



Eidgenössische Technische Hochschule Zürich  
Swiss Federal Institute of Technology Zurich

# **Design and Implementation of Functionality, Security and Maintainability Enhancements for SCIONLab Coordination Service**

Bachelor Thesis

Claude Hähni

October 7, 2017

Advisors: Prof. Dr. Adrian Perrig, Prof. Dr. David Hausheer, Dr. Ercan Ucan  
Department of Computer Science, ETH Zürich



---

## Abstract

In order to increase SCION's [1] availability and distribute it further, the SCIONLab project was created. SCIONLab is a publicly available version of SCION that allows research institutions and other interested parties to easily join the SCION testbed environment, thus making it possible to experiment with its unique capabilities. One goal of SCIONLab is to reduce the administration overhead for an institution to join and manage their own ASes. This management is done through the *SCIONLab Coordination Service* which serves two purposes. First, it offers an easy-to-use interface, enabling interested parties to register and download AS configurations to deploy onto their own hardware in order to become part of the SCION network. Moreover, *SCIONLab Coordination Service* is designed to be a global intermediary between the local management services of different ASes, managing connections between one another. The new features and components developed in this thesis aim to render the pre-existent Coordination Service much more robust and secure and extend its functionality to a point where it can be made available to the public. New features include the ability to send emails from *SCIONLab Coordination Service*, a mechanism to verify and activate new users, a counter measurement against bot abuse as well as new mechanisms to improve testing and deployment.

---

## **Acknowledgments**

First and foremost, I would like to thank Prof. Adrian Perrig and the whole Network Security Group at ETH Zurich for the opportunity of working on this project.

I would like to express my sincere gratitude to my supervisors Prof. David Hausheer and Dr. Ercan Ucan for the many discussions and their support throughout this thesis. Without their input, this thesis would not have been possible.

Finally, I would like to express my profound gratitude to my family for the continuous encouragement and the unconditioned support throughout my studies. Thank you.

---

# Contents

---

Abstract . . . . .	i
Acknowledgments . . . . .	ii
<b>Contents</b>	<b>iii</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Background</b>	<b>3</b>
2.1 SCION - A Future Internet Architecture . . . . .	3
2.1.1 Network Structure . . . . .	3
2.1.2 Isolation . . . . .	3
2.1.3 Path Selection . . . . .	4
2.1.4 Scalability . . . . .	5
2.2 SCION Architecture . . . . .	5
2.2.1 Border Routers . . . . .	5
2.2.2 Beacon Servers . . . . .	6
2.2.3 Path Servers . . . . .	6
2.2.4 Certificate Servers . . . . .	6
2.2.5 Name Servers . . . . .	6
2.3 SCION AS Management Framework . . . . .	6
2.3.1 Local Management Service . . . . .	7
2.3.2 SCIONLab Coordination Service . . . . .	8
2.4 SCIONLab Experimentation Environment . . . . .	8
<b>3 Related Work</b>	<b>11</b>
3.1 PlanetLab . . . . .	11
3.1.1 PlanetLab Node Management . . . . .	11
3.2 GENI . . . . .	12
3.3 Fed4FIRE . . . . .	13
<b>4 Requirements Engineering</b>	<b>15</b>

4.1	Functional Requirements . . . . .	15
4.2	Non-Functional Requirements . . . . .	16
<b>5</b>	<b>Architecture Overview</b>	<b>17</b>
5.1	Design Overview . . . . .	17
5.2	Components . . . . .	18
5.2.1	Database . . . . .	18
5.2.2	Back End Server . . . . .	18
5.2.3	Web Interface . . . . .	18
5.3	Services . . . . .	19
5.3.1	Postmark . . . . .	19
5.3.2	Google reCAPTCHA . . . . .	19
5.4	Continuous Integration . . . . .	19
5.5	Deployment . . . . .	20
<b>6</b>	<b>Implementation</b>	<b>21</b>
6.1	Initial State . . . . .	21
6.2	Email Address Verification . . . . .	21
6.2.1	Email package . . . . .	23
6.2.2	Adapted Components . . . . .	25
6.3	Manual User Activation . . . . .	26
6.3.1	Role based access control . . . . .	26
6.3.2	Adapted Components . . . . .	26
6.4	CAPTCHA Integration . . . . .	29
6.4.1	Google reCAPTCHA . . . . .	29
6.4.2	Adapted Components . . . . .	31
6.5	CircleCI Integration . . . . .	32
6.5.1	Environment . . . . .	32
6.5.2	Dependencies . . . . .	33
6.5.3	Testing . . . . .	33
6.6	Deployment with Ansible . . . . .	34
6.6.1	Hosts . . . . .	34
6.6.2	Files . . . . .	35
6.6.3	Parameters . . . . .	35
6.6.4	Tasks . . . . .	35
<b>7</b>	<b>Evaluation</b>	<b>37</b>
7.1	Deployed Components . . . . .	37
7.1.1	Robust User Registration . . . . .	37
7.1.2	Invitation Based Registration . . . . .	40
7.1.3	Email Functionality . . . . .	41
7.1.4	Integrated Testing . . . . .	42
7.1.5	Effortless Deployment . . . . .	42
7.2	Requirements Evaluation . . . . .	43

<b>8</b>	<b>Conclusion</b>	<b>47</b>
8.1	Summary of Achievements . . . . .	47
8.2	Future Work . . . . .	47
<b>A</b>	<b>Security and Maintainability Enhancements</b>	<b>49</b>
A.1	Ensuring Secure Operation . . . . .	49
A.1.1	Server Crash on Login . . . . .	49
A.1.2	Nil Pointer When Accessing User Information . . . . .	49
A.2	Maintainability . . . . .	50
A.2.1	Duplicate Configurations . . . . .	50
A.2.2	Obsolete HTTP Handling Code . . . . .	50
A.2.3	Obsolete Database Queries . . . . .	50
A.2.4	Outdated Dependencies . . . . .	50
	<b>Bibliography</b>	<b>53</b>





## Chapter 1

---

# Introduction

---

SCION [1] is a proposed Future Internet Architecture (FIA) with the goal to improve on today's Internet which suffers from various shortcomings. Many of these shortcomings are related to the IP and BGP protocols which together form the narrow waist in the protocol stack. In particular, the Internet lacks transparency and control over packet routing and the global nature of its protocols makes it scale badly and suffer from outages. Instead of keeping on building on this unstable foundation, SCION tackles these problems with a fundamental re-design of the Internet's core protocols. It aims to provide high availability, control, transparency and secure end-to-end communication in networks. [2]

As such, SCION offers convincing opportunities especially for the industry which can profit greatly from its unique features. The adoption of SCION is not an obstacle as it interfaces nicely with today's architecture. Still, it is necessary that businesses, research institutions and other interested parties are able to test its unique features. For this exact purpose the SCIONLab Experimentation Environment was created. It's a project aiming to provide a testbed environment where users join the SCION network with their own computation and actively contribute to the network, thus allowing for realistic testing scenarios. To facilitate the process of joining SCIONLab and managing nodes, SCIONLab provides an easy-to-use tool. The *SCIONLab Coordination Service* offers an intuitive interface enabling interested parties to register and download AS configurations to deploy onto their own hardware in order to become part of the SCION network. Moreover, *SCIONLab Coordination Service* is designed to be a global intermediary between the local management services of different ASes, managing connections between one another. Additional information about SCIONLab and *SCIONLab Coordination Service* is available in Sections 2.4 and 2.3.2 respectively.

This thesis aims to improve on the existing Coordination Service, making it easier and more appealing for end users as well as significantly improv-

ing its functionality, security and maintainability. These improvements in particular consist of the following enhancements and extensions: (1) An email verification system used to validate user registrations, (2) an administrator panel that allows to review registrations and activate users, (3) the implementation of a CAPTCHA, in order to protect the service against bot accounts, (4) interfacing with a continuous integration solution for faster testing of *SCIONLab Coordination Service*, (5) leveraging an automation engine for easier deployment onto multiple hosts and (6) enhancing the service with numerous improvements addressing speed, cleanliness and security of the system.

The rest of the thesis is structured as follows: Chapter 2 contains an overview of the SCION architecture, briefly describing its components. In Chapter 3, testbed management systems of other network experimentation environments are discussed. Chapter 4 presents design objectives constrained by functional and non-functional requirements. Then, based on the requirements elicited in the previous chapter, we outline the proposed *SCIONLab Coordination Service* architecture with its associated components and services in chapter 5. Chapter 6 describes the process of implementing enhancements dictated by the gathered requirements. It's shown how *SCIONLab Coordination Service* was transformed to match the proposed architecture. To sum up, Chapter 7 evaluates the enhancements made in chapter 6 and discusses achievements of design objectives from Chapter 4. Finally, in chapter 8 we come to a conclusion and provide an outlook on future work.

## Chapter 2

---

# Background

---

This chapter gives an overview of SCION, briefly describing the core concepts, the network structure and essential components, all required to understand the context of this thesis. For an in-depth understanding please refer to [1].

## 2.1 SCION - A Future Internet Architecture

SCION<sup>1</sup> is a future Internet architecture with the goal to offer a "highly available", secure and transparent "point-to-point packet delivery" infrastructure. [1, Page 17] SCION tackles problems with respect to political and economical issues, today's internet suffers from. These properties can even be fulfilled in presence of malicious network members.

To realize above ambitions SCION makes use of sophisticated techniques. The most important of these concepts are described below:

### 2.1.1 Network Structure

SCION organizes Autonomous Systems (ASes) into entities called Isolation Domains (ISDs). A selection of ASes in an ISD are designated core ASes responsible for administering the ISD. For example, the core ASes negotiate the roots of trust used for authentication. An AS can join an ISD by connecting to an AS that already is a member of the ISD. Joining an ISD implies agreeing to the policies governing the ISD. [1, Chapter 2]

### 2.1.2 Isolation

SCION divides the network into isolated entities called Isolation Domains and lets these ISDs manage their own part of the network, including the elec-

---

<sup>1</sup><https://github.com/netsec-ethz/scion>

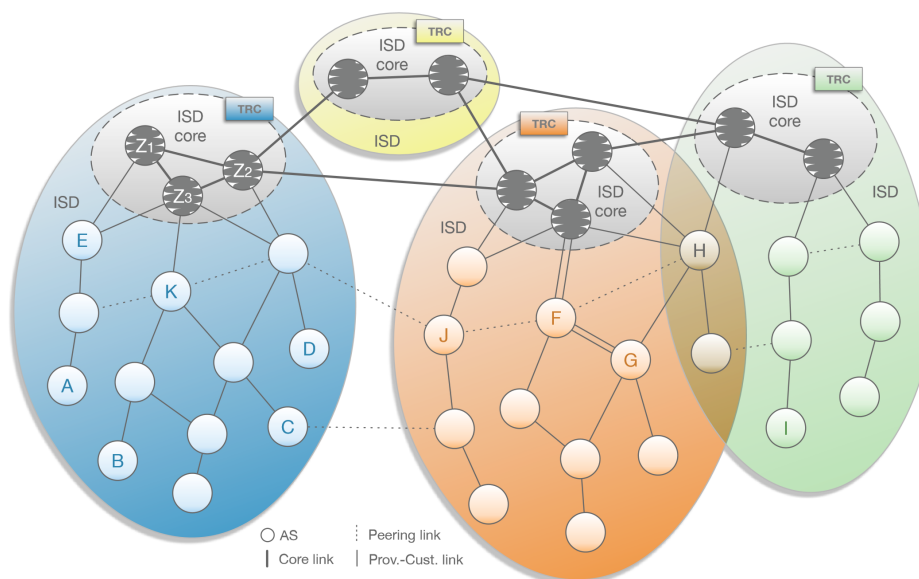


Figure 2.1: Schematic overview of a SCION network consisting of four ISDs [1]

tion of authorities and properties like routing policies and key agreement. This enables SCION to divide the control plane in a way such that one ISD is not influenced by changes in any other ISD. As an example, this mitigates outages (therefore increasing availability as desired) caused by accidentally or even maliciously misconfigured ASes, which make BGP announcements for addresses they have no control over. whereas In today's Internet, potentially every host could be affected. [1, Chapter 3] Figure 2.1 shows a network with four ISDs, each consisting of multiple ASes.

