Enhanced Authentication Approach to the GENeSIS System for Social Sciences

Scott Simpson
Computing (Industry)
2009/2010

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student)__________________________
Summary

The aim of the project was to develop a system that enhanced an existing security mechanism of a Social Science project. The project would attempt to improve on the existing username/password implementation by performing an evaluation of existing techniques and utilising one for use within the project.

The project developed the system in an iterative manner, allowing for specific deliverables to be set out, ensuring that the end goal of an iteration was known. The end of the iteration would allow for the developed application to be tested and evaluated before proceeding onwards. The result of the final iteration would be a complete solution that solved all the requirements of the original problem.
# Contents

1 Introduction

1.1 Aim .................................................. 1
1.2 Objectives ............................................ 1
1.3 Requirements ........................................ 2
1.4 Deliverables ......................................... 2
1.5 Schedule ............................................. 2

2 The Problem

2.1 Problem Description ................................. 3
2.2 Computer Security ................................. 4
2.3 Grid Research ....................................... 4
   2.3.1 Grid Computing ................................. 5
   2.3.2 Grid Users ....................................... 5
   2.3.3 Grid Challenges ................................. 5
   2.3.4 Grid Middleware ................................. 6
   2.3.5 Grid Security Infrastructure ................. 7
   2.3.6 Storage Resource Broker .................... 7
2.4 Science, Social Science and Grids ............... 8
   2.4.1 e-Science ....................................... 8
   2.4.2 e-Social Science ............................... 9
2.5 Current GENeSIS System ......................... 9
   2.5.1 GENeSIS architecture ......................... 10
   2.5.2 System Model .................................. 11
2.5.3 Interface ......................................................... 12
2.5.4 SRB Access .................................................. 12
2.5.5 Tools used within the Web application ..................... 13
2.5.6 Summary ..................................................... 15
2.6 Security Methods ............................................. 15
  2.6.1 Password/Username ........................................ 15
  2.6.2 Public Key Infrastructure ................................ 16
  2.6.3 Kerberos .................................................. 18
  2.6.4 Shibboleth ............................................... 19
  2.6.5 Security Comparison .................................... 20
2.7 Existing Implementations .................................... 21
  2.7.1 MyProxy .................................................. 22
  2.7.2 VOMS ...................................................... 23
  2.7.3 ShibGrid .................................................. 24
  2.7.4 SHEBANGS ............................................... 24
  2.7.5 SARoNGS ................................................ 25
2.8 Overview of Implementations ................................ 26
2.9 Methodology .................................................. 26
3 Implementation Choice & Methodology ....................... 27
  3.1 Comparison of Technologies ................................ 27
    3.1.1 Authentication .......................................... 27
    3.1.2 Authorisation .......................................... 28
    3.1.3 Single Sign-On ........................................... 28
    3.1.4 Advantages .............................................. 28
    3.1.5 Required ................................................ 29
    3.1.6 Disadvantages .......................................... 29
    3.1.7 MyProxy Server ......................................... 29
    3.1.8 Existing Authentication ................................ 30
    3.1.9 CTS ...................................................... 30
    3.1.10 Authorisation ......................................... 30
3.1.11 Certificates ................................. 30
3.1.12 How the implementations solve the initial problem .... 31
3.1.13 Choice ................................. 31

3.2 SARoNGS Process ................................. 31
3.2.1 Certificate-less access .................. 31
3.2.2 Obtaining an NGS Account using SARoNGS ............. 32
3.2.3 Retrieving a SARoNGS Credential .......... 32

3.3 Methodology ................................. 33
3.3.1 Iteration One ................................. 33
3.3.2 Retrieving a SARoNGS Credential .......... 33
3.3.3 Connecting to and Retrieving a File from the SRB .... 34

3.4 Iteration Two ................................. 34
3.4.1 Portlet Creation ................................. 35
3.4.2 Create Login Portlet ................................. 35
3.4.3 Create Get File Portlet .................. 35
3.4.4 Portlet Process ................................. 36
3.4.5 Time to Live ................................. 36

3.5 Iteration Three ................................. 37
3.5.1 Create a Web Service .................. 37
3.5.2 User Authorisation ................................. 38
3.5.3 Integrate with GENeSIS .......... 38

4 Iterative Implementation & Evaluation ................................. 39
4.1 Iteration One ................................. 39
4.1.1 Retrieving SARoNGS Credential .......... 39
4.1.2 Connecting to and Retrieving a File from the SRB .... 40
4.1.3 Problems ................................. 41
4.1.4 End of Iteration Testing .................. 42
4.1.5 Evaluation of the First Iteration .... 42

4.2 Iteration Two ................................. 43
4.2.1 Create JSP Login Portlet .......... 43

iv
7 Appendix B - Test Results
  7.1 End of First Iteration Testing ................................................. 68
  7.2 End of Second Iteration Testing .............................................. 70
  7.3 End of Third Iteration Testing ................................................ 72

8 Appendix C - Diagrams ............................................................. 74
List of Figures

2.1 GENeSIS Architecture. Dashed box contains items within scope of the project 10
2.2 GENeSIS System Model 11
2.3 Table of Comparison for Security Methods 21
3.1 Portlet Process; Retrieving a File 36
4.1 Code that retrieves a credential from a MyProxy server 40
4.2 Code to retrieve a file from the SRB using credential authentication 41
4.3 Web Service Code 47
4.4 Example VOMS entry 49
6.1 Revised Project Schedule 66
8.1 GENeSIS User Interface Interaction 74
8.2 SARoNGS Process. Where Are You From? (WAYF) 75
8.3 SARoNGS Process. Local Authentication 75
8.4 SARoNGS Process. SARoNGS Account Registration 76
8.5 SARoNGS Process. Which VO Are You From? (WVOAYF) 76
8.6 SARoNGS Process. Credential Upload 77
8.7 Application. SARoNGS Login Process 77
8.8 Web Service. Hosting test page 77
8.9 Web Service. Client code to call web service 77
Chapter 1

Introduction

1.1 Aim

To evaluate existing methods and techniques for Grid-based authentication and implement an advanced authentication service for an e-Social Science application to enhance data security.

1.2 Objectives

- Investigate and examine the current system (i.e. GENeSIS) and review literature outlining existing authentication methods. Have a good understanding of the limitations and problems associated with the authentication operations used by GENeSIS currently

- Demonstrate the stages of the methodology in the process of designing an enhanced authentication solution for GENeSIS

- Deliver a novel service for solving the authentication problem, including its design and prototype implementation

- Compare the security of the new solution against the existing approach through analytic evaluation and experimental assessments, e.g. simulation of potential attacks
1.3 Requirements

The minimum requirements are:

- Existing authentication methods and techniques will be evaluated
- The system will successfully authenticate a valid user
- The system will prevent an invalid user from accessing secure data
- The system will be integrated into the existing e-Social Science application

Further Enhancements are

- The system will be created as a service, allowing for use by other applications
- The system will enhance security by implementing a 'time-to-live' on any authentication method(s) used
- The system will implement a method that will allow for the authorisation of a user, on top of the authentication

1.4 Deliverables

- After each iteration of the software, a prototype shall be delivered
- Each prototype will be tested and evaluated against its requirements

1.5 Schedule

See Appendix A
Chapter 2

The Problem

2.1 Problem Description

Grid computing [24] helps enhance available computational power by connecting multiple, geographically dispersed machines to coordinate resources that are not subject to centralised control. The resultant Grid gives the user access to vast amounts of data and processing speeds to match supercomputers. However, it is necessary that the user does not abuse these features. It is therefore necessary to implement security mechanisms into the Grid.

The value pair of username/password is the historical method of ensuring user authentication and without question still plays a relevant role in today’s computing security; but this method is compromised if a password is discovered by a third-party. In a day-to-day environment, a hacked email account can be rectified by a quick change of password. In a more data sensitive environment, losing a password can have much more serious ramifications.

GENeSIS is an ongoing e-Social Science project which aims to ”develop models of social systems...to build environments and cities” [19]. The modelling involved requires a large amount of sensitive data, stored on the National Grid Service (NGS). The NGS ”aims to enable coherent electronic access for UK researchers to all computational and data based resources and facilities required to carry out their research, independent of resource or researcher location” [33]. The data stored for use by GENeSIS includes Census information and medical records, all of which are covered under the Data Protection Act, and therefore any leak of this data can have huge implications for those involved.

The GENeSIS system currently uses password and username pairs to access and retrieve the sen-
sitive data. The use of passwords, which are inherently long-lived, means that any malicious attempt to steal password information could result in sensitive data being accessed by an unauthorised body. Therefore, this project will look into enhancing the existing GENeSIS security, by analysing and comparing existing security technologies and introducing a new method of user authentication to help ensure the integrity and security of the data.

To help inform the implementation decisions to be made, the following section will look at related technologies and issues surrounding Grid computing, e-Science and computer security.

2.2 Computer Security

Security, when related to computing, is defined as "the ability of a system to protect information and system resources with respect to confidentiality and integrity" [37]. As well as confidentiality and integrity, authentication can also be included as a major area of computer security. Confidentiality is concerned with ensuring that information is not accessed by an unauthorised person. Integrity is in place to ensure data is not modified by an unauthorised person and authentication attempts to ensure a person is who they say they are. All three elements are vital to keeping data secure within an e-Social Science environment.

Of the three elements, authentication should be the first security step, as users who fall at this stage are more likely to be unauthorised. It is important to state that a user can be authenticated successfully, but not be authorised to perform particular tasks. For example, a user can login (authenticate) successfully into a system, but there are certain tasks they are not allowed to carry out (e.g. administration tasks). Within a Grid environment, an authenticated user may have limited roles they can perform. Linking back to confidentiality, a user may not be authorised to access a specific, sensitive, dataset. On the other hand, they may have authority to access the data, but to ensure integrity, they do not have rights to modify the data.

2.3 Grid Research

Before detailing the current GENeSIS system and its authentication mechanism, this section will discuss fundamentals of Grid Computing and computing security.
2.3.1 Grid Computing

Grid computing [5] is the term used to describe a large number of co-ordinated computing resources that are geographically diverse. The difference between a standard network and a Grid is that the resources are not under central control. Each resource is controlled by the owning organisation; therefore any third-party requiring access will be required to authenticate itself. To achieve this authentication and to allow the user to access the diverse, heterogeneous resources, the Grid must encapsulate standard, open, general-purpose protocols. The standards [24] allow for resource sharing between entities, irrespective of heterogeneity. To ensure standardisation when developing applications to interact and interface with Grid resources, the Globus Toolkit has become the de-facto. This also fills another criterion of being 'open', as this toolkit is open-source allowing for continuous, diverse development.

2.3.2 Grid Users

A Virtual Organisation is a "Temporary or permanent coalition of a geographically dispersed individuals, groups, organisational units or entire organisations that pool resources, capabilities, and information to achieve common objectives" [12]. An individual Grid user will be part of one or more VOs. A VO can be dynamic, with members constantly joining and leaving.

The access of Grid resources by users can be done through numerous methods. A user can use command line tools like Scommands or programmatically access resources. One of the main methods however, is using a Grid Portal [16]. This is popular due to its non-technical interface allowing non-technical users access to the power of the Grid. The portal usually implements single sign-on, allowing the user to login once and then have transparent access to numerous resources across multiple VOs, without the need to re-authenticate.

2.3.3 Grid Challenges

Grid Computing inherits numerous challenges [5], which are emergent from its architecture and goals. The challenges can be split up into four areas; trust, performance, programming and resource coordination.

Trust is required to allow machines to have the confidence to share and exchange resources. Grid resource sharing relies upon one machine, the client, using resources (data, computing power or software) that are hosted on single or multiple machines. The administration that controls the hosting resources
must be confident that the client can be trusted to use the resources in a fair manner. As, for example, data being accessed could be highly sensitive and be controlled under the Data Protection Act. To ensure a level of trust, a trusted third-party is utilised as a middleman between the two entities, and it, as an entity itself, has to be trusted.

Performance is another challenge for the Grid community. If a client is using a distributed resource, they need to have reassurance that the resource is stable and will provide the performance necessary. If there is a problem however, the client must be either made aware directly or the server should have a mechanism to ensure any problem fails gracefully.

