Investigating and evaluating the authentication and authorisation of the JAliEn grid middleware framework

Vilde Kristine Fossum

Master's thesis in Software Engineering at

Department of Computer Science, Electrical Engineering and Mathematical Sciences,
Western Norway University of Applied Sciences

Department of Informatics,
University of Bergen

July 2023

Western Norway
University of
Applied Sciences

Abstract

Grid computing involves a distributed collection of resources worldwide, involving many components. As a result, it presents numerous attack surfaces for potential security threats. This thesis explores the security aspects of grid middleware by examining the JAliEn middleware, a grid framework utilised within the ALICE collaboration at CERN for data distribution throughout the grid.

The primary focus lies in identifying potential vulnerabilities that may compromise data integrity and enable unauthorised access to the system. The study involves discussions with developers, testing the system's behaviour, and analysing token certificates used for authentication and authorisation in JAliEn.

The research identifies two vulnerabilities. Firstly, users can execute more jobs than allowed, potentially leading to system overload, and secondly, job tokens may be exploitable by how easy it is to obtain them. However, a more structured approach and further testing are necessary to assess these vulnerabilities' full extent.

Acknowledgements

First and foremost, I'd like to thank the Grid research group at HVL and my primary supervisor, Bjarte Kileng, for their invaluable technical knowledge and support in understanding the intricacies of the JAliEn system. Bjarte gave me a comprehensive technical introduction to the subject, which was essential throughout this journey.

I thank my second supervisor, Håvard Helstrup, whose guidance and mentorship were invaluable in navigating the complexities of writing a master's thesis. Despite setbacks, Håvard always remained optimistic, offering valuable insights and solutions. Having such support and encouragement made the challenges of writing a thesis more manageable.

I want to offer special thanks to Maksim Storetvedt, whose assistance was invaluable during his PhD work at CERN. Maksim's extensive knowledge of the system and ability to explain concepts in a way I could easily grasp was instrumental in my research.

I thank Costin Grigoras for providing insights into the system, presenting the security requirements, and offering guidance throughout the project. Thank you for sharing your knowledge and helping me with my thesis work.

Finally, I would like to thank family and friends for their unwavering love, support, and encouragement throughout this project and my entire education. Mainly, I want to thank Emil, your unwavering support and belief in my abilities have been a constant source of strength.

Vilde Kristine Fossum
July 28, 2023

Contents

List of Figures

Listings

Acronyms

ALICE A Large Ion Collider Experiment.

AliEn ALICE Environment.

CA Certificate Authority.

CE Computing Element.

CERN The European Council for Nuclear Research.

CLI command-line interface.

DN distinguished name.

JAliEn Java ALICE Environment.

JDL Job Description Language.

LDAP Lightweight Directory Access Protocol.

LHC Large Hadron Colliders.

NIST National Institute of Standards and Technology.

PKI Public Key Infrastructure.

VO Virtual Organization.

WLCG Worldwide LHC Computing Grid.

Chapter 1

Introduction

Distributed computing has been an active research field for over 60 years and has played a significant role in shaping many of the everyday technologies we use, such as the Internet, the World Wide Web, and cloud computing, to name a few [21]. Such systems appear unified and coherent but consist of a collection of computing elements known as nodes. These nodes collaborate and coordinate their efforts to accomplish a shared objective or solve complex problems [33].

One revolutionary form of distributed computing is cloud computing. It has transformed the field by providing access to hardware and software resources over a network. It has given an efficient way to process, store and manage large amounts of data over the Internet [3]. Users can utilise virtualised resources delivered as a service over the Internet [27]. These are easily available and scalable resources. Regarding security, the issues lie in infrastructure, data and storage, and access control in cloud environments [3].

This thesis will explore the security in distributed computing in the context of grid computing, considered the predecessor to modern cloud computing. While grid and cloud computing share some similarities, they operate in distinct ways. Grid computing differs from the cloud regarding the architectural models, approaches and user experiences. Grid typically shares its resources with specific research communities and organisations, while cloud computing is widely adopted across various industries and offers a more standardised and scalable approach.

Grid computing is known for its heterogeneity regarding the underlying infrastructure and the organisations involved. Unlike cloud computing, which typically operates within a single provider's environment, grid computing

involves diverse resources and entities from different organisations. This introduces additional security considerations due to multiple attack surfaces and numerous administrators.

1.1 Grid Computing

Grid computing provides high computing capacity to a distributed system using geographically distributed resources. In grid computing, a user gives a large task, divided into smaller tasks, and then submitted to the grid application for computation. Here, a resource broker finds a match for resource and job in the grid and submits the subtasks. The processed job/task is returned to the user when the job is complete. [30]

1.1.1 Grid middleware

Grid middleware is a software application to connect computing resources. The primary objective is to enable users to access and utilise distributed resources conveniently [34]. Grid middleware plays a role in software engineering by providing an evolving infrastructure layer between the network and applications. Basic Grid services encompass job execution, security, information, and file transfer services. [22]

A grid middleware comprising multiple components, services, and protocols forms the grid's core by interfacing with participating resources or clusters. It provides users with a platform, manages job distribution among computing nodes, and provides each component with necessary tools, including data management, secure transfer, and relevant APIs. [31]

1.2 Software security

Software security is an important field in software engineering as it protects software from malicious attacks and user errors. Neglecting to prioritise security leaves a system vulnerable to attacks, increasing the risk of data loss and compromise from corrupt malware.

1.2.1 Security in distributed systems

Regarding security in distributed systems, two critical aspects are communication and authorisation. Communication involves interactions between

users and processes, which can be located on separate machines. Authorisation should ensure that a process is granted access only to the necessary resources within the distributed system. [33]

There are four security threats to consider; interception, interruption, modification, and fabrication [33]. Interception through unauthorised access to a service or data. Interruption by making services or data unavailable or unstable (denial of service attacks). Modifications involving unauthorised changes to data or tampering with services. Lastly, fabrication entails generating additional data or activity. These security threats can be seen as forms of data falsification.

To build a secure system, a well-defined security policy is essential. This policy outlines the permitted and prohibited actions for various entities, such as users, services, data, machines, etc. Once the security policy has been developed and established, the focus shifts to selecting and implementing the appropriate security mechanisms that align with the policy's objectives. There are four important security mechanisms; encryption, authentication, authorisation and auditing [33].

Encryption is fundamental for computer security in the form of data confidentiality and integrity, transforming information into an unreadable format. Authentication is used to verify the claimed identity of users or entities before accessing services, often using passwords or alternative methods. After authentication, it is necessary to check authorisation. Authorisation ensures that the authenticated clients have permission to perform requested actions, such as modifying specific fields in a database. Lastly, auditing tools are used to trace and record client access, aiding in security breach analysis and deterring attackers from leaving traces. By logging access, the risks associated with attacks are decreased. [33]

1.2.2 Grid Security

Due to the heterogeneous nature of the grid and the multitude of components and resources involved, the entire infrastructure becomes vulnerable once the module is compromised. This complexity makes grid security challenging, requiring constant monitoring to maintain a secure environment. [16]

The point of grid computing is resource sharing, making authentication and authorisation crucial for accessing grid resources. A common approach for user usability is implementing a single-sign-on system, which often relies on Public Key Infrastructure (PKI) based on X509 certificates, as described in Section 2.2.1. [16]

Grid Security Infrastructure

The Grid Security Infrastructure (GSI), formerly known as the Globus Security Infrastructure, has its roots in the deprecated Globus Toolkit middleware [4, 14]. It includes tools, libraries, and protocols that establish the foundational elements required to ensure a secure grid environment. This includes authentication, authorisation, data integrity, and data confidentiality.

1.3 CERN

The European Council for Nuclear Research (CERN) was founded in 1954 as one of Europe's first joint ventures. There are now 23 member states, and it is situated on the French-Swiss border near Geneva, Switzerland. This is where the LHC are, which is the world's largest and most powerful particle accelerator. It consists of a 27-kilometre ring of superconducting magnets with two high-energy particle beams inside. These collide with the corresponding particle detectors at four locations: ATLAS, CMS, ALICE and LHCb. [5, 11]

1.3.1 ALICE

A Large Ion Collider Experiment (ALICE) is one of the four particle detectors at CERN. It is one of the world's largest experiments devoted to studying the physics of strongly interacting matter at high energy densities. The experiment aims to characterise the physical properties of the Quark-Gluon Plasma (QGP). The particle detector is located in the Sergy community, in France, 60 metres below ground. [6]

The raw data collected from LHC for Run 3 and 4 are estimated to be about 3.5 TB/s, but it is reduced to about 100 GB/s of data to store permanently, which is still a lot of data coming through by the second [35]. To store, distribute, and analyse all this data, ALICE uses the Worldwide LHC Computing Grid (WLCG), which uses the storage and computing power contributed by around 170 computing centres in more than 40 countries [12]. ALICE as of today have 74 sites up and running provided by the WLCG.

Different middlewares enable access to the distributed computers within the WLCG across the network, bridging the gap between operating systems and physics-applications software for problem-solving purposes [1].

ALICE Environment (AliEn) was initially developed in 2003 as a prototype middleware system written mainly in Perl, C, and C++ [15]. However, over

time, the system faced challenges in managing the increasing data storage and scaling the infrastructure. As a result, there was a recognised need for modernisation and improvements to address these issues. This resulted in Java ALICE Environment (JAliEn), a grid framework that is used as middleware, on top of WLCG within ALICE [23].

1.4 Problem Description

The security of grid middleware is an interesting topic; to further explore it, an analysis of the JAliEn grid middleware will be done. The JAliEn framework operates globally, interacting with numerous users and computers worldwide, similar to other grid systems. In the past, the primary focus of JAliEn development was on performance, with security and safety considerations, other than authentication and authorisation, sometimes overlooked. However, due to the importance and unique nature of the data transmitted through the system, there is now a focus towards further ensuring data integrity.

While data confidentiality is important, the current emphasis lies on maintaining data integrity, which means preventing unauthorised editing or deletion of data. To address this, the system employs strict authorisation mechanisms facilitated by token certificates. It would be great if someone used the data and discovered something the researcher has yet to find. However, they must never be able to change or delete anything from the grid. Therefore, the data are signed for authenticity rather than being encrypted for confidentiality.

The JAliEn grid middleware is a recently deployed system that undergoes continuous evolution to transition securely from the legacy middleware, AliEn. In the course of Steffen Schreiner's dissertation titled *"A Security Architecture for e-Science Grid Computing"*, a security analysis of the legacy system was conducted to establish a new, enhanced security architecture [29]. This architecture aimed to surpass the security measures of the previous system. Additionally, due to the inefficiencies of the old system in processing much larger datasets, the JAliEn system was developed.

