**ZHAW - MAS Informatik**

**MAS Thesis**

**Controlled communication of seized mobile devices in the IT Forensics Unit of Zurich Metropolitan Police**

A controlled network environment that completely blocks the data communication of seized mobile phones and only allows essential communication through whitelisting.

Alexander Wüst

Kronenwis 19

8864 Reichenburg

**Abgabe** 15. May 2025

**Betreuer:in** Peter Heinrich

**Abstract**

Dies ist ein Platzhalter-Text. Dies ist ein Platzhalter-Text. Dies ist ein Platzhalter-Text. Dies ist ein Platzhalter-Text. Dies ist ein Platzhalter-Text. Dies ist ein Platzhalter-Text. Dies ist ein Platzhalter-Text. Dies ist ein Platzhalter-Text.

**Table of Contents**

[1. Introduction 1](#_Toc198211645)

[1.1. Preface 1](#_Toc198211646)

[1.2. Goal of the work 2](#_Toc198211647)

[1.3. Scope Limiations 2](#_Toc198211648)

[2. Background 3](#_Toc198211649)

[2.1. Evolution of Firewalls 3](#_Toc198211650)

[2.2. Overview of current firewall solutions 3](#_Toc198211651)

[2.2.1. System Level Firewalls 3](#_Toc198211652)

[2.2.2. Open Source Firewalls 4](#_Toc198211653)

[2.2.3. Enterprise Firewalls 4](#_Toc198211654)

[2.2.4. Requirements for the firewall solution 4](#_Toc198211655)

[2.2.5. Compare firewall solutions based on requirements 5](#_Toc198211656)

[2.3. Hardware evaluation 5](#_Toc198211657)

[2.3.1. Final Decision for Protectli V1410 6](#_Toc198211658)

[2.4. Remote Wipe 7](#_Toc198211659)

[2.4.1. Differences between Wiping and Deletion 7](#_Toc198211660)

[2.4.2. Apple iOS remote wipe 7](#_Toc198211661)

[2.4.3. Google Android 7](#_Toc198211662)

[3. Implementation 8](#_Toc198211663)

[3.1. Overview of the System Architecture 8](#_Toc198211664)

[3.2. Prototyping and Preliminary Testing 9](#_Toc198211665)

[3.2.1. Firewall Configuration During Prototyping 9](#_Toc198211666)

[3.2.2. API Integration and Log Retrieval 10](#_Toc198211667)

[3.3. Backend Implementation 10](#_Toc198211668)

[3.4. Frontend Implementation 10](#_Toc198211669)

[3.5. Database Design 10](#_Toc198211670)

[3.6. Integration with OPNsense 10](#_Toc198211671)

[3.7. Rule Application Logic 10](#_Toc198211672)

[3.8. Logging 10](#_Toc198211673)

[3.9. Challenges Encountered 11](#_Toc198211674)

[3.10. Summary 11](#_Toc198211675)

[4. Analysis / Results 11](#_Toc198211676)

[5. Discussion / Conclusion 11](#_Toc198211677)

[Appendix 13](#_Toc198211678)

[Declaration of Originality 14](#_Toc198211679)

# Introduction

## Preface

Electronic evidence plays a central role in today's criminal prosecution. Almost no criminal proceedings can do without the analysis of digital data - be it data from mobile phones, computers, IoT devices, cloud data and many other data sources. Digital forensics plays a crucial role in this: IT forensic experts prepare the seized devices according to forensic standards to ensure that the data collected can be used in court.

It's important does the collected devices are completely isolated from data networks after seizing them to maintain data integrity. If this is not possible due to security settings, the Zurich Metropolitan Police use a Faraday room[[1]](#footnote-1). This prevents data communication from the mobile phone and data on the device from being changed. As soon as a mobile phone is reconnected to the internet it will updating apps and synchronising data, such as cloud services, messages and other application data and this of course will trigger write processes on the data storage. Also, there is the risk of a remote deletion of the device which needs to be avoided.

A fundamental principle of forensic work is the reproducibility of the results. Once write operations occur on the data storage, this reproducibility can no longer be guaranteed. So, it would be clear, establishing a network connection or even simply powering on a device seems not to be an option and needs to be avoided.

However, best practices at the Metropolitan Police Zurich have shown that powering on a device is necessary to verify whether the data acquired by the forensic hardware and software has been processed correctly. A defined protocol is followed to check for any apps or entries that may not have been parsed correctly – In fact there are often problems during parsing in practice. Especially apps just known in Switzerland like “Twint” are not parsed automatically.

Without powering on the device, still in Airplane mode, it is not possible to detect parsing errors. For example, when no WhatsApp messages are shown at all or to identify if any media files such as photos are missing. In recent years, when physical access to a device was possible, like with a known or brute-forced[[2]](#footnote-2) passcode, it has proven useful to use this access to validate the acquisition and ensure no data was lost or misinterpreted during the process. The described procedure, although it also causes write operations, but it is considered as essential to ensure a good and complete data acquisition.