Thirdly, the programming of applications is necessary to create distributed, Grid based, system. To facilitate this, Grid-specific Software Development Kits (SDKs) and Application Programming Interfaces (APIs) have been developed. Solutions include JavaCoG [28] and Jargon API [3], which provide libraries and API functions to facilitate the development of Grid enabled applications.

The last problem to be explained here is that of resource co-ordination. Within a Grid computing environment, resources (data, computing power and software) can be distributed across multiple, heterogeneous machines, which in turn could be under the administration of numerous entities. Therefore, there have to be tools implemented that can co-ordinate the sharing of resources. For example, a user wishes to analyse three sets of data together; instead of the user going to three different locations to retrieve the data, it would be more useful if they had a single entry point, and that the Grid itself could retrieve the relevant data sets.

2.3.4 Grid Middleware

As explained previously, the architecture of the Grid creates challenges alluding to the distributed nature of resources. Grid middleware helps solve this problem by bringing together the client wishing to access the resources and the institutions making their resources available. Over the years, the Globus Toolkit has emerged as the de facto standard for Grid middleware [25]. There are four main components that constitute the toolkit and help provide solutions to the challenges of Grid computing. Firstly, information services allow for access to the information and data stored within the Grid. Globus also provide the Grid Security Infrastructure, which deals with authentication and authorisation within the Grid. In addition, there is a component to deal with the actual transfer of data and tools for administrators to manage resources.
2.3.5 Grid Security Infrastructure

The Grid Security Infrastructure, GSI, refers to the "portion of the Globus Toolkit that provides the fundamental security services needed to support Grids" [42]. The main purpose of the GSI is to allow members of a VO to share data securely [13]. The ‘securely’ feature is split up into authentication and authorisation. Authentication performs a role to ensure the user logging in is who they say they are. The most common example is the username-password pair. It is presumed that the username is commonly known (e.g. an email) but the password is secret to the owner of the username. Therefore a system can authenticate the user by checking that the username and password pair matches. Authorisation is the stage after authenticating. This process checks to see if the logged in user is authorised to use a particular resource, or set of resources. The GSI aims to provide the infrastructure to meet these security goals.

For the user, the GSI provides a point of access to resources via a single-sign on (SSO). This ensures that if a user needs to access multiple resources across multiple VOs they need only to login once and the GSI provides features to handle passing the login information between these VOs. [13]

The main feature of the GSI is its ability to handle credentials, which encapsulate information to distinguish a user’s identity. They include X.509 certificates, which are explained later. The movement of these credentials is performed over Transport Level Security (TLS). This provides both protection and privacy to the integrity of the credentials by encrypting their transport. GSI also supports using Certificate Authorities (CAs) which can act as trusted third parties. [13]

2.3.6 Storage Resource Broker

The diverse, dispersed nature of the Grid means that data and resources are spread globally. The Storage Resource Broker (SRB) [4], developed by the San Diego Supercomputer Center, provides a way of accessing this shared data. The overall goal is to make available a uniform interface into the diverse and distributed data storage resources of the Grid. This goal has been realised with the interface being implemented as middleware. This allows the multiple data locations to seem as one large resource to the client, hiding any unnecessary complication.

A standard way of accessing data is to go directly to the location of the data, e.g. pointing a browser towards a File Transport Protocol (FTP) Uniform Resource Locator (URI) and retrieving the data. With SRB, access is available to multiple data source, like archived resources, compressed files, databases and HTTP. All these data sources are viewed as just one data store to the user.
The architecture of SRB servers is that of a federated server, with each server responsible for a set of resources [7]. This feature allows for transparency to the user of the location of the data, as the user can connect to one SRB server, then 'behind-the-scenes’ the SRB can look up the location of the data and if on other SRB servers, point to and retrieve the data from those servers and return all the data back to the client. The server arrangement also increases reliability and availability, with the replication of data and being able to balance incoming SRB requests across numerous servers.

There are however limitations with SRB [38]. Firstly, it is not completely open source. Without opening up the source code to the development community, SRB has failed to allow different people with different ideas to enhance and upgrade the existing software. Secondly, SRB is not based on any agreed upon standards. Many technologies used within Grid systems have a standards basis (e.g. X.509 certificates, SAML assertions etc.).

2.4 Science, Social Science and Grids

This section will look at the role played by Grids in helping to support the research work performed within the fields of science and social science.

2.4.1 e-Science

e-Science is described as "Large scale science...carried out through distributed global collaborations enabled by the Internet” [6]. Large scale science relates to the necessity to collect and analyse vast amounts of data using a similarly large amount of computing resources; processing power and memory storage. These requirements fit with the solutions provided by the Grid architecture.

e-Science can be more specifically explained by the methods involved [2]. The aim is to use computational methods (i.e. Grid computing) to perform three main tasks. The first is based around generating and analysing data used within research. This data can be garnered from experiments and computational simulations. The next task is to utilise the computational power to implement and expand scientific models and simulations. Simulations with huge amounts of data can be parallelised to ensure run-time is timely and suitable for the user’s needs. The final task is to use the distributed nature of a Grid environment to create dynamic Virtual Organisations. The VOs can then be used to share and collaborate towards a common goal.
2.4.2 e-Social Science

Social Science is the science related to the human aspect of geography. It covers a wide range of areas including demographics, politics and medicine. Within social science [16], it is common place for related data to be geographically dispersed and therefore this brings its own challenges, which are aligned with the tasks involved with e-Science. Firstly, with such diverse and dispersed data, there is a need to have tools available to gather the data and manage it, so that it is accessible by those who need it. Many different people, institutions and organisations will then need access to the data, so the infrastructure must be in place to suit a many-to-many architecture, where many clients will be attempting to access data and resources on many different machines. Furthermore, the client should be able to access their relevant data from anywhere geographically and therefore not be tied to accessing data from one machine or from within one VO. Additionally, this brings with it the need for access control. The data held within social science can be sensitive and subject to the Data Protection Act 1999 and therefore controlling who and what has access to data is extremely important.

e-Social Science can simply be described as e-Science for the social sciences and aims to solve the social science problems outlined previously by employing Grid Computing. A more thorough definition of e-Social Science is given as the "Conduct of social science through global collaborations, drawing on computational power of Grid technology" [45]. The infrastructure provided by the Grid enables social science to "collect, process, integrate, share and disseminate data" [39]. However, the infrastructure itself does not solve security issues and therefore extra measures must be implemented to ensure the integrity and privacy of sensitive data.

As mentioned previously, both authentication and authorisation together enable the security of resources within a Grid environment. For e-Social Science, and this project in particular, authentication is the most important aspect; to ensure that the user accessing the data is who they say they are and that the data being exchanged is secure. However, this is not to say that authorisation is not required, as with this element, access to the data will be more secure.

2.5 Current GENeSIS System

This section will take a detailed look at the current GENeSIS system, outlining the areas of interest for this project.
2.5.1 GENeSIS architecture

Figure 2.1: GENeSIS Architecture. Dashed box contains items within scope of the project

- **Users.** The users of GENeSIS will mainly be non-technical users. These include Social Scientists and Governmental Planners who will use the services to build future models of social systems. The non-technical nature of these users brings with it considerations for the implementation of the authentication service. The implementation must be designed at a certain level of abstraction that makes the authentication procedure as simple as possible. Also, the authentication procedure must be as transparent as possible to the user, hiding any unnecessary technical complexity.

- **Virtual Organisations and Authorisation.** The current GENeSIS system does not utilise Virtual Organisations as a way of creating a temporary group of individuals and institutions. Also, GENeSIS currently does not require authorisation of users to access resources. It is assumed currently that if a user is successfully authenticated then they have rights to use any of the GENeSIS services. However, as GENeSIS goes forward as a project, this requires altering.

- The utilisation of Virtual Organisations and user authorisation within GENeSIS will allow for certain access rules to be applied. This will allow for authorisation of users to occur, enhancing data and resource security.
2.5.2 System Model

The architecture provides a low-level view of the components of GENeSIS. To view a higher level of abstraction, a system model is outlined below. The 'Users' entity is a direct copy from the architecture diagram. The 'Interface' incorporates the End User Interfaces listed in the architecture. The 'Web Service' entity is used to provide an example of a user’s interaction with one of the GENeSIS web services, either a main or utility service. The 'SRB' data store is the Storage Resource Broker element of the architecture diagram. Local Store is ignored as it is assumed if the user has access to data on a local storage that they must have had rights to retrieve that data initially.

![GENeSIS System Model Diagram](image)

Figure 2.2: GENeSIS System Model

The Current Process:

a) The user accesses the GENeSIS login screen through a portal, entering their details and what service and data they require
b) The user details (including the password) are serialised and passed to the relevant web service(s)
c) The web service deserialises the user details and passes them to the SRB to gain connection. The web service then requests the relevant data from the SRB
d) The SRB retrieves the data and passes it back to the web service
e) The web service performs any operations required on the data and passes the results back to the interface to be rendered
f) The data is rendered as a map for example, and shown to the user
2.5.3 Interface

There are two main interfaces for a standard, non-technical user of GENeSIS. The first is a lightweight desktop tool, used by the user to set their user configuration values and retrieve data files (.genesis files) from the SRB. The values set include the necessary details for the user to connect to the SRB, including SRB account details. After retrieving the .genesis file, the second interface, a web application, allows for the upload of the retrieved file. Web services can then be called to perform multiple operations on the data. (See figure 8.1 in Appendix C)

There are emergent properties from these operations which negatively affect the security and usability of the system. Firstly, the above process can be seen as cumbersome, as the user needs to use two tools to perform a particular action. Secondly, when the user retrieves the file from the SRB, it contains both sensitive data and information relating to the users account details including password information. The password being shown in clear text is not an issue to the user in terms of it being their password and so they know it anyway, however the problem exists that if a third-party gains access to the user’s machine then they will be able to read the user’s password and then have full access to the SRB and therefore any data stored on there. With regards to the data, it is serialised with a unique ID and so only readable if this ID is known. However, if a third-party somehow found the ID they could deserialise the data and have access to extremely sensitive data.

2.5.4 SRB Access

The SRB used by GENeSIS is that which is supplied by the National Grid Service. To access the SRB, a user must have an SRB account, which comes from gaining an NGS account. To obtain an NGS account, a user must first request a UK e-Science certificate from the NGS. An example process is outlined below:

a) User submits a request over the internet (using a compatible browser) for a UK e-Science certificate

- The standard certificate URL is https://ca.grid-support.ac.uk/cgi-bin/pub/pki?cmd=getStaticPage&name=index
- The request contains the user’s name, a PIN and the Registration Authority (RA) that they wish to be authenticated at
• As the request is submitted, a public key is generated for the user and sent to the UK e-Science Certificate Authority (CA)

b) The user must then visit, in person, their chosen RA to prove their identity

• This involves the user showing a form of identification and then entering their PIN that they entered when submitting the request

c) Once the RA has confirmed to the CA the user’s identity, the CA generates the user’s certificate
d) The user can then use the browser they made the request from to directly import the certificate
e) Again using the same browser, the user can request an NGS and SRB account

• The URL to request an account is https://uas.ngs.ac.uk/apply.php

• The user must enter basic contact information, as well as their reasoning for needing an NGS account

f) The NGS then performs peer-review on the request before approving it
g) Once approved, the user is sent an email, containing an SRB username and password

• This password must be changed within five days

h) The SRB username and password can be used to access the NGS SRB, to download and upload data files

2.5.5 Tools used within the Web application

The main user interface of GENeSIS is based around a web application and is developed using two main technologies. Firstly, portlets are used as containers for each of the services that are available to the user. The portlets themselves have to be deployed within a portal, which in the case of GENeSIS is implemented using Liferay.

• Portlets. There are two flavours of portlet, JSR 168 and JSR 286. The difference between them is not within the scope of this project. The portlets used within GENeSIS are JSR 168. Portlets are described as "Web-based components managed by portlet containers that supply dynamic content" [36]. In modern web environments, portlets are similar to the ubiquitous widget, seen on the more advanced homepages like iGoogle. They allow for the user to select what content they require by adding the required portlet to the portal. The data within the portlet can then be
either manually or dynamically updated. GENeSIS utilises portlets by attaching a service to each portlet. For example, there is a portlet that deals with creating charts from data sets.