Now that JAliEn is deployed and operational, it is crucial to assess the effectiveness of its new architecture and security features in ensuring the desired level of security. This assessment involves mapping out potential security threats and determining the extent to which they are mitigated or need improvement.

To conduct a thorough analysis, input from the ALICE developers at CERN and the ALICE research group at HVL/UiB is important. Through collaboration with them, the key focus areas for the analysis are determined. One of the main requirements for the system is ensuring data integrity, meaning that unauthorised users or entities should not be able to modify the data.

This thesis will investigate the evolution of authentication and authorisation mechanisms, starting from Schreiner's dissertation and progressing to the implementation of Token Certificates as elaborated in Section 4.3 of this thesis. This research aims to highlight the improvements brought about by the new system and evaluate the efficacy of the authorisation process.

1.5 Research methodology

The research methods employed in this thesis primarily focus on analysing authentication and authorisation within the JAliEn system.

Firstly, a literature study was conducted to better understand the system and the evolutionary aspects of security. This analysis will serve as the basis for further examination and testing of the current implementation.

Subsequently, meetings with the developers at CERN have been arranged to discuss specific security features they consider important or require. These discussions will provide valuable insights into their ongoing work and security-related needs. It is crucial to map out the requirements and identify the necessary security features, which will be accomplished through interviews and meetings with the developers.

Lastly, tests have been conducted to evaluate whether the system meets the required security measures. These tests will consider the feedback the developers provided and the features already implemented during the transition from the old system, considering its previous flaws.

1.6 Research Questions

Research questions are the basis of any research project. It helps to keep the focus on the objective of the thesis and guide the research from start to finish. This thesis aims to evaluate and potentially improve the security of the JAliEn framework. Here are the research questions of this thesis:

Research Question 1: What security protocols are necessary in JAliEn?

Research Question 2: What are the greatest security threats to JAliEn?

Research Question 3: What vulnerabilities can be found within the JAliEn middleware?

Research Question 4: How are token certificates authenticated/work in JAliEn?

Research Question 5: Can a token certificate be misused or go astray?

1.7 Outline

The rest of the thesis is structured into eight chapters. This section will introduce each chapter.

Chapter 2: Background
This chapter provides an overview of the background and objectives of the thesis, focusing on the security analysis and enhancement of the JAliEn system. It introduces JAliEn as a middleware framework within ALICE and its core services and uses of Grid certificates for authentication. The chapter also mentions the significance of Public Key Infrastructure (PKI) and Certificate Authorities in ensuring a secure environment.

Chapter 3: Related Work
Chapter 3 presents an overview of previous work conducted in the context of security within AliEn and JAliEn, as well as in the broader field of grid security. The chapter emphasises grid security, focusing on authentication, authorisation, and data integrity.

Chapter 4: JAliEn
This chapter provides a comprehensive overview of the JAliEn system, offering detailed insights into the process of running a job and the corresponding system operations. Furthermore, it introduces the implementation of Token Certificates.

Chapter 5: Tests performed on JAliEn and their results
This chapter presents the tests performed on JAliEn and the correlating results. There was performed three types of tests.

Chapter 6: Discussion
This chapter presents the research results and the methodology employed to obtain them.

Chapter 7: Conclusion

This chapter will summarise the findings and the research objectives and discuss the results against the research questions. It provides a conclusion of the research's significance and impact.

Chapter 8: Further Work
The final chapter explores the next steps in this research and outlines potential directions for future investigations. It discusses the path forward and identifies areas that require further exploration.

Chapter 2

Background

This chapter provides the background on technologies and systems the thesis discusses. The thesis aims to analyse and evaluate the security of JAliEn. That requires some understanding of the system and the technologies involved.

2.1 JAliEn

Java ALICE Environment (JAliEn) is a middleware framework used within ALICE as mentioned earlier. It is derived from the earlier AliEn middleware. The purpose of JAliEn is to coordinate the computing operations and data management of the collaboration (ALICE) on the grid [23]. JAliEn consists of many components with distinct functions. Some worth mentioning are JCentral, Computing Element, VOBox, Job Agent, and JBox. Figure 2.2 is an illustration of JCentral and its many grid sites. Grid sites are geographically dispersed data centres around the world. Figure 2.1 shows an excerpt of the current sites in mostly Europe.

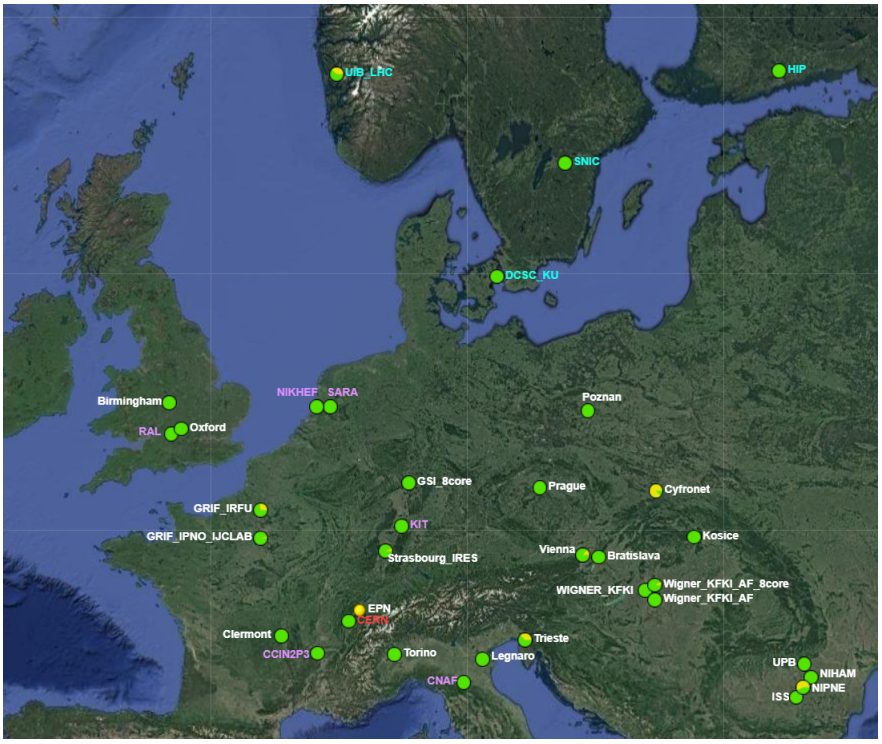

Figure 2.1: ALICE grid sites monitoring map [2]

JCentral, also known as central services, is vital in managing direct access to the core databases [15]. It efficiently handles all incoming requests, performing necessary lookups in the file catalogue, database, or task queue [31]. In Figure 2.2, you can observe components like Catalogue, task and transfer queue, and LDAP. LDAP is further elaborated in Section 2.1.1. The Catalogue, or file catalogue, contains annotations for every file in the grid's distributed storage, including pointers to the files' physical locations [23]. The task queue contains all jobs sent to JAliEn for processing, and the transfer queue contains a list of transfer requests between storage elements.

The Computing Element (CE) is responsible for distributing jobs to worker nodes on a grid site by communicating with JCentral. It resides in the VOBox. The VOBox is a dedicated machine that serves as a cluster entry-point for all the computing activities as a grid site. [31]

The Job Agent has several responsibilities in the middleware. It performs Job Matching, which finds a job in the JCentral Task queue that matches the capabilities of the current computing site. It also handles Environment Preparation, examining the job's JDL to determine the necessary packages

required and then launching a container with the correct environment. Moreover, the Job Agent monitors the job's execution, providing trace information and status updates. [31]

The job is executed on a worker node residing on the grid site.

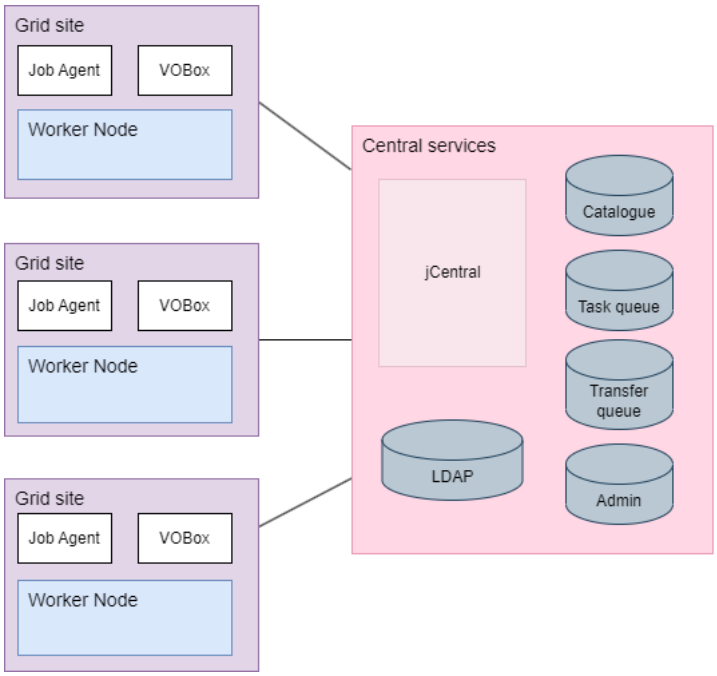

Figure 2.2: JAliEn components

JBox is an end user or a job service that manages client-side authentication and upstream connections [15].

To connect to JAliEn, the client must use their standard Grid certificate [23]. Grid certificates are personal digital certificates, also known as Public Key Infrastructure (PKI) or X.509 certificates. These are signed by CERN or other acceptable certificate authorities so that clients can self-authorise to gain access to the JAliEn grid [28].

Chapter 4 will present the necessary approach for connecting to the JAliEn framework, running jobs, and the implemented security measures.

2.1.1 LDAP

JAliEn uses Lightweight Directory Access Protocol (LDAP), a mature, flexible, and well-supported standards-based mechanism for interactions with

directory servers [19]. It can also be used to authenticate and store information about users, groups, and applications [19]. JAliEn uses LDAP to give permissions to users and sites, storing information on users and sites, as well as site configurations for each VOBox [31]

2.2 Security

This section serves as an introduction to security concepts to better understand the research.

2.2.1 Public key cryptography

A cryptographic algorithm is a set of mathematical functions and procedures for encryption and decryption [20]. It aims to achieve three security objectives: Confidentiality, Data Integrity and Authentication.

Public key cryptography, or asymmetric cryptography, utilises two different keys to encrypt and decrypt. These are generated as a pair and are called a public and a private key.