### 2.1.3 Path Selection

SCION allows each host to control routes for outgoing and incoming packages. For each AS different paths, so called up-paths, are constructed using *Path Construction Beacons* (PCBs). These PCBs reflect the constraints imposed by the ISD routing policy. ASes then announce over which of these paths they want to be reached. This technique, amongst other advantages, makes it possible to deploy effective mechanisms against Denial of Service attacks. Additionally, SCION offers a secure way of revoking failed paths and paths that do not conform to the route policy any more. [1, Chapters 7, 10]

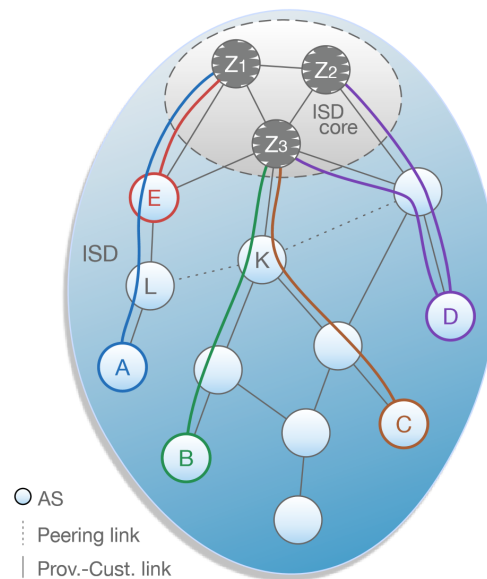


Figure 2.2: ISD with path segments for ASes A, B,C, D, and E [1]

### 2.1.4 Scalability

The controlled path selection approach outlined above ensures great scalability, as opposed to source-based routing where every host needs to know the entire network topology. At the same time, the freedom of path selection is preserved. Furthermore, since routes are pre-determined, all the forwarding information is encoded in the package itself. This renders badly scaling routing and forwarding tables obsolete, all to the benefit of increased scalability. [1]

## 2.2 SCION Architecture

Following, important components of the SCION architecture are described. All the information is taken from [1].

### 2.2.1 Border Routers

Border routers are responsible for inter-AS communication. They carry out packet forwarding based on the pre-determined path, encoded in each packet. Border routers work efficiently since the costly look up in a forwarding table is omitted.

### 2.2.2 Beacon Servers

The main function of the Beacon Server is to process path construction beacons (PCBs). The mechanism starts with a core AS generating initial PCBs and distributing them over the network. Beacon servers of non-core ASes upon receive propagate the PCBs down to their child ASes, such that the whole network is flooded. Through these beacons, ASes learn paths on which they can reach the ISD core. The AS then selects a subset of these paths and registers them using its Path server.

### 2.2.3 Path Servers

After processing PCBs, an AS registers at its local path server down-paths over which it wants to be reachable. This information is then used by other ASes to determine routes to that AS.

In reverse, the path server also functions as look up service via which an AS can retrieve path segments to reach another AS. When queried for an AS, the path server returns the paths available to reach the desired AS.

### 2.2.4 Certificate Servers

Local certificate servers manage and cache certificates of all members inside their AS, as well as the certificate issued by the ISD core to be used by the AS itself. For example, certificate servers support path servers by validating PCBs. [1, Chapter 2]

A certificate server in the ISD core manages all certificates it handed out to ASes in its ISD and provides a validation service ASes can query.

### 2.2.5 Name Servers

Similar to the DNS system currently used in the Internet, name servers perform the translation from human-friendly names to addresses understood by SCION infrastructure. The path server is then queried to obtain end-to-end paths reaching the resolved address. [1, Chapter 2]

## 2.3 SCION AS Management Framework

For easy deployment and maintenance, SCION offers the SCION AS Management Framework which provides an intuitive web interface for deploying and managing ASes. The framework consists of a *Local Management Service* per AS and a global coordinator, the *SCIONLab Coordination Service*. This framework facilitates inter-AS communication by relaying connection requests between ASes. In Section 2.1.1 we explained how an AS can join

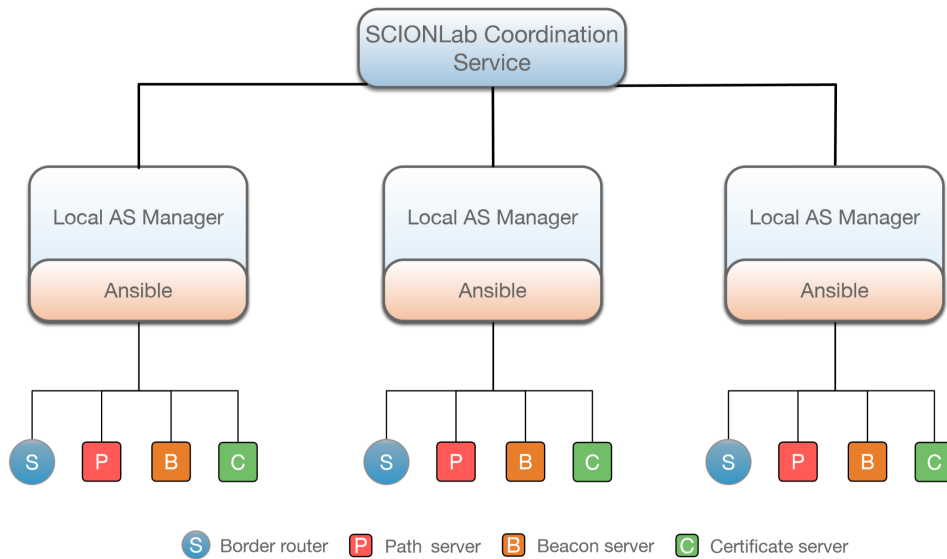


Figure 2.3: Overview of deployment architecture showing the role of *SCIONLab Coordination Service* as mediator [1]

an ISD by connecting to an AS that already is in said ISD. Such ISD join requests, for example, are handled by the framework. Figure 2.3 shows three deployed ASes connected through the SCION AS Management Framework.

The following sections describe the two components of the framework in more detail.

### 2.3.1 Local Management Service

The *Local Management Service*<sup>2</sup> is the local component of the SCION AS Management Framework. It's run by the authority managing the AS, used to monitor and configure that AS. It offers a web interface through which, amongst other functionality, administrators can make connection requests to other ASes they wish to connect to. These requests, containing all the necessary data to set up a new link between the initiator and the remote AS, are then relayed via *SCIONLab Coordination Service* to the recipient AS. [1, Chapter 10] In a similar fashion join requests for integrating an AS into an ISD can be issued using the *Local Management Service*. Other functionality involves generating the AS topology and the configurations to be deployed onto local servers.

Figure 2.4 shows the *Local Management Service* interface presented when making a connection request.

<sup>2</sup><https://github.com/netsec-ethz/scion-web>

## 2. BACKGROUND

SCION AS manager Home Submitted requests Network graph Logged in as: admin (logout) || Admin panel

### Create a new connection request

Connecting to: AS 1-2  
This AS is CLOSED. Your connection request will be reviewed by AS admins.

**Info**

Info

**Router public ip**

193.247. . /30
193.247. . /30
193.247. . /30

50000

Send message

Figure 2.4: *Local Management Service* used to make a connection request [3]

### 2.3.2 SCIONLab Coordination Service

*SCIONLab Coordination Service*<sup>3</sup> serves two purposes. For one it is part of the SCION AS Management Framework where it serves as a mediator between different *Local Management Service* instances. It provides information about available ISDs and ASes making it easy for new ASes to be created and connected to the SCION network. *SCIONLab Coordination Service* in the role of a mediator is not required by SCION. It's merely a tool to facilitate the deployment process in the early stages of adoption. Later, inter-AS connections will be negotiated between the involved parties directly. [1, Chapter 10]

*SCIONLab Coordination Service's* second responsibility is the support of the *SCIONLab Experimentation Environment* which lets interested organisations and institutions experiment with the unique capabilities of SCION. More detail about *SCIONLab Experimentation Environment* and the role *SCIONLab Coordination Service* takes in it is available in Section 2.4.

## 2.4 SCIONLab Experimentation Environment

The *SCIONLab Experimentation Environment* is a project started with the goal to provide a unique testbed environment, enabling researches to experiment

<sup>3</sup><https://github.com/netsec-ethz/scion-coord>



with SCION and the features it offers. At the same time, it allows the SCION network to naturally grow by opening up the infrastructure. In SCIONLab, participants join by deploying their own ASes in the SCION network and then connecting to other SCIONLab ASes. Hence, SCIONLab forms a subset of the entirety of ASes deployed in the SCION network. Participants become an integral part of the network, actively partaking in routing. This allows researchers to conduct highly realistic experiments, not possible with other popular testbed environments. [1, Chapter 10]

SCIONLab builds on the foundation of SCION AS Management Framework. It uses the same tools to offer a clean, easy-to-use way of joining the SCIONLab network. As an example, *SCIONLab Coordination Service* facilitates the deployment of SCIONLab ASes for interested parties. It is envisioned to offer built in user administration capabilities, where users can register accounts and download SCIONLab configurations to be run on their hardware. Moreover, it also provides status updates about services running in the AS and notifications about new SCION versions. [1, Chapter 10]

Summarized, the goals of *SCIONLab Experimentation Environment* are:

- Providing a unique testbed environment
- Opening up SCION to allow organic growth
- Allowing for realistic conduction of experiments
- Achieving above points with low overhead using user-friendly tools such as the *SCIONLab Coordination Service*

Figure 2.5 shows an example of an envisioned SCIONLab environment.

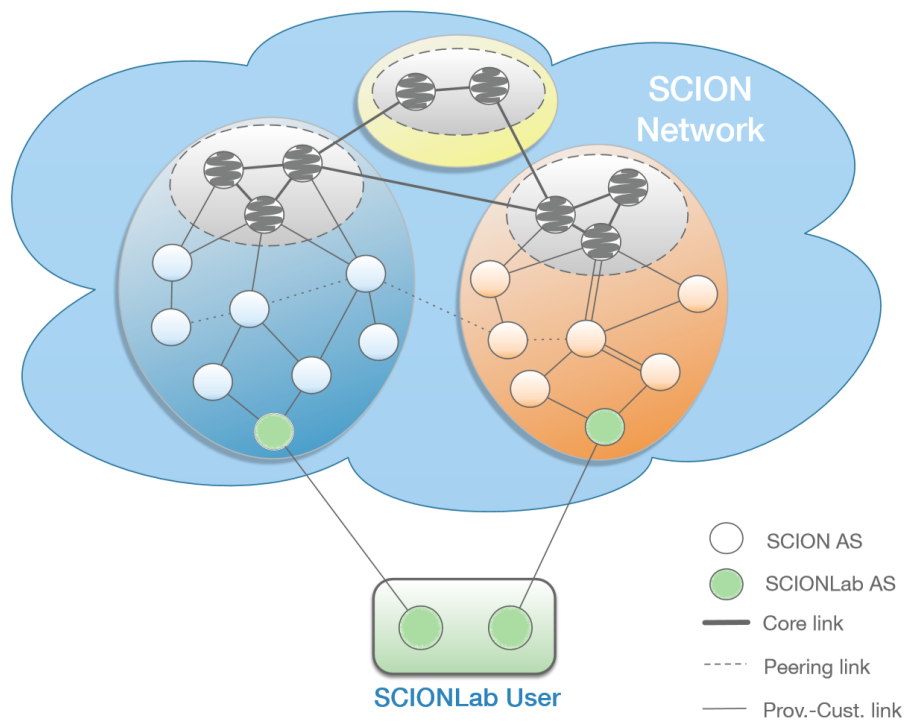


Figure 2.5: An envisioned SCIONLab testbed environment [1]

# Related Work

---

The following sections provide a brief overview of other testbed environments for running network and distributed systems experiments. In particular, their approach of organising and managing network nodes and their way of handling users enrolled in experiments is highlighted.

## 3.1 PlanetLab

**PlanetLab**<sup>1</sup> is an open platform for computer networking and distributed systems research. It allows the deployment and testing of newly developed network protocols, distributed algorithms, peer-to-peer software and CDNs<sup>2</sup>. [4] For conducting tests, each project has access to a set of globally distributed virtual machines. These sets, called slices in PlanetLab terminology, run on physical nodes provided and maintained by users of PlanetLab, mainly research institutions.