- **Portal.** To allow the portlets to be accessible and manipulated by the users, a portlet container is required. GENeSIS uses Liferay [29] to perform this task. Liferay provides an interface that allows a user to add applications (portlets) into their work area. The existing use of Liferay and portlets provides a user interface for utilising services deployed by GENeSIS and therefore could be utilised when creating enhanced security.

- **Java and JSP.** Java and JSP are combined to create the portlets used within GENeSIS. The Java code is used to deal with the result of a particular action, i.e., what should happen when a button is pressed. The JSP’s role is to provide the dynamic HTML code used to display portlet content.

- **Jargon API.** To interact with the SRB, Jargon API [3] is used. The API, written in Java, provides methods to allow for numerous interactions with Grid-stored data, including data retrieval. The classes of interest within the API with regards to GENeSIS are contained within the edu.sdsc.grid.io.srb package. This package contains the classes and methods that enable the creation of an SRB account object, which can be used to create an instance of an SRB File System. From here, files held on the NGS SRB can be retrieved by the user. Irrespective of the implementation developed, interaction with the SRB will be necessary, therefore utilising the Jargon API will be necessary.

- **Web Services.** The web services within GENeSIS perform tasks such as analysing data and displaying results in charts and geographical maps. The web services are developed using Java and deployed to a server within the Leeds domain. The web services are then contacted via the client application to return results to the user. Further details on the web services and deployment methodology are outside the scope of the project as they do not affect the implementation of an enhanced security system.

In summary, Jargon API provides suitable Java methods for interacting with the SRB and retrieving files stored on the Grid and therefore can be utilised in the security implementation. Similarly, the presentation layer, currently handled by Liferay and its contained portlets, can be utilised for the users’ interaction with the implemented security service.
2.5.6 Summary

The system architecture and model have been outlined, with the major components discussed in relation to the problem. The issues arisen fall into two categories; functionality and usability. Usability relates to how suitable a service is to its users and in this project, the authentication method must be suitable for a non-technical user. Therefore, the process must be simple to follow and transparent, hiding any technical implementations. The current process is also cumbersome and so an implemented service could improve upon this, by streamlining the process.

Functionality relates to the jobs the implementation performs to enhance system security. The use of username/password as an authentication method is unsuitable and passing this value around in human readable files is clearly not secure. Therefore, any implementation will have to provide a more secure authentication service. To enhance security, authorisation must also be considered within the implementation, to enhance data security and correct availability. Finally, the GENeSIS system already has a clear architecture and structure and so it is reasonable that any implementation reduces, as much as possible, its impact on the current state of the system. To better inform the choice of implementation, the next section will look into existing security methods.

2.6 Security Methods

This section looks at existing security techniques used to implement authentication of a user and, in some cases, authorisation of that user as well.

2.6.1 Password/Username

As explained in the problem description, the current authentication system used is the value pair of username and password. This ubiquitous method of authentication is a simple, efficient method of logging a user into a system.

Password protection is easy to implement and requires no extra software or implementation. Standard login controls are common place and software development tools, like Visual Studio, come with login tools as a standard control of their Development Kits. The only hardware required is somewhere to store username and password pairs (or the storage of just the username, encrypted by the password). This information can be stored alongside customer information in a standard database. Usernames and passwords can be changed at the user’s discretion and the pervasive nature means that users know how
Passwords however have obvious weaknesses [34]. The source of the weakness is the user that owns the password. The security depends entirely on the user keeping their password safe, secure and hidden. If an untrusted third party gains access to the password, the application/data that the password is protecting is then vulnerable. To alleviate this issue, the user can follow good practice. Ensuring that they change passwords regularly and not repeating passwords can help, but again this requires an effort from the user’s behalf that can seem an unnecessary overhead. Also, there is an increasing trend of websites having a password security strength rating, helping the user choose a strong password. These strong-passwords [30] contain numbers instead of letters, a mixture of upper and lower case and special characters. This brings an issue with regards to remembering the password, with a greater likelihood of users writing down strong-passwords and therefore negating the security advantages that these give.

The project currently uses username/password pair as its only means of authentication. The value pair is sent as part of a request to the ’server’. The server attempts to authenticate the user, and if successful returns the requested resource back to the user. This issue arises in the form of the returned resource. The username and password are stored as clear, human-readable text as part of the file/resource returned. This security flaw could lead to a third party gaining access to the password and username, allowing them to access the server data.

2.6.2 Public Key Infrastructure

The Public Key Infrastructure allows for the use of digital certificates instead of passwords as a way of authenticating users

Public Key Cryptography [44] is a method used to solve security issues emanating from communications within heterogeneous environments. The issues revolve around confidentiality, integrity, authentication and non-repudiation. Confidentiality is ensuring that data being sent from one entity to another remains secret and private for the duration of the journey and that the intended recipient is the only body able to read the incoming data. To ensure the integrity of the data during communication, Public Key Cryptography must support methods that ensure the data cannot be tampered with or modified. Additionally, the entities involved must be verified by means of authentication. An impartial body must be present to ensure the entities are who they say they are to avoid data being obtained by a false identity. Finally, Public Key Cryptography must support non-repudiation. This ensures that one entity cannot deny that a transaction was performed or that data was sent. The Public Key Infrastructure utilises
Public Key Cryptography to satisfy the above requirements.

The Public Key Infrastructure (PKI) is built upon a framework containing numerous components. These components combine to support the use of Public Key Cryptography, utilising public and private keys and certificates. The distribution of certificates is handled by a Certificate Authority (CA) and a Registration Authority (RA). Combined, they allow for certificates and keys to be created, distributed and managed. The combination of the certificates and keys constitutes a Digital Certificates, which can then be used as a tool to confirm the identity of entities.

Certificate Authorities are third-parties that act as a body of trust for end-entities communicating with each other. These end-entities individually establish trust with the CA. To perform this, the end-entities both make (electronic) request to the CA. The CA must then authenticate the different entities and determine any attributes required for the certificate. From this, the CA can issue each of end-entities with a digital certificate. The certificate is signed with the private key of the CA, which acts as a validation ‘stamp’, verifying the entities identity. The two entities are now able to trust each other as they have a common trust with the CA. The authentication of an end-entity can be designated to a Registration Authority (RA). They act to verify the identity of an entity. The entity, usually a human, will be asked to visit their local RA and provide proof of ID. Due-diligence is then carried out to determine the entities reasoning for requiring a digital certificate.

The certificates involved are constructed of elements to define an entity and provide other useful information. The distinguishing name (DN) of the entity is contained within the certificate, an example of which is:

\[ C=UK, O=eScience, OU=Leeds, L=ISS, CN=\text{scott simpson} \]

The expiration period is also present so that the server using the certificate to authenticate a user can first check to see if the certificate is itself valid. To ensure a standard for the certificate and how it handles the information contained within, X.509 certificates [41] are used.

An example process of obtaining a digital certificate is the requesting of a UK e-Science certificate via the National Grid Service (NGS). The end-entity submits a request electronically using a web form. A public key is generated and sent, along with the entered information, to the UK e-Science CA. The end-entity must then go to their local RA (based in Leeds University for example) and prove their identity. At this stage, the end-entity must also type in their PIN code that was entered in the original form, to act as a method of verifying the person proving their identity is the person who placed the request. Once the RA process is complete, they can inform the CA that the identity has been proven.
The CA then provides the end-entity with their certificate, containing the distinguishing name, any other attributes and the public key signed by the UK e-Science CA with their private key. The certificate can then be used to access numerous restricted resources, without the need for usernames and passwords.

A limitation of the Public Key Infrastructure is the lack of transparency between users and certificates. Handling certificates can be unwieldy, including backing-up and creating proxies. Also, certificates are just another piece of the security puzzle and another 'at-risk' element, which if it is not looked after correctly could cause a security issues; it is therefore ideal that the user is as unaware as possible of the use of certificates.

Schneier outlines ten risks of the Public Key Infrastructure [8]. The majority of these risks revolve around the huge importance the Certificate Authority plays in the infrastructure. Firstly, the major assumption about the CA is that it is a trusted body, and therefore suitable to provide certificates to users. However, if an untrusted body could gain the trust of clients, then clients could be susceptible to supplying personal details or retrieving an untrusted certificate. Also, the CAs private key plays a major role in Public Key Cryptography, as it is used to sign the public-key of the client and so the CA must ensure extremely tight security of their key.

2.6.3 Kerberos

Kerberos is a "network authentication system" [18]. Its main aim is to provide an implementation to verify the identity of entities using an unsecured/unprotected network and provides another alternative to username/passwords by using tickets to authenticate users to use a service/resource. Kerberos is a Trusted Third-Party authentication service, using the standard public/private key cryptography to encrypt details.

The Kerberos implementation is outlined below:

a) The client sends a request to the authentication server (AS) to gain a ticket for a given server [18]
b) The AS responds with the credentials encrypted with the client’s private key [18]. These credentials are referred to as a Ticket Granting Ticket, which contain a ticket for the server and a temporary encryption key (session key)
c) The client transmits the TGT to the TGS (Ticket Granting Server) [32], containing the clients identity and session key (from step 2), all encrypted in the server’s key
d) The TGS creates an encrypted key with a timestamp, granting the client a Ticket Granting Service
e) The client can then use the Ticket Granting Service to authenticate itself with the service [31]. The session key involved in this process is used to authenticate the client and can be used to encrypt further communication between client and server for as long as the key is valid. The Authentication Server outlined above has a main duty to maintain a database of entities. This database contains a list of entity names along with their secret keys.

There are several limitations with Kerberos. Firstly, authorisation is needed to ensure the user has specific privileges to be able to use the service/resource, and Kerberos does not provide this. In addition, compared to a standard username and password method, extra hardware is required. For example, a Key Distribution Centre is needed to "supply tickets and temporary session keys" [18]. Authentication and Ticket-Granting servers also need to be in place. Another problem is that the tickets granted have a long time-to-live, and so if untrustworthy entity gains access to the ticket, they would have a method of authenticating themselves for a long period of time. Finally, Kerberos is built "for use with single-user client systems" [43] and in today’s world, Grid and Cloud computing have client systems with multiple users and so Kerberos is not suitable to be used 'out-of-the-box' with a distributed computing technology and therefore is usually combined with other methods.

2.6.4 Shibboleth

Shibboleth utilises username and password to perform a single sign-on action for authentication. As well as single sign-on, Shibboleth has an advantage over standard username/password authentication by allowing attributes to be passed to resources, allowing for authorisation to occur as well. This extra level of authorisation is an advantage in the e-Social Science environment, putting an extra layer of security on sensitive resources and data.

Shibboleth is a system built for "secure access to resources among multiple organizations" [22]. The architecture allows for single sign-on to be used for access within and between organisations [27], using the OASIS Security Assertion Markup Language (SAML). The Shibboleth software is split into two distinct packages. The Identity Provider (IdP) and the Service Provider (SP). The IdP is operated by an organisation containing the users who are wishing to access the services or resources. The SP, as the acronym suggests, is run by the service provider.

The following outlines the procedure of single sign-on using the Shibboleth architecture [22]
a) The user navigates to the site of the resource/service they wish to use
b) The SP redirects the user to the Where are you from? (WAYF) page. This is a separate webpage, listing all the organisations that can access the resource
c) The user selects their organisation and is redirected to the webpage running the IdP software and can now login as normal using their home institutions authentication system
d) The IdP software directs the user to the originally requested resource site, with a message containing a SAML 'Assertion' proving that the user is valid
e) The SP software on the resource site validates the assertion and then requests any additional information about the user (by making a request to the IdP)
f) The SP receives the additional attributes and passes them to the resource
g) The resource deciphers the users access policy (authorisation) and can then grant the user the appropriate access

Benefits
There are numerous benefits to using Shibboleth for authentication [27]. Firstly, the single-sign on it provides ensures that users do not need multiple login details for different resources. Shibboleth is lightweight in terms of its data requirements for authentication. Shibboleth only needs the data necessary for authentication, therefore organisation only need to send the minimum data in a just-in-time way.