The public key is available to the public and can be freely distributed. The private key is kept secret and is only known to the key pair's owner. Any data encrypted with the public key can only be decrypted using the corresponding private key. Using a public key is a form of secure communication since only the private key owner can decrypt and access the original message. An example of how this is done is illustrated in Figure 2.3.

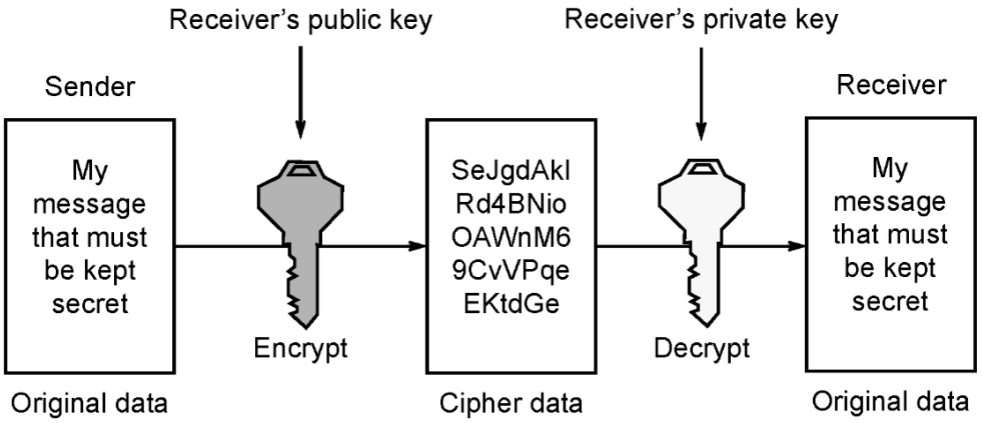

Figure 2.3: PKI [34]

On the other hand, the private key can encrypt data that can be decrypted using the public key. This is often used for digital signatures and authentication, as the receiver can verify the authenticity and integrity of the data.

Public Key Infrastructure

A Public Key Infrastructure (PKI) combines policies, procedures, and technology required to manage digital certificates within a public key cryptography scheme [26]. A digital certificate is a data structure that binds an entity to its public key and ensures secure communication using public key cryptography and digital signatures. A PKI aims to ensure a certificate is trustworthy. [26]

The digital certificate used in JAliEn is X.509 certificates. It is a standard format for public key certificates. Such certificate typically includes fields containing issuer distinguished name (DN), typically the issuing CA, validity period, subjects distinguished name (DN) and Subject public key information among others.

2.2.2 Certificate Authority

According to National Institute of Standards and Technology (NIST), a Certificate Authority is defined as "A trusted entity that issues and revokes public key certificates" [13]. Further details remain found in "NIST special publication 1800-16" volume D, page 18, where it says, "When a PKI has been implemented in an organisation, a CA is used to validate the identity of users and computers.".

A Certificate Authority (CA) is a trusted third-party specialising in issuing and administering digital signatures. A CA can distribute/issue certificates in two ways. The use of a "central CA" and issuing certificates directly to a client or "certification chain" where the CA authorises another entity to do so. The latter is the most common, even though a central CA reduces the number of third parties necessary and ensures that proper professional procedures are followed. However, relying on one single point of failure can have catastrophic consequences if it is compromised. [26]

CERN employs a certificate chain to accommodate the acceptance of different types of CAs due to the various contributing countries and systems involved. This approach allows for flexibility and interoperability in the complex grid environment.

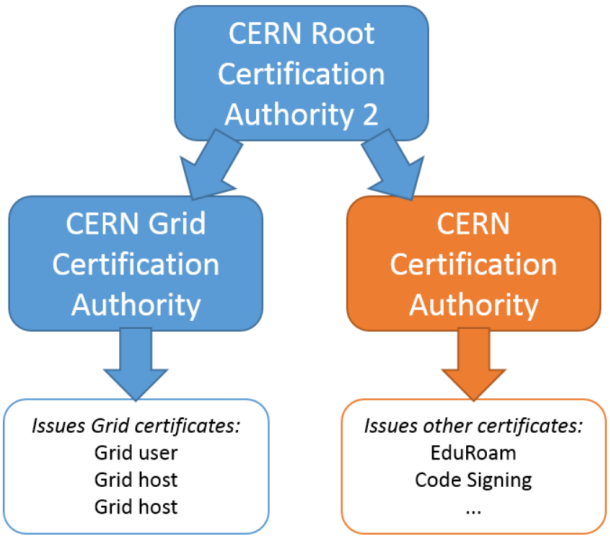

Figure 2.4: CERN CA [7]

At CERN, the digital certificate infrastructure consists of a root CA and two intermediate authorities, CERN Grid Certification Authority and CERN Certification Authority [7]. The CERN Grid CA issues different Grid Certificates such as user, host and robot certificates. The CERN CA issues other certificates such as EduRoam Certificates, Code Signing Certificates and CERN Host Certificates and cannot be used for Grid authentication. This is illustrated in Figure 2.4.

The JAliEn CA works much the same but allows a broader range of CAs, including different national CAs such as the Nordugrid CA [24].

Chapter 3

Related Work

The security of the ALICE grid middleware on CERN has undergone a thorough examination in the past, specifically on the now legacy framework known as AliEn. The most notable investigation was conducted by Steffen Schreiner, a PhD student, in 2014. Schreiner authored a dissertation titled *"A Security Architecture for e-Science Grid Computing"* [29].

In his dissertation, Schreiner analyses the security threats associated with the old AliEn grid framework and proposes a new security architecture that effectively addresses these threats. The study employs established security principles and frameworks to identify the primary security concerns in the AliEn grid system. Based on the analysis, Schreiner proposes a new security architecture, which forms the foundation of today's JAliEn, offering a more robust and comprehensive security framework for e-Science grid computing. Schreiner's dissertation provides valuable insights into the design and implementation of secure grid systems.

Schreiner's analysis of AliEn focuses on various aspects of grid security, including examining security characteristics, potential attacker motives and identifying security objectives to be achieved. Because of this, several important security characteristics were recognised and of interest to this thesis as JAliEn still is a grid with many of the same security characteristics. Addressing security characteristics and understanding the potential attacker's motives helps establish robust security protocols for the grid. [29, p. 35]

Firstly, the grid operates within a public and insecure environment, such as the Internet, where data and communications are exposed to potential threats from malicious actors. [29, p. 39]

Secondly, resource providers, also known as sites, maintain complete control over their infrastructure, encompassing physical and administrative aspects. This control over resources allows sites to regulate access and usage policies and decide how resources are shared with other VOs or individual users. Furthermore, sites play a significant role in facilitating the distribution of resources across the grid, making them responsible for serving multiple consumers or users. [29, p. 39]

Another important consideration is that grid users can introduce arbitrary data and code into the grid and subsequently request its execution within grid jobs. This introduces the possibility of malicious code being executed. The grid lacks direct control over user environments, and users' devices may not always adhere to security standards. This introduces an additional challenge for ensuring data integrity and protection. [29, p. 39-40]

Lastly, the grid involves collaborations between independent organisations and institutions across international borders. Such widespread collaboration requires a high level of trust and efficient security mechanisms to safeguard sensitive data and ensure smooth operations. [29, p. 40]

The potential attacker motives and targets mentioned in Schreiner's work are VO as the primary target (data manipulation or service disruption), the secondary target, such as attacks on individuals, sites or third parties connected to the VO, and the system as a target. [29, p. 40-41]

Lastly, five security objectives were defined as follows [29, p. 41-44]:

- Data integrity, authenticity, and authorship: Ensuring the integrity, authenticity, and provenance of all data within the grid.
- System integrity: Protecting the grid and its infrastructure from unauthorised access and alteration.
- Availability: Ensuring the grid's availability and usability even with affected sub-systems or Sites.
- Non-repudiation of grid jobs: Verifying the authenticity and origin of grid jobs, even after a user's access has been revoked.
- Confidentiality and data privacy: Protecting data and information from unauthorised access, retrieval, or copying.

These security objectives serve as general assessment criteria to guide the security efforts and measures within a grid infrastructure.

In the paper *The Security model of the ALICE next generation Grid Network*

from 2019, the most current security model of JAliEn is presented, based in part on Schreiner's research [25]. The paper introduces Token Certificates as a significant aspect of this model as it aims to unify authentication and authorisation mechanisms for Grid users, jobs, and payload isolation in a way Schreiner's work does not do.

In this context, a token certificate refers to an X.509 certificate combined with an attribute indicating the entity's role. This is done through the certificates DN field is used to separate tokens into groups that correspond to grid roles [31]. Token Certificates are used in the current system and will be explained in detail in Chapter 4 JAliEn in Section 4.3.

Token certificates are time-limited and can represent three types of identity: user, job, or job agent, each with specific permissions based on their roles and responsibilities. These certificates are signed by the JAliEn CA and can only be issued by JCentral. A user must possess a valid Grid Identity certificate to obtain a token certificate.

By default, user tokens have the same rights as the grid user identity certificate and expire after 48 hours, but their validity can be extended. User tokens can be shortened to one hour or extended to match the lifetime of the grid user's identity certificate. To extend the validity, the user needs to authenticate again to limit the amount of time a stolen token could be exploited.

Furthermore, JCentral will also close idle connections that have been inactive for more than two hours. This is an additional security measure to ensure that connections are monitored and controlled, reducing the window of opportunity for any potential unauthorised access.

In another paper published in 2019, titled *JAliEn: the new ALICE high-performance and high-scalability Grid framework*, the authentication and authorisation model of JAliEn is described [23]. This paper also discusses Token certificates and how they incorporate additional functions, such as the role assigned in the system, by encoding them in the DN and extensions of the X.509 certificate. While the paper provides more detailed information in some aspects, it ultimately conveys the same information concerning the authentication and authorisation mechanisms in JAliEn.

Chapter 4

JAliEn

This chapter will provide a more in-depth description of how the current JAliEn implementation operates, how to run a job, and the performance in place concerning security and Token Certificates. In JAliEn, there are two options for submitting a job: alien.py and JShell, both of which serve as a user interface and command-line interface (CLI).

JShell is a Java-based shell that is an integral part of the JAliEn framework. It operates within the JAliEn environment and enables communication with the central services by exchanging serialised objects. JShell is included within the bundled jar files used in the framework, allowing users to interact with the system and perform various operations through a command-line interface.

alien.py serves as a Python reimplementation of JShell within the JAliEn framework. It offers an alternative to JShell by utilising WebSockets for communication instead of Java and serialised objects. This Python-based module, named "xjalienfs", with the program "alienv" is a distinct package or module separate from the main JAliEn components. alien.py provides similar functionality to JShell but with the flexibility and advantages associated with using Python and WebSockets for communication within JAliEn. It was designed to address potential issues arising from different Java and JDK versions associated with JShell. Using alien.py, users can bypass compatibility concerns related to Java.