### 3.1.1 PlanetLab Node Management

For an institution to be granted access to PlanetLab, it has to add its own computational resources to the PlanetLab network. [5] These nodes are remotely managed by PlanetLab operational staff. Local administrators are not given root access and are merely allowed to modify certain parameters, such as outgoing network bandwidth. [6] A Principal Investigator (PI) at each site is responsible for approving accounts and assigning them to slices. Furthermore, the PI locally enforces the PlanetLab Acceptable Use Policy (AUP). After instantiating a slice, users are given SSH<sup>3</sup> access with root priv-

---

<sup>1</sup><https://www.planet-lab.org/>

<sup>2</sup>Content Delivery Networks

<sup>3</sup>Secure Shell

ileges to nodes in their slice. However, this root access is restricted as it does not allow changes to hardware and network configurations. [7]

Comparing to *SCIONLab Experimentation Environment*, there are some fundamental differences:

- In *SCIONLab Experimentation Environment*, each project joins with its own resources. Neither are there restrictions regarding system configuration, nor are nodes managed by SCIONLab operational staff.
- There is no assignment to slices by a PI in SCIONLab. The deployment of nodes in SCIONLab is left entirely to the project team.
- Deployed nodes do not necessarily need to be virtualized. There are configurations for running SCION directly on physical machines.
- Since there are no PIs in SCIONLab, the intermediate step of creating accounts via PI is omitted. Institutions directly register their accounts via *SCIONLab Coordination Service*.
- SCIONLab uses a novel approach of pre-authenticated accounts with pre-established connections to get research institutions aboard. This allows invited parties to start using SCIONLab with minimal overhead.
- Developed from ground up with simplicity in mind, the *SCIONLab Experimentation Environment* offers easy-to-use node management tools.

## 3.2 GENI

Similar to PlanetLab, **GENI**<sup>4</sup> (Global Environment for Network Innovations) is a testbed environment designed for networking and distributed systems research. Like PlanetLab, GENI uses the notion of slices to describe a set of geographically distributed virtual and physical hosts, instantiated for an experiment. The unique feature of GENI is its "deep programmability" capability which lets users connect compute resources on the link layer and replace above layers with custom protocols. [8]

In GENI, each project is led by a single individual; the project lead. The project lead is responsible for allocating slices and assigning project members to them. Slices consist of different resource providers available in the GENI network, so called aggregates.

Aggregates are hosted by institutions and managed by local operators. In order to start using GENI, an account must be registered. However, thanks to tight collaboration with many institutions, the institutions account may be used to register for GENI. Additionally, since experiments on GENI often

---

<sup>4</sup><http://www.geni.net/>

revolve around new services, GENI allows end users, who are not affiliated with the project, to opt in, in order to bring real traffic to experiments. [8]

## 3.3 Fed4FIRE

**FED4FIRE**<sup>5</sup> strives for creating a large federation of experimentation facilities and network testbeds in Europe. The main idea behind this initiative is to simplify the use of already existing testbed environments, thus making it possible for researchers to collaborate by sharing existing test facilities in the broad field of ICT<sup>6</sup>. This also allows researchers to use multiple testbeds for their experiments. [9]

FED4FIRE offers its facilities in two ways: Open Calls are selected projects which receive financial support to be carried out. The other possibility, Open Access, allows every interested party to run their projects, without being funded.

Accounts are registered at a FED4FIRE authority and then used to create new projects, similar to PlanetLab and GENI. The architecture consists of multiple testbed environments, all aggregated under the FED4FIRE initiative. By the nature of this very heterogeneous design, the management of the environment is much more complicated than in SCIONLab. [9]

---

<sup>5</sup><https://www.fed4fire.eu/>

<sup>6</sup>Information and Communication Technology



# Requirements Engineering

---

With the introduction of SCIONLab, *SCIONLab Coordination Service* now plays a significant role in the *SCIONLab Experimentation Environment*. Because of this, a set of new requirements arose, focusing on making the service an intuitive, user friendly tool, while at the same time increasing the robustness of the system. This shift from a mediator to an online account management tool, allowing users to register accounts and download their SCION configurations in order join the network with minimal overhead, demands an extension of initial requirements.

## 4.1 Functional Requirements

To make *SCIONLab Coordination Service* meet the functionality outlined above, the following requirements were gathered:

1. *A mechanism to verify users' email addresses:*  
This increases the authenticity of registered accounts. This also ensures that users can be contacted by email.
2. *A mechanism to manually activate users who signed up successfully:*  
Not all users should immediately be granted access to *SCIONLab Coordination Service*. Administrators need to be able to audit registration requests.
3. *A mechanism to protect the service against automated account creation:*  
This safety measurement protects the service from spam and the creation of fake accounts.
4. *New functionality is validated using CircleCI's testing environment:*  
This increases development productivity and ensures new additions do not break *SCIONLab Coordination Service*.

5. *A fast and easy way to deploy the service onto multiple machines:*  
Since *SCIONLab Coordination Service* is evolving quickly, it is required that it can be deployed onto the target machine effortlessly.
6. *A system for sending notifications to users:*  
Users should be notified if the status of their ASes change or when new versions of SCION are available.
7. *Implementation of missing APIs between Coordination Service and the Local Management Service:*  
This ensures flawless operation of *SCIONLab Coordination Service* in its role as mediator.
8. *A visualization of the SCIONLab Experimentation network accessible by users:*  
This makes it easy for new users to get an overview of the SCIONLab network and helps them join an ISD.

### 4.2 Non-Functional Requirements

Non-Functional requirements comprise the following points:

1. *SCIONLab Coordination Service* aims to be an easy to use tool.
2. None of its functionality should require the user to invest a great amount of work.
3. It is preferred to hide as much complexity as possible from users.
4. The web interface of the service is required to be fast, clean and responsive, since it will be amongst the first impressions users get of SCION.
5. The web interface needs to be visually appealing, both on desktop and mobile devices.
6. In terms of maintainability, *SCIONLab Coordination Service* aims for easy extensibility as the project evolves fast and new requirements need to be integrated with minimal effort.



---

## Architecture Overview

---

### 5.1 Design Overview

Figure 5.1 shows an overview of the architectural design of *SCIONLab Coordination Service*. All the components and services are described in detail in the following sections.

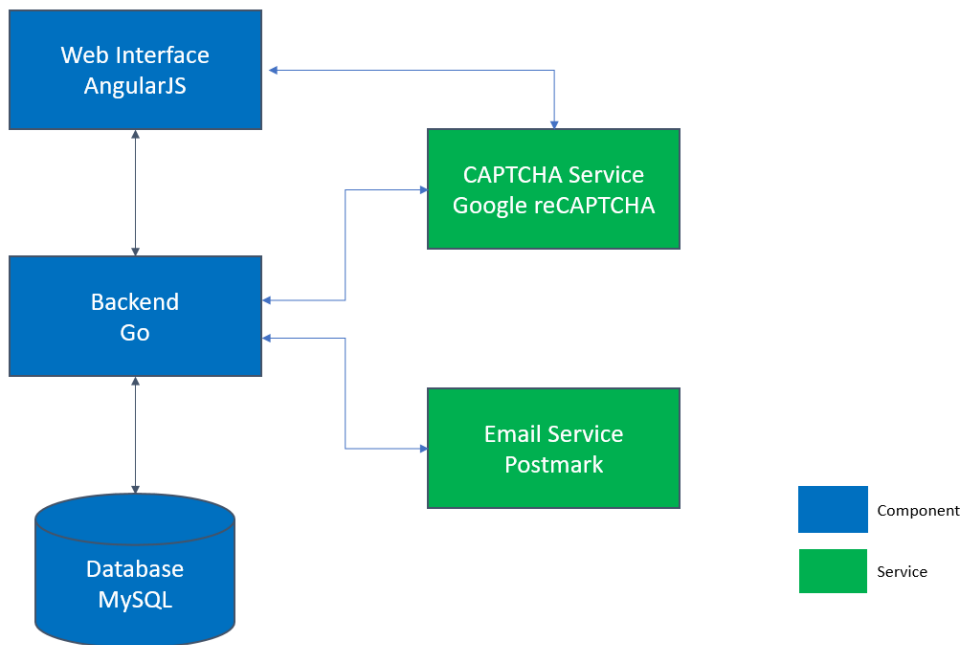


Figure 5.1: Schematic overview of *SCIONLab Coordination Service* architecture, showing its components and the services it uses

### 5.2 Components

In the following sections, component denotes a part that directly belongs to the *SCIONLab Coordination Service* architecture.

#### 5.2.1 Database

*SCIONLab Coordination Service* uses a MySQL database as storage system. MySQL is one of the most widely used relational database management systems. In this project a free to use, open-source version is used. It stores basic entities such as users, their corresponding accounts, ASes deployed by users and connection requests sent between ASes. If a user makes use of a virtual machine to run SCION, configurations for this set up are stored as well. Another purpose is to keep track of the state aforementioned entities are in. It directly interfaces with the back end web server which is the only component it receives queries from.

#### 5.2.2 Back End Server

The centre piece of the *SCIONLab Coordination Service* is the back end server. It interfaces with all the other components via a set of well defined APIs, therefore functioning as a mediator. It provides the following functionality:

- Serving web pages to the *SCIONLab Coordination Service* web interface
- Processing of API calls received from the *SCIONLab Coordination Service* web interface
- Processing of API calls received from *SCION Local Management Service*
- Controlling access to resources
- Preparing emails and handing them over to Postmark servers for sending
- Processing and manipulating data retrieved from the database

#### 5.2.3 Web Interface

The web interface is the front end part of the *SCIONLab Coordination Service*. It is the graphical interface between users and the system. It closely interacts with the back end server, to which it makes API calls and from which it receives all required data. The following functionality is offered by this component:

- Ability to register new users
- Login for existing users

- A personal page per user showing account details and allowing to download a SCIONLab VM image
- An administrator panel for managing pending user registrations
- A landing page for users, successfully verifying their email address

## 5.3 Services

*SCIONLab Coordination Service* makes use of multiple services to outsource certain tasks. These services are described in the subsections that follow.

### 5.3.1 Postmark

*SCIONLab Coordination Service* uses [Postmark](https://postmarkapp.com/)<sup>1</sup> for email sending. An early requirement was to use a local MTA<sup>2</sup>. However, as outlined in Section 6.2.1, this proved to be very unreliable. The decision was then to switch to an external provider that handles the complexity. Postmark guarantees fast and reliable delivery of all transactional emails sent by *SCIONLab Coordination Service*. [10]

### 5.3.2 Google reCAPTCHA

As a measurement against bots that create fake SCIONLab accounts the web interface of *SCIONLab Coordination Service* contains a CAPTCHA<sup>3</sup> widget which needs to be solved in order to create an account. *SCIONLab Coordination Service* makes use of the Google reCAPTCHA service for implementing this functionality.

## 5.4 Continuous Integration

[CircleCI](https://circleci.com/)<sup>4</sup> is a continuous integration utility that, once set up for a project, automatically runs unit tests for newly added code using cloud technologies. The feedback produced includes a comprehensive list of all tests it ran, showing which tests failed and for what reasons. This enables the *SCIONLab Coordination Service* development team to validate pull requests before they are merged into the master branch. CircleCI tightly ties into [GitHub](https://github.com/)<sup>5</sup>, showing the outcome of the test suite directly on the pull request page.

---

<sup>1</sup><https://postmarkapp.com/>

<sup>2</sup>Mail Transfer Agent - The software running on an email server

<sup>3</sup>Completely Automated Public Turing test to tell Computers and Humans Apart

<sup>4</sup><https://circleci.com/>

<sup>5</sup><https://github.com/>

Additionally, new code is reviewed using [Reviewable](https://reviewable.io)<sup>6</sup>. Reviewable keeps track of changes, as pull requests evolve with new commits. To be merged, all tests on CircleCI must pass and all discussions on Reviewable must be resolved.

### 5.5 Deployment

Manually setting up a *SCIONLab Coordination Service* instance on a remote machine requires many steps. Hence, doing it often and for multiple machines very quickly becomes a cumbersome task. To automate this process we use [Ansible](https://www.ansible.com/)<sup>7</sup>. Ansible is an agentless IT automation engine with the goal to end repetitive tasks. It modifies a system in such a way that it matches the state described in configuration files, the playbooks. This makes the deployment, if the playbook is written carefully, idempotent.

---

<sup>6</sup><https://reviewable.io>

<sup>7</sup><https://www.ansible.com/>

## Chapter 6

---

# Implementation

---

The following sections contain in-depth descriptions of the design and implementation process of major additions developed throughout this thesis. Less extensive improvements, addressing the security and maintainability of *SCIONLab Coordination Service*, are described in appendix A.