Limitations
Although Shibboleth is created for cross-organisation security, it is not built specifically for Grid computing. The dynamic nature of the Grid and more specially its users ensures that the Virtual Organisations will constantly change, with different users and different VOs themselves. Therefore, items like attributes within the assertion will be constantly change, as will individual user’s rights. Therefore, enhancements must be made to make it ’Grid-ready’.

2.6.5 Security Comparison
The table below provides an overview of the major security techniques that have been outlined above. Specific categories have been selected to outline the major security features for an e-Social Science system. (Modified from [1]). The choice of comparing authentication and authorisation are obvious choices, considering their previously outlined implications with regards to user security. Single sign-on
has also been chosen as a category as it is a key factor with regards to security and usability. Utilising single sign-on requires the use of only one set of login details to access numerous Grid resources, therefore reducing the need for users to remember, or more likely write-down, numerous login details. Also, the transparency that a single sign-on brings will increase usability, from the simplicity of fewer buttons clicks and hidden nature of the authentication.

<table>
<thead>
<tr>
<th>Category</th>
<th>Authentication</th>
<th>Authorisation</th>
<th>Single Sign-on</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username/Password</td>
<td>Yes – through the value pair</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PKI</td>
<td>Yes – through certificates and keys</td>
<td>No – not directly, but changes to certificates can be made to enable authorisation</td>
<td>No – not directly, but can be solved using proxy certificates</td>
</tr>
<tr>
<td>Kerberos</td>
<td>Yes – through the use of tickets</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Shibboleth</td>
<td>No – requires the use of an existing authentication mechanism</td>
<td>Yes – using attributes passed in the SAML assertion</td>
<td>Yes – through federated identity</td>
</tr>
</tbody>
</table>

Figure 2.3: Table of Comparison for Security Methods

As the table shows, there is no one specific mechanism that solves all the requirements of Grid security for an e-Social Science system. Both username/password and Kerberos only provide authentication, with no direct support for authorisation and single sign-on. Kerberos is advantageous over username/password as it allows for the communication between a service and a client to be encrypted, which is especially advantageous within e-Social Science as the data being exchanged is particularly sensitive.

Shibboleth provides solutions to two of the three factors; however it fails to provide an implementation of the most important feature of authentication. Conversely, the Shibboleth architecture does account for the need of authentication and specifies that it use the local authentication method of the user.

The following chapter looks at implementations that have utilised these security mechanisms to help redress any security downsides and align them with Grid computing.

2.7 Existing Implementations

This section outlines software solutions that aim to utilise the security methods mentioned in section 2.5 to solve general Grid security issues and problems linked to e-Social Science resources.
2.7.1 MyProxy

MyProxy is an "online credentials repository" [11] system. It consists of a MyProxy client and MyProxy server, the latter of which stores users’ credentials. The standard setup for MyProxy is for the "MyProxy server to hold credentials with lifetimes of weeks, months, or years, with policies that restrict the lifetime of delegated credentials to 12 h or less" [14]. The system allows for credentials to be delegated to multiple resources via a single sign-on method. The term credential used within the MyProxy infrastructure is used to describe the combination of a public key certificate and the corresponding private key.

MyProxy uses a type of digital certificate called a 'proxy' [11], which acts as a standard X.509 certificate. The 'copying’ of a standard certificate to create a proxy allows the owning entity to delegate their privileges to numerous services and resources. A standard X.509 certificate has a long time-to-live, upwards of a year in some cases, however a proxy of the standard certificate can be assigned shorter period of time to help improve security of resource access, which is vitally important within e-Social Science.

The proxy certificate can be used in the single sign-on process. To help improve security, this method reduces the need for entities to exchange private key information. The process begins with the user manually authenticating once, which results in the creation of a proxy certificate. This proxy can then be delegated to authenticate the user for a short period of time (specified in the attributes of the proxy). To achieve this, the long-term standard private key is used to create the shorter lived proxy certificate.

The process of storing a proxy certificate is outlined in [11]. The resultant proxy can now be used from any host in the network, without the need to use long-time credentials. Also, the password used to encrypt the long-time credential is only sent between the user and the client application and therefore there is no need to transmit the password over a network.

To use the proxy it must be retrieved from the server as outlined in [15] and [11]. The retrieved proxy certificate will last up until the expiration time. The process of retrieving a credential can be repeated up until the point in which the credentials held within the MyProxy server expire. At this point, the delegation/storage process must be repeated.

There is one obvious limitation to the MyProxy system, linked with the MyProxy server [15]. If an attacker gains access to the repository, then the stored credentials are vulnerable to maltreatment. This risk however is mitigated by the knowledge that credentials are secured by passwords and have a limited
time-frame in which they can be used.

2.7.2 VOMS

Virtual Organisation Membership Service (VOMS) is an "attribute authority issuing attributes describing a user’s affiliation to a virtual organization (VO)" [21]. The attributes returned allow a service to determine the access rights of a particular user, by describing their affiliation to a specific VO.

The attributes returned by VOMS are encapsulated within a digitally signed Attribute Certificate (AC). The attributes are split into two categories. The first category is an attribute than define the user’s membership with the VO. The second lists the user’s group, roles and capabilities with regards to the VO. These details are encapsulated within the AC and embedded into a proxy certificates that is sent to the service. The roles of the user and their capabilities are stored in a database provided by VOMS, which also supplies "a set of tools for accessing and manipulating the database" [26]. These tools allow the administrators to update and make changes to a user’s rights or assign or remove a user from a VO. Additional VOs can also be added to the database. From a user’s point-of-view, Globus allows the user to generate a proxy credential (in which the AC gets encapsulated). This proxy can then be used to access services (as long as the services are ‘VOMS-aware’).

The hardware required by VOMS is composed of four components [20]. The user architecture is split into client/server architecture, as is the administration side. The server in both scenarios is the VOMS database, and the two clients provide different views into the database. For a standard user, the client uses their interface into VOMS to retrieve their VOMS-certificate, a proxy containing the AC. The admin client however, is used to add/change users and to create roles/groups etc.

The major benefit of using VOMS is that is covers both the authentication and authorisation elements of Grid security. The user must be authentic to retrieve a proxy certificate and the AC embedded in the certificate allows the VOMS enabled service to check what rights the user has to use a service of a set of services, i.e. authorisation. Also, the process for the user compared to a standard creation of a proxy certificate (in MyProxy) is disrupted as little as possible.

There are three obvious drawbacks of this implementation. Firstly, the start-up costs involved of introducing a new server to a network and the required training for administrators. Secondly, the extra work required by administrators to initially add the VO data for each user and then needing to keep this data timely could be quite lengthy. Finally, the service or resource the user is trying to contact needs to
be VOMS-aware and be able to extract the AC data.

2.7.3 ShibGrid

ShibGrid [10] is a project that aims to “integrate the future UK national Shibboleth infrastructure with the UK’s National Grid Service (NGS)”; in short it aims to provide Shibboleth Access for the NGS.

The ShibGrid project aims to meet many requirements. With respect to the architecture created, ShibGrid was designed to allow access to Grid services provided by the National Grid Service, using existing Shibboleth federations. These existing federations are provided by JISC and UK higher-education institutes (e.g. Leeds University provides a Shibboleth federation). In terms of the requirements for a user, they are linked to the two classes of user explained earlier. If certificates are used, ShibGrid aims to make them as transparent as possible, hiding any unnecessary details and attempting to reduce the need for a user to know what is happening to their certificate. However, ShibGrid suggests that where possible, certificate-less access is attained.

As well as being able to retrieve a certificate, ShibGrid must also deal with users wishing to upload their X.509 certificates to the ShibGrid MyProxy server. The main requirement is that there has to be a way of linking the user to their certificate. To achieve this, ShibGrid, at the point of upload, links the Shibboleth identification of a user with their X.509 distinguishing name. These two values therefore act similarly to a database composite primary key. The process involved in uploading is as explained fully in [10], as is the retrieval process.

2.7.4 SHEBANGS

SHEBANGS [17] is an implementation designed to allow access to National Grid Service resources and services without the need to use digital certificates. To achieve this, Shibboleth is used for interactions between the client requesting and the server holding the resource. Assertions are used as credentials in place of digital certificates, with SAML being used to create Grid Security Infrastructure credentials.

A Credential Translation Service (CTS) is the pivotal element in the architecture. This acts as a middleman service, sitting between the Shibboleth infrastructure and the Grid infrastructure. To the user, the CTS appears as the Shibboleth Service Provider (SP) and is accessible via an Internet browser. From a credential aspect, the CTS simulates a MyProxy Client, a Credential Authority and a Virtual Organisation Membership Service (VOMS). The Credential Translation Service has been designed as a lightweight middleware component. The implementation is a CGI Perl script which can be added to the
cgi-bin directory of a Web Server, which itself is protected by Shibboleth. The full SHEBANGS process is outlined in [17].

2.7.5 SARoNGS

Shibboleth Access for Resource on the National Grid Service (SARoNGS) [23] is a project used to integrate Shibboleth authentication with the X.509 certificate infrastructure. It is accomplished by users being authenticated to acquire low assurance credentials that can then be used to access Grid resources held on the National Grid Service (NGS). The SARoNGS project emanates from ShibGrid and SHEBANGS projects.

The architecture involved in SARoNGS and the process is outlined below. The process involves the scenario of retrieving a credential which can then be used to access a Grid resource.

a) The user requests access to a Grid resource and through SARoNGS and is redirected to the Which VO Are you From (WVOAYF) service. This is a webpage allowing the user to specify their VO memberships

b) The user chooses their VO from the WVOAYF. The standard WAYF service then redirects the user to select the IdP of their local institution

c) The user can then authenticate using the local institution’s authentication method

d) Credential delegation process:

(a) IdP redirects the user’s browser to the Credential Translation Service (CTS). The signed SAML assertions are passed in this redirect. The Shibboleth SP asks the IdP’s Attribute Authority for attributes about the user, and passes these attributes along to the CTS

(b) The username and SAML assertions are sent to the MyProxyCA server. This server returns back to the CTS a generated low-assurance certificate

(c) The Virtual Organisation Membership Service (VOMS) verifies the user’s ID based on the low-assurance certificate. If the verification successful, a VOMS-enabled credential is delivered back to the CTS

(d) The CTS delegates this credential to the MyProxy Server (allowing for the credential to be stored for future use)

(e) The CTS passes to the NGS portal a username:password:server triplet. The user can then access
NGS resources using a credential gained from the triplet values

2.8 Overview of Implementations

A full comparison of the above technologies is outlined at the start of section 3

2.9 Methodology

The short timescale of implementing a solution to the problem lends itself to an incremental methodology. Iterative and Incremental Development [9] involves performing numerous iterations, with each iteration being considered as a mini-project in its own right. Each iteration will go through the process of identifying objectives, then designing, implementing and evaluating the produced software.

The initial iterations will deliver a prototype as their end product. A prototype is a "system or partially complete system that is built quickly to explore some aspect of the system requirements" [9]. The 'quickly' aspect is suited to this project, as it allows for rapid development of software solutions. The four steps of the prototype delivery fit in with the four categories of the Boehm Spiral. Initially, the objectives of the prototype are defined, the prototype is then specified (designed), then constructed and finally evaluated. The purpose of the evaluation is to test that the selected objectives have been met and that the prototype works correctly to fulfil the objectives. After the first prototype has been evaluated it can then be used in the second iteration as the basis for the next prototype. Therefore, after each iteration, the software solution will continue to grow in functionality.

After all iterations have been completed, all requirements will have been covered to ensure the final software solution solves completely the requirements set-out. The next chapter sets out the aims of each iteration and what the required outcome of each will be.
Chapter 3

Implementation Choice & Methodology

The first part of this chapter looks at comparing the security options outlined in chapter two and outlining the reasons for choosing one as the basis for an implementation.

