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 29, 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

1 First Part Thesis

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 [?].

As the economic and operational costs of launching cyberattacks continue to decrease due to AI automation [?], 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 [?].

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. 510 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 ńStand der Forschungż 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 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 (onboard)
Banana Pi R3 Mini	MT7986A, Quad-core ARM Cortex-A53 @ 1.3 GHz	2 GB DDR4, 8 GB eMMC, microSD	2 Œ 2.5 GbE	57 W	MT7976C, Wi-Fi 6 (AP/Client/Monitor)
Banana Pi R3	MT7986A, Quad-core ARM Cortex-A53 @ 1.3 GHz	24 GB DDR4, eMMC, microSD	1 Œ 1 GbE, 2 Œ 2.5 GbE, 4 Œ 1 GbE	710 W	MT7976C, Wi-Fi 6
Banana Pi R4	MT7988A, Quad-core ARM Cortex-A73	4 GB DDR4, NVMe option	4 Œ 2.5 GbE, 2 Œ 10 GbE (SFP+)	1015 W	None (M.2 Wi-Fi module required)
Banana Pi R5	MT7988B, Quad-core ARM Cortex-A73	4 GB DDR4, NVMe option	2 Œ 10 GbE, 2 Œ 2.5 GbE	1218 W	None (M.2 Wi-Fi module required)
Raspberry Pi 4	BCM2711, Quad-core ARM Cortex-A72 @ 1.5 GHz	28 GB LPDDR4, microSD	1 Œ 1 GbE (second via USB dongle)	68 W	Wi-Fi 5 (AP/Client only)
Raspberry Pi 5	BCM2712, Quad-core ARM Cortex-A76 @ 2.4 GHz	48 GB LPDDR4X, microSD	1 Œ 1 GbE (second via PCIe card)	812 W	Wi-Fi 5 (AP/Client only)
NanoPi R76S	Rockchip RK3588S, Octa-core (4Œ Cortex-A76 @ 2.4 GHz + 4Œ Cortex-A55 @ 1.8 GHz)	16 GB LPDDR4X / LPDDR5, NVMe (option via M.2)	3 Œ 2.5 GbE (RJ45)	1015 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	✓ (2Œ)	✓ (2Œ)	✓ ✓ (4Œ)	✓ (2Œ)	✗	✗	✓
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 theNanoPI 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.