### 6.1 Initial State

At the start of this thesis, a basic version of *SCIONLab Coordination Service* had already been implemented. The implementation consisted of the back end server written in [Go](https://golang.org/)<sup>1</sup>, the web interface implemented in [AngularJS](https://angularjs.org/)<sup>2</sup> and the [MySQL](https://www.mysql.com)<sup>3</sup> database. While operational, *SCIONLab Coordination Service* was lacking desired functionality. Relevant for this thesis were particularly the weak user registration process, which did not perform any checks to ensure validity of submitted data. When logging in, users were presented with a simple page, merely showing credentials needed to connect to SCIONLab. It was not possible to download configurations and virtual machine images for running SCION. Furthermore, *SCIONLab Coordination Service* had no way of communicating with users via email.

This basic implementation was enhanced with new functionality as described in the rest of this chapter.

### 6.2 Email Address Verification

Two options were evaluated for email verification. Email callback verification and verification based on email exchange.

---

<sup>1</sup><https://golang.org/>

<sup>2</sup><https://angularjs.org/>

<sup>3</sup><https://www.mysql.com>

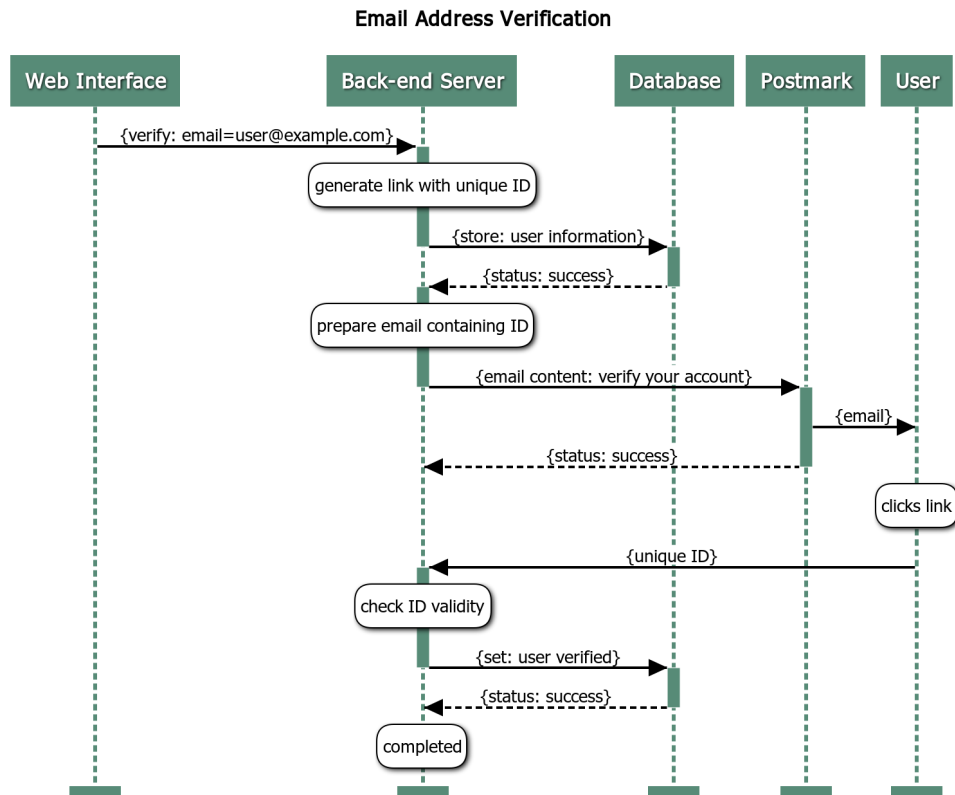


Figure 6.1: Sequence diagram showing the messages exchanged for the Email Verification feature

Email callback verification finds its use mostly as anti-spam measure in SMTP<sup>4</sup> servers. The verification is carried out in the same way as sending an email to the target email address. However, instead of following through and sending actual email content, the process is aborted as soon as the remote mail exchanger accepts or rejects the recipient address as invalid. The response is the same "OK" or "Unknown user" as when running the protocol with the intent to send an email. [11] Email callback verification, however, suffers from unreliability [11] which makes it not suitable for *SCIONLab Coordination Service*.

Instead, a robust approach based on email exchange was chosen. Upon registration users receive a link with a unique identifier. When they follow the link this is registered by the system and the users' email addresses are marked as verified. Figure 6.1 shows a message sequence diagram of the process implemented.

<sup>4</sup>Simple Mail Transfer Protocol

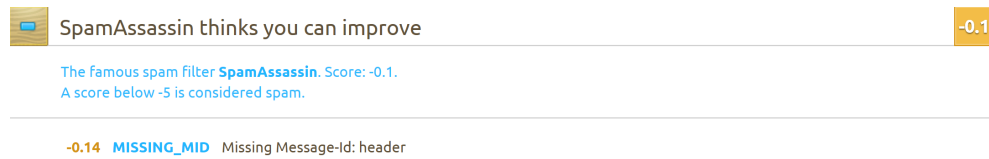


Figure 6.2: Result of SpamAssassin ran against local Exim SMTP server (Run on **Mail-Tester**)

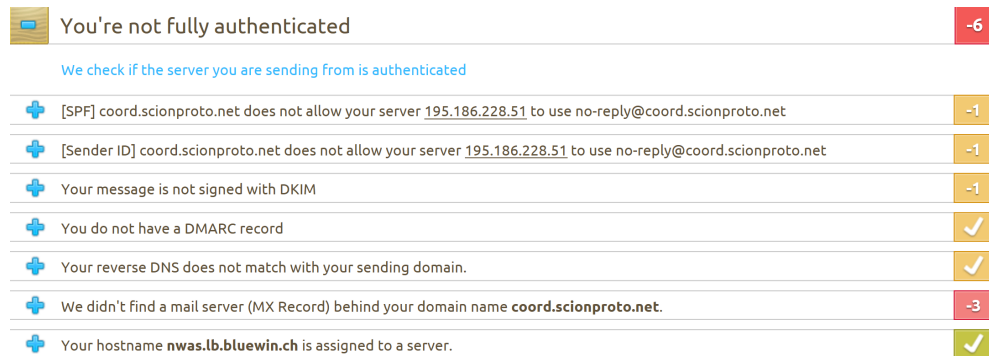


Figure 6.3: Authentication report for the local Exim SMTP server (Run on **Mail-Tester**)

### 6.2.1 Email package

As already mentioned in Section 5.3.1, the initial requirement was to support a local SMTP server for sending emails. Since *SCIONLab Coordination Service* originally did not have the ability to send emails, a new, self-written email package was added to the back end server. It was communicating with an Exim<sup>5</sup> instance running on the same server. Extensive testing showed that the approach of supporting and maintaining a local MTA is too complex to implement in a reliable fashion for this project. While tweaking the email headers and content was sufficient for the email not to be marked as spam by **SpamAssassin**<sup>6</sup> (see Figure 6.2), missing MX Records<sup>7</sup> and absent DKIM<sup>8</sup> authentication (see Figure 6.3) often caused emails to not reach the inbox of users. It was then decided to outsource this complexity to Postmark, as described in Section 5.3.1. As a result, the email package was changed to make use of Postmark's APIs, instead of interfacing with the local MTA. The outcome of the same spam detection and authentication tests ran with Postmark as mail provider are shown in Figures 6.4 and 6.5.

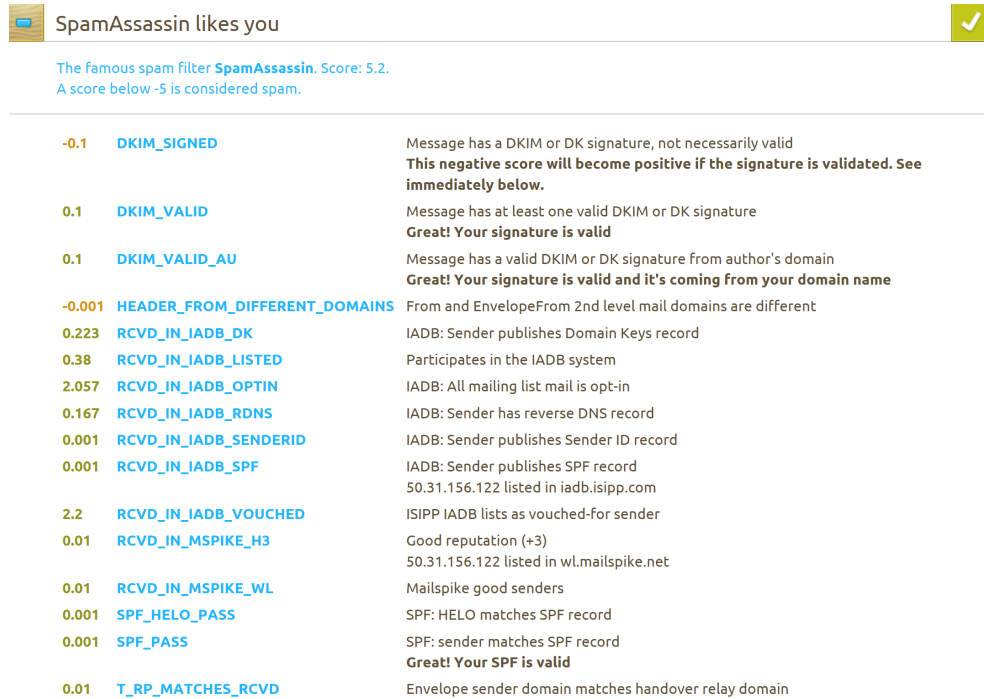
<sup>5</sup>Exim is a popular MTA for UNIX based systems

<sup>6</sup>A sophisticated spam filter: <http://spamassassin.apache.org/>

<sup>7</sup>Mail Exchange Resource Record - A record specifying a mail server in the DNS system

<sup>8</sup>DomainKeys Identified Mail - a method to detect email spoofing

## 6. IMPLEMENTATION

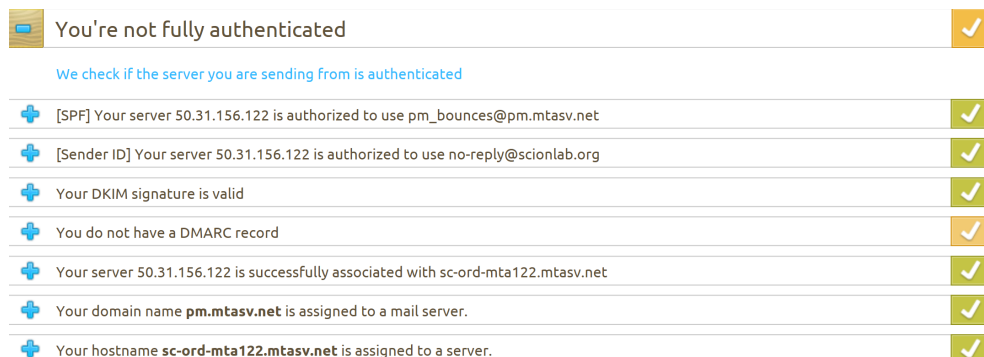


SpamAssassin likes you ✓

The Famous spam filter **SpamAssassin**. Score: 5.2.  
A score below -5 is considered spam.