3.1 Comparison of Technologies

3.1.1 Authentication

<table>
<thead>
<tr>
<th>Meth.</th>
<th>Authentication Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>MyProxy</td>
<td>Utilises the PKI by using digital certificates</td>
</tr>
<tr>
<td>VOMS</td>
<td>Utilises the PKI by using digital certificates</td>
</tr>
<tr>
<td>ShibGrid</td>
<td>The user’s home institution authentication methods is used to authenticate the user within the Shibboleth architecture. X.509 certificates are then used to authenticate the user with the service</td>
</tr>
<tr>
<td>SHEBANGS</td>
<td>The user’s home institution authentication methods is used to authenticate the user within the Shibboleth architecture. SHEBANGS provides certificate-less access by using GSI credentials, which are used to authenticate the user with the service</td>
</tr>
<tr>
<td>SARoNGS</td>
<td>The user's home institution authentication methods is used to authenticate the user within the Shibboleth architecture. SARONGS uses VOMS-enabled credentials to authenticate the user with the service</td>
</tr>
</tbody>
</table>
### 3.1.2 Authorisation

<table>
<thead>
<tr>
<th>Meth.</th>
<th>Authorisation Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>MyProxy</td>
<td>-</td>
</tr>
<tr>
<td>VOMS</td>
<td>The credentials delegated to services contain attribute details which can be used for authorisation</td>
</tr>
<tr>
<td>ShibGrid</td>
<td>Attributes passed in SAML assertions from the IdP can be used for authorisation</td>
</tr>
<tr>
<td>SHEBANGS</td>
<td>Attributes passed in SAML assertions from the IdP can be used for authorisation</td>
</tr>
<tr>
<td>SARoNGS</td>
<td>Attributes passed in SAML assertions from the IdP can be used for authorisation</td>
</tr>
</tbody>
</table>

### 3.1.3 Single Sign-On

<table>
<thead>
<tr>
<th>Meth.</th>
<th>Single Sign-On Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>MyProxy</td>
<td>Proxy certificates can be created that allow for the delegation of one certificates credentials to multiple resources</td>
</tr>
<tr>
<td>VOMS</td>
<td>Proxy certificates can be created that allow for the delegation of one certificates credentials to multiple resources</td>
</tr>
<tr>
<td>ShibGrid</td>
<td>Shibboleth’s federated identity allows for Single sign-on</td>
</tr>
<tr>
<td>SHEBANGS</td>
<td>Shibboleth’s federated identity allows for Single sign-on</td>
</tr>
<tr>
<td>SARoNGS</td>
<td>Shibboleth’s federated identity allows for Single sign-on</td>
</tr>
</tbody>
</table>

### 3.1.4 Advantages

<table>
<thead>
<tr>
<th>Meth.</th>
<th>Advantages Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>MyProxy</td>
<td>Password not transmitted over the network</td>
</tr>
<tr>
<td>VOMS</td>
<td>Authorisation ability</td>
</tr>
<tr>
<td>ShibGrid</td>
<td>Authorisation ability. Attempts to keep certificates transparent from the user</td>
</tr>
<tr>
<td>SHEBANGS</td>
<td>Authorisation ability. Uses credentials instead of certificates</td>
</tr>
<tr>
<td>SARoNGS</td>
<td>Authorisation ability. Uses credentials instead of certificates</td>
</tr>
</tbody>
</table>
### 3.1.5 Required

<table>
<thead>
<tr>
<th>Meth.</th>
<th>Required Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>MyProxy</td>
<td>MyProxy server</td>
</tr>
<tr>
<td>VOMS</td>
<td>VOMS database containing user attribute information. VOMS server</td>
</tr>
<tr>
<td>ShibGrid</td>
<td>Requires Shibboleth architecture (including IdP and SP), MyProxy Server</td>
</tr>
<tr>
<td>SHEBANGS</td>
<td>Requires Shibboleth architecture (including IdP and SP), MyProxy Server, Credential Translation Service</td>
</tr>
<tr>
<td>SARoNGS</td>
<td>Requires Shibboleth architecture (including IdP and SP), MyProxy Server, Credential Translation Service, VOMS server</td>
</tr>
</tbody>
</table>

### 3.1.6 Disadvantages

<table>
<thead>
<tr>
<th>Meth.</th>
<th>Disadvantages Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>MyProxy</td>
<td>Central storage of certificates can lead to the MyProxy server being vulnerable to attack. User is aware of using certificates. No authorisation</td>
</tr>
<tr>
<td>VOMS</td>
<td>Services being accessed need to be VOMS aware. User is aware of using certificates</td>
</tr>
<tr>
<td>ShibGrid</td>
<td>Initial authentication reliant on existing implementation</td>
</tr>
<tr>
<td>SHEBANGS</td>
<td>Initial authentication reliant on existing implementation</td>
</tr>
<tr>
<td>SARoNGS</td>
<td>Initial authentication reliant on existing implementation. Services need to be VOMS aware</td>
</tr>
</tbody>
</table>

### 3.1.7 MyProxy Server

The National Grid Service, used by the GENeSIS project, provides its own MyProxy server for certificates to be uploaded to and consequently proxy credentials downloaded from. The server is accessible from myproxy.ngs.ac.uk and requires a username and password to access the server. This extra layer of authentication adds additional security issues. With the MyProxy and VOMS implementation (where a VOMS server replaces the MyProxy server), the user must enter their username/password pair to gain access to the server and retrieve credentials. The Shibboleth implementations aim to hide the connection to the MyProxy server from the user and therefore reduce security issues with accessing the server. With ShibGrid, the SAML assertion values are used to connect to the MyProxy server to both upload and retrieve credentials. This method hides the authentication details from the user. SHEBANGS and
SARoNGS perform the authentication with the MyProxy server in the same way as each other, by returning to the client application a username:password:server triplet. These values can be extracted, ideally in a way that is hidden from the user, and then used to connect to the MyProxy server to retrieve the credential. This is similar to ShibGrid in hiding any MyProxy authentication issues from the user; however it differs slightly by the fact that it does not use X.509 certificates and just provides credentials at the time the user requires them. This reduces the need for the upload process prevalent in the other implementations, helping to hide any complications that certificates could bring to users.

3.1.8 Existing Authentication

All the Shibboleth based security implementations require an existing authentication technique at the user’s institution layer. Organisations, like the University of Leeds, are part of the UK Shibboleth Federation and as such provide an authentication layer for users. In the case of Leeds, this is done via a web login form, requiring the user’s local username and password details. This allows the Shibboleth techniques to be viable options when choosing the method to implement for this project.

3.1.9 CTS

Similarly to the MyProxy Server, the National Grid Service provides the Credential Translation Service. To use the CTS, it is deployed over the internet, accessible via a web browser.

3.1.10 Authorisation

Authorisation of a user is not a critical aspect of this project. However, it is seen as a possible enhancement and would add another layer of security to the GENeSIS system and help improve data security and integrity. Therefore, it is sensible to factor this into the final decision of an implemented service.

3.1.11 Certificates

The initial requirements of this project within GENeSIS do not outline whether digital certificates should be used as a means of authenticating the user with their required service/resource. For an experienced technical user, the handling of certificates should not be that difficult; however for a user less experienced with certificates, or for a non-technical user, certificates can be cumbersome and quite unwieldy. The whole process of requesting and retrieving a certificate, especially for those users just requiring ad-hoc
access could seem excessively time-consuming and unnecessarily complex. Therefore, it can be seen as more advantageous to select a method that hides the use of certificates as much as possible.

3.1.12 How the implementations solve the initial problem

The original problem is to improve upon the existing username/password authentication within GEnESIS and the implementations outlined do so. Firstly, all the methods remove the use of username and password as the authentication to the requested service. The Shibboleth based methods still use username/password, however they do not authenticate the user to the service directly, but perform authentication between the user and their local institution first, before performing an extra layer of authentication to the service.

3.1.13 Choice

SARoNGS is the implementation method of choice. This incorporates certificate-less access to Grid resources, unlike MyProxy, VOMS and ShibGrid. This removes the need for users to handle technical certificates and helps with hiding authenticating details. SARoNGS also allows for the use of VOMS-enabled credentials to facilitate user authorisation; this is not provided by MyProxy. There is very little difference between SARoNGS and SHEBANGS, however SARoNGS is the successor of SHEBANGS and therefore the intelligent option is to choose SARoNGS.

3.2 SARoNGS Process

Technical information about how SARoNGS works has been detailed in Chapter 2. This section will look at SARoNGS from a user’s point of view, outlining the steps involved in utilising SARoNGS.

3.2.1 Certificate-less access

The use of the term certificate-less is somewhat misleading with regards to SARoNGS. The process of accessing Grid resources via SARoNGS does require the use of certificates; however they are hidden from the user. This is in contrast to just using MyProxy, for example, in which the user handles directly an end-entity X.509 certificate by retrieving it from the NGS and then uploading it to a MyProxy server. The use of certificates within SARoNGS is contained within the retrieval of a SARoNGS certificate from the MyProxy server.
3.2.2 Obtaining an NGS Account using SARoNGS

A prerequisite of accessing the SRB, where GENeSIS data is retrieved from, is for the user to have an NGS account. This process of creating an NGS account is slightly different from the standard process (outlined in 2.7.1) and is described below:

A standard request for an NGS account would return to the user an X.509 long-lived certificate, with a DN of, for example:
/C=UK/O=eScience/OU=Leeds/L=ISS/CN=scott simpson

The certificate is installed into the user’s browser. Therefore, the user can use the browser to access the NGS account request page and the user’s DN can be retrieved automatically from the installed certificate. With SARoNGS however, there is now no step of retrieving and installing a certificate, and therefore the process of getting an NGS account is changed.

The process for SARoNGS is as follows (for illustrations see Appendix C, SARoNGS Process):

a) The user points their browser to https://cts.ngs.ac.uk/Login/RegNGS.pl
b) After selecting the institution, the user then needs to authenticate themselves within their local institution
c) SARoNGS then generates a unique CN (Common name) for the username of the user logging in via that particular institution
d) A DN is then created, incorporating the CN, and is auto-populated into the NGS account application form
e) From here the user can request an NGS account, which will automatically create the user’s SRB account

3.2.3 Retrieving a SARoNGS Credential

a) The user points their browser to https://cts.ngs.ac.uk/Login/
b) The browser is redirected to the WAYF service and the user must log into their home institution
c) After a successful login, the browser is redirected to the shows WVOAYF service, allowing the user to select their VO
d) After continuing, the NGS creates a VOMS-enabled credential and delegates it to the NGS MyProxy server. To retrieve the credential, a MyProxy username, password and server is pro-
vided. The user can use these details to connect to the server and retrieve the proxy certificate.
e) This proxy (SARoNGS) certificate will have the same DN that was created when requesting an NGS account. Allowing for there to be a link between the certificate and the NGS account and therefore the SRB account

3.3 Methodology

This section is concerned with outlining the aims and objectives of each of the iterations of the application to be created. Each iteration will aim to add increased functionality and adherence to the requirements outlined.

3.3.1 Iteration One

The first iteration will aim to provide a solution to the following requirements:

- The system will successfully authenticate a valid user
- The system will prevent an invalid user from accessing secure data

The chosen technology of SARoNGS must be utilised in this iteration to produce a solution to the outlined requirements. To help with building a solution, the iteration will be split up into smaller deliverables, namely retrieving a SARoNGS credential and retrieving a file from the SRB. Also, to help with implementing a solution, this initial iteration will be coded as business-logic only, with no end-user interface. This allows for quicker implementation, therefore aligning to the advantages of iterative development. In addition, coding the business-logic allows for confirmation that SARoNGS is a feasible solution to the original problem.

3.3.2 Retrieving a SARoNGS Credential

The purpose of this deliverable is to enable the retrieval of an uploaded SARoNGS credential from the NGS MyProxy server. From a user’s point of view, the upload process is a simple login form with the upload being performed transparently. To retrieve this credential from the server, the MyProxy fields of username, password, host and port are needed. In a user-interface driven environment, these variables will be extracted automatically, however for the purpose of this iteration, the values will be hardcoded.
An API providing MyProxy classes can then be used to retrieve the SARoNGS credential, using the MyProxy fields.

To deal with an invalid user attempting to retrieve a SARoNGS credential, there are two layers of checking that should prevent it from happening. Firstly, the web browser upload process requires the user to login via their home authentication, where only a successful login will allow the user to continue. If a distrusted user does have authentic login information, they still need to have an NGS account to be able to upload/retrieve a credential.