It is more and more common that not everything that is accessible on the device is saved on the device. Common examples are images and videos that are stored directly on provider servers (cloud). When manual reviewing chat conversations trough a police investigator, it can happen that the content of chats clearly goes in one direction, but the images and videos taken are missing because they are no longer or have never been stored locally on the device. For example, an image that could be identified as an offence or a prohibited media file containing violence or even child pornography may be missing from the device and because that it will not allow any clear conclusions. Or the communication seems to be clear, but the pictures are harmless? So, it is important to give during a search the best possible picture. But how to gain this extra information without getting the device online? The best way is to access the data directly with a cloud acquisition method. For this method the service needs to be supported, and a valid token needs to be extracted from the device. Sometimes even that method is not possible due to different reasons. Then it will normally give a consultation between the public prosecutor, the police officer who is in charge for the case and the forensic examiner. One of the biggest problems is the possibility of a remote deletion when taking the device online. As the acquiring trough forensic hard- and software took already place it will still be possible to have the original copy of the state of the device when it was seized.

There is no easy solution that makes it possible to take a device online in a controlled environment without risking the remote deletion of data and at the same time only allowing the necessary connections to the Internet. This MAS thesis aims to close this gap.

## Goal of the work

The primary objective of this thesis is to implement a wireless network with an integrated firewall, which by default blocks all internet access for connected devices. A central focus of the project is the development of a custom web interface that interacts with an firewall solution via its API. Through this interface, the forensic examiner will be able to monitor and configure firewall settings, define individual access rules for connected devices.

The solution is designed to improve usability and control by allowing the forensic examiner to make informed decisions about which connections should be allowed and which should remain blocked. All connections will be logged, enabling transparency and accountability. This is especially important in a forensic context.

To meet these goals, the project includes the development of an application with the following core capabilities:

* Centralized management of device-based firewall rules
* Monitoring of network activity
* Logging of all connection attempts and allowed traffic
* Easy addition and removal of rules through a user-friendly interface

Therefore suitable hardware will be evaluated and bought to ensure compatibility with the firewall solution and to meet performance and environmental requirements (e.g., rugged form factor, multiple network interfaces).

Reference to use cases ???…

## Scope Limiations

This master's thesis explicitly excludes the following topics from its scope:

* Implementation or evaluation of two-factor authentication methods using mobile networks (e.g., MobileID, SMS-based authentication)
* Analysis of write operations on mobile devices that may occur as a result of simply powering them on, even if considered best practice in forensic handling
* Integration of Deep Packet Inspection (DPI) or content-level traffic analysis within the firewall
* Examination of legal considerations or compliance issues, such as data protection laws, admissibility of evidence, or regulatory frameworks

The focus of this work remains on the technical implementation of a device-aware firewall management system for controlled network access and forensic transparency.

# Background

## Evolution of Firewalls

As computer networks began to connect to each other, the need to protect one network from the other and avoid external threats became increasingly important. The name “Firewall” came from physical barriers used in the architecture and history: Walls that protected cities like the Great Wall of China or structural firewalls that prevented a fire in the kitchen to spread out on other parts of the building. [2]. Similar, in networking, a firewall serves as a barrier to keep not wanted traffic outside and to allow other traffic to go through. Even if the “external side” is burning the firewall is intended to keep the internal network safe and unaffected.

The development of firewalls has progressed through several generations. First-generation firewalls in the 1990s used basic packet filtering to allow or block traffic based on IP addresses and ports.

In the Early 2000s the second generation, stateful inspection firewalls emerged, introducing the ability to track the state of connections and inspect traffic in context, offering enhanced control. Then application-layer firewalls enabled content-aware filtering and protocol-specific inspection, further increasing security.

Around 2008 the third Generation or also called Next-Generation Firewalls (NGFW) integrated traditional firewall functions with deep packet inspection (DPI), intrusion prevention systems (IPS), and application awareness.

The fourth firewall generation was launched around 2020. These firewalls use machine learning to detect zero-day threats in real time, going beyond traditional signature-based methods. Key features include zero-delay signature updates, automated security policy recommendations, and IoT device visibility based on behavioral analysis. It is continously learning from the network traffic. It aims to reduce the manual intervention. [2], [3]

## Overview of current firewall solutions

This section provides an overview of actual avaiable firewall solutions focused on Open Source and Enterprise solutions.

### System Level Firewalls

Within the Linux systems there are essential tools to control network traffic and indivudal hosts. The most common solutions are iptables, nftables, ufw and firewalld. They are varying from complexity, flexibility and how user friendly they are.

Iptables – Is a tradtional packet-filtering framework which is included directly in the Linux kernel.

Nftables – Is like a new version of Iptables with many advantages like rule changes at once. All rules like Ipv4 and Ipv6 are possible in the same code base. [4]

ufw

### Open Source Firewalls

An open-source firewall is a firewall solution whose source code is publicly available and freely accessible. The source code can be reviewed and modified. These firewalls are typically community-driven and cost-effective. One of a big advantage is also the opportunity to customize the product. Therefore many projects offer commercial services like professional support and additional enterprise features. This helps the organisation behind to maintain the software and to fund the needed infrastructure. [5]

**pfSense -** Is a flexible, open-source firewall and router platform that offers NAT, packet filtering, and next-generation firewall features. It supports multiple interfaces, scales well, and provides a command-line interface for advanced configuration.

