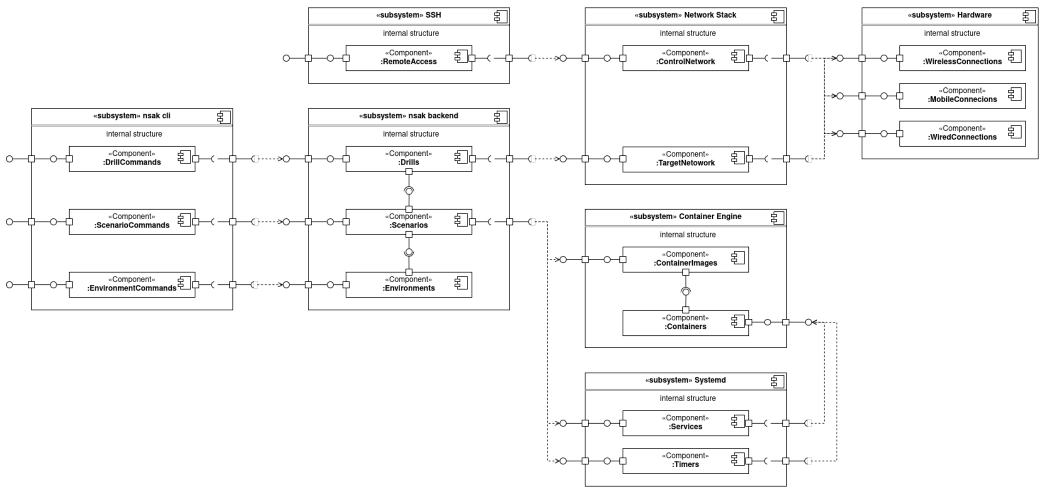
«subsystem» SSH
internal structure
«Component» :RemoteAccess
«subsystem» Network Stack
internal structure
«Component» :ControlNetwork
«Component» :TargetNetowork
«subsystem» Hardware
internal structure
«Component» :WirelessConnections
«Component» :MobileConnecions
«Component» :WiredConnections
«subsystem» nsak cli
internal structure
«Component» :DrillCommands
«Component» :ScenarioCommands
«Component» :EnvironmentCommands
«subsystem» nsak backend
internal structure
«Component» :Drills
«Component» :Scenarios
«Component» :Environments
«subsystem» Container Engine
internal structure
«Component» :ContainerImages
«Component» :Containers
«subsystem» Systemd
internal structure
«Component» :Services
«Component» :Timers

Figure 3.2

4 Second Part Thesis: Implementation

2.4 Methoden Im Methodenkapitel wird aufgezeigt, was (Material, Daten) wie (Methode) untersucht wurde, um die Fragestellung zu beantworten. Das Ziel des Kapitels ist dabei, die Nachvollziehbarkeit und Überprüfbarkeit der Untersuchung für spätere oder weiterführende Arbeiten zu gewährleisten. Dafür muss das methodische Vorgehen wiederholbar festgehalten werden. Das Methodenkapitel hat dabei einen ebenso hohen Stellenwert wie das Ergebniskapitel, denn nur nachvollziehbar erhobene Ergebnisse sind verlässliche Ergebnisse. Was genau im Methodenkapitel festgehalten wird, hängt stark von der methodischen Ausrichtung der Arbeit ab. Je nach gewählter Methodik sollten folgende Inhalte festgehalten werden:

– Welches Material wurde untersucht? (Proben, Prüfkörper, Objekte, Bauteile, Elemente, Software usw.) – Mit welchen Geräten, Vorrichtungen, Werkzeugen usw. wurde gearbeitet? – Welcher Probenumfang wurde untersucht? (Stichprobe, Gewährspersonen usw.) – Wie wurden die Daten erhoben? – Wie wurden die Daten ausgewertet? (statistische Testverfahren, Software usw.) – Wo und wie wurde Literatur gesucht, welche wichtigen Quellen wurden verwendet, wie und warum wurden sie ausgewählt?

4.1 Implementation

4.2 Network Environments

Each environment represents a practical setup in which the NSAK device can be deployed for traffic analysis, performance testing, or security evaluation.