The one used in this paper is alien.py.

4.1 Connecting to alien.py

To use the alien.py, you can either install CVMFS and download the JAliEn repository from GitLab and run with the command */cvmfs/alice.cern.ch/bin /alienv enter xjalienfs* or download the xjalienfs repository from GitLab and run it locally [17, 18].

To access the service, you need a CERN account, a CERN certificate, and to register with ALICE's virtual organisation [9, 10].

The certificate provided when requesting a certificate is of the type p12. This needs to be converted to PEM format Key pairs, one with the key, userkey.pem, and one for the certificate, usercert.pem [8]. This can be done with OpenSSL. Finally, the last step is to add your certificate and key to the .globus folder in your home directory, typically found at /home/username/.globus. Once this is done, when you enter alien.py, the expected output should resemble Listing 4.1, shown below.

```
[vilde@vilde ~]$ /cvmfs/alice.cern.ch/bin/alienv enter xjalienfs
[xjalienfs] ~ > alien.py
Welcome to the ALICE GRID
support mail: adrian.sevcenco@cern.ch

AliEn[vfossum]:/alice/cern.ch/user/a/v/vfossum/ >
```

Listing 4.1: alien.py connection

4.2 Running a job

Two essential components are required to execute a job: a script and a .jdl file. The .jdl file, which stands for Job Description Language, serves as a job description, outlining various aspects of the job. It declares the input script, potential arguments, output location, and splitting instructions, particularly when initiating a masterjob. When interacting with JAliEn/alien.py, it is the .jdl files that are submitted containing all the necessary information to define and execute the desired job.

When the job is submitted, it progresses through various statuses. The most common initial status is 'I' for inserted, followed by 'ASG' for assigned, 'R' for running, 'SV' for saving, and 'D' for done. In the event of an error, the status will begin with 'E' to indicate an error occurrence. For instance, 'ESV' signifies an error during the saving process. To monitor the status of jobs, the alien.py script incorporates a 'ps' command, which lists all jobs currently

running under your user. Additionally, you can utilise the 'jobInfo' command followed by the 'job id' to obtain specific information about a particular job."

You write the command *submit '.jdl file'* to submit a job. Listing 4.2 shows an example of submission of the .jdl file "sample" that declares testscript.sh as input script and the "ps" command done right after.

```
AliEn[vfossum]:/alice/cern.ch/user/a/v/vfossum/ >submit sample.
    jdl
Submitting /alice/cern.ch/user/a/v/vfossum/sample.jdl
Your new job ID is 2849439259
AliEn[vfossum]:/alice/cern.ch/user/a/v/vfossum/ >ps
vfossum 2849439259   ___   ASG                          testscript.sh
```

Listing 4.2: Submitting a job

Now what happens in the system when a job is submitted? Figure 4.1 shows a simplified model of the process of running a job. A job is submitted, sent to central services via Web Sockets, and added to the Task queue in JCentral.

Subsequently, sites worldwide have their CE check if there are any pending jobs. If a site finds there is an available job, it generates a Job Agent startup script and places it in the site's batch queue. To do this, the CE needs to request a Job Agent token to put in the startup script, among other things.

This is then sent to a free worker node that starts a Job Agent. At this stage, the Job Agent matches and fetches the JDL and starts a container with the job id, as mentioned in Section 2.1, where it begins execution. It is the Job Wrapper that is in charge of the execution. During execution, the container communicates with JCentral if it is necessary, as shown in the figure. This is done through a component called TJAliEn, but this is not relevant to this research.

As described, there are many components to the system and thereby have several attack surfaces, as mentioned in Chapter 1.

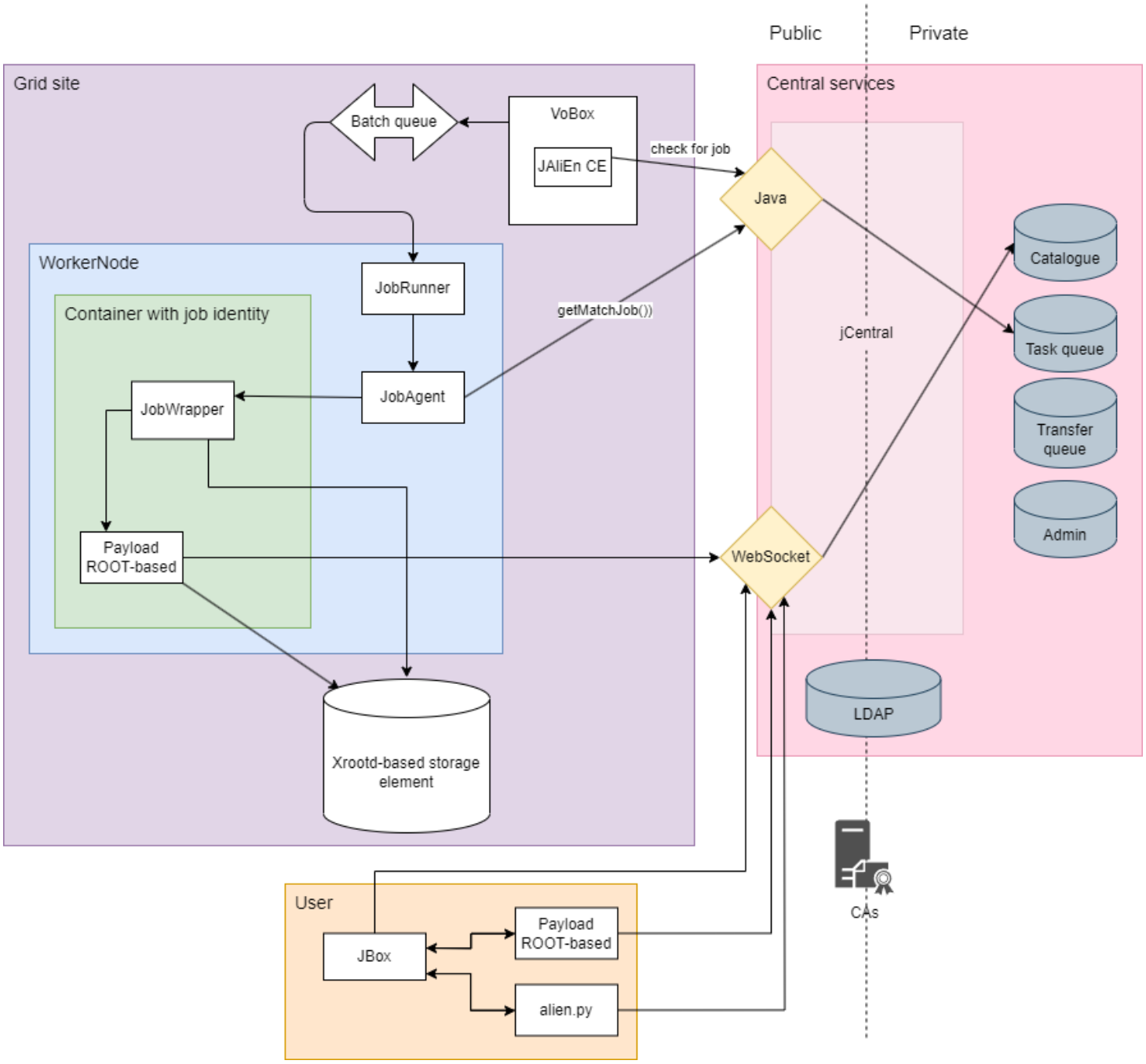

Figure 4.1: Flow of JAliEn

4.3 Token Certificate

This thesis emphasises the utilisation of Token Certificates as a key security feature, a new addition to the JAliEn framework compared to the previous AliEn implementation. A Token Certificate, as used in JAliEn, comprises an X.509 certificate and a specific role assigned within the system, as described in Chapter 3. This differs from the legacy AliEn implementation, which relied solely on the X.509 certificate without role assignment.

The system defines three primary roles: user, job, and job agent, each associated with different permissions and access levels. Additionally, there is the host token certificate. Though the host token certificate is not the only way for a host to authenticate itself.

The list of Token Certificates is as follows:

- Host token certificate
- Job Agent token certificate
- Job token certificate
- User token certificate

A user token has the same permissions as the user grid identity, listed in LDAP. A job token can only be requested by a Job Agent and only have the permission to execute the payload, as in the job token gets all the permissions of the user who submitted the job and work on the job as that user. A Job Agent token can only be requested by VOBox services and does job matching, updating job status, and uploading job trace, as described in 2.1.

Figure 4.2 illustrates the token flow during the execution of a job. The user possesses a user token certificate, which is utilised when submitting a job to the central services. Similarly, a VOBox can hold a host token certificate, which it employs to check for jobs. Another way is for the host to use its "normal" grid certificate/grid identity and authenticate with the permissions kept in LDAP.

After the site finds available jobs, the Job Agent token is included in the startup script, as mentioned in the previous section. The script is then moved to the batch queue and handed to the worker node that starts the Job Agent. When the Job Agent identifies a suitable match for the site, it fetches the job and requests a Job token specific to that job. The Job token is required by the container environment responsible for executing the job, as it needs the necessary permissions to communicate with JCentral. Additionally, to ensure

that the container Client does not interfere with unauthorised operations, it acquires the job-specific Job token. This way, the container Client can safely perform its tasks with the required permissions, and the job execution remains secure and isolated from other processes.