**OPNsense** - Is a user-friendly, web-managed next-generation firewall with built-in intrusion detection (IDS), web filtering, and VPN support. It combines strong security features with an intuitive interface, making it suitable for diverse network environments.

**VyOS** - Is a community-driven, fully open-source firewall that aims for high availability and uptime. It includes stateful inspection, NAT, and routing features, and is often used in hardware appliances for continuous performance.

**ClearOS** - Is a simple, stateful firewall solution aimed at users with basic network protection needs. It is easy to manage and configure, though it lacks advanced features like NAT and packet filtering.

### Enterprise Firewalls

Enterprise firewalls are typically closed-source and unlike open-source alternatives, not freely accessible or modifiable. These solutions often come with a wide range of additional services, including professional support, subscription-based threat intelligence, and service-level agreements (SLAs), making them particularly attractive for organizations that require guaranteed uptime, vendor accountability, and integrated security management. Well known brands are Palo Alto, Fortinet, Cisco, Check Point, Sophos.

### Requirements for the firewall solution

Early in the project phase, it was essential to determine the firewall platform for the project. The Primary goal was to build an extended interface for our specific need than developing a complete firewall solution from scratch. The selected system needed to provide a robust and flexible base. For evaluation purpose a minimal set of functional and technical requirements was defined:

**Open-Source Licensing** The firewall must be distributed under an open-source license. This ensures the solution is free of licensing costs and avoids vendor lock-in. Furthermore, open-source access enables future customization, which may become necessary in advanced stages of the project.

**Self-Hosting on Local Infrastructure** The solution must be fully deployable on the organization’s own infrastructure without relying on any external cloud components or third-party services. This is essential for maintaining full control over the processed data, meeting forensic standards, and following with the information security and data protection policies (ISDS) of the City of Zurich.

**API Support** A modern well documented API. The project aims to integrate firewall control into a custom web interface, without the need to interact directly with the firewall’s native user interface. API support is also essential for any automation workflows.

**Device-Aware Rule Management** The firewall must support rules that can be defined per device, based on identifiers such as IP or MAC addresses. This capability is necessary for implementing granular access policies. For example, permitting one device to access a specific IP while restricting another.

**Active Development** The firewall must have an ongoing development with regular updates and security patches.

**Feature-Rich Environment** The solution should include a wide range of network security features out of the box. Minimize the need for third-party tools and to have the capability for later processing steps.

### Compare firewall solutions based on requirements

As the enterprise firewall solutions (e.g., Palo Alto, Fortinet, Cisco, Check Point, and Sophos) are proprietary and do not meet the open-source licensing requirement, they were excluded from the comparison table below. These products are therefore considered out of scope for this project, which is focused solely on open-source, self-hosted solutions.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Firewall** | **Open Source** | **Self-Hosting** | **API Support** | **Device-Aware Rules** | **Active Development** | **Feature-Rich** |
| iptables[6] | Yes | Yes | No | Limited | Yes | Moderate |
| nftables[7] | Yes | Yes | No | Limited | Yes | Yes |
| pfSense[8] | Yes | Yes | Limited | Yes | Yes | Yes |
| OPNSense[9] | Yes | Yes | Yes | Yes | Yes | Yes |
| VyOS[10] | Yes | Yes | Limited | Limited | Yes | Yes |
| ClearOS[11] | Yes | Yes | Limited | No | No | Decreasing with time as no active development is done |

Table 1: Comparison of different firewall solutions

After evaluating the available open-source firewall projects and consulting their official documentation and community resources, **OPNsense** was selected as the source firewall for this project. OPNsense has an actively maintained platform with a huge user and developer community.

A great advantage of OPNsense is the officially supported and documented API, which already covers most when not all needed functions for the later project. The project offers regular updates, an open roadmap and detailed release notes.

Although the project would be possible on a base like nftables which is integrated directly into the Linux kernel and with the needed implementation time it would be possible as well. However, the expected additional development time was looked at as too extensive, this is why a ready to use solution was selected.

## Hardware evaluation

Hardware selection was not a primary focus at the outset of the project. After the decision was made to use OPNsense as the firewall platform, further research was made using online sources, including general searches (e.g., Google) to identify suitable hardware configurations for a cost-effective setup.

The official OPNsense hardware appliance offerings [12] were reviewed but ultimately looked as out of scope due to their cost. These systems are primarily targeted at enterprise customers, with prices often exceeding the budget. The project aimed to stay within a budget of just a few hundred Swiss francs, ideally utilizing small-form-factor, low-cost hardware such as a Raspberry Pi 5.