### 3.3.3 Connecting to and Retrieving a File from the SRB

This deliverable completes the aims of the first iteration. The purpose of this deliverable is to use the retrieved SARoNGS credential to authenticate a user with the NGS’s SRB. Once authentication has been completed, the user can retrieve files from the SRB that have been previously uploaded (the uploading is outside of this project’s scope).

There are three distinct stages in completing this deliverable. Connecting to the SRB, authenticating the user with the SRB and retrieving a file from the SRB. Connecting to the SRB will require both SRB and user specific information. The SRB information required is that of the host and port that is used as the point of connection and will be hard-coded into the application.

Successful authentication should only occur if the user has a valid NGS account and their DN from the SARoNGS credential matches that of a valid SRB account. If either of those are invalid, then authentication should not occur.

### 3.4 Iteration Two

The second iteration will build upon the work from iteration one, and in addition, aim to provide a solution for the following requirements:

- The system will demonstrate authentication within an e-Social Science environment
- The system will enhance security by implementing a 'time-to-live' on any authentication method(s) used

Similarly to the first iteration, to help deliver the additional functionality, the development will be divided into deliverables. The first deliverable will focus on providing the functionality for a user to
login via SARoNGS and upload a SARoNGS credential. The second deliverable will then use the credential to retrieve a file from SRB. These deliverables differ from those in iteration one as this iteration will introduce a presentation layer, providing a user-interface to allow for dynamic interaction.

3.4.1 Portlet Creation

To align with the current e-Social Science environment, namely GENeSIS, portlets will be created to match the technology of those already implemented. The development tool chosen for the creation of the portlet will be Netbeans, because of two factors. Firstly, the existing GENeSIS project has been developed using this IDE and therefore any support required during implementation can be garnered from the original developer. Also, Netbeans allows for the installation of add-ons to enhance the IDE. One of these add-ons is the Portlet Pack, which enables easy creation of the skeleton classes required for portlet development and deployment.

3.4.2 Create Login Portlet

This deliverable will provide an interface for the business-logic created in iteration one. The idea of the login portlet is for the user to click a button which automatically starts the SARoNGS login process. After selecting their required VOs to complete the login, the user should be returned back to the browser page containing the portal and contained portlets. The returned URL will then contain the MyProxy username:password:server triplet, which can be extracted and then used to connect to the MyProxy server and retrieve the SARoNGS credential. This functionality mirrors “Retrieving a SARoNGS Credential” from iteration one, however the MyProxy values are created dynamically by the user on Shibboleth login and therefore no hard-coding of the values is needed.

3.4.3 Create Get File Portlet

To ensure the existing system model is adhered to, this portlet will allow for the user to select the .genesis file(s) they require from the SRB and therefore perform an action upon the data in the file. The functionality provided by this portlet utilises the business-logic created in the first iteration. The returned MyProxy login details are used to retrieve the SARoNGS credential, which can then be used to authenticate the user with the SRB. From here dynamic file selection and download can be performed.

To allow for the portlet to be used, the user must first use the ‘Login Portlet’ and therefore this ‘Get File’ portlet must provide suitable information to the user that this is the path of action to be taken. Once
the path has been followed correctly, the portlet will provide, to the user, a list of the files stored in the SRB. The user can choose one or more of the files to download for use within an existing GENeSIS portlet (e.g., analyse and create a chart from the data).

### 3.4.4 Portlet Process

The process outlined above in the portlets is illustrated below on the diagram. This is the path the users should take to retrieve an SRB file:

![Figure 3.1: Portlet Process; Retrieving a File](image)

- a) User login via SARoNGS to generate credential and store on MyProxy server
- b) MyProxy details sent back to Login Portlet
- c) SARoNGS credential retrieved from MyProxy server
- d) Credential passed to SRB along with file(s) required by user
- e) SRB performs authentication on the credential
- f) If authentication in 'e' is successful, then the files are returned to the user

### 3.4.5 Time to Live

The above deliverables demonstrate the application, including authentication, being utilised within an e-Social Science environment. However, one of the issues initially raised with the existing security implementation was that of using a long-lasting method of authentication i.e., a password. To overcome this problem, this new implementation will attempt to include two levels of expiration. Firstly, the credential created and stored on the MyProxy server will have its own default expiration before it is (a) no longer accessible from the server, and (b) no longer usable after being retrieved from the server. The second level of expiration is concerned with the MyProxy login values returned from the server after
SARoNGS login. To enable the values to be used to retrieve an SRB file for example, they must be stored for at least the length of the user’s session, and therefore the implementation will need to deal with being able to store the values.

### 3.5 Iteration Three

The third iteration will build upon the work from the previous two iterations, and in addition, aim to provide a solution for the following requirements:

- The system will be created as a web-service, allowing for use by other applications
- The system will implement a method that will allow for the authorisation of a user, on top of the authentication
- The system will be integrated into GENeSIS

This final iteration will aim to deliver the final elements of functionality, to help solve the original authentication problem. The first deliverable will be to utilise web services in the implementation, allowing for any application to utilise the service. In addition, the second deliverable will aim to add an extra layer of security by creating a function that will allow an administrator to determine the rights of a particular authenticated user, to ensure they have sufficient privileges to access a particular file for example. Finally, the overall implementation will then be integrated into the existing GENeSIS system.

#### 3.5.1 Create a Web Service

This deliverable will aim to create a web service from the existing application code created in iterations one and two. A web service is a standalone function or method that can be called over HTTP, with SOAP being used for message communication and transfer. The format of the messages is that of XML, which is platform independent; the result of which allows for a service to be written in one language and then be consumed by an application utilising a completely different hardware and software setup. The purpose of a web service in relation to this project is that a function or method can be created as web service and then utilised by other users who require similar functionality.

The function chosen for the web service will allow the user to login via SARoNGS, creating and uploading the SARoNGS credential. The service would then use the returned MyProxy server values to retrieve the credential and return it as a string to the user. This does have a problem however, in that the MyProxy password would be sent as part of a SOAP request.
The purpose of this service is that an application developer could use the returned string to create programmatic credential object in whatever language they require, using any suitable API. The credential object can then be used to authenticate a user with their required service, decoupling the link between a credential object only being used to authenticate with SRB, as in previous iterations.

3.5.2 User Authorisation

This deliverable will aim to display how authorisation can be used within GENeSIS to add an extra layer of user security. User authorisation is based upon a user’s role within a certain VO (Virtual Organisation). For example, user Bob Smith could be a member of the NGS VO, with a role set as ‘member’. Frank Brown could also be a member of the same VO, but be in an ‘admin’ role. Therefore, Frank would have higher authority and for example, have access to data that Bob shouldn’t be able to see. This scenario could be used within GENeSIS, where one class has a higher clearance of access to data compared to another class of user (or members of different VOs have different levels of access).

VO memberships are stored within a VOMS-certificate. This certificate is then embedded within the parent credential, i.e., the SARoNGS credential. This embedding is performed automatically by the NGS when the SARoNGS credential is created after user login.

GENeSIS is not VOMS-aware, and therefore has no concept of dealing with one, a VOMS credential and two, different users and their related roles. Creating roles and user groups is beyond the scope of this project and therefore performing full user authorisation is not completely possible. However, the created application will attempt to display the VOMS details for a logged in user and can then perform dummy authorisation tasks (e.g., only displaying certain data is a user is a member of a certain VO).

3.5.3 Integrate with GENeSIS

The final deliverable is centred on ensuring the above deliverables, for all iterations, can be deployed successfully within the existing GENeSIS project. The main integrating feature is that of user authentication when retrieving a GENeSIS file. To be successfully incorporated into GENeSIS, a user should be able to access the Liferay portal, and then use portlets to login into SARoNGS, retrieve credentials and access and download relevant .genesis files. The retrieved files should then be able to be uploaded into the existing GENeSIS portlets, for analysis, charting and other services.
Chapter 4

Iterative Implementation & Evaluation

The implementation will be based upon the previously discussed iterative methodology. Each iteration will look to add increased functionality and adherence to the requirements of the system, with the ultimate goal of producing a system that solves the original problem. The deliverables to be implemented have been set out in Chapter 3.

4.1 Iteration One

The aim of the first iteration is to programmatically authenticate a user to the SRB using SARoNGS. This iteration focuses on coding the business logic of the authentication service separate to any user interface. This allows for the feasibility of programmatically combining SRB and SARoNGS to be determined before progressing with any kind of user interface.

4.1.1 Retrieving SARoNGS Credential

The first stage is to retrieve the proxy SARoNGS credential from the NGS’s MyProxy Servers. To retrieve the credential, details of the MyProxy server holding the credential need to be known. The server details are the username, password, host and port. The ideal way to retrieve these items is for them to be passed in a URL back to the web browser being used for a user’s interaction with GEneSIS. However, as this iteration doesn’t implement an interface, the values are hard-coded into the application. To get the MyProxy values that are to be hard-coded, the SARoNGS process is followed via https://cts.ngs.ac.uk/scgi-bin/Login.pl; the result of which shows a web page listing the username,
The next stage is to utilise the MyProxy values to retrieve the SARoNGS credential and store it programmatically. To enable this deliverable development, existing Grid APIs are utilised to provide classes to stored instances of Grid ‘objects’. The first ‘object’ required is that of a MyProxy server. To create a MyProxy server object, the Globus Toolkit is used. An instance of the MyProxy class can be created by passing in the MyProxy host and port values as arguments. The stored SARoNGS credentials can then be retrieved from the server by calling the MyProxy object’s get method. The retrieved credentials are then stored as a Java GSSCredential object.

4.1.2 Connecting to and Retrieving a File from the SRB

The next stage is to use the GSSCredential to connect to the SRB and retrieve the file required, during which, authentication will occur.

To connect to the SRB, Jargon API is required to programmatically instantiate SRB classes. To utilise the Jargon API, a relevant JAR file is required. This JAR file allows for, initially, the creation of an SRBAccount object. The constructor of this class requires the retrieved GSSCredential, along with the host and port of the SRB server (different from the host and port of the MyProxy server). The port for connecting to SRB is default to 5544 and therefore there is no need to change this value. The actual choice of the SRB host doesn’t matter because of the way SRB works, allowing for a user to connect to any host (server) to retrieve/upload their files. For implementation purposes, a default host of srb1.ngs.rl.ac.uk is used. The host, port and credential can then be used to create an SRBAccount object.

Currently, the created SRBAccount is just a local object with parameters and no connection to the

```
public static GSSCredential getProxyCredential() throws MyProxyException, GSIServiceException, MalformedURLException, IOException {
    String myProxyUsername = "<user>;"
    String myProxyPassword = "<password>"
    String myProxyUrl = "myproxy.ngr.ac.uk:7511"
    String myProxyHost = myProxyUrl.substring(0, myProxyUrl.indexOf(':'));
    int myProxyPort = Integer.parseInt(myProxyUrl.substring(myProxyUrl.indexOf(':')+1, myProxyUrl.length()));
    int lifetime = 1;
    MyProxy proxyServer = new MyProxy(myProxyHost, myProxyPort);
    GSSCredential credential = proxyServer.get(myProxyUsername, myProxyPassword, lifetime);
}
```
SRB. To setup the connection, an SRBFileSystem object needs to be created. The SRBFileSystem requires the account object as an argument, and at this point, the user authentication occurs. The authentication looks at the SARoNGS credential and related DN that has been passed in the account object. The credential itself will be signed by the SARoNGS CA, and so when the credential reaches the SRB, the SARoNGS CA can decrypt the credential and ensure that it is authentic, valid and has not been tampered with. Once this has been performed, the DN is extracted. SRB will then check to see if an account is linked to that DN. If so, authentication has been successfully completed between the user and the NGS’s SRB.

After successful authentication, the SRBFileSystem object can be used to perform numerous actions on the contained files (view, delete, copy etc.). For the purpose of this deliverable, a specified file is retrieved by accessing it as an SRBFile object, then a local copy of the file is created within the client’s local file store.

```java
String srbHost = "srbi.ngs.rl.ac.uk";
int srbPort = 5644;
GSSCredential cred = getProxyCred();
SRBAccount account = new SRBAccount(srbHost, srbPort, cred);
SRBFileSystem fileSys = new SRBFileSystem(account);
SRBFile file = new SRBFile(fileSys, "hello.txt");
file.copyTo(new LocalFile("C:\temp\hello.test"), true);
```

Figure 4.2: Code to retrieve a file from the SRB using credential authentication

### 4.1.3 Problems

The first problem with SARonGS arose when attempting to use a retrieved SARoNGS credential to authenticate with the NGS’s SRB. After storing the retrieved credential as a GSSCredenial object, it is then used to authenticate with the SRB. However, initial attempts to access the SRB returned authentication errors. After discussions with NGS support, it was determined that the certificate directory on the SRB server did not have the required CA (Certificate Authority) details, as this was the first time SARoNGS was being utilised in this manner. This was resolved by installing the SARoNGS CA on the SRB server.