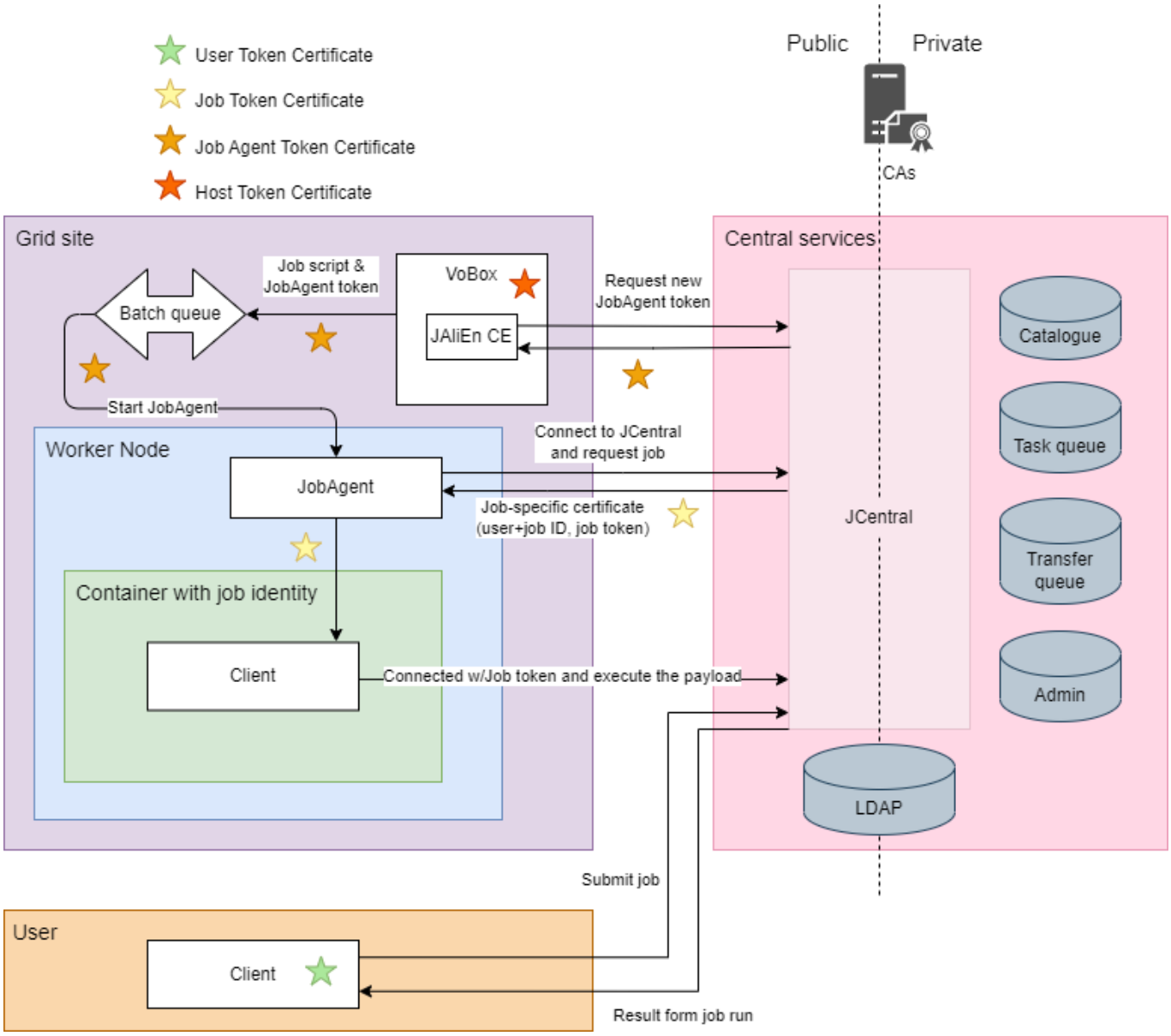

Figure 4.2: Flow of token in JAliEn

Chapter 5

Tests performed on JAliEn and their results

In this chapter, the tests performed on the system to address the research are described, as well as their results.

5.1 Token requests

The assessment involved the evaluation of token-related commands, aiming to gain familiarity with the system and explore the range of function calls. The main objective focused on comprehending the system and assessing the capability to request tokens.

One of the commands in the alien.py script is 'token-init,' which allows users to request tokens of specific types. However, it should be noted that the user 'vfossum' does not possess any roles associated with being a host, JobAgent, or VOBox. Therefore, they should not have the capability to request such tokens. Listing 5.1 shows an excerpt from one of the inquiries. In this example, the command includes the options: -u (username), -t (tokentype), -jobid (jobDN extension).

```
1 AliEn[vfossum]:/alice/cern.ch/user/a/v/vfossum/ >ps
2 vfossum 2849439259   ___    D                    testscript.sh
3 AliEn[vfossum]:/alice/cern.ch/user/a/v/vfossum/ >token-init -u
    vfossum -jobid
4 2849439259
5 Remote I/O error : Server didn't execute your request, reason
    was: Only a JobAgent can ask for a Job token, [vfossum] is
    not one
```

```
6  The token could not be created! check the logfile /home/vilde/
       alien_py.log
7  Remote I/O error : Server didn't execute your request, reason
       was: Only a JobAgent can ask for a Job token, [vfossum] is
       not one
8  DN >>> C=ch/O=AliEn2/CN=Users/CN=vfossum/OU=vfossum
9  ISSUER >>> C=ch/O=AliEn2/CN=AliEn CA
10 BEGIN >>> 2023-05-12 14:15:23
11 EXPIRE >>> 2023-06-12 16:15:23
```

Listing 5.1: requesting job token

In each test, the request was denied, as expected.

5.2 Overloading test

One initial concern was the potential for a user to submit an excessive number of jobs, which could overload the system. While it was mentioned that efforts were made to address this issue, the current status of the resolution remains to be discovered [32].

To mitigate the risk of system overload, there are restrictions in place regarding the number of jobs a user can submit. Typically, users are not allowed to submit more than approximately 2000 jobs; in most cases, the limit is set to around 100 jobs. In this specific test case, the user had a limit of 100 jobs.

This limitation is crucial to prevent CPU resources from being wasted on potentially incorrect or less critical jobs. The system can prioritize and allocate resources efficiently by enforcing job submission limits.

5.2.1 Splitting and submitting a masterjob above limit

To begin to test this problem, the goal is to try submitting 2500 jobs using what is called a masterjob. A masterjob splits a task and submits it an explicit number of times. This is declared in the .jdl file shown in Listing 5.2.

```
1 Executable="/alice/cern.ch/user/a/v/vfossum/sleeper-test.sh";
2 Split="production:1-2500"; #submit job 2500 times
3 Output = {"stdout@disk=1"};
```

Listing 5.2: sleeper-masterjob.jdl

To have minimal effect on the system and to reduce the risk of compromising the system, the actual job is a script that sleeps. Because it takes time to

submit all the jobs, the sleep time, as seen in Listing 5.3, is three days. The goal is to test if every job starts to run, not if they run its entire course.

```
#!/bin/bash
sleep 3d #waits 3 days to be sure all jobs start.
```

Listing 5.3: sleeper-test.sh

Surprisingly all jobs started, even when splitting the job into 5000 jobs. However, when trying to submit the same masterjob again while the previous masterjob was running, the job failed. Upon sharing the results with developers at CERN, it was revealed that one of them had started working on a solution to the problem. It seemed that when a user submitted a job while under the limit, every split worked, but when already over the limit, the job was stopped. As a result, the user who submitted the job could run more than the permitted limit of 100 jobs simultaneously if committing the master job while under the limit.

5.2.2 Submitting individual jobs above limit

The second part of the overloading test was to do the same thing but by submitting individual jobs and not using masterjob and split. This will further check if the statement in the previous section stands. Will the job be stopped when passing 100 this time?

This time the jdl did not include a split, as shown in Listing 5.4, but the sleeper script was the same, shown in Listing 5.3.

```
Executable = "/alice/cern.ch/user/a/v/vfossum/sleeper-test.sh";
Output = {stdout@disk=1};
```

Listing 5.4: sleeper.jdl

To avoid having to submit each job manually, a submit script was created, shown in Listing 5.5. This script was run from outside of alien.py.

```
#!/bin/bash
for i in {1..2500}
do
    alien.py submit sleeper.jdl
done
echo "Done, I'm a submit test"
```

Listing 5.5: submit-file.sh

Initially, the test was conducted with 200 job submissions, and they all successfully started running. Subsequently, a test with 2500 job submissions was performed, and once again, every job started running. This outcome was unexpected and raised concerns. While it is not a common practice to submit such a large number of jobs without using a masterjob, it highlighted a breach in the system's rules and restrictions. Although the developers were not overly alarmed, this vulnerability could potentially be exploited and pose a risk to the system's integrity.

5.3 Exploitation of environment variables

This test attempts to gain access to keys or tokens through the environment variables of the machine that runs the job. To do this, the script enviro-test.sh, Listing 5.6, was created with the corresponding jdl file, Listing 5.7. The script prints all the environmental variables, and the .jdl file stores the output in the output_dir_enviro folder in the submitting user's home directory, in this case, vfossum.

```
1 #!/bin/bash
2 printenv
3 echo "Done, I'm an environment test"
```

Listing 5.6: enviro-test.sh

```
1 Executable = "/alice/cern.ch/user/a/v/vfossum/enviro-test.sh";
2 Output = {stdout@disk=1};
3 OutputDir = "/alice/cern.ch/user/a/v/vfossum/output_dir_enviro/"
      ;
```

Listing 5.7: enviro.jdl

The results from the run resulted in the file shown in Appendix A. What was interesting in the results was the variable JALIEN_TOKEN_KEY, holding the location to the token key, */workdir/jobtoken7377736111310526252.pem*, and JALIEN_TOKEN_CERT variable pointing to the token certificate at */workdir/jobtoken5712500697710563999.pem*.

Now the next step is to try printing the content of the files to output and check if a user has access to them through the job. A test to print the files is performed to check if a user can access the key and certificate files. Listing 5.8 shows the source code of the test, and Listing 5.9

```
#!/bin/bash
echo "JALIEN_TOKEN_KEY: "
cat $JALIEN_TOKEN_KEY

echo "JALIEN_TOKEN_CERT: "
cat $JALIEN_TOKEN_CERT

echo "Done, I'm an environment get test"
```

Listing 5.8: enviro-get-test.sh

```
Executable = "/alice/cern.ch/user/a/v/vfossum/enviro-get-test.sh
    ";
Output = {stdout@disk=1};
OutputDir = "/alice/cern.ch/user/a/v/vfossum/
    output_dir_enviro_get/";
```

Listing 5.9: enviro-get.jdl

This resulted in both the key and certificate being printed to the job output seen in Appendix B. Now the question is: What can be done with this information? Can the job token be used to exploit the system?

To test if a token could be exploited, the key and certificate were added to two PEM files, usercert.pem and userkey.pem. One potential exploit could be using these credentials to authenticate when entering alien.py, as described in Section 4.1, and run a test job.

The result of this test was successful, as it allowed the user to authenticate and enter alien.py. This outcome might not be surprising, as the token inherits the permissions of the user who provided the job. However, what remains to be investigated is whether the token possesses more permissions than it should, potentially enabling unauthorised actions, such as editing or deleting files in other users' repositories. This aspect warrants further examination to assess the extent of the token's privileges and any possible security implications.