While the Raspberry Pi 5 initially appeared to be a promising candidate due to its affordability, it was ultimately excluded. Although it meets the official recommended hardware requirements for OPNsense [13], including:

**Processor:** Minimum 1.5 GHz multi-core CPU

**RAM:** 8 GB

It failed to meet the additional project-specific requirements, which were essential for practical deployment in a forensic lab environment:

**Industrial enclosure:** The device must be housed in a durable case suitable for continuous operation in a professional setting.

**Networking capabilities:** The device must support at least one WAN port and two or more LAN ports (e.g., one for Wi-Fi and one or more for Ethernet-connected devices).

However rugged enclosures for Raspberry Pi exist, they are still part of a DIY solution and often require additional adapters or USB-to-Ethernet dongles, which compromise reliability and performance. When evaluating networking options specifically, it became clear that no off-the-shelf Raspberry Pi configuration could fulfil the requirements.

In the OPNsense forum [14] many community members shared links to low-cost firewall hardware available on platforms such as AliExpress. However, this option was not considered further due to concerns regarding warranty coverage, lack of technical support, and the potential for long delivery times - often several weeks. These factors would have introduced unnecessary delays and risk to the project timeline.

### Final Decision for Protectli V1410

Finally, the decision was made to purchase hardware from Protectli, a US company with a location in Rossdorf, Germany. Protectli is specialized in hardware for open-source firewall appliances and is well-regarded for its compatibility with OPNsense. After evaluating several of the company’s models, the Protectli V1410 was selected, as it fulfilled all functional requirements for the project, including port availability, compact form factor, hardware durability, and full compatibility with the OPNsense software platform.

Selected Device: **Protectli V1410**

**CPU**: Intel® N5105 Quad-Core, 2.0 GHz (Turbo up to 2.9 GHz)

**Networking**: 4 × Intel® I226-V 2.5 GbE RJ-45 Ethernet ports

**Memory**: 8 GB LPDDR4 (on-board)

**Storage**: 32 GB onboard eMMC and 250 GB Kingston NVMe (NV2-250G)

**Expansion**: M.2 slots for optional Wi-Fi or LTE modules

**Power Supply**: 12 V with screw-in connector (included)

**Other**: Fanless, silent operation, coreboot-supported, compact form factor

**Price**: €284.55 (excluding VAT, as of February 17, 2025)

This hardware provides sufficient performance and operational flexibility for use in a forensic laboratory environment. It fully meets the hardware requirements for OPNsense and includes additional storage capacity for extended logging or future use cases. The fanless design and industrial-grade build further support long-term maintainability and stability under continuous operation.

## Remote Wipe

«Remote wipe is a security feature that allows a network administrator or device owner to send a command that remotely deletes data from a computing device. It's primarily used to erase data on a device that has been lost or stolen, so the data won't be compromised if it falls into the wrong hands.» [15]

### Differences between Wiping and Deletion

Wiping means the process which ensures that the data stored on the device is irreversibly destroyed and to make the recovery impossible. This is typically achieved by overwriting the specific storage area or the entire storage with random data. On the other hand, a simple file deletion involves the system removing only the references to the data, while the actual data remains intact on the storage device. This data can often be recovered and made visible again using specialized tools and software, provided that the storage has not been overwritten due to the need for disk space.

### Apple iOS remote wipe

To understand how the remote wipe of an Apple iOS device works it is important to understand how the apple system works. When an iOS device first set up or after each factory reset it generates a random volume key, “media key”. The key is stored locally on the device in a secure location called “Secure enclave”.

All data on the device including app data, system files and user data will be encrypted using the media key. Without the media key it is not possible to access the stored data on the device. If now a remote wipe command is triggered by the Mobile Device Management (MDM) or directly through the iCloud account. The device will as soon the command is received respond with an Acknowledgment, and it will instantly perform the wipe command. [16]

It is important to know does with the MDM capability the wipe command can not only come from Apple himself it also can come from various companies if the device is a managed device. Just to list some of the companies: Invanti, Jamf, Microsoft, MobileIron.[17]

The remote wipe command is designed to remove the media key only. The data himself is not wiped as this would be a time intensive process. However, by removing the media key the stored data on the file system is not accessible anymore, as the decryption key is missing. This prevents access to the data even for the owner himself as there is no known possibility to decrypt the data anymore.

### Google Android

The Android operating system works slightly different than the iOS operating system. With Android 10 the system was no longer using full-disk encryption (FDE) the system changed to file-based encryption (FBE). After the first set up or after a factory reset, the system creates a set of encryption keys. The Credential Encrypted key (CE) and the Device Encrypted key (DE) which are stored in a secure hardware storage like Trusted Execution Enviroment (TEE) or Secure Element. The Android ecosystem uses a layered key system, where all data on the device, including user data, app data and most system files are encrypted with its own unique key. These unique keys using the CE or DE key for encryption. The CE key is only available after the user has authenticated. Without the CE key it is not possible to decrypt or access the stored data on the device.[18]

The remote wipe command can be triggered with Google’s Find My Device Network or also like Apple over an MDM system. When the device receives this command, it may respond with an acknowledgment if done via MDM, after it will immediately initiate the remote wipe process. This process is primarily designed to destroy the CE and DE keys, the actual data is not overwritten. Without the encryption keys, the file data becomes inaccessible and unrecoverable[19], [20].