-0.1	<b>DKIM_SIGNED</b>	Message has a DKIM or DK signature, not necessarily valid <b>This negative score will become positive if the signature is validated. See immediately below.</b>
0.1	<b>DKIM_VALID</b>	Message has at least one valid DKIM or DK signature <b>Great! Your signature is valid</b>
0.1	<b>DKIM_VALID_AU</b>	Message has a valid DKIM or DK signature from author's domain <b>Great! Your signature is valid and it's coming from your domain name</b>
-0.001	<b>HEADER_FROM_DIFFERENT_DOMAINS</b>	From and EnvelopeFrom 2nd level mail domains are different
0.223	<b>RCVD_IN_IADB_DK</b>	IADB: Sender publishes Domain Keys record
0.38	<b>RCVD_IN_IADB_LISTED</b>	Participates in the IADB system
2.057	<b>RCVD_IN_IADB_OPTIN</b>	IADB: All mailing list mail is opt-in
0.167	<b>RCVD_IN_IADB_RDNS</b>	IADB: Sender has reverse DNS record
0.001	<b>RCVD_IN_IADB_SENDERID</b>	IADB: Sender publishes Sender ID record
0.001	<b>RCVD_IN_IADB_SPF</b>	IADB: Sender publishes SPF record 50.31.156.122 listed in iadb.isipp.com
2.2	<b>RCVD_IN_IADB_VOUCHEDED</b>	ISIPP IADB lists as vouched-for sender
0.01	<b>RCVD_IN_MSPIKE_H3</b>	Good reputation (+3) 50.31.156.122 listed in wl.mailspike.net
0.01	<b>RCVD_IN_MSPIKE_WL</b>	Mailspike good senders
0.001	<b>SPF_HELO_PASS</b>	SPF: HELO matches SPF record
0.001	<b>SPF_PASS</b>	SPF: sender matches SPF record <b>Great! Your SPF is valid</b>
0.01	<b>T_RP_MATCHES_RCVD</b>	Envelope sender domain matches handover relay domain

Figure 6.4: Result of SpamAssassin ran against Postmark (Run on Mail-Tester)



You're not fully authenticated ✓

We check if the server you are sending from is authenticated

+	[SPF] Your server 50.31.156.122 is authorized to use pm_bounces@pm.mtasv.net	✓
+	[Sender ID] Your server 50.31.156.122 is authorized to use no-reply@scionlab.org	✓
+	Your DKIM signature is valid	✓
+	You do not have a DMARC record	✓
+	Your server 50.31.156.122 is successfully associated with sc-ord-mta122.mtasv.net	✓
+	Your domain name <b>pm.mtasv.net</b> is assigned to a mail server.	✓
+	Your hostname <b>sc-ord-mta122.mtasv.net</b> is assigned to a server.	✓

Figure 6.5: Authentication report for Postmark (Run on Mail-Tester)



Listing 6.1: API signature for verifying an email address

```
1 //email validation
2 router.Handle("/api/verifyEmail/{uuid}", loggingChain.ThenFunc(
3     registrationController.VerifyEmail))
```

### 6.2.2 Adapted Components

Apart from the added email package, other components had to be adapted to implement the feature. The following functionality was added:

#### User information on Web Interface

On the web interface, an alert box was integrated on the registration page. Upon successful registration it asks the user to check his emails. Otherwise it displays an error corresponding to the problem that occurred.

#### Email creation

The existing code to register a new user was extended to additionally prepare a personalized email, containing the verification link. This email then gets sent using the email package. The unique identifier contained in the link is stored in the database together with the user information. This allows for a direct mapping between user and identifier.

#### Verification API

A new API was added for handling email address verification requests. This API gets called by the user's web browser when following the confirmation link. Listing 6.1 shows the signature of the API. `"uuid"` denotes the per-user unique identifier.

#### Handler Function

A new handler function which gets invoked by the above API call. It checks that the identifier sent in the request is valid and belongs to a user with a not yet verified email address. If the identifier is valid, the corresponding user is set to verified and a confirmation page gets served to the web browser. If not, an error is logged and forwarded to the web browser.

#### Confirmation Page

A confirmation page informing the user about the successful verification process was added to the web interface. This is the page served by the

handler function on successful verification. It offers a shortcut to the login page.

### 6.3 Manual User Activation

The manual user activation feature builds on top of the email verification system. After verifying the email address a user should not immediately be granted access to *SCIONLab Coordination Service*. A second verification step is required. The user needs to be activated. To make the process of user activation as automated as possible we distinguish between two cases. Either the user's email address is on a list of pre-defined, trusted domain names. In this case the activation is done automatically. In case the email's domain name is not on the list, the user has to be manually approved by an administrator. Manual activation happens through a newly designed administrator panel added to the web interface. The process with its two cases is outlined in Figure 6.6. The notification emails sent to administrators and users make use of the new email package introduced for the email verification system.

#### 6.3.1 Role based access control

In order to activate a regular user by an administrator the concept of different user groups had to be introduced to *SCIONLab Coordination Service*. For this purpose the system was adapted to support a simple role based access control model (RBAC). Instead of granting special rights to individual users each user gets instantiated with a role that reflects the rights this user should possess. This model easily supports many different roles with distinct sets of permissions. However, for the user activation feature only two roles were needed; an administrator role and a regular user role. The regular user role is the role assigned to standard users when signing up through the web interface of *SCIONLab Coordination Service*. The administrator role possesses all the permissions of the regular user role and additionally the ability to log in to the administrator panel via the web interface, from where regular users can be activated.

#### 6.3.2 Adapted Components

On top of the changes made to implement the RBAC model the following enhancements were made to support the user activation feature:

##### **Admin User Creation**

We mentioned above that creating a user through the web interface assigns the regular user role. In order to bootstrap administrator creation, *SCIONLab*

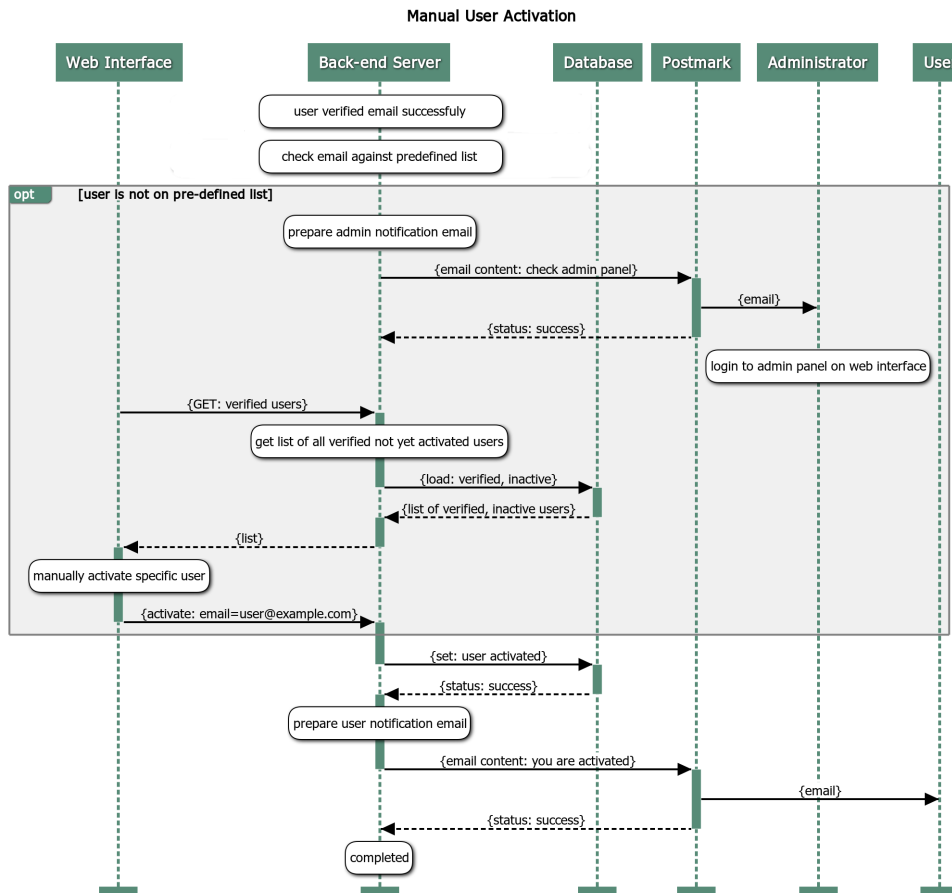


Figure 6.6: Sequence diagram showing the messages exchanged for the Manual User Activation feature

*Coordination Service* was extended to accept command line arguments that allow the creation of administrators on system start up. Administrators could then be used to create other administrators. However, such functionality is not in place yet.

### Activation API

Activating users through the administrator panel requires the web interface to issue two types of requests to the back end server (see Figure 6.6). First, it retrieves a list of all verified, not yet activated users. Then, for a selected user, it issues an activation command. The signature of these two API calls are shown in Listing 6.2.

Listing 6.2: API signatures for loading unactivated users and for activating users

```
1 // user activation
2 router.Handle("/api/loadUnactivatedUsers", loggingChain.ThenFunc(
3     loginController.LoadUnactivatedUsers))
4 router.Handle("/api/activateUser", loggingChain.ThenFunc(
5     loginController.ActivateUser)).Methods("POST")
```

### Handler Functions

Corresponding to the APIs in Listing 6.2, *SCIONLab Coordination Service* was extended with two new handler functions. `"LoadUnactivatedUsers"` first checks if the request was issued by an administrator. If this is the case it retrieves a list of all verified, inactive users from the database and sends it back in the HTTP response. Otherwise an error is sent to the web interface.

The second handler function, `"ActivateUser"`, performs the same administrator authentication check and, on success, activates the user corresponding to the email address contained in the request. If the authentication fails, the handler responds with an error.

### Email Notifications

Using the email package introduced in Section 6.2.1, new notification emails were added to *SCIONLab Coordination Service*:

An email gets sent to all administrators when there are new users waiting to be activated in the administrator panel. This ensures that users who can not be activated automatically are not waiting too long for their activation. This email is sent once there are pending users, rather than for every individual user.

When a user gets manually activated, an email informing about the changed status gets sent to that user. It contains a link to the login page of the web interface.

### Pre-defined List

In Section 6.3 we mentioned that users with a trusted email address get activated automatically after they verify their email address. In order to implement this feature, trusted email domain names are collected in a list that comes as part of the *SCIONLab Coordination Service* configuration.

The following formats are possible:

- example@domain.tld (full email address)

- domain.tld (domain & TLD<sup>9</sup>)
- sub1...sub2.domain.tld (subdomains & TLD)
- tld (TLD only)

Upon successful verification, it is checked whether or not the email is on the list. If the address matches an entry, it is immediately activated. Otherwise the user must go through the manual activation process.

### Administrator Panel

A new administrator panel was added to the web interface. It displays a table containing all verified, inactive users together with their relevant information like name, email address and organisation. Individual users can be activated via an activation button.

### Confirmation Page

The confirmation page displayed when users follow the email verification link was revamped to provide users who can not be activated automatically with information about the extended activation process.

## 6.4 CAPTCHA Integration

Fake accounts can negatively impact the performance of a system, and with email sending involved in the registration process, bots can abuse *SCIONLab Coordination Service* for spamming. As counter measurement against such malicious behaviour, a CAPTCHA was placed on the registration page. The interaction of users with the CAPTCHA and its overall integration into *SCIONLab Coordination Service* can be seen in Figure 6.7.

### 6.4.1 Google reCAPTCHA

For *SCIONLab Coordination Service*, Google **reCAPTCHA**<sup>10</sup> was chosen as CAPTCHA provider for a variety of reasons: Since reCAPTCHA is the most popular CAPTCHA service many users will be familiar with how it works. [12] Also, thanks to its advanced risk analysis which observes a user's interaction with the website to infer whether a user is human, often users are not forced to solve a challenge to pass the Turing test. A mere click on a check box is enough to pass. [12] This effortless verification for human users complies with the non-functional requirements of *SCIONLab Coordination Service* outlined in Section 4.2. Furthermore, reCAPTCHA is very well supported and documented by Google which makes implementation straight forward.