With permission from two other users, both an admin and a regular user, a test on its repository was done. Testing editing on their files. Listing 5.10 shows an illustration of the outcome.

```
...
-rwxr-xr-x    username2 username2           203 Mar 22 09:31
    test1.jdl
...
AliEn[vfossum]:/alice/cern.ch/user/u/username2/ >mv test1.jdl
    test1-exploited.jdl
Input/output error : Failed to move test1.jdl to /alice/cern.ch/
    user/u/username2/test1-exploited.jdl
AliEn[vfossum]:/alice/cern.ch/user/u/username2/ >nano hello.sh
Specified source /alice/cern.ch/user/u/username2/hello.sh not
    found!
Error downloading /alice/cern.ch/user/u/username2/hello.sh,
    editing could not be done.
```

Listing 5.10: Exploitation test illustration 1

In Listing 5.11, there is an attempt to change a file by adding a comment saying *"Exploitation worked"*, but when saving, the result showed an error as expected if the permissions were correct.

```
AliEn[vfossum]:/alice/cern.ch/user/u/username2/ >nano test1.jdl
Input/output error : Failed to move /alice/cern.ch/user/u/
    username2/test1.jdl to /alice/cern.ch/user/u/username2/test1.
    jdl~
Error uploading
```

Listing 5.11: Exploition test illustration 2

Chapter 6

Discussion

This chapter will discuss the results from the research and the method used to get these results.

The results from testing revealed two significant findings: firstly, a user can execute a higher number of jobs than they are authorised to, and secondly, users can easily access the certificate and key associated with a job token. While it is important to acknowledge the testing methodology's limitations, both statements remain valid. However, drawing definitive conclusions regarding the implications for the system is challenging. Although initial tests were conducted to assess the potential impact of these vulnerabilities and indicated that the risks were not severe, a more comprehensive testing approach would be necessary to fully evaluate the extent of the vulnerability in the JAliEn system.

Regarding the result indicating that a user cannot request a job certificate, it is important to note that drawing definitive conclusions based on a single test is difficult. While the test result may indicate that the user could not request a job certificate, it does not necessarily imply that the system is entirely secure against exploitation. Alternative approaches to exploit the system were not discovered during these tests. Therefore, further investigation and additional tests are required to assess the overall vulnerability of the system to unauthorised job certificate requests.

As described in Chapter 5, three different types of tests were conducted, focusing on a broad range of system security aspects. However, to obtain deeper insights, it would have been beneficial to have a more interconnected approach that specifically targeted a particular breach or aspect of security. This would have provided more detailed results that carry greater weight

when assessing the system's overall security.

Unfortunately, the overloaded masterjob test results were unsurprising to the developers, as a PhD student was already investigating the issue. However, there needed to be more effective communication within the group regarding this matter, though they newly started using a change log to inform of the latest changes to the code and the system. On the other hand, the unexpected results of the manual submission test provided valuable insights, even though it did not pose a critical breach. The results obtained may be related to job quotas, but they could also indicate other issues within the system that are not functioning correctly. It is unclear which specific entity was affected based solely on the test conducted.

Another point of improvement regarding understanding the system was the difficulty of locating documentation, even though it did exist scattered across various places and sites. However, frequent and accessible communication with the developers benefited the research.

In hindsight, dedicating more time to information gathering would have been beneficial, as the research encountered multiple halts due to insufficient knowledge of the system and its inner workings. Consequently, a significant portion of the work was devoted to understanding the system, which ultimately consumed time that should have been dedicated to testing and in-depth research on the JAliEn framework.

As grid computing is considered a predecessor to cloud computing, it becomes challenging to find modern or recent documentation and research in the field, especially since grids are primarily used for large computations, often related to physics data.

Chapter 7

Conclusion

This chapter serves as the conclusion of the research, summarising the efforts to investigate each research question and presenting the corresponding answers.

Research Questions 1 and 2: What security protocols are necessary for JAliEn? What are the greatest security threats to JAliEn?

Discussions with developers revealed that data integrity is one of the most important security features for JAliEn, but not the only one needed. To protect the data, good authentication and strict authorisation methods are necessary.

The greatest threat would be someone with not necessarily but possibly malicious intent attempting to delete or remove files from the system. This emphasises the necessity of data integrity and proper authorisation to prevent unauthorised access and tampering with data.

This statement would benefit from more support from research. While it is not entirely false, additional documentation and proof could further strengthen the validity of the research findings and eliminate any doubts regarding their accuracy.

Research Question 3: What vulnerabilities can be found within the JAliEn middleware?

The discussions with developers and research identified two possible vulnerabilities in the system, which had been breached before and could still present a threat. The first vulnerability involves token exploitation or tokens with unwarranted permissions, which could lead to unauthorised access to sen-

sitive resources. The second vulnerability is related to users being able to submit more jobs than their allowed limit, potentially causing an overload on the system. These vulnerabilities require careful attention and mitigation to ensure the security and stability of the JAliEn framework.

The testing revealed that the job submission process posed a threat that was not completely mitigated. Although efforts were underway to address the issue, the testing uncovered a vulnerability the developers were unaware of. This highlights the importance of testing and continuous improvement in the security measures of the JAliEn framework to enhance its resilience against potential threats and attacks.

Research Question 4: How are token certificates authenticated/ work in JAliEn?

As for research question three, a substantial amount of information has been presented. With that, the conclusion would have to be that the question was answered. This aspect has been a major portion of the research, providing valuable insights into the inner workings of token certificates used in JAliEn and the benefits they offer to the system.

Research Question 5: Can a token certificate be misused or go astray?

This was a question brought up during a discussion with the developers. Could a token be stolen and misused? To try to find out the test described in Section 5.3, "Exploitation of environment variables" was initiated. This resulted in access to the job token certificate and key to the environment of the job. Due to insufficient testing of the implication, this question can not be categorised as answered completely. Though a user can gain access to a job token and log in with it, it is still within the user's permissions and not really a completely different token than the user token.

As discussed in Chapter 6, the research could have benefited from a narrower scope and dedicated more time and resources to each research question. By doing so, a more comprehensive and detailed analysis could have been conducted, leading to more precise and robust answers to the research questions. Focusing on fewer or smaller questions would have allowed for deeper investigations and a more in-depth understanding of the security aspects of the JAliEn framework. Future research endeavours could consider this approach to enhance the quality and effectiveness of the research outcomes.

Chapter 8

Further Work

This chapter outlines the areas for further investigation and work, considering the research's limitations in addressing all the research questions and the need for more comprehensive testing. The following aspects should be explored in future work:

- The significance and potential risks of the environmental variable should be thoroughly assessed, despite some initial testing being conducted.
- Investigate the implications of obtaining the token and key together.
- Verify if the token possesses more permissions than it should, as this could be a potential vulnerability in the system.
- Explore alternative and more sophisticated methods of acquiring a job token, analysing if there are potential loopholes or vulnerabilities in the process. Ensure that a job token is restricted to only enabling file upload and download operations within the context of the submitting user's permissions.
- Determine the scope and extent of actions that can be carried out with a job token, including the specific permissions it grants.
- Investigate the relationship between the site, job agent, and job components, understanding how they interact and impact security measures.
- Analyse if any tokens possess unauthorised permissions, identifying potential security gaps that must be addressed.
- Research to which extent it is possible to take control of a machine or a site, either through tokens and permissions or by submitting a job

containing malicious code.

In future work, as discussed in Chapter 6, adopting a more structured approach is necessary to ensure reliable and accurate scientific results. This could involve focusing on smaller parts of the system and examining each component individually. Exploring other security aspects and attack surfaces would also be beneficial.

Finally, based on the experience gained during this thesis, an interesting aspect to consider in future work is gathering all information and documentation about the system in one centralised location. This would facilitate easier access and understanding of the system for future researchers and users and less time used to get familiar with the system.