The wipe command can be issued not only by Google, but also by various enterprise providers if the device is enrolled in their MDM system. These include providers like Microsoft, VMware Workspace ONE, IBM MaaS360, Invanti, MobileIron, Samsung Knox Manage and many others.[21]

# Implementation

This chapter describes the implementation of the system developed in the context of this thesis. The goal was to create a practical solution that allows centralized control. The core of the system is a custom-built web application that communicates with an open-source firewall (OPNsense) via its API.

The implementation covers several key components: The design and development of the web interface, the backend logic that processes user input and interacts with the firewall, the integration with the OPNsense API, the logging and storage of all relevant data in a database, and the deployment of the system on suitable hardware.

## Overview of the System Architecture

The prototype developed in this thesis consists of several integrated components that together form a modular system. The main components of the system are:

**Firewall:** The open-source firewall OPNsense (OS 24.7 - Thriving Tiger) acts as the foundation of the system. It enforces all network traffic rules and blocks all connections by default unless an explicit PASS rule is defined. All interactions with the firewall are done via its officially supported API, which enables dynamic rule management and log queries. Installed on a Protectly V1410.

**WLAN Access Point:** A dedicated wireless access point in this case a “TP-Link Omada EAP610GP-DESKTOP WLAN Access Point 1201” provides connectivity for mobile devices.

**Web Interface / Web Framework:** The user interface is implemented using the Django web framework (5.1.6). It allows forensic examiner to manage devices, view logs and manage firewall rules. The interface is quite minimalistic, focused on the core use cases.

**Backend Logic:** The core application logic is written Python (3.13.1) and organized into multiple files within the Django framework. It processes user input from the web interface, manage database operations and communicates with the OPNsense API.

**Database:** The system uses SQLite as a lightweight relational database to store device metadata, firewall rules and network logs. SQLite was choosen to simplify development and deployment. Django’s model-view architecture provides abstraction over the database layer, making future migration to a more robust database system (such as PostgreSQL or MySQL) straightforward if needed.

Figure 1 illustrates how the individual components interact with each other:

A diagram of a router and a computer

AI-generated content may be incorrect.

Figure 1: Overview about the system setup

## Prototyping and Preliminary Testing

Before beginning the implementation of the full system, several exploratory prototyping steps were undertaken using simple Python console scripts. As the project domain was entirely new to the author, this phase was important for identifying potential challenges related to device behavior, network resolution, and firewall integration.

One of the key uncertainties at the beginning was how connected devices would resolve hostnames - whether they would store and reuse IP addresses or resolve them dynamically for each request. Additionally, it was unclear how large-scale services (e.g., Google, Apple) would behave in terms of load balancing, content distribution, or dynamic IP usage, which could affect firewall rule effectiveness and logging clarity.

To address these uncertainties, early prototypes focused on:

* Creating simple scripts to parse firewall logs
* Observing device behavior when placed in a restricted network
* Testing hostname resolution patterns
* Understanding how network traffic is structured under different conditions

### Firewall Configuration During Prototyping

For the initial testing environment, the following configuration strategy was applied to the OPNsense firewall:

**Internal Access Rule** - All traffic from the internal LAN\_net to the firewall itself was allowed. This ensured continued access to the web GUI and API during testing, regardless of other firewall restrictions.

**Global Deny Rule** - A final block-all rule was implemented, denying all traffic not explicitly allowed, both inbound and outbound. This ensured that newly connected devices would not have any internet access unless a rule was programmatically defined.

### API Integration and Log Retrieval

A key objective of the prototype phase was to verify whether the OPNsense API could be used reliably to:

* Configure firewall rules
* Retrieve log entries, particularly for blocked connections

Initial attempts successfully returned a list of firewall rules that were previously created via the API. Rules configured manually through the web interface, however, were not visible through the API, this seemed to be acceptable as the final system will manage all rules programmatically.

Early API calls for log retrieval were unsuccessful, but after additional testing with alternative endpoints and improved request formatting, access to firewall logs was successfully achieved. This confirmed that sufficient data could be programmatically accessed to support the planned functionality of the final system.

### MAC Address Handling and Device Identification

During early tests, a real-world scenario involving an Apple iPhone and Apple Watch highlighted an important challenge for the planned firewall rule management system: the dynamic and unpredictable behavior of MAC address assignments.

When connecting an iPhone to the forensic WLAN network for the first time, it was observed that the device appeared with two distinct IP addresses and MAC addresses within the same time frame. Initially, this raised concerns about MAC address randomization[22], a known privacy feature in modern mobile operating systems. However, further investigation revealed that the second MAC address and IP pair belonged to a paired Apple Watch, which had automatically joined the network shortly after the iPhone.

After manually deleting the DHCP leases in OPNsense, the iPhone reconnected and maintained a consistent MAC address (02:89:49:2b:5c:f2). Not as expected that it would change its MAC after each session due to iOS's default privacy settings. Nevertheless, this incident illustrated the need for the firewall management system to gracefully handle MAC address changes, particularly for devices that may exhibit multiple addresses over time.