---

<sup>9</sup>Top Level Domain

<sup>10</sup><https://www.google.com/recaptcha/intro/android.html>

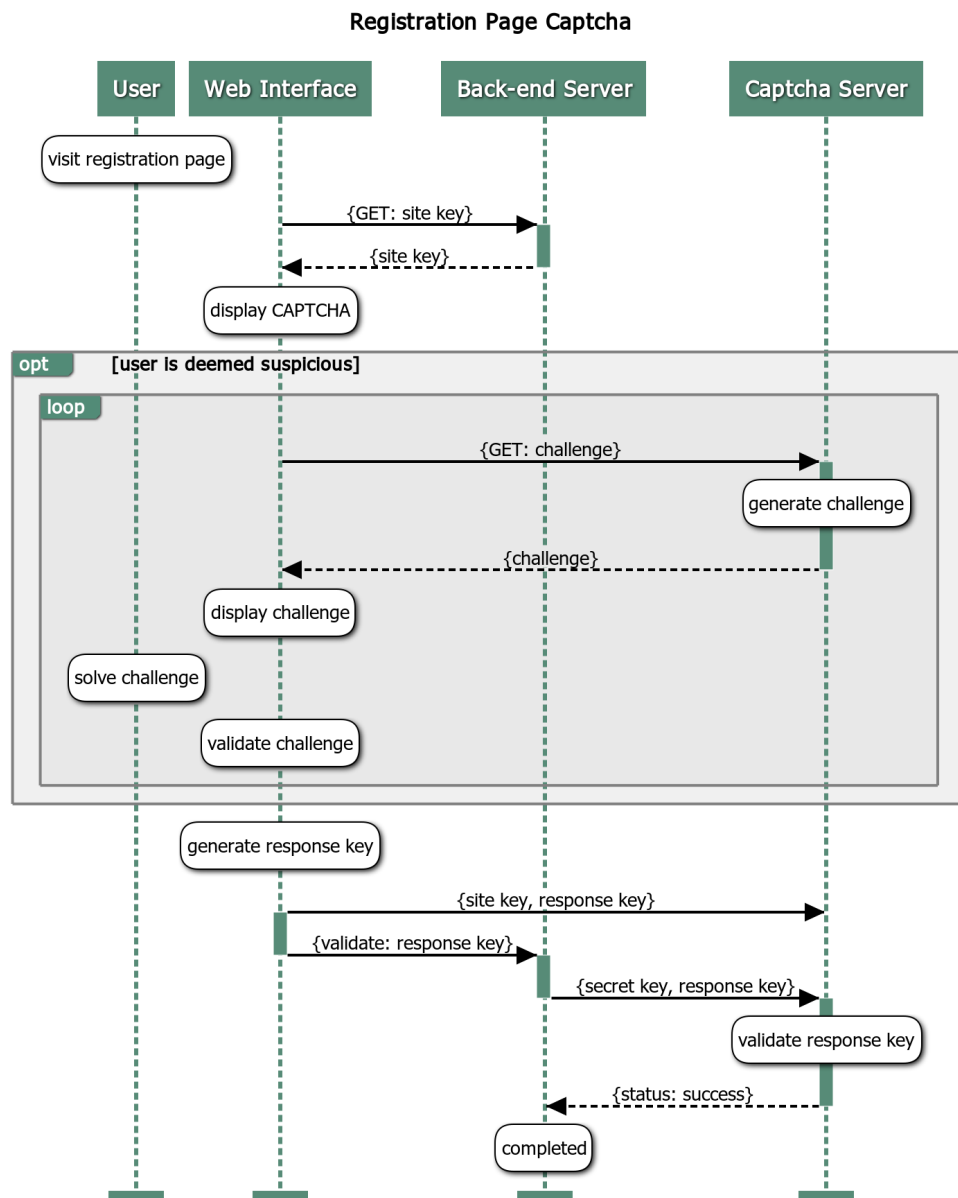


Figure 6.7: Sequence diagram showing the exchanged messages for the CAPTCHA implementation

Listing 6.3: API signatures for registering users and loading the captcha site key

```
1 // user registration
2 router.Handle("/api/captchaSiteKey", loggingChain.ThenFunc(
3     registrationController.LoadCaptchaSiteKey))
4 router.Handle("/api/register", loggingChain.ThenFunc(
5     registrationController.Register)).Methods("POST")
```

reCAPTCHA uses a set of keys to fulfill its task. The web page containing the reCAPTCHA must be registered to obtain a site key and a secret key. The site key is used to create reCAPTCHA response keys for users passing the Turing test. This response key then gets verified by sending it to a reCAPTCHA server together with the secret key. Additionally, the reCAPTCHA server provides the reCAPTCHA widget on the web page with challenges to be solved.

### 6.4.2 Adapted Components

In order to implement the reCAPTCHA for the registration page of the web interface the following changes had to be made to the system:

#### CAPTCHA API

On the web interface reCAPTCHA must be instantiated with the site key corresponding to the web page it is placed on. When the page loads this site key is retrieved via an API call from the back end server. This offers the convenience of having all keys stored in the main configuration file of *SCIONLab Coordination Service* rather than hard coding the site key into the web page. Like this, if new keys need to be installed or *SCIONLab Coordination Service* is deployed onto another machine, only the configuration file has to be changed.

The response key gets appended to the user information as an attribute and is sent using the existing registration API call to the back end server where it is validated together with other user information.

The described endpoints are shown in Listing 6.3

#### Handler Functions

The newly added `"LoadCaptchaSiteKey"` handler retrieves the site key from the configuration and hands it over to the web interface where it is used to instantiate the reCAPTCHA.

When a user submits his registration information, the existing `"Register"` handler is triggered. It reads the user data, including the response key, from the request and performs validity checks on the submitted information. The following new check was added to this phase: The site key is sent together with the secret key to the reCAPTCHA server, which in turn verifies whether or not the response key is valid for the site specified by the secret key. Here, communication with the reCAPTCHA server is done using the third party [haisum/recaptcha](https://github.com/haisum/recaptcha)<sup>11</sup> package. If all checks pass, the user's information is stored in the database.

### CAPTCHA Widget on Registration Page

Using the [VividCortex/angular-recaptcha](https://github.com/VividCortex/angular-recaptcha)<sup>12</sup> AngularJS directive, the reCAPTCHA widget was placed on the registration page. Solving the reCAPTCHA was made mandatory for sending registration data to the back end.

## 6.5 CircleCI Integration

In Section 5.4 we talked about CircleCI and how it helps with the software verification process. Every time a pull request is updated with new code CircleCI runs a defined suite of tests in the cloud. For this to work CircleCI must know about the project environment, what dependencies are used and what tests it has to run. This information is pulled from a configuration file that resides in a special folder inside the *SCIONLab Coordination Service* code base. These configurations are processed and used to create a minimal, virtual environment in the CircleCI cloud, which is able to run the code to be tested. The contents of the configuration file is shown in Listing 6.4.

The following subsections describe how CircleCI is set up in order to run the test suite.

### 6.5.1 Environment

The easiest way to set up the required environment is to use a pre-defined [Docker](https://www.docker.com/)<sup>13</sup> image that comes bundled with all the tools needed for the testing and debugging process.

As can be seen on line 6 and 12 in Listing 6.4, two Docker images are used: The `"circleci/golang:1.8"` image is an image built by CircleCI specifically for testing Go code. It comes with the Go 1.8 environment and numerous other useful tools, such as Git and SSH, pre-installed.

---

<sup>11</sup><https://github.com/haisum/recaptcha>

<sup>12</sup><https://github.com/VividCortex/angular-recaptcha>

<sup>13</sup>Docker bundles software in isolated containers, then runs these containers on a host system using virtualization: <https://www.docker.com/>



Listing 6.4: CircleCI project configurations

```

1  —
2  jobs:
3    build:
4      docker:
5        -
6          image: "circleci/golang:1.8"
7        -
8          environment:
9            - MYSQL_ROOT_PASSWORD: development_pass
10           - MYSQL_DATABASE: scion_coord_test
11
12           image: "circleci/mysql:latest"
13      steps:
14        - checkout
15        - run: "cp conf/development.conf.default conf/development.conf"
16        - run:
17          name: Wait for db
18          command: dockerize -wait tcp://localhost:3306 -timeout 1m
19        -
20          run: "go get -v -t -d ./..."
21        -
22          run: "go test -v ./..."
23      working_directory: "/go/src/github.com/netsec-ethz/scion-coord"
24  version: 2

```

`"circleci/mysql:latest"` additionally installs the latest version of MySQL in the test environment, which is crucial since tests involve storing to and loading from the database. This image also allows setting up the database in a way such that it conforms to what *SCIONLab Coordination Service* is expecting. (see lines 8 to 10 of Listing 6.4)

### 6.5.2 Dependencies

*SCIONLab Coordination Service* uses many dependencies which all need to be installed for tests to run successfully. Dependencies are downloaded using the Go command `"go get -v -t -d ./..."`. This downloads all needed packages and writes the name of each package to the test log.

### 6.5.3 Testing

`"go test -v ./..."` looks for all test files in the code base and then runs the test cases one by one. For each test case it writes status information to the log file stating whether the test failed or not. In case of failure, a corresponding error message is logged as well.

Listing 6.5: Excerpt of Ansible tasks, used to set up *SCIONLab Coordination Service*

```
1 ---
2 - name: Install Scion Coord Software
3   apt: name={{ item }}
4   with_items:
5     - git
6     - golang-go
7     - python-mysqldb
8     - mysql-server
9
10 - name: Create MySQL root user
11   mysql_user:
12     name: root
13     password: "{{ mysql_pass }}"
14     host: localhost
15     login_password: "{{ mysql_pass }}"
16
17 - name: Create database
18   mysql_db:
19     name: "{{ mysql_db }}"
20     login_password: "{{ mysql_pass }}"
```

Listing 6.6: Excerpt of the Ansible parameter file used for *SCIONLab Coordination Service*

```
1 mysql_db: scion_coord_test
2 mysql_pass: development_pass
```

## 6.6 Deployment with Ansible

In Section 5.5 we described how Ansible supports the deployment process. As with CircleCI, Ansible needs instructions to put the target machine into the right state and to deploy *SCIONLab Coordination Service* correctly. The most important configurations are described below.

### 6.6.1 Hosts

The host configuration file contains sections for different roles to deploy. Within each section, the addresses of the machines assigned to this role are specified; therefore creating a one-to-many relation between roles and machines.

In the case of *SCIONLab Coordination Service* there is only one role, as the complete architecture runs on a single machine. However, by specifying multiple addresses, the service could easily be replicated for redundancy.

### 6.6.2 Files

Ansible is capable of copying entire files to a target machine. In *SCIONLab Coordination Service* this is used to apply SSH keys and a sudoers file<sup>14</sup> to the deployment environment. This sets up the environment with the necessary user account permissions, required to run *SCIONLab Coordination Service*.

### 6.6.3 Parameters

Certain parameters used in tasks are confidential, do change frequently or are not the same for every target machine. To avoid duplication of playbooks, Ansible allows the use of place holders in tasks.

For simplicity, place holders used in *SCIONLab Coordination Service* are compiled in a dedicated parameter file, together with their corresponding values. Lines 13 and 15 of Listing 6.5 show how place holders are used in a task to confidentially set the password for the MySQL root user. Listing 6.6 shows the corresponding excerpt of the parameter file.

### 6.6.4 Tasks

Ansible runs for each target machine a set of tasks (a playbook), dependant on the assigned role. If tasks are written carefully, the process of running them is idempotent, which means that running the tasks multiple times has no further effect. Idempotence is one of the core concepts of Ansible. It allows tasks to be written in a descriptive way, such that they only get executed when the current state of the machine violates the desired end state. This speeds up the deployment process and prevents data loss caused by writing files which already exist. After completing all tasks, the target machine is in a state that conforms to the state described by the playbook.

Tasks for deploying *SCIONLab Coordination Service* consist of installing required packages, setting up the database, applying SSH keys and checking out the code base. Listing 6.5 shows an excerpt of the playbook used to deploy *SCIONLab Coordination Service*. Lines 2-8 show how required packages are installed. As described above, if a package already exists it will not be downloaded again. This speeds up the deployment. Similarly, tasks for setting up the database (lines 10-20) only run when needed to reach the end state, preventing data loss in case the database already exists.

---

<sup>14</sup>A file containing rules that users must obey when the sudo command is used



## Chapter 7

---

# Evaluation

---

As of writing, *SCIONLab Coordination Service* is running with features developed throughout this thesis actively in use. It is deployed on an [AWS](#)<sup>1</sup> instance and reachable via <https://www.scionlab.org> and <https://coord.scionproto.net>. Several beta testers successfully used *SCIONLab Coordination Service* to register, download and install their own SCION AS; not least enabled by functionality introduced as part of this thesis. These enhancements contribute to the goal of *SCIONLab Coordination Service* being a key component for opening up the *SCIONLab Experimentation Environment* while at the same time maintaining its simplicity.

### 7.1 Deployed Components

The following sections evaluate the enhancements and additions described in Chapter 6 based on the effect they take on *SCIONLab Coordination Service* in its current state and how they are enablers for future improvements.

#### 7.1.1 Robust User Registration

The process of registering an account on *SCIONLab Coordination Service* was greatly improved with respect to authenticity of users, by incorporating the email verification system (Section 6.2) as well as the reCAPTCHA (Section 6.4). This leads to better control over resources made available to users. Figure 7.1 shows the new registration form with the integrated reCAPTCHA widget and the process of validating it. If a user is not exhibiting behaviour suspicious to the reCAPTCHA algorithms, no challenge is required to be solved. Therefore, this protection mechanism does not interfere with the registration process the user goes through.