Appendix A

enviro-test.sh output

```
1  AliEn[vfossum]:/alice/cern.ch/user/a/v/vfossum/ >less
       output_dir_enviro/stdout
2  JALIEN_TOKEN_KEY=/workdir/jobtoken7377736111310526252.pem
3  JALIEN_HOME=/alice/cern.ch/user/a/v/vfossum/
4  SSL_CERT_FILE=/cvmfs/alice.cern.ch/el7-x86_64/Packages/Python-
       modules/1.0-371/share/python-modules/lib/python/site-packages
       /certifi/cacert.pem
5  ROOT_INCLUDE_PATH=/cvmfs/alice.cern.ch/el7-x86_64/Packages/
       OpenSSL/v1.1.1m-13/include
6  APPTAINER_COMMAND=exec
7  TMPDIR=/workdir/tmp
8  ALICE_TARGET=linuxx8664gcc
9  X509_CERT_DIR=/cvmfs/alice.cern.ch/el7-x86_64/Packages/AliEn-
       Runtime/v2-19-le-138/globus/share/certificates
10 LIBXML2_ROOT=/cvmfs/alice.cern.ch/el7-x86_64/Packages/libxml2/v2
       .9.3-84
11 LD_LIBRARY_PATH=/cvmfs/alice.cern.ch/el7-x86_64/Packages/XRootD/
       v5.5.3-14/lib:/cvmfs/alice.cern.ch/el7-x86_64/Packages/
       libxml2/v2.9.3-84/lib:/cvmfs/alice.cern.ch/el7-x86_64/
       Packages/Python-modules/1.0-371/share/python-modules/lib:/
       cvmfs/alice.cern.ch/el7-x86_64/Packages/OpenSSL/v1.1.1m-13/
       lib:/cvmfs/alice.cern.ch/el7-x86_64/Packages/Python/v3
       .9.12-31/lib:/cvmfs/alice.cern.ch/el7-x86_64/Packages/libffi/
       v3.2.1-59/lib64:/cvmfs/alice.cern.ch/el7-x86_64/Packages/
       sqlite/v3.15.0-69/lib:/cvmfs/alice.cern.ch/el7-x86_64/
       Packages/libpng/v1.6.34-148/lib:/cvmfs/alice.cern.ch/el7-
       x86_64/Packages/zlib/v1.2.8-112/lib:/cvmfs/alice.cern.ch/el7-
       x86_64/Packages/FreeType/v2.10.1-107/lib:/cvmfs/alice.cern.ch
       /el7-x86_64/Packages/AliEn-Runtime/v2-19-le-138/lib:/cvmfs/
       alice.cern.ch/el7-x86_64/Packages/GCC-Toolchain/v7.3.0-alice1
       -9/lib64:/cvmfs/alice.cern.ch/el7-x86_64/Packages/GCC-
       Toolchain/v7.3.0-alice1-9/lib:/.singularity.d/libs
```

```
12 JALIEN_USER=vfossum
13 GCC_TOOLCHAIN_ROOT=/cvmfs/alice.cern.ch/el7-x86_64/Packages/GCC-
       Toolchain/v7.3.0-alice1-9
14 SINGULARITY_NAME=centos7
15 SQLITE_ROOT=/cvmfs/alice.cern.ch/el7-x86_64/Packages/sqlite/v3
       .15.0-69
16 ALIEN_JOB_TOKEN=oo^zt)Rf54^RmtgcsPVofPA3sI)uQrz{
17 ALICE_TARGET_EXT=linuxx8664gcc
18 PATH=/cvmfs/alice.cern.ch/el7-x86_64/Packages/JAliEn/1.7.2-1/bin
       :/cvmfs/alice.cern.ch/el7-x86_64/Packages/xjalienfs/1.4.5-22/
       bin:/cvmfs/alice.cern.ch/el7-x86_64/Packages/XRootD/v5
       .5.3-14/bin:/cvmfs/alice.cern.ch/el7-x86_64/Packages/libxml2/
       v2.9.3-84/bin:/cvmfs/alice.cern.ch/el7-x86_64/Packages/Python
       -modules/1.0-371/share/python-modules/bin:/cvmfs/alice.cern.
       ch/el7-x86_64/Packages/OpenSSL/v1.1.1m-13/bin:/cvmfs/alice.
       cern.ch/el7-x86_64/Packages/Python/v3.9.12-31/bin:/cvmfs/
       alice.cern.ch/el7-x86_64/Packages/sqlite/v3.15.0-69/bin:/
       cvmfs/alice.cern.ch/el7-x86_64/Packages/libpng/v1.6.34-148/
       bin:/cvmfs/alice.cern.ch/el7-x86_64/Packages/AliEn-Runtime/v2
       -19-le-138/bin:/cvmfs/alice.cern.ch/el7-x86_64/Packages/GCC-
       Toolchain/v7.3.0-alice1-9/bin:/cvmfs/alice.cern.ch/el7-x86_64
       /Packages/JDK/12.0.1-6/bin:/usr/local/sbin:/usr/local/bin:/
       usr/sbin:/usr/bin:/sbin:/bin:/cvmfs/alice.cern.ch/bin
19 APPTAINER_APPNAME=
20 APPTAINER_ENVIRONMENT=/.singularity.d/env/91-environment.sh
21 _=/usr/bin/printenv
22 JALIEN_HOST=127.0.0.1
23 ALIEN_PROC_ID=2849464181
24 PWD=/workdir
25 _LMFILES_=/cvmfs/alice.cern.ch/el7-x86_64/Modules/modulefiles/
       BASE/1.0:/cvmfs/alice.cern.ch/el7-x86_64/Modules/modulefiles/
       JDK/12.0.1-6:/cvmfs/alice.cern.ch/etc/toolchain/modulefiles/
       el7-x86_64/Toolchain/GCC-v7.3.0:/cvmfs/alice.cern.ch/el7-
       x86_64/Modules/modulefiles/GCC-Toolchain/v7.3.0-alice2-30:/
       cvmfs/alice.cern.ch/el7-x86_64/Modules/modulefiles/AliEn-
       Runtime/v2-19-le-138:/cvmfs/alice.cern.ch/el7-x86_64/Modules/
       modulefiles/FreeType/v2.10.1-107:/cvmfs/alice.cern.ch/el7-
       x86_64/Modules/modulefiles/zlib/v1.2.8-112:/cvmfs/alice.cern.
       ch/el7-x86_64/Modules/modulefiles/libpng/v1.6.34-148:/cvmfs/
       alice.cern.ch/el7-x86_64/Modules/modulefiles/sqlite/v3
       .15.0-69:/cvmfs/alice.cern.ch/el7-x86_64/Modules/modulefiles/
       libffi/v3.2.1-59:/cvmfs/alice.cern.ch/el7-x86_64/Modules/
       modulefiles/Python/v3.9.12-31:/cvmfs/alice.cern.ch/el7-x86_64
       /Modules/modulefiles/OpenSSL/v1.1.1m-13:/cvmfs/alice.cern.ch/
       el7-x86_64/Modules/modulefiles/Python-modules/1.0-371:/cvmfs/
       alice.cern.ch/el7-x86_64/Modules/modulefiles/libxml2/v2
       .9.3-84:/cvmfs/alice.cern.ch/el7-x86_64/Modules/modulefiles/
       XRootD/v5.5.3-14:/cvmfs/alice.cern.ch/el7-x86_64/Modules/
       modulefiles/xjalienfs/1.4.5-22:/cvmfs/alice.cern.ch/el7-
```

```
    x86_64/Modules/modulefiles/JAliEn/1.7.2-1
JAVA_HOME=/cvmfs/alice.cern.ch/el7-x86_64/Packages/JDK/12.0.1-6
APPTAINER_NAME=centos7
LANG=C
JALIEN_PID=133
LOADEDMODULES=BASE/1.0:JDK/12.0.1-6:Toolchain/GCC-v7.3.0:GCC-
    Toolchain/v7.3.0-alice2-30:AliEn-Runtime/v2-19-le-138:
    FreeType/v2.10.1-107:zlib/v1.2.8-112:libpng/v1.6.34-148:
    sqlite/v3.15.0-69:libffi/v3.2.1-59:Python/v3.9.12-31:OpenSSL/
    v1.1.1m-13:Python-modules/1.0-371:libxml2/v2.9.3-84:XRootD/v5
    .5.3-14:xjalienfs/1.4.5-22:JAliEn/1.7.2-1
SYSTEM_LIBPATH=lib64
PYTHONHOME=/cvmfs/alice.cern.ch/el7-x86_64/Packages/Python/v3
    .9.12-31
SINGULARITY_ENVIRONMENT=/.singularity.d/env/91-environment.sh
APPTAINER_CONTAINER=/cvmfs/alice.cern.ch/containers/fs/
    singularity/centos7
APMON_CONFIG=alicebox.farm.particle.cz
SINGULARITY_BIND=/cvmfs,/workdir,/tmp
SHLVL=3
HOME=/workdir
ALIEN_JDL_CPUCORES=1
ALIEN_USER=vfossum
JALIEN_TOKEN_CERT=/workdir/jobtoken5712500697710563999.pem
PYTHONPATH=/cvmfs/alice.cern.ch/el7-x86_64/Packages/xjalienfs
    /1.4.5-22/lib/python/site-packages:/cvmfs/alice.cern.ch/el7-
    x86_64/Packages/XRootD/v5.5.3-14/lib/python/site-packages:/
    cvmfs/alice.cern.ch/el7-x86_64/Packages/Python-modules
    /1.0-371/share/python-modules/lib/python/site-packages:/cvmfs
    /alice.cern.ch/el7-x86_64/Packages/Python/v3.9.12-31/lib/
    python/site-packages
TMP=/workdir/tmp
ALIEN_MASTERJOB_ID=2849464181
CLASSPATH=/cvmfs/alice.cern.ch/el7-x86_64/Packages/JAliEn
    /1.7.2-1/lib/alien-users.jar
PROMPT_COMMAND=PS1="Singularity> "; unset PROMPT_COMMAND
SINGULARITY_CONTAINER=/cvmfs/alice.cern.ch/containers/fs/
    singularity/centos7
JALIEN_WSPORT=42057
ALIEN_SITE=Prague
APPTAINER_BIND=/cvmfs,/workdir,/tmp
LIBXML2_VERSION=v2.9.3-84
BASEDIR=/cvmfs/alice.cern.ch/el7-x86_64/Packages
Done, I'm an environment test
payload-2849464181
```