In addition, an issue was found at the stage of creating an NGS account and ultimately an SRB account. When requesting the NGS account, the SRB account username and password are automatically generated. The username created was a direct copy of the generated CN, therefore creating a unique identifier for the account. However, the NGS SRB could not cope with the length of the generated CN (a
string of 40 hexadecimal characters). This issue was referred back to the National Grid Service, whose solution was to manually create an SRB account with a shorter username. This created account is then linked to the DN of the SARoNGS certificate, creating a chain, and therefore the certificate can be used to authenticate access to the SRB account.

4.1.4 End of Iteration Testing

See Appendix B

4.1.5 Evaluation of the First Iteration

The criterion for the first iteration was for the resultant system to have solved the following criteria:

- The system will successfully authenticate a valid user
- The system will prevent an invalid user from accessing secure data

**The system will successfully authenticate a valid user**

This requirement was resolved in the first iteration by implementing Storage Resource Broker (SRB) authentication of a user via a SARoNGS credential. To achieve this, a SARoNGS credential was first created and uploaded to a MyProxy server, via the provided NGS SARoNGS web interface. The credential is then retrieved programmatically and used to authenticate to and access the SRB. To confirm that authentication has been successful, a file is retrieved from the SRB.

The 'Authenticate Valid User' test sequence then proved that the implemented solution performed as required.

**The system will prevent an invalid user from accessing secure data**

This requirement was met by ensuring that only those users’ who are valid to authenticate can access the SRB. This was achieved by utilising the security methods chosen to block access to those users who are not valid. Firstly, the local institution authentication is used as the first barrier of security, with only those people who have login details for an institution that is part of the UK Federation able to continue. Once past this stage, the user must then have a registered SARoNGS NGS account linked to an SRB account to then utilise a SARoNGS credential to access the SRB.

The 'Prevent Invalid User from Authenticating' test sequence then proved that the implemented solution performed as required.
Overall, it has been shown that the implemented deliverables have solved the first two requirements. Therefore, the coded business-logic can be utilised in the development of the next iteration.

4.2 Iteration Two

The aim of this iteration is to incorporate the business logic created in the first iteration into a service which has a user front-end, allowing for the retrieval of SARoNGS credentials to be dynamic. The interface will allow for the user to login via Shibboleth, and then transparently create and upload a SARoNGS credential. The credential can then be retrieved from the MyProxy server to be used as the authentication method into the SRB.

4.2.1 Create JSP Login Portlet

There are two necessary components that work together to create a dynamic portlet. The first component is a JSP page, which enables the creation of the actual interface to the portlet. The second component is a Java class, which captures the events that are fired from within the JSP portlet. The two main events captured are the processing of an action, e.g., pressing a button. This combination allows for the events triggered by a user’s action to be captured and used to dynamically update the content of the portlet. To achieve this dynamic updating, variables are defined in the JSP page and can be updated from the Java class.

The first step was to add a button to the created portlet, whose action pointed to NGS SARoNGS login page [https://cts.ngs.ac.uk/scgi-bin/Login.pl](https://cts.ngs.ac.uk/scgi-bin/Login.pl). The standard result of pointing to this page would be for the MyProxy details to be displayed to the screen and not returned to the GENeSIS user interface, which is a problem that would prevent the process from being dynamic and flowing in a user-friendly way. To overcome this, a hidden input is added to the form, specifying the URL of the current page. The Perl script at the SARoNGS end can then extract this URL and be aware to redirect back to that page. The resultant process is very similar to the standard SARoNGS process. However, the confirmation button click from the WVOAYF page redirects the user back to the Liferay GENeSIS page. Within this redirect, the MyProxy server, host, username and password details are added to the URL as query string parameters. This now allows the application to retrieve these MyProxy details and utilise them, transparently, for user authentication with the SRB.

The retrieved MyProxy details can be stored as variables which are local to the Login Portlet. For a
test scenario, this causes no problems, as the values can be extracted and used to perform authentication. However, for the purpose of this project this was not a suitable solution, as other portlets would require the MyProxy values. To solve this issue, sessions were utilised. Sessions are objects used within an HTTP environment to maintain state in a stateless environment. They are passed in the HTTP header, allowing for values to be passed between web pages or portlets. The session objects can contain numerous key-value pairs, which can be accessed and manipulated by the Java code that deals with a user interaction, e.g., a button press or a browser refresh. For the purpose of this project, the session object was used to store four key-value MyProxy pairs (host, port, username and password). To ensure the values were accessible by all portlets, the session object had to be set as 'ApplicationScope'.

The use of sessions also allows for a test to be performed, to decide what to display to the user. If there are MyProxy values in the session, the message 'successfully logged in and retrieved details' message will be displayed. If there are no values in the session, the 'Login' button is shown.

4.2.2 Create Get File Portlet

This deliverable makes use of the session data created in the Login Portlet to perform SARoNGS based SRB authentication and then retrieve selected files. As this iteration is adding a user interface to the existing business-logic from the first iteration, another portlet was created to handle the user interaction. The portlet attempts to list all the SRB files available to the user in a list box and provide a button to download selected files.

The initial stage was to create the portlet, adding a list box and button. A variable was created within the JSP to allow the list of files to be passed between the presentation and application layers. To ensure the list box is populated automatically, without the user having to click a button, the Java method that deals with the refresh of the portlet was utilised. At the point of a refresh, the method investigates the session object to check to see if it contains MyProxy keys added from the Login Portlet. If there are no such MyProxy values, this point is made to the user, advising them to login via SARoNGS first. However, if the session values do exist, they are extracted and then used within the MyProxy get method to retrieve the SARoNGS credential (using the code from iteration one). The process then continues to produce an SRBFileSystem object. Each file name is returned back to the application and stored as an ArrayList. The JSP portlet can then iterate through this List, adding each file to the list box. The user can then select one or more files from the list box, then click the button and download the files to a local store.
4.2.3 Time-to-live

This deliverable aims to provide a suitable solution to the 'time-to-live' issue by providing two levels of expiration of credential details. Firstly, when logging in through SARoNGS, the password generated to access the credential on the MyProxy server is only valid for seven days, so thus reducing the chance of an unscrupulous third-party having long term access to the MyProxy server and therefore credentials used for authentication. Secondly, the implemented application itself deals with the time-to-live. The credential generated is only a programmatic object, generated 'on-the-fly’ by the code itself; therefore no third-party should be able to gain access to the credential itself. In addition, the returned MyProxy values are only stored as long as the session is active; therefore, realistically, their time-to-live will be a matter of minutes or a few hours at most.

4.2.4 End of Iteration Testing

See Appendix B

4.2.5 Evaluation of the Second Iteration

The purpose of the second iteration was for the following requirements to be met:

- The system will demonstrate authentication within an e-Social Science environment
- The system will enhance security by implementing a 'time-to-live' on any authentication method(s) used

The system will demonstrate authentication within an e-Social Science environment

This requirement was achieved in the second iteration by utilising the functionality developed in the first iteration to create a user interface that was aligned with the existing interface of GENeSIS. This required the development of web portlets for human-computer interaction, displaying dynamic content and reacting to user commands. The implemented portlets allow the user to login through a portlet and use a separate portlet to authenticate with the SRB and ultimately retrieve a file.

The test sequence contained within 'Retrieve SRB File Dynamically via user interface' demonstrated that the functionality of the user interfaces perform as expected and therefore solve the requirement successfully.
The system will enhance security by implementing a 'time-to-live’ on any authentication method(s) used

This requirement was realised by a two-fold approach of utilising existing expiration methods. Firstly, the expiration provided by SARoNGS was utilised to take advantage of the short-lived SARoNGS credential property, ensuring that a retrieved credential could not be utilised for any great length of time. Secondly, sessions were used to maintain state, by storing the MyProxy values for the duration of a user’s browsing session. This ensures that the values will only be 'active' for a short length of time and that each new browsing session will require new login details. To ensure the time-to-live functionality was dealt with in a user-friendly manner, checks are performed by the portlets to relay any necessary messages to users about expired credentials, and the subsequent need to login.

The 'Dealing with Time-to-Live’ tests were performed to ensure that the expiration functionality worked correctly and that relevant messages were shown and actions were performed by the portlets, all to ensure that the requirement was met successfully.

Overall, the second iteration built upon the business-logic implemented in the first iteration by adding a user interface to allow for dynamic interaction; which in turn has helped solve the requirement and enhancement. Therefore, with the minimum requirements covered, the third iteration can commence to implement additional enhanced features.

4.3 Iteration Three

This final iteration will aim to deliver the final elements of functionality, to help solve the original authentication problem. The first deliverable will be to utilise web services in the implementation, allowing for any application to utilise the service provided for their own needs. In addition, the second deliverable will aim to add an extra layer of security by creating a function that will allow an administrator to determine the rights of a particular authenticated user, to ensure they have sufficient privileges to access a particular file for example. Finally, the overall implementation will then be integrated into the existing GENeSIS system.

4.3.1 Create a Web Service

As explained in the methodology section, the function that is to be created as a web service is that of the retrieval of a previously uploaded SARoNGS credential. The returned credential is stored in a
String format, contained within the SOAP response, allowing for the consuming application to deal with the response in a method suiting their needs. The code that deals with retrieving a credential from the MyProxy server has already been written in the first iteration and therefore it can be re-used for this deliverable.

To create the web service operation itself, inputs and outputs were required. The input parameters are the MyProxy server details (host, port, username and password) and the output format is a String object. In the source code, the values require annotations to make them web service specific. The @WebMethod is used to specify that the created operation is a web service operation. Similarly, @WebParam is used to annotate the input parameters, to notify that the parameter is part of a web service and therefore should be included in the SOAP request message.

The design-view creates the skeleton of the code; however the business logic needs to added. The code that retrieves the credential is the same as that that has been previously created, however, this web service needs to deal with converting the credential details into a String, which initially caused an issue. The credential retrieved is a GSSCredential object and the GSSCredential class does not contain a method to convert the credential into a String, or at least bytes which can be turned into a String. Therefore to achieve the conversion to String, additional features of the Globus Toolkit were required. The returned credential object needed to be converted into an ExtendedGSSCredential object, upon which a method could be used to convert the credential into bytes and then into a String. The first stage of the credential conversion was to turn the GSSCredential into a GlobusCredential, using the getGlobusCredential() method. The resultant GlobusCredential could then be converted into ExtendedGSSCredential, by creating a new ExtendedGSSCredential object, and passing in the GlobusCredential as an argument. The resultant ExtendedGSSCredential could then have the export method called upon it, which returns an array of bytes. From this, simple byte to String conversion occurs, with the resultant String the return value.

```
MyProxy proxyServer = new MyProxy(proxyHost, proxyPort);
GSSCredential credential = proxyServer.get(proxyUsername, proxyPassword, 1);
GlobusCredential globCred = ((GlobusGSSCredentialImpl)credential).getGlobusCredential();
ExtendedGSSCredential cred = new GlobusGSSCredentialImpl(globCred, ExtendedGSSCredential.IMEXP_GPAQTHE);
byte[] byteArray = cred.export(ExtendedGSSCredential.IMEXP_GPAQTHE);
String credString = new String(byteArray);
return credString;
```

Figure 4.3: Web Service Code
For the purpose of testing the web service, it was deployed locally to localhost. To handle the deployment, GlassFish was used as the server. Upon running GlassFish, the web service is deployed and GlassFish starts listening at a specified address for interactions with the service. For example, listening at http://localhost:15612/GetCredentialServiceService.