2.4 Specification

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

2.4.1 Category I — Inline:

Diagram: Laptop ↔ NSAK device ↔ Router

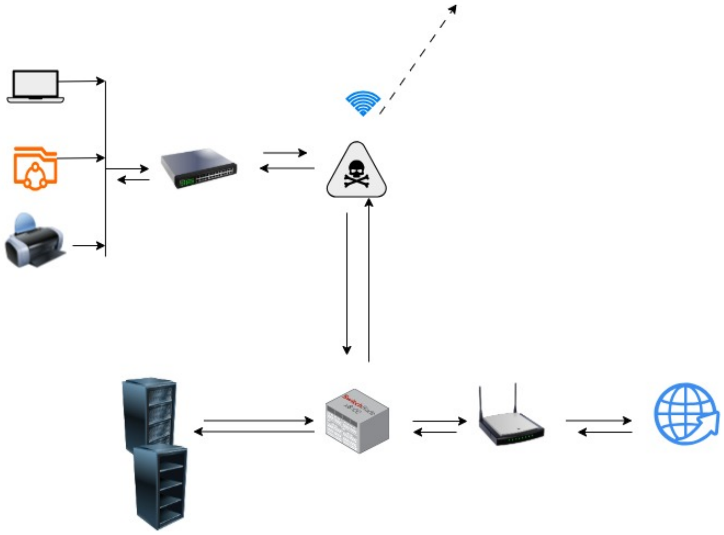

Figure 2.1

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

2.4.2 Category II — Wireless:

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

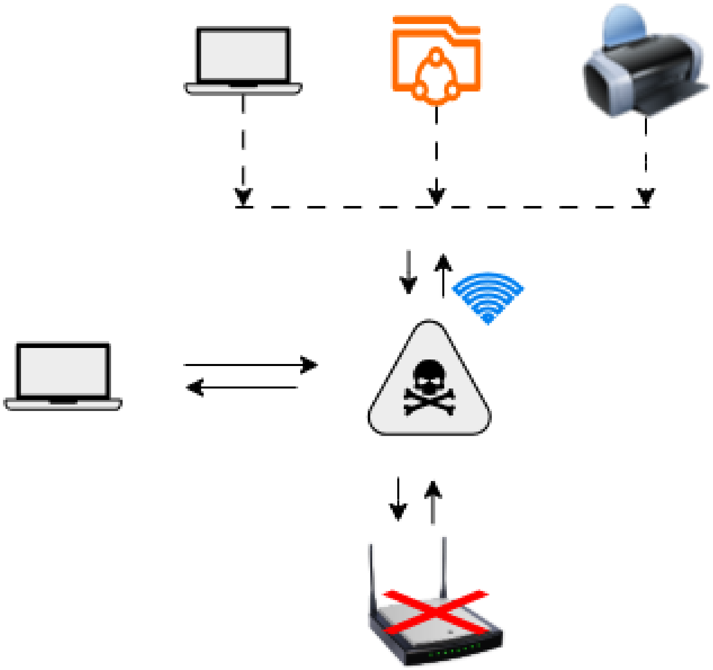

Figure 2.2

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

3 Architecture and Design

3.1 Use-Cases

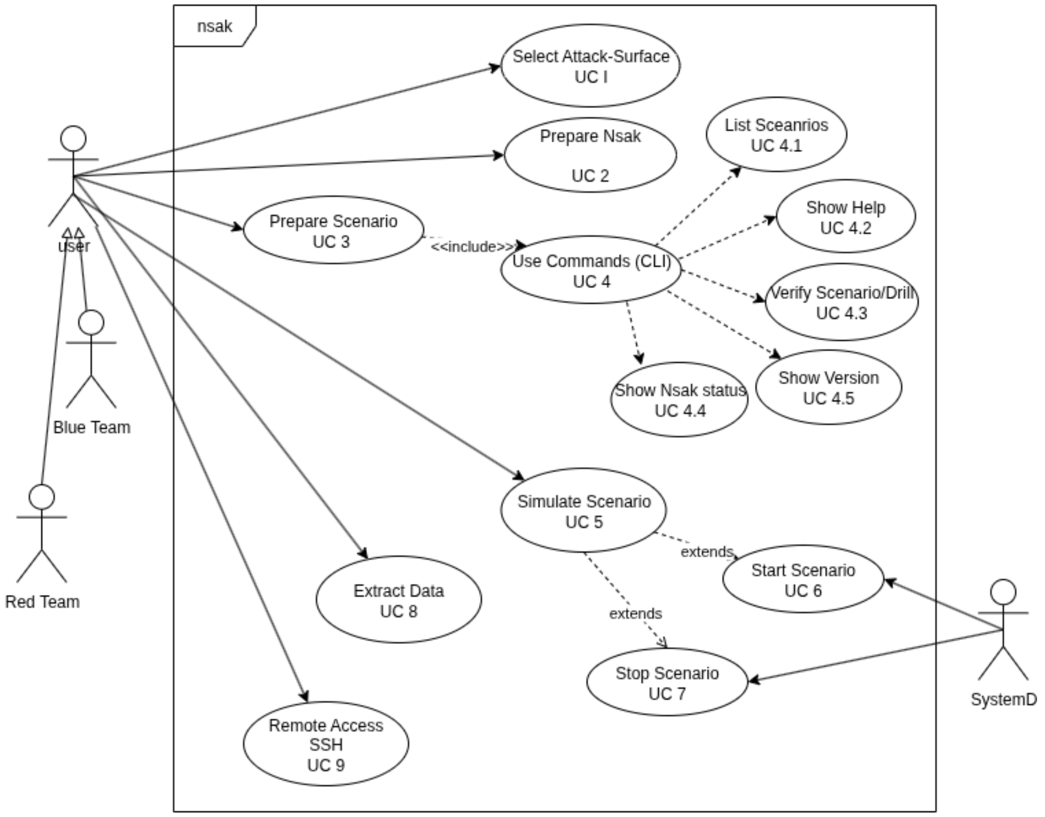

Figure 3.1

Table 3.1: Use Cases Specification (NSAK)