To mitigate this, the concept of a centralized device identifier was noted for later implementation: each device is assigned a unique “Asservaten Nummer” (i.e., an evidence ID). All known MAC addresses associated with a device are then linked to this ID in the database.

### DNS Rules set to Access

During initial firewall logging tests, it was observed that outbound traffic from devices such as the test iPhone appeared to target only internal Ips. For example, repeated connection attempts from 192.168.5.155 (the test iPhone) to 192.168.5.1 (the OPNsense firewall). These requests were being blocked, and no external IP addresses were visible in the logs, which initially suggested minimal or idle network behavior.

Upon further investigation, it became clear that these blocked requests were a result of the iPhone's attempt to resolve DNS queries using the default gateway IP address (i.e., the firewall). This is standard behavior in most networked devices when no external DNS server is explicitly defined. Apple mobile phones, by default, automatically use the network's assigned DNS server. The system automatically assigns the router as the DNS resolver, expecting it to forward queries or to act as a local caching DNS server.

This highlighted an important architectural requirement:

DNS resolution must be permitted for connected devices, otherwise no hostname lookups can occur, and as a result, applications will not initiate any outbound connections to services.

To address this, a decision was made to allow outbound DNS queries to a limited set of public DNS resolvers. This enables name resolution while still tightly controlling what happens after resolution.

The following DNS providers were selected for the prototype setup:

|  |  |  |
| --- | --- | --- |
| **IP Address** | **DNS Provider** | **Notes** |
| 1.1.1.1 | Cloudflare |  |
| 8.8.8.8 | Google | No Android Devices |
| 9.9.9.9 | Quad9 |  |

Table 2: Public DNS resolvers overview

These resolvers were explicitly allowed through the firewall, and the connected devices were configured to use them. This ensured that DNS lookups succeeded, allowing applications to discover the IP addresses of their services and initiate traffic.

### Impact of Apple Private Relay on Firewall Logfiles

During further tests, it was observed that when accessing websites such as 20min.ch or google.ch using the Safari browser on an iPhone, the expected destination IPs were not present in the firewall logs. Instead, the logs consistently showed traffic directed toward Apple-owned IP addresses.

Upon further analysis, it was determined that the test device had Apple iCloud Private Relay[23] enabled. This feature actually just available to users with a paid iCloud subscription, acts as a privacy-preserving proxy system. When active, it masks the destination IP addresses of web requests made through Safari, routing traffic through Apple-operated and third-party relay servers. As a result, the true destination of a web request is obscured, and only the intermediate Apple-controlled IP addresses are visible to the firewall.

### IP Enrichment and Company Identification

During testing, it quickly became evident that a raw list of IP addresses alone often exceeding 100 entries per device, was insufficient for effective analysis. Simply presenting numeric IP addresses without any contextual information made it difficult to determine which blocked connections was ok and should be added to the allowed firewall rules list. Without additional data such as hostnames, ISP information, or service ownership - manual evaluation was not a deal. As a result, a automatic ip enrichment with additional information was needed.

**Initial Attempt - Reverse DNS Lookup**

The first approach implemented was a reverse DNS lookup, using the built-in socket library in Python. In some cases, this method generated helpful results, with hostnames clearly indicating the associated service or company (e.g., whatsapp-chatd-edge-shv-01-zrh1.facebook.com, a80-67-82-201.deploy.static.akamaitechnologies.com). However, this method proved unreliable:

* Many IP addresses did not resolve to any hostname at all
* Others resolved to generic or uninformative names
* Reverse DNS lacks consistency across regions and providers

**Improved Approach: IP Metadata API**

To overcome these limitations, the system was extended to query external IP metadata APIs in order to gather more reliable and structured information per IP address. Several services were evaluated see Table 3, based on functionality, rate limits and pricing the sercice ip-api.com was selected for developing. The service ipinfo.io has also a great offer but with a really limited query answer on the free tier.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Service** | **Free** | **Commercial use** | **Requests** | **Note** |
| ipregistry.co[24] | Limited | Yes | 100’000 | 100’000 requests in free account |
| ipapi.com[25] | Yes | Yes | 0.8h / 100m |  |
| ipwhois.io[26] | Yes | No | 13.8h / 10’000m |  |
| geo.ipify.org[27] | Limited | Yes | 500 | 500 requests in free account |
| ipgeolocation.io[28] | Yes | Yes | 41.6h / 30’000m |  |
| ipdata.co[29] | Yes | No | 62.5h / 45’000m |  |
| ip-api.com[30] | Yes | No | 2700h / 1’944’000m | 45 requests per minute |
| ipapi.co[25] | Yes | No | 41.7h / 30’000m |  |
| ipinfo.io[31] | Yes | Yes | Unlimited | Limited details |

Table 3: IP API Service overview

ip-api.com was chosen due to:

* Detailed ISP data (useful for grouping by company)
* A clear and usable free tier (45 requests per minute)
* And an affordable premium option ($13.30/month) with no rate limits, suitable for future production use.