Appendix B

enviro-get-test.sh output

```
JALIEN_TOKEN_KEY:
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAkqZpy7nSbs0SyFquA5eypp+enb1f00/d4KFSXs6vtyHCI6+L
BTqJ8qQq9BrdY0WY0QgdGkue5J8Xi35anEppkwddikxngkS32ncxaHYY3biyUxVD
rl6RDXtxas8lKtRObZyZj7C/dL+eX+vMw9UJ83O5v8Pz2TLbVbQOYD6JPKXnj0NK
7RL6pSfhC3ASHewVMi86ZZATB+NqpKOoZmI4iiqOzWu658GDXTXwhPlanP5YBw2m
tsygvHf0dB3T6aNUMtnh9X2M6UumPoepGcL0Dj3Mutv90DoZXpCtjXb646GwleQr
s4YzVnQfCxufNelrfaf6a2CwQyS5yU42mMiGOwIDAQABAoIBADkgHAwhCYjh15Q2
RpzdpdjkSsxVbZOKA/sxvD10M9yZiN+PzQ6vW/cp3hWStXSrMrkSeQu6M14JXwmW
ocNJ9VAxyU1LL/L9w8SVM6jmgA82mDhnCyNMcSKYWHjr0iWZ6CconmTry9i1oALr
fqs2PJcaGFxL/5w3BFuhNQxmr9zwiHwr4Rn1R9Bp4Zvr2DLN51ifYr5gb76Rdmr/
RaJtb8xwj9czUQLiNaUUsQjGoO353xlknKMxQ0lKRBdT1VBIG1Jdmc/bVF9H4qoH
iyJ6Amo08xEUQxoay8IPxn/xoibhqVrlykSLk6VNcjvNUH//7dfu3v+pGN6NH619
m7VrxKkCgYEAvEy5DV+gNnQL7P+CLMPVAB05kHg5OAGPiVNzguFLy8EMXPnrn+f9
PonHIlN85JRBzuQLEZHpvlKEy6XhVuiHgTHS9cUJruahwmWUch8U1Wsn5u3Oo8DT
W/noqPTy5pBqH+qU95/CnQm1Lj/svxkngUNV3qpYZM19VFK0VXQrNDcCgYEAx2A3
+sTZH14lq4cGsYgd5lZDb5FP72az78rtxrMsuXrEXNkN4vYwGkZl+EPrsUhLc4jB
UkAiATQnthn/RajhOcY5h/yeD9zQ6zgAaOMu/Pi1mQw+fwyQgcyhHPnnmr4irMBg
WW1smpzq07NU25YuEQRLMy8GyQgHqTmefYGjRB0CgYEAg8dJ5rMao6E5S2ZRitSO
d6L8Sz/oA2sMMfYmK91SN0lmRpddMC6iC6dOppY5HJspJwpvOJ+eLuT/LKptdm4L
jv34fqmjqjSb1jlS1z8rTGTOqw2YhP+Iq6DNE5lBeNxueOGp2VaEU3ScyL5SCDcp
b4xZRxh9/iXVZTLwI7HGHqcCgYEAsTwcp2TkKplT3WZcEeeRxNf8fKptp8DhxhHc
S92trXiwJGWjjZYSbtkT/p3KYL+Gqz7vtAFk+TL29k+n+LHG/cf0DqoHjMxcQC15
V8RVWfZx4Irc564wq+JPeuXA+HN7eZxOSlEW0V8a9XSew9F+RAxQGUD9xCnKPaoF
9UYP8BUCgYBD5ApK4JsbzbSkDTvZQPMDhKTCvUtyvZS4rMFADQB1tJIaz9oedsXg
t7Ig4XOmRUX84Vbw6/LPWat3Bc/jAIZEFGGOnDwkVBqkGFzMFBM3QlDlYR89YwDn
Pe+bhLDjvQnczWKsyVONmMP8sb6D/kCfNyHb2k+y6cJt9Y/Ggd2wVw==
-----END RSA PRIVATE KEY-----

JALIEN_TOKEN_CERT:
-----BEGIN CERTIFICATE-----
MIID5TCCAs2gAwIBAgIQYVoelua+jZn6lprUgDF8wjANBgkqhkiG9w0BAQsFADAx
```

```
MQswCQYDVQQGEwJjaDEPMA0GA1UEChMGQWxpRW4yMREwDwYDVQQDEwhBbGlFbiBD
QTAeFw0yMzA1MjMxMjE2MTBaFw0yMzA1MjQxNDE2MTBaMH0xCzAJBgNVBAYTAmNo
MQ8wDQYDVQQKDAZBbGlFbjIxDTALBgNVBAMMBEpvYnMxEDAOBgNVBAMMB3Zmb3Nz
dW0xEDAOBgNVBAsMB3Zmb3NzdW0xKjAoBgNVBAsMIXF1ZXVlaWQ9Mjg1NjY3MzEy
OS9yZXN1Ym1pc3Npb249MDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEB
AJKmacu50m7NEshargOXsqafnp29X9NP3eChUl7Or7chwiOviwU6ifKkKvQa3WNF
mNEIHRpLnuSfF4t+WpxKaZMHXYpMZ4JEt9p3MWh2GN24slMVQ65ekQ17cWrPJSrU
Tm2cmY+wv3S/nl/rzMPVCfNzub/D89ky21W0DmA+iTyl549DSu0S+qUn4QtwEh3s
FTIvOmWQEwfjaqSjqGZiOIoqjs1ruufBg1018IT5Wpz+WAcNprbMoLx39HQd0+mj
VDLZ4fV9jOlLpj6HqRnC9A49zLrb/dA6GV6QrY12+uOhsJXkK7OGM1Z0HwsbnzXp
a32n+mtgsEMkuclONpjIhjsCAwEAAaOBrDCBqTAfBgNVHSMEGDAWgBQMYbqlSS9N
Z+9PhcYLOfqY+cEq4DAdBgNVHQ4EFgQUAcxRRg1mlZAKmPAbfPViL4cEzGEwHQYD
VR0lBBYwFAYIKwYBBQUHAwIGCCsGAQUFBwMBMAsGA1UdDwQEAwIF4DA7BgNVHREE
NDAygglsb2NhbGhvc3SCFWxvY2FsaG9zdC5sb2NhbGRvbWFpboIJMTI3LjAuMC4x
ggM6OjEwDQYJKoZIhvcNAQELBQADggEBAHZvxKlNS+RQ8mSWcV79yDXWSss9Vj94
uD47sZZe1qntKZnVCpqXy8reNiZl3Wj+cAuoUm1ity5mBVSeMb+2Y70vUik1rUoM
+LFLuK6nbWnz9HAJPFIXzWop/5Ok7z31YrBPglqHEzQOnNYJHoT9wAt/zBkmiLQX
ZZtko4695lHnbn8QcMZ9s9NiMxQuY0YjpWXcpEy04F4Fd8xNqZkdwFwdyXUi05Uy
cNOXSEi2L4gQ9diAT1dlnaQ12x6pxypTJ7Wsf6Naz8xxKLzZ7HG5Xmamforoax4W
0nGac7zQ7OrFHOH56xIjWPBlo8rkxFgKqVXmo3Lte8MDf5Q8K0N1RBI=
-----END CERTIFICATE-----
Done, I'm a environment get test
payload-2856673129
```

Bibliography

[1] en. June 2023. URL: https://home.cern/science/computing/grid-software-middleware-hardware.

[2] *ALICE Grid sites monitoring map*. en. URL: http://alimonitor.cern.ch/map.jsp.

[3] Lokesh B. Bhajantri and Tabassum Mujawar. "A Survey of Cloud Computing Security Challenges, Issues and their Countermeasures." In: *2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*. Dec. 2019, pp. 376–380. DOI: 10.1109/I-SMAC47947.2019.9032545.

[4] Randy Butler et al. "National-scale authentication infrastructure." In: *Computer* 33 (Jan. 2001), pp. 60–66. DOI: 10.1109/2.889094.

[5] CERN. *About — CERN*. en. URL: https://home.web.cern.ch/about (visited on Sept. 7, 2022).

[6] CERN. *ALICE Experiment — CERN ALICE*. en. URL: https://alice.cern/ (visited on Sept. 7, 2022).

[7] CERN. *CERN Certification Authority*. URL: https://ca.cern.ch/ca/Help/?kbid=020000.

[8] CERN. *CERN Certification Authority - How to install a Grid Host certificate*. URL: https://ca.cern.ch/ca/Help/?kbid=024100.

[9] CERN. *CERN Certification Authority - Request a new Grid User certificate*. URL: https://ca.cern.ch/ca/user/Request.aspx?template=EE2User.

[10] CERN. *Register with the ALICE Virtual Organization — JAliEn - ALICE Environment Grid Framework*. URL: https://alien.web.cern.ch/content/register-alice-virtual-organization.

[11] CERN. *The Large Hadron Collider*. en. URL: https://home.web.cern.ch/science/accelerators/large-hadron-collider (visited on Sept. 7, 2022).

[12] CERN. *WLCG*. URL: https://wlcg.web.cern.ch/ (visited on Mar. 3, 2022).

[13] CSRC Content Editor. *Certificate Authority (CA) - Glossary — CSRC*. EN-US. URL: https://csrc.nist.gov/glossary/term/certificate_authority.

[14] Ian Foster et al. "A Security Architecture for Computational Grids." In: *Proceedings of the ACM Conference on Computer and Communications Security* (Feb. 2000). DOI: 10.1145/288090.288111.

[15] A G Grigoras et al. "JAliEn – A new interface between the AliEn jobs and the central services." en. In: *Journal of Physics: Conference Series* 523 (June 2014), p. 012010. ISSN: 1742-6588, 1742-6596. DOI: 10.1088/1742-6596/523/1/012010.

[16] Muhammad Asif Habib and Michael Thomas Krieger. "Security in Grid Computing." In: *Seminar aus Netzwerke und Sicherheit: Communication Infrastructure*. 2008.

[17] *JAliEn / JAliEn*. en. June 2023. URL: https://gitlab.cern.ch/jalien/jalien.

[18] *JAliEn / xjalienfs*. en. June 2023. URL: https://gitlab.cern.ch/jalien/xjalienfs.

[19] LDAP. *LDAP.com*. en-US. URL: https://ldap.com/.

[20] Dwi Liestyowati. "Public Key Cryptography." en. In: *Journal of Physics: Conference Series* 1477.5 (Mar. 2020), p. 052062. ISSN: 1742-6596. DOI: 10.1088/1742-6596/1477/5/052062.

[21] Dominic Lindsay et al. "The evolution of distributed computing systems: from fundamental to new frontiers." en. In: *Computing* 103.8 (Aug. 2021), pp. 1859–1878. ISSN: 1436-5057. DOI: 10.1007/s00607-020-00900-y.

[22] Qusay H Mahmoud. *Middleware for communications*. Vol. 73. Wiley Online Library, 2004, pp. 109–113.

[23] M Martinez Pedreira, C Grigoras, and V Yurchenko. "JAliEn: the new ALICE high-performance and high-scalability Grid framework." In: *EPJ Web Conf.* 214 (2019), p. 03037. DOI: 10.1051/epjconf/201921403037. URL: https://cds.cern.ch/record/2701497.

[24] *NorduGrid Certification Authority*. URL: http://ca.nordugrid.org/.

[25] Miguel Pedreira et al. "The Security model of the ALICE next generation Grid framework." In: *EPJ Web of Conferences* 214 (Jan. 2019), p. 03042. DOI: 10.1051/epjconf/201921403042.

[26] *Public Key Infrastructure (PKI)*. en. URL: https://www.enisa.europa.eu/topics/incident-response/glossary/public-key-infrastructure-pki.

[27] Aaqib Rashid and Amit Chaturvedi. "Cloud computing characteristics and services: a brief review." In: *International Journal of Computer*

Sciences and Engineering 7.2 (2019), pp. 421–426. DOI: https://doi.org/10.26438/ijcse/v7i2.421426.

[28] Even Berge Sandvik. "Site Sonar - A monitoring tool for ALICE's Grid Sites." en. Dec. 2021, p. 92.

[29] Steffen Schreiner. "A Security Architecture for e-Science Grid Computing." In: (2015).

[30] Manjeet Singh. "An Overview of Grid Computing." In: *2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*. 2019, pp. 194–198. DOI: 10.1109/ICCCIS48478.2019.8974490.

[31] Maksim Melnik Storetvedt. "A new grid workflow for data analysis within the ALICE project using containers and modern cloud technologies." eng. Accepted: 2023-04-04T08:40:01Z. Doctoral thesis. Høgskulen på Vestlandet, 2023. ISBN: 9788284610429. URL: https://hvlopen.brage.unit.no/hvlopen-xmlui/handle/11250/3061978.

[32] Maxim Storetvedt. private conversation. 2023.

[33] Andrew S. Tanenbaum and Maarten van Steen. *Distributed Systems*. en-US. 3rd edition. Maarten van Steen, 2017. ISBN: 978-90-815406-2-9.

[34] Barry Wilkinson. *Grid Computing: Techniques and Applications*. en. New York: Chapman and Hall/CRC press, 2009. ISBN: 978-0-429-14558-2. DOI: 10.1201/9781420069549.

[35] Chiara Zampolli. *ALICE data processing for Run 3 and Run 4 at the LHC*. 2020. DOI: 10.48550/ARXIV.2012.04391. URL: https://arxiv.org/abs/2012.04391.