NR	Details
UC-02	Use-Case: Select Scenario 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 a list of scenarios. 2. System displays available scenarios with metadata (impact, noise). 3. User selects one or more scenarios and sets the 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).
UC-03	Use-Case: Prepare Scenario Description: Load, validate and prepare the selected scenarios: collect drills, resolve dependencies, build container. Actor: User `nsak scenario build <scenario-name>`. 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 the 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.

NR	Details
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 a 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.
UC-04.1	Use-Case: List Scenario Description: Show available scenarios for an environment with key metadata. Actor: User Trigger: `nsak scenario list`. Preconditions: Environment known or provided. Main Scenario: 1. System reads scenario specifications/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:

NR	Details
UC-04.3	Use-Case: Verify Scenario / Drill Description: Validate `scenario.yaml`/`drill.yaml` structure and declared dependencies (syntax and 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 results 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.
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 a state file and inspects container runtime. 2. System prints summary and optional details. Alternative Scenarios: `-json` for monitoring. Error Scenarios: Corrupt state warning and fallback inspection. Result: User sees current state. Postconditions:

NR	Details
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 and key tools versions. Alternative Scenarios: Error Scenarios: Return a report. Result: Version info available for troubleshooting/reporting. Postconditions:
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.

NR	Details
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.
UC-08	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 a 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.

NR	Details
UC-09	Use-Case: Remote Access SSH Description: Allow secure SSH access to the NSAK device 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.

3.2 Component-diagram

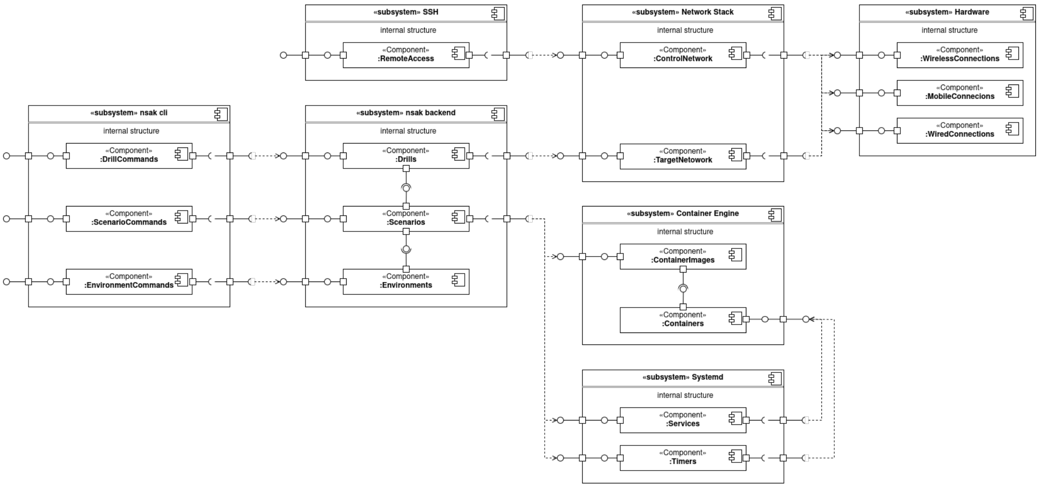
«subsystem» SSH
internal structure
«Component» :RemoteAccess
«subsystem» Network Stack
internal structure
«Component» :ControlNetwork
«Component» :TargetNetowork
«subsystem» Hardware
internal structure
«Component» :WirelessConnections
«Component» :MobileConnecions
«Component» :WiredConnections
«subsystem» 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: Method

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 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 29, 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