To respect the free tier limits during development, API requests were sent with a 1.5-second delay between them. This strategy ensured stability and prevented running into the rate limit.

The following query shows the response from the API for "52.97.201.242":

{

"query": "52.97.201.242",

"status": "success",

"continent": "Europe",

"continentCode": "EU",

"country": "Switzerland",

"countryCode": "CH",

"region": "ZH",

"regionName": "Zurich",

"city": "Zurich",

"district": "",

"zip": "",

"lat": 47.3768,

"lon": 8.5416,

"timezone": "Europe/Zurich",

"offset": 7200,

"currency": "CHF",

"isp": "Microsoft Corporation",

"org": "Microsoft Corporation",

"as": "AS8075 Microsoft Corporation",

"asname": "MICROSOFT-CORP-MSN-AS-BLOCK",

"mobile": false,

"proxy": false,

"hosting": true

}

The implementation showed excellent results in practice. For most IP addresses, clear metadata could be retrieved, including ISP name, autonomous system (ASN), organization, and country. The ISP field in particular proved to be reliable for associating IPs with companies.

Prototyping Insights: Apple iCloud, IP Grouping, and Application-Level Control

Further tests were conducted using an iPhone connected to the forensic WLAN with all traffic blocked by default. The phone was logged into the same iCloud account as a MacBook, which had unrestricted internet access. Both devices had Bluetooth enabled, raising the question of whether the iPhone could receive "Find My Device" commands via indirect Bluetooth or cloud-assisted channels, even when isolated from direct internet access.

iCloud Find My Device Behavior

Contrary to initial expectations, the iPhone did not behave like an AirTag. While the MacBook issued a "play sound" command via iCloud, the iPhone—being network-isolated—did not update its location or play a sound. This suggests that Find My Device commands are queued and require a direct internet connection to be received. No background peer-to-peer Bluetooth relay was observed in this context.

IP Visibility and Company Grouping

During extended testing, the system successfully captured connection attempts from the mobile device, even when all traffic was initially blocked. The destination IP addresses were collected and grouped by associated companies, based on WHOIS data and IP metadata. Notably:

The iPhone attempted to reach 41 IP addresses, all within the 17.0.0.0/8 range, which belongs to Apple Inc..

No connections outside this range were initiated while the device remained idle or under system-level Apple processes.

This capability confirms that the minimal viable functionality of the system has been achieved: the prototype is able to log destination IPs per device, resolve their ownership, and allow or deny access selectively.

Application-Level Access Tracing

When launching new mobile applications, the system clearly captured initial communication attempts. These could then be individually reviewed and selectively permitted. This enables granular, app-specific firewall management, allowing forensic analysts to observe and control traffic without enabling full internet access.

This approach presents a powerful method for:

Isolating specific apps during testing or evidence collection.

Approving only necessary connections to enable functionality.

Maintaining a closed network perimeter while still interacting with cloud-based apps on a case-by-case basis.

Limitations with Safari and CDNs

During practical browsing tests using Safari (e.g., accessing 20min.ch or Blick.ch), it was noted that many Content Delivery Network (CDN) IPs must be allowed to successfully load the site. This poses a usability challenge, as CDNs typically use a large number of dynamic IPs, making fine-grained allowlisting more complex.

While this behavior may not be problematic in structured forensic use cases (where browser access is limited or controlled), it may require additional rule sets or IP range allowances for more general usage scenarios.

Some considerations out of the tests

Physical access to the device

* Apple Relay off
* MAC Anonimazer off
* DHCP set to

## Backend Implementation

## Frontend Implementation

## Database Design

## Integration with OPNsense

## Rule Application Logic

## Logging

## Challenges Encountered

## Summary

# Analysis / Results

# Discussion / Conclusion

Remote Wipe… some research business device…References

[1] M. Mohler, ‘Der Faradaysche Käfig’. Accessed: Jan. 13, 2025. [Online]. Available: https://lp.uni-goettingen.de/get/text/833

[2] K. Ingham and S. Forrest, ‘Network Firewalls’, University of New Mexico, 2002. [Online]. Available: http://iar.cs.unm.edu/~forrest/publications/firewalls-05.pdf

[3] Palo Alto Networks, ‘The History of Firewalls | Who Invented the Firewall?’ Accessed: Aug. 05, 2025. [Online]. Available: https://www.paloaltonetworks.com/cyberpedia/history-of-firewalls

[4] A. Dubey, ‘Comprehensive Guide to Linux Firewalls: iptables, nftables, ufw, and firewalld’, Medium. Accessed: Aug. 05, 2025. [Online]. Available: https://medium.com/@amandubey\_6607/comprehensive-guide-to-linux-firewalls-iptables-nftables-ufw-and-firewalld-9e86e0a49979

[5] C. Dilmegani, ‘Top 7+ Open Source Firewall Options in 2025: Features & Types’, AIMultiple Research. Accessed: Jun. 02, 2025. [Online]. Available: https://research.aimultiple.com/open-source-firewall/