4.2.1 Category I: Basic / Local

E 1: Point-to-Point

Diagram: Laptop ↔ nsak ↔ Router

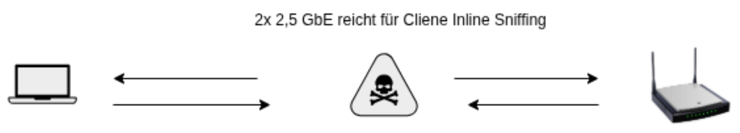

Figure 4.1

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

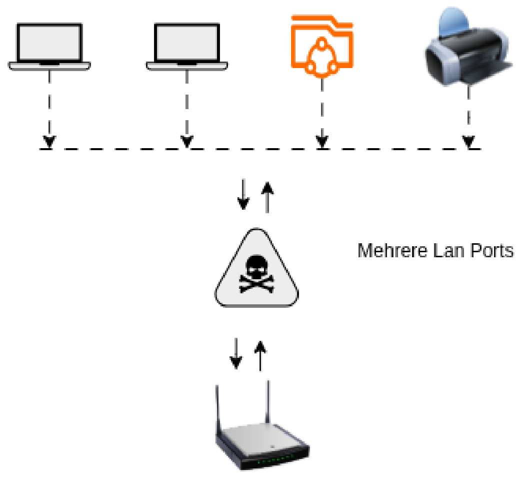

Figure 4.2

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

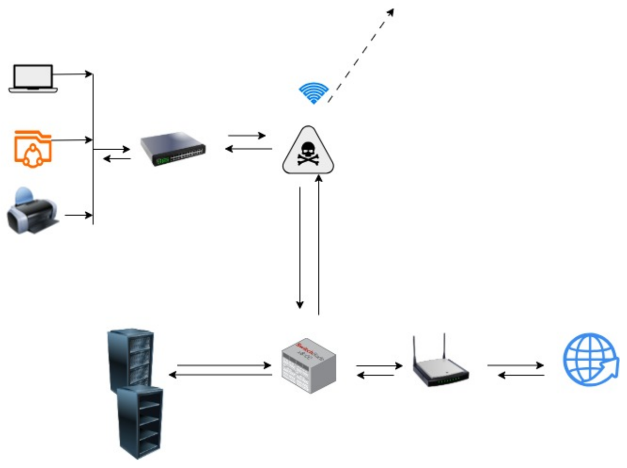

Figure 4.3

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

4.2.2 Category II: Mobile / Wireless

E 4: Access Point Mode

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

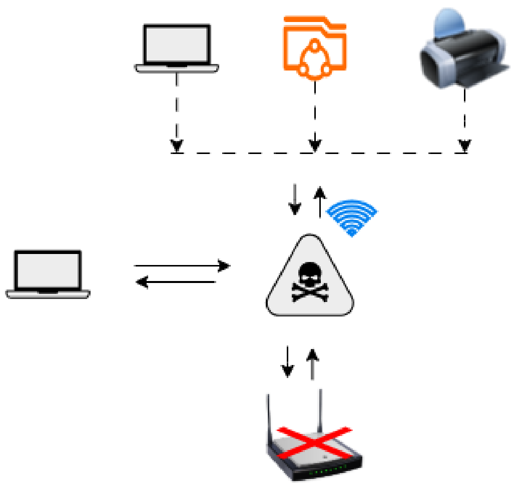

Figure 4.4

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.

4.2.3 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.

4.2.4 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.

4.2.5 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 $\leftrightarrow$ VPN/SSH $\leftrightarrow$ 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.

4.3 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. despiteManintheMiddle (MITM)
4. Exploits
5. Brute Force
6. Denial of Service (DoS)
7. Physical (TBD?)

4.3.1 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.

4.3.2 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

4.3.3 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.

4.3.4 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

4.3.5 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

4.3.6 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

4.4 Scenarios

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

4.5 Conclusion

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.

December 30, 2025

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

Bibliography

[1] Jeyavijayan Rajendran, Vinayaka Jyothi, and Ramesh Karri. Blue team red team approach to hardware trust assessment. In *Proceedings of the IEEE 29th International Conference on Computer Design (ICCD)*, pages 285–288, Amherst, MA, USA, 2011. IEEE.

[2] Vaibhav Garg and Jayati Dev. Artificial intelligence and the new economics of cyberattacks. USENIX ;login: online article, August 2024. Article shepherded by Rik Farrow.

[3] Polina Zilberman, Rami Puzis, Sunders Bruskin, Shai Shwarz, and Yuval Elovici. SoK: A Survey of Open-Source Threat Emulators. arXiv preprint, 2020.

List of Figures

2.1 6
2.2 7

3.1 10
3.2 16

4.1 18
4.2 18
4.3 19
4.4 19

List of Tables

2.1 Comparison of Board Variants 4
2.2 Requirements Fulfillment by Candidate Boards 4

3.1 Use Cases Specification (NSAK) 11

Listings

.1 First Appendix Chapter

.1.1 Project 2 Proposal