---

<sup>1</sup>Amazon Web Service: <https://aws.amazon.com/de/>

## 7. EVALUATION

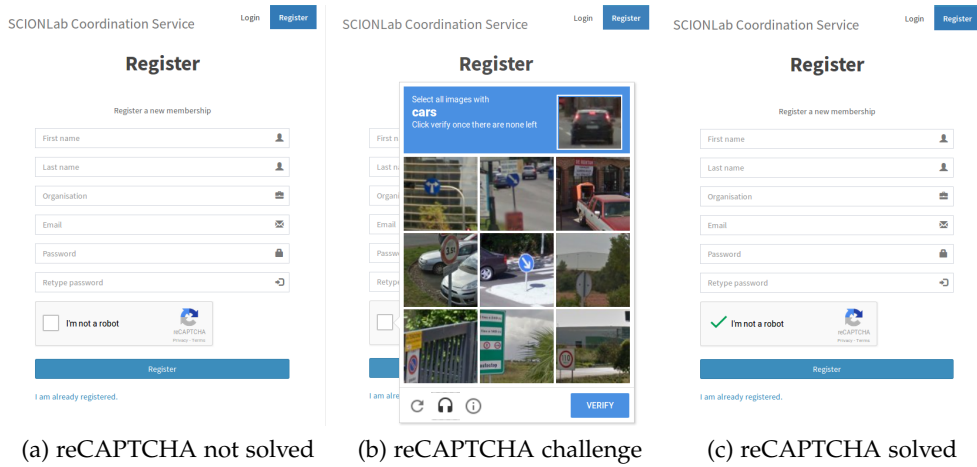


Figure 7.1: reCAPTCHA verification process; challenge (b) is not necessary if the user behaves humanly

For users who have not been invited to use SCIONLab, a registration without solving the reCAPTCHA is not possible; neither via web interface nor by exploiting the relevant API directly. Figure 7.2 shows the error message presented when a user does not solve the reCAPTCHA in the registration form. Figure 7.3 shows the HTTP responses received when addressing the registration API directly; once with an invalid key and once by performing a replay attack.

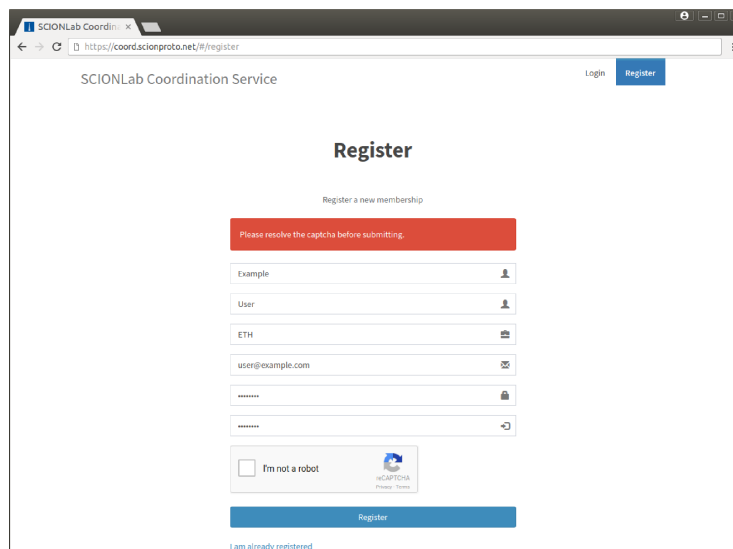


Figure 7.2: User with unsolved reCAPTCHA prevented from registering

## 7.1. Deployed Components

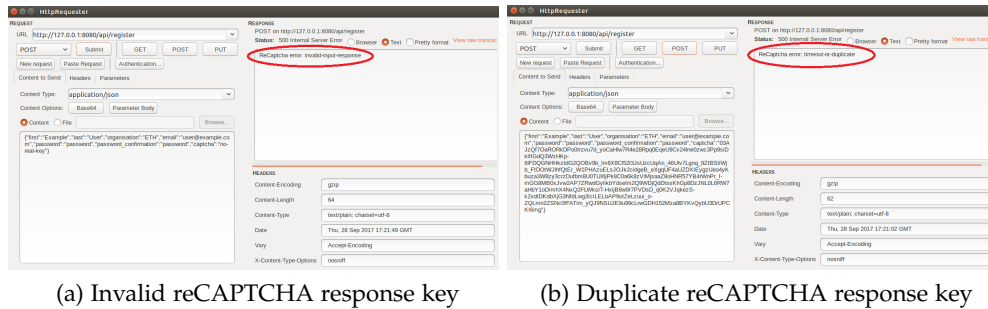


Figure 7.3: Error messages returned when trying to circumvent the reCAPTCHA with invalid or duplicate keys. The generated POST requests mimic the behaviour of a bot trying to register an account. (requests issued using [HttpRequister](#))

When a user does sign up successfully a prompt with further instructions is shown. (see Figure 7.4). Figure 7.5 shows the email containing the verification link sent to the user and the confirmation page loaded upon following the link.

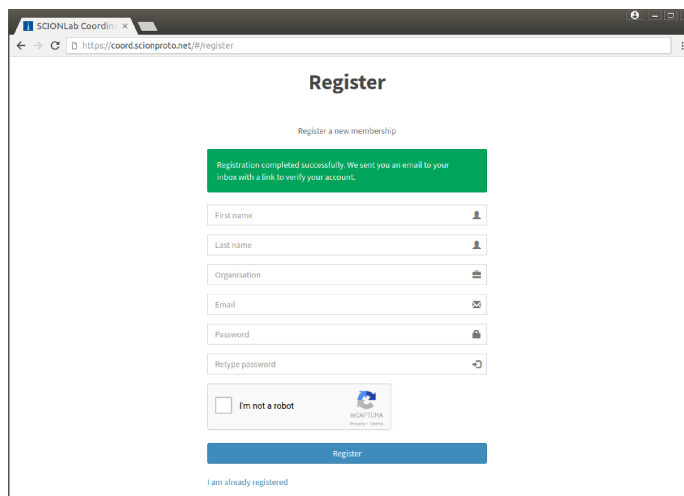


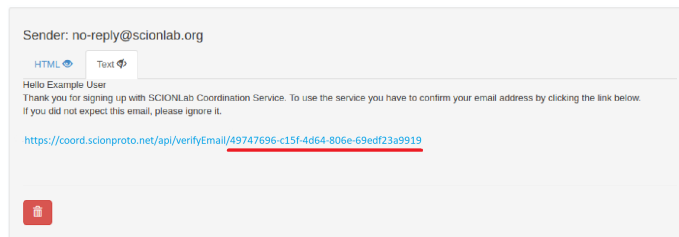
Figure 7.4: Instructions presented to users upon successful registration

Users who registered with an email address that can not be verified are not granted access to *SCIONLab Coordination Service*. (see Figure 7.6)

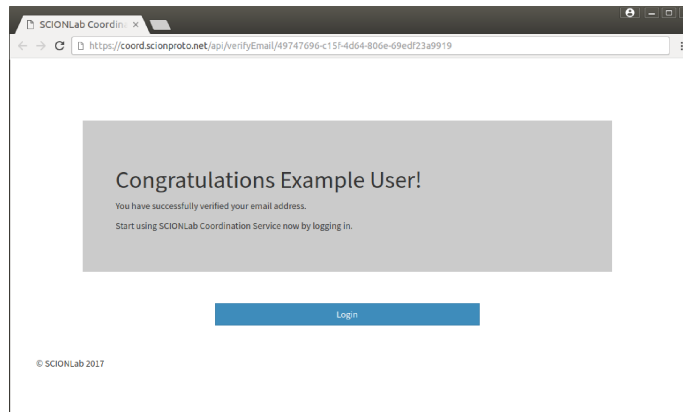
This renovated process stands in contrast to the initial basic implementation that allowed users to register with an email address they do not own or which does not exist at all. Also, by exploiting the exposed API endpoint, it was easily possible to automate the registration process in order to spam

## 7. EVALUATION

---



(a) Verification email sent to user (underlined in red the personalized identifier)



(b) Confirmation page when following personalized link

Figure 7.5: Email verification process with (a) email sent to user and (b) confirmation page

the system. Combined, the email verification system and the reCAPTCHA implementation actively prohibit system abuse.

### 7.1.2 Invitation Based Registration

The manual user activation feature outlined in Section 6.3 is not being used in *SCIONLab Coordination Service* as of writing. The process requires new users to go through yet another round of verification. While originally intended, this proves to be cumbersome both for users as well as system administrators. Since every account has to be activated manually an administrator must log in to the administrator panel frequently in order to not cause delays. This interferes with the requirement of *SCIONLab Coordination Service* being a low overhead and responsive management tool. Instead, a more automated process is envisioned. Figure 7.7 shows the administrator panel for the web interface as developed for the manual user activation feature.

For above reasons, core components such as the role based access control model and the administrator web panel were leveraged by the development



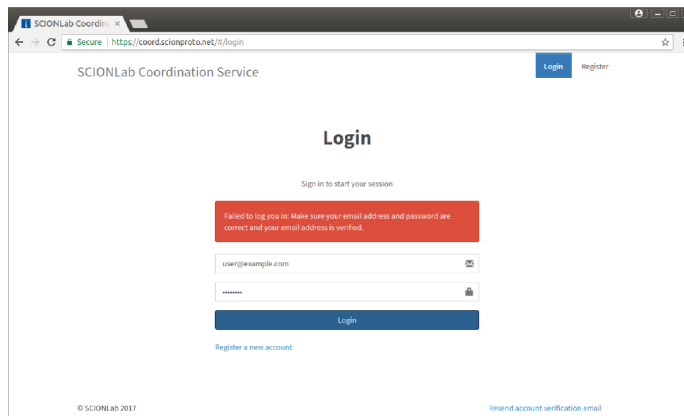


Figure 7.6: User without verified email address prevented from logging in

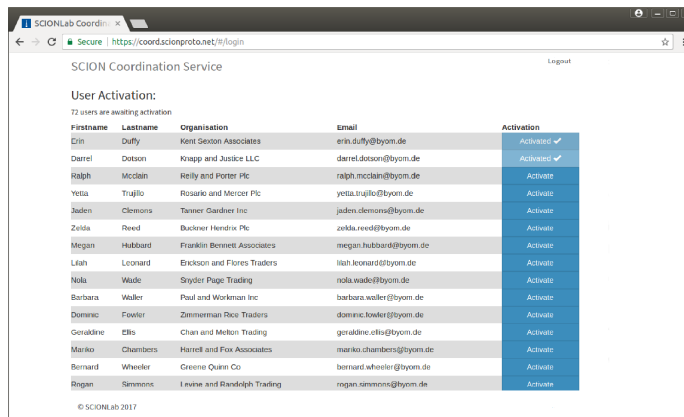


Figure 7.7: Administrator panel on web interface used to manually activate users

team to build a new, invitation based registration model. Via the web panel, administrators can send out invitation emails to specific users. These users are pre-approved and do not go through a verification process.

The components developed for manual user activation (Section 6.3) are extensible. The role based access control model provides a framework for adding additional roles with different sets of permissions to *SCIONLab Coordination Service* as needed. Also, the administrator panel can be extended to handle other administration related tasks.

### 7.1.3 Email Functionality

Apart from the email verification process multiple new improvements made to *SCIONLab Coordination Service* rely on the email package introduced in Section 6.2.1. For one, other developers were able to implement a notifi-

cation system, informing users about status changes of their running ASes. Additionally, the aforementioned invitation based registration system makes use of the email package for sending invitations to recipients.

Email sending in *SCIONLab Coordination Service* is easily extensible with new email templates for use in future enhancements. The email package provides high reliability for sending transactional emails as already discussed in Section 6.2.1. (see Figures 6.4 and 6.5)

### 7.1.4 Integrated Testing

CircleCI integration (Section 6.5) has been set up for *SCIONLab Coordination Service* and used by the development team to validate code contributions. Currently, the test suite of *SCIONLab Coordination Service* consists of 3 unit tests with a total of 15 test cases. CircleCI takes 45-70 seconds to process these tests. This includes building the environment, downloading source code and dependencies and then running the tests. Figure 7.8 shows the output generated for a failed test on CircleCI.

This integration does not require any work for adding additional tests, apart from writing the tests themselves, as they will be picked up and run by CircleCI automatically.