[6] ‘iptables(8) - Linux manual page’. Accessed: May 14, 2025. [Online]. Available: https://man7.org/linux/man-pages/man8/iptables.8.html

[7] ‘nftables wiki’. Accessed: May 14, 2025. [Online]. Available: https://wiki.nftables.org/wiki-nftables/index.php/Main\_Page

[8] ‘pfSense® - World’s Most Trusted Open Source Firewall’. Accessed: May 14, 2025. [Online]. Available: https://www.pfsense.org/

[9] OPNSense, ‘OPNsense Documentation’, OPNsense. Accessed: Feb. 12, 2025. [Online]. Available: https://docs.opnsense.org

[10] ‘VyOS – Open source router and firewall platform’, VyOS. Accessed: May 14, 2025. [Online]. Available: https://vyos.io/

[11] ‘ClearOS’. Accessed: May 14, 2025. [Online]. Available: https://clearos.com/

[12] ‘Hardware – OPNsense® Shop’. Accessed: May 14, 2025. [Online]. Available: https://shop.opnsense.com/product-categorie/hardware-appliances/

[13] ‘Hardware sizing & setup — OPNsense documentation’. Accessed: May 14, 2025. [Online]. Available: https://docs.opnsense.org/manual/hardware.html

[14] ‘Hardware and Performance’, OPNsense Forum. Accessed: May 14, 2025. [Online]. Available: https://forum.opnsense.org/index.php?board=21.0

[15] ‘What is a Remote Wipe? | Definition from TechTarget’, Search Mobile Computing. Accessed: May 14, 2025. [Online]. Available: https://www.techtarget.com/searchmobilecomputing/definition/remote-wipe

[16] Apple Inc, ‘Apple Platform Security’, p. 302, Dec. 2024.

[17] ‘List of mobile device management software’, *Wikipedia*. May 10, 2025. Accessed: May 14, 2025. [Online]. Available: https://en.wikipedia.org/w/index.php?title=List\_of\_mobile\_device\_management\_software&oldid=1289694990

[18] ‘File-based encryption’, Android Open Source Project. Accessed: May 15, 2025. [Online]. Available: https://source.android.com/docs/security/features/encryption/file-based

[19] ‘Find, secure or erase a lost Android device - Android Help’. Accessed: May 15, 2025. [Online]. Available: https://support.google.com/android/answer/6160491?hl=en-GB

[20] ‘Wipe corporate data from a device - Google Workspace Admin Help’. Accessed: May 15, 2025. [Online]. Available: https://support.google.com/a/answer/173390?hl=en

[21] ‘Android Business Device Solutions Directory - Android Enterprise - EMMs’, Android Enterprise. Accessed: May 15, 2025. [Online]. Available: https://androidenterprisepartners.withgoogle.com/emm/

[22] ‘Privacy features when connecting to wireless networks’, Apple Support. Accessed: May 15, 2025. [Online]. Available: https://support.apple.com/en-gb/guide/security/secb9cb3140c/web

[23] Apple Inc, ‘iCloud Private Relay Overview’, p. 11, Dec. 2021.

[24] Elaunira SARL, ‘Affordable IP Geolocation and Threat Intelligence Pricing - Ipregistry’. Accessed: May 15, 2025. [Online]. Available: https://ipregistry.co

[25] Kloudend, Inc., ‘ipapi - IP Address Lookup and Geolocation API | No SignUp’. Accessed: May 15, 2025. [Online]. Available: https://ipapi.co

[26] ipwhois.io, ‘IP Geolocation API - Pricing’. Accessed: May 15, 2025. [Online]. Available: https://ipwhois.io

[27] ipify.org, ‘IP Geolocation API - Try Our IP Location API Free Of Charge’. Accessed: May 15, 2025. [Online]. Available: https://geo.ipify.org

[28] JFreaks Software Solutions, ‘IP Geolocation API Pricing’, IP Geolocation API Pricing. Accessed: May 15, 2025. [Online]. Available: https://ipgeolocation.io

[29] ipdata.co, ‘IP Geolocation API with Threat Intelligence’. Accessed: May 15, 2025. [Online]. Available: https://ipdata.co

[30] ip-api.com, ‘IP-API.com - Geolocation API’. Accessed: May 15, 2025. [Online]. Available: https://ip-api.com

[31] IPinfo, ‘Trusted IP Data Provider, from IPv6 to IPv4’. Accessed: May 15, 2025. [Online]. Available: https://ipinfo.io

# Appendix

Hier sind die in der Arbeit referenzierten Anhänge aufzuführen.

# Declaration of Originality

Bitte Wortlaut aus «Merkblatt Erstellung Abschlussarbeit in CAS, DAS und MAS» übernehmen.

1. Faraday room or Faraday cage, which prevents communication from or into the room. A great and quite simple setup was made by the University of Gottingen [1]. [↑](#footnote-ref-1)
2. Brute force refers to methods that systematically attempt all possible passcode combinations. [↑](#footnote-ref-2)