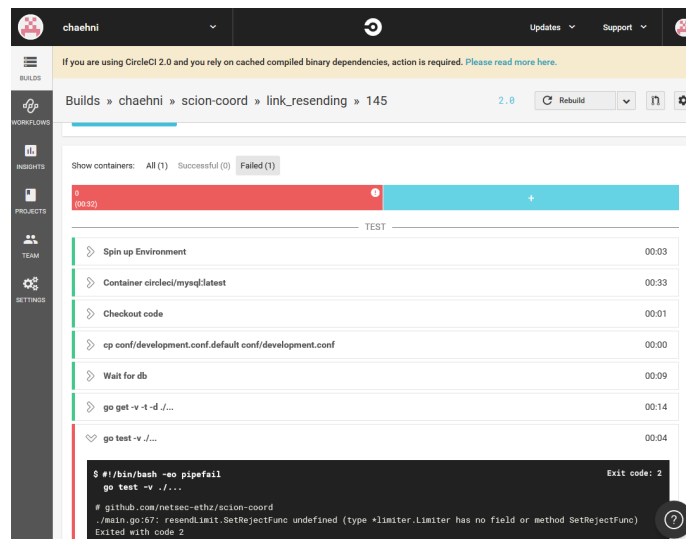


Figure 7.8: Failed test on CircleCI

### 7.1.5 Effortless Deployment

The Ansible playbook for smooth deployment of *SCIONLab Coordination Service* onto remote machines was successfully used throughout this thesis to

deploy newly developed code onto an AWS instance set up for testing purposes. However, the playbook has not been used in production. In the meantime adjustments are necessary to adapt the playbook to the fast evolving *SCIONLab Coordination Service* code base. The reason these adjustments are necessary is that changes made to the service have not been reflected in the Ansible playbook. Most notably, Ansible must be advised to install additional files, such as encryption keys for VPN based connections, to produce the required environment for the service to function correctly with newly introduced VPN connections.

Nonetheless, if maintained, the framework provided could prove useful for future deployments, especially when moving towards a replicated set up of *SCIONLab Coordination Service* where, manual deployment becomes too time consuming.

## 7.2 Requirements Evaluation

This section summarizes to what extent the requirements gathered in Chapter 4 have been satisfied by the work done in this thesis.

### Functional Requirements

1. *A mechanism to verify users' email addresses:*  
This requirement is considered achieved by implementation of the email address verification system. (Section 6.2). The evaluation of this system has been discussed as part of Section 7.1.1
2. *A mechanism to manually activate users who signed up successfully:*  
A mechanism satisfying this requirement has been implemented (Section 6.3). Even though the feature has not been deployed, for reasons depicted in Section 7.1.2, it nonetheless had been functional and, more importantly, provided components to be reused for the invitation based registration process. Therefore, this requirement has been achieved.
3. *A mechanism to protect the service against automated account creation:*  
This feature has been implemented as presented in Section 6.4 and its functional evaluation has been shown in Section 7.1.1. This requirement is thereby fully achieved.
4. *New functionality is validated using CircleCI's testing environment:*  
Implemented (Section 6.5) and deployed, this functionality has been proven to work as described in Section 7.1.4. By these means, the requirement is regarded as fulfilled.
5. *A fast and easy way to deploy the service onto multiple machines:*  
An Ansible playbook had been designed (Section 6.6) and successfully

used to deploy *SCIONLab Coordination Service* onto remote machines (Section 7.1.5). Newer changes introduced to *SCIONLab Coordination Service*, however, have not considered adapting the playbook accordingly. This has left it in a state where it needs adjustments to be fully functional again.

6. *A system for sending notifications to users:*  
By implementing the package for sending emails (Section 6.2.1) the foundation for this requirement had been laid. However, as mentioned in Section 7.1.3, the remaining parts have been added by other developers, not part of this thesis. While overall fulfilled, only parts of the solution have been provided by this thesis.
7. *Implementation of missing APIs between Coordination Service and the Local Management Service:*  
No work has been done with respect to this requirement throughout this thesis. Therefore, it has not been achieved.
8. *A visualization of the SCIONLab Experimentation network accessible by users:*  
This requirement has not been worked on and is as a consequence not achieved by this thesis.

### Non-Functional Requirements

For the non-functional evaluation, rather than specifying whether or not requirements have been achieved, we are interested in what effect changes made to *SCIONLab Coordination Service* have on the given constraints.

1. *SCIONLab Coordination Service aims to be an easy to use tool:*  
The improved registration process (functional evaluation in Section 7.1.1) makes the process of registering an account slightly more involved. Apart from that, no changes have been made regarding ease of use. Overall, *SCIONLab Coordination Service* retains its simplicity.
2. *None of its functionality should require the user to invest a great amount of work:*  
The introduction of the email verification process (Section 6.2) increases the overhead for users by requiring them to check their email inbox. While making use of email callback verification (see Section 6.2.1) would have led to less overhead, this solution has been deemed infeasible because of its unreliability. The approach taken is considered the right balance between simplicity and reliability of the feature.  
  
The CAPTCHA provider has been chosen with the goal of simplicity in mind. As described in Section 7.1.1, reCAPTCHA does not require users to invest a great amount of work.

3. *It is preferred to hide as much complexity as possible from users:*  
All implementations made have been designed to require as little user interaction as possible. As a result, the complexity of added features is mostly hidden from users. An exception is the email verification system which has been discussed above.
4. *The web interface of the service is required to be fast, clean and responsive, since it will be amongst the first impressions users get of SCION:*  
Apart from the added reCAPTCHA widget, the visual appearance of SCIONLab Coordination Service has not been altered by changes made in this thesis. It therefore remains clean and user-friendly.
5. *The web interface needs to be visually appealing, both on desktop and mobile devices:*  
There have been no changes made regarding visual appearance on different screen sizes in this thesis.
6. *In terms of maintainability, SCIONLab Coordination Service aims for easy extensibility as the project evolves fast and new requirements need to be integrated with minimal effort:*  
As described in Appendix A, a variety of improvements have been made with respect to maintainability. Additionally, the implemented components (Chapter 6) reduce the complexity for developing future extensions.



## Conclusion

---

### 8.1 Summary of Achievements

The improvements developed in this thesis are a step towards the envisioned *SCIONLab Experimentation Environment*, where interested parties are provided with a user-friendly management tool to register and download their own SCIONLab ASes. In particular, the enhancements described in this thesis allow the *SCIONLab Coordination Service*, said management tool, to be opened up for public use. The achievement of this goal was supported by contributions made as part of this thesis.

The set of implemented features includes email sending functionality, a new, robust user registration mechanism, a new access control model, improvements for the engineering team regarding testing, deployment and maintainability, as well as many small additions such as error corrections and code re-factorizations to meet new standards.

Most developed software components were tested and proved to work in a live environment. Others were leveraged by the development team to implement additional innovations.

### 8.2 Future Work

From here, *SCIONLab Coordination Service* can be further improved by working on the requirements not addressed by this thesis. These are the following (numbered as in Section 4.1):

7. Implementation of missing APIs between *Coordination Service* and the *Local Management Service*
8. A visualization of the SCIONLab Experimentation network accessible by users

## 8. CONCLUSION

---

In addition to above requirements, the *SCIONLab Coordination Service* could benefit from these enhancements:

- Currently, only one AS can be managed per account. It is a new requirement to increase this limit to support multiple ASes.
- It could be beneficial to have a rate limiting middleware for protecting APIs exposed by *SCIONLab Coordination Service* against DoS<sup>1</sup> attacks.
- Not all functionality of *SCIONLab Coordination Service* is covered by unit tests. Additional tests could minimize manual testing overhead.
- Adapting the Ansible playbook to be used in production would facilitate deployment.
- A task runner for front end code could be leveraged to automate repetitive tasks, such as minifying and unit testing the web interface.

---

<sup>1</sup>Denial of Service



## Appendix A

---

# Security and Maintainability Enhancements

---

### A.1 Ensuring Secure Operation

The sections below outline less extensive but nonetheless critical enhancements made to ensure a secure and stable operation of *SCIONLab Coordination Service*.

#### A.1.1 Server Crash on Login

Attempting to log in to *SCIONLab Coordination Service* with either an empty email address or an empty password caused the back end server to crash, effectively taking *SCIONLab Coordination Service* offline. This was caused by malfunctioning error handling code, resulting in a nil pointer dereference. This defect exposed the service to accidental and malicious attacks, threatening its uninterrupted operation.

The security hole was fixed by correcting the handling of empty user information submitted to *SCIONLab Coordination Service*.

#### A.1.2 Nil Pointer When Accessing User Information

For loading and displaying data corresponding to the right user on the web interface, *SCIONLab Coordination Service* depends on user sessions. The code responsible for validating these sessions contained malfunctioning error handling code which could have led to server crashes caused by nil pointer dereferences.

To secure uninterrupted operation, the faulty code was repaired by correcting the specific error handling.

### A.2 Maintainability

The following enhancements were made to increase *SCIONLab Coordination Service's* maintainability. At the same time, they make sure the service operates flawlessly and retains its compatibility to SCION.

#### A.2.1 Duplicate Configurations

On startup, *SCIONLab Coordination Service* loads configurations from a configuration file and stores them in program variables. The `goconf`<sup>1</sup> package used for this functionality allows configurations to be initialize with default values in case they are not set in the configuration file. This regularly caused confusion since certain settings were set in two different places, where changing it in the wrong place didn't have any effect.

This duplication was resolved by removing default values for configurations.

#### A.2.2 Obsolete HTTP Handling Code

Alongside code used in production, *SCIONLab Coordination Service* contained dead legacy code for serving web pages, including the web page resources themselves. This constantly led to confusion since changes made in the wrong place didn't have any effect on the system.

The code base was cleaned and restructured by removing unused code and resources.

#### A.2.3 Obsolete Database Queries

The package used to interface with the MYSQL database contained queries which were not needed any more. For the sake of a cleaner code base, these queries were deleted.

#### A.2.4 Outdated Dependencies

*SCIONLab Coordination Service* uses a variety of libraries to offer its functionality. These libraries are downloaded and supplied by the vendoring<sup>2</sup> software `govendor`<sup>3</sup>. Because packages were not updated regularly, *SCIONLab Coordination Service* was unaware of changes in the main `SCION`<sup>4</sup> code base it depends on. This led to incompatibility of SCION and *SCIONLab Coordination Service*.

---

<sup>1</sup><https://github.com/sec51/goconf>

<sup>2</sup>In the context of Go, vendoring denotes the process of copying the software packages a project relies on and storing them in the project repository.

<sup>3</sup><https://github.com/kardianos/govendor>

<sup>4</sup><https://github.com/netsec-ethz/scion>

In order to retain compatibility, govendor was instructed to vendor more recent packages. The code base was adapted to work with the changes introduced in SCION.



---

## Bibliography

---

- [1] Adrian Perrig, Pawel Szalachowski, Raphael M. Reischuk, and Laurent Chuat. *SCION: A Secure Internet Architecture*. Springer Verlag, Aug 2017.
- [2] David Barrera, Raphael M. Reischuk, Pawel Szalachowski, and Adrian Perrig. Scion five years later: Revisiting scalability, control, and isolation on next-generation networks. *arXiv e-prints*, pages 1–3.
- [3] François Wirz. Testbed management and network monitoring system for future internet architectures, Aug 2016.
- [4] Anton Ovchinnikov. *Future Internet Architecture Testbed Management System*. Aug 2015.
- [5] PlanetLab. Node requirements. <https://www.planet-lab.org/node/225>, . [Online; accessed 01.10.2017].
- [6] PlanetLab. Hosting requirements. <https://www.planet-lab.org/hosting>, . [Online; accessed 01.10.2017].
- [7] PlanetLab. User’s guide. <https://www.planet-lab.org/doc/guides/user>, . [Online; accessed 01.10.2017].
- [8] Mark Berman, Jeffrey S. Chase, Lawrence Landweber, Akihiro Nakao, Max Ott, Dipankar Raychaudhuri, Robert Ricci, and Ivan Seskar. Geni: A federated testbed for innovative network experiments. *Computer Networks*, 61:5 – 23, 2014.
- [9] Martin Serrano, Nikolaos Isaris, Hans Schaffers, John Domingue, Michael Boniface, and Thanasis Korakis. *Building the Future Internet through FIRE*. River Publishers, Jun 2017.
- [10] Wildbit, LLC. Why postmark? <https://postmarkapp.com/>. [Online; accessed 04.09.2017].

## BIBLIOGRAPHY

---

- [11] Free Software Foundation, Inc. Sender verification. <http://www.fsf.org/about/systems/sender-verification>. [Online; accessed 09.09.2017].
- [12] Google. Tough on bots easy on humans. <https://www.google.com/recaptcha/intro/android.html>. [Online; accessed 17.09.2017].



## Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

**Title of work** (in block letters):

Design and Implementation of Functionality, Security and Maintainability Enhancements for SCIONLab Coordination Service

**Authored by** (in block letters):

*For papers written by groups the names of all authors are required.*

**Name(s):**

Hähni

**First name(s):**

Claude

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the '[Citation etiquette](#)' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

**Place, date**

Zürich, 07.10.2017

**Signature(s)**

C. Hähni

*For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.*