To test the deployed service and provide an example of the use of the returned value from the service, a client portlet was created. This can utilize the NetBeans IDE to automate the web service client code. The code generated is used to call the getCredential service, passing in the MyProxy values into the constructor. The returned value from the service is the SARoNGS credential in String format.

To be able to use the String credential a procedure is created that is similar to the opposite of the procedure utilised by the service to create the String object. Initially, the String is converted into bytes. These bytes are then used as a constructor argument in the creation of a GSSCredential object. To test authentication, the GSSCredential object is used to create an SRBAccount object, which is then used to attempt to create an SRBFileSystem object. If this is created successfully, authentication itself has been successful.

4.3.2 User Authorisation

This deliverable looks at the how authorisation can be implemented within GENeSIS to add another layer of security, on top of user authentication using a SARoNGS credential.

When a user performs SARoNGS login, the final stage is to select their VO and their membership to it. The VO details are encapsulated in a VOMS-credential, which is stored within the SARoNGS credential. To extract the details, existing middleware in the form of Grix [40] was used to provide the classes to extract the details. Grix provides a VomsProxy class as part of its JAR file, which allows for the VOMS credential to be extracted from a GlobusCredential.

To create the GlobusCredential, the work performed in the web service deliverable was re-used to convert the web service String to an array of bytes. This array was converted to a GlobusCredential by instantiating a new instance, and passing in the bytes as a constructor argument. The VomsProxy class can then be instantiated, using the created GlobusCredential as an argument. The resultant object is that of a VOMS credential, which can be investigated to extract individual VOMS details (VO and membership relations).

The important key within the VOMS entry is 'attribute'. This shows that the user is a member of the UKFederation VO, within which the user is a member of the leeds.ac.uk group. This information is
4.3.3 Integrate with GENeSIS

This final deliverable ensures that the authentication method is suitable for GENeSIS by checking that a .genesis file can be retrieved from a the SRB, then used in a standard GENeSIS way, i.e., perform analysis or create a graph from the data. To enable this deliverable, no extra functionality was needed, as it is just a combination of previous deliverables working as a complete system.

End of Iteration Testing

See Appendix B

4.3.4 Evaluation of the Third Iteration

The expected outcome of the third iteration was for the following enhancements to have been met:

- The system will be created as a web-service, allowing for use by other applications
- The system will implement a method that will allow for the authorisation of a user, on top of the authentication
- The system will be integrated into GENeSIS

The system will be created as a web-service, allowing for use by other applications

This enhancement was achieved by the creation of a web service whose purpose is to retrieve a SARoNGS credential as a String object as part of a SOAP response. Thus, the created web service does not exactly match the original requirement, as the ‘system’ as a whole has not been created as a web service, as this is impossible due to the nature of the application and the importance of a web service being a function, not a system.
The service takes the form of a simple input-process-output mechanism. The inputs are the MyProxy server values, which are then utilised in the processing to retrieve a credential object, which is then converted into a String. The resultant string is then passed as an output. The resultant web service can then be used by other applications who are utilising a MyProxy server to store credentials and require the extraction of the credential and, depending on the implementation, a system not utilising SARoNGS could be used, as the actual service itself is not reliant on SARoNGS.

The ’Web Service Test’ was performed to both show that the web service performed correctly and that the resultant data can be consumed by a client.

The system will implement a method that will allow for the authorisation of a user, on top of the authentication

To fully achieve this deliverable would have required the availability of user groups and roles for the GENeSIS project, which would have allowed for full authorisation to occur. However, as they were not implemented by the GENeSIS project and time did not allow for any such implementation, example functionality was created that displayed the availability of VOMS and VO information about users that could be used for authorisation.

The developed function showed how the VOMS information can be extracted from a credential and the individual attributes extracted to analyse the attribute values. The resultant values can that be interrogated, to ensure that a user has sufficient rights to access, for example, a file or data item. This was tested in ‘User Authorisation Testing’ to illustrated how a different ‘role’ value can affect the access rights of a user, which in turn demonstrated how authorisation can be performed within GENeSIS in the future.

The system will be integrated into GENeSIS

The overall system developed from the individual deliverables was with the aim of allowing the developed authentication system to be utilised by GENeSIS. The delivered application allows for the developed portlets to be added directly into the GENeSIS environment, ensuring authentication can occur directly and that the subsequent retrieval of files from the SRB is possible. The retrieved files can then utilised by the existing GENeSIS services to perform required actions including analysis and charting.

The following system evaluation section discusses, in detail, the effectiveness of the use of SARoNGS as an authentication method and then evaluates the value of the delivered solution to GENeSIS.
Chapter 5

System Evaluation

5.1 SARoNGS Evaluation

The purpose of this chapter is to follow on from the iterative evaluations section and to discuss the system as a whole. Initially, the choice of SARoNGS as an authentication methodology will be analysed with regards to its features, with any drawbacks outlined. Secondly, the produced solution will be evaluated with regards to the original system, comparing the solutions in terms of their security benefits and their usability.

5.1.1 Performance

The first criterion is related to reliability and availability, under the umbrella of performance. SARoNGS must be relied upon to be available when the user requires, as without the ability to login through SARoNGS, they will be unable to obtain a credential and therefore unable to authenticate themselves with the Storage Resource Broker.

SARoNGS has been created, and therefore is supported, by a third party, the National Grid Service (NGS). The use of a third-party system means that there is a reliance on an external body beyond the control of the GENeSIS project. Therefore, the GENeSIS project team must be confident that SARoNGS is a reliable mechanism that is available as and when required.

On the National Grid Service, for SARoNGS to be available, it must be possible for a user to visit https://cts.ngs.ac.uk/scgi-bin/Login.pl via the portlet Login button, and go through the process of logging in through Shibboleth and selecting their VO memberships. If this is not possible,
either due to the Where Are You From service not allowing for Institution selection or that the web server
hosting cts.ngs.ac.uk is unavailable, then the user cannot complete their login, therefore not allowing
them to authenticate with the SRB and download files. The obvious issue that arises is that a user
or set of users’ cannot perform a task; however this can be mitigated by the users’ having some files
downloaded to local store for use. The probability of SARoNGS being unreliable and unavailable
should be relatively slim, as it is hosted and supported by a national body, whose expertise is providing
and hosting Grid resources. However, as the technology of SARoNGS is new and the use of a CTS is
in its infancy, there is a chance that downtime could be required to perform updates or that the service
becomes inaccessible at times due to load or other unforeseen circumstances. To help mitigate the risk of
this occurring, a future extension of this project could be to incorporate a backup mechanism that would
allow for authentication if SARoNGS is down; this is discussed in more detail in future extensions.

5.1.2 Transparency

The next element to help evaluate SARoNGS is that of how transparent SARoNGS is in helping user
authentication, especially to aid usability for non-technical users. The original system utilises username
and password, which in itself is not transparent, however, how the values are utilised is kept secret from
the user. Also, the ubiquitous nature of passwords helps increase usability and recognisability.

SARoNGS offers transparency in its dealings with certificates and credentials, by hiding detail from
the user. The credential is been created and uploaded to a MyProxy server, transparently. This trans-
parency helps hide any complexity, increases usability and user-friendliness. Comparing this to the
original system, there is very little difference in the technical nature of the process. The original system
required the user to enter their username and password, along with any details regarding their SRB.
The new process is similar in terms of having to enter the username and password; however the need to
enter SRB details is replaced by the step to enter VO details. Therefore, the overall transparency of the
authentication process is very similar.

When comparing SARoNGS to other mechanisms with regards to transparency, it performs ad-
mirably with hiding authentication details from the user. If MyProxy or VOMS was chosen solely as
the authentication method, then the level of transparency would have been reduced significantly. With
MyProxy, the user would have had to go through the full process of requesting a certificate and proving
their identity to retrieve the certificate. The next stage would have involved manually uploading the cer-
tificate to a MyProxy server. With SARoNGS, the stage of having to prove identification is removed and
the uploading process is performed transparently. Also, using MyProxy or VOMS would have required the user to look after their long-lasting end-entity certificate, ensuring it is kept safe and secure behind a usually difficult to remember high-strength password.

One drawback of using SARoNGS with regards to transparency is the method of retrieving the MyProxy server details. When a user confirms their login into SARoNGS, the MyProxy details are sent back to the user’s browser as part of the URL. There is an obvious lack of transparency here, with the user able to see that a username and password has been generated, even though they are not likely to know what it is being used for. One way of solving the issue in the future could be for the password to be encrypted and then sent in the URL, so that when the password is sent back to the NGS’s MyProxy server for credential retrieval, the password could be decrypted before use. This however, would need to be implemented by the NGS on the ‘server’ side.

5.1.3 Security

The third item, and conceivably the most important, is that of security. The security of the data used within GEneSIS is paramount and therefore evaluating SARoNGS’s ability to effectively provide this security is necessary, including how it compares to the original system.

The method of authentication that SARoNGS deploys is two-fold, utilising an existing local authentication mechanism to initially verify the identity of a user, before then creating a credential that can be used to authenticate a user with a grid resource. This dual-layer authentication helps enhance data security over the existing implementation, which only requires one level of authentication to access a grid resource, namely username and password. The local authentication in SARoNGS is also username and password, however this is just an authentication step to create a credential, whereas in the original mechanism, the username and password directly authenticate with the grid resource (SRB).

With regards to both confidentiality and integrity, SARoNGS can play a huge role in improving these factors over the original mechanism. Confidentiality relates to ensuring that data is only accessed by those who have the rights to access it, therefore ensuring that the data is kept hidden from those who should not see it. This also relates to integrity, which is concerned with guaranteeing that data is not modified by an unauthorised person. SARoNGS provides authorisation features that allow it to promote both integrity and confidentiality. The use of Attribute Certificates in the form of VOMS credentials means that user roles within groups and organisations can be defined, with specific access rights being assigned to individuals. Therefore, once a user has been authenticated, there is another level
of checking to make certain that the user has the specific rights to access data (confidentiality). Also, what actions that user can perform can also be defined, whether it be just read or read and write for example (integrity). This level of authorisation is not possible with the existing mechanism.

There are however still security issues to resolve, specifically regarding SARoNGS. Discussed later will be the remaining security issues, but this section will deal with items that are prevalent with SARoNGS, irrespective of the environment it is used within. The main issue with SARoNGS is its method of returning the MyProxy password in the URL of the user’s web browser. This was discussed earlier in terms of a lack of transparency, but as well as that, there is an issue with confidentiality due to the way the value is transmitted from server to client. With the transparency issue, the password value is being sent back to the user, after authentication, so it is presumed the user is legitimate and therefore them having access to the password is not a massive security issue. However, the URL containing the password is communicated over HTTP not HTTPS (Hypertext Transfer Protocol Secure), which means that the URL is not encrypted and therefore could be susceptible to eaves-dropping or packet sniffing, the result of which would be a untrusted third-party gaining access to a password that can be used to retrieve credentials. The risk is however mitigated by the short-lived time of the password, reducing long term security impacts. To help solve the communication issue, the SARoNGS web page that returns the password could put in place a rule only allowing for the returning URL to be HTTPS, therefore encrypting the entire URL response.

5.1.4 Summary

Overall, in terms of the most important factor of security, SARoNGS provides:

- A more effective mechanism than the existing method of username and password
- A mechanism that is more advanced, but not too dissimilar in terms of user interfacing, than a standard username and password authentication
- Tangible benefits over the existing method

There are however still issues, including:

- The use of a third party tool does bring performance and reliability challenges
- Additional security challenges regarding secure communications
5.2 Solution Evaluation

This segment of the evaluation reviews the solution delivered to solving the original problem. It will look at the system as a whole, rather than the requirements and enhancements, which as discussed in section 4, have been met individually. The system will be reviewed against specific criteria, including items that were emergent during the research and development.

5.2.1 Comparison of Existing Methods

The requirement "Existing authentication methods and techniques will be evaluated" was accomplished in section 3.1 when technologies were compared. This allowed for direct comparison of technologies, alongside the chosen important categories including authentication and authorisation. The result of this table, along with the extensive background research into the technologies, provided a platform for deciding upon the technology to use within this project. The resultant choice of SARoNGS allowed for a successful solution to be developed, in terms of both usability and security.

5.2.2 Usability

The first item under consideration is that of the usability of the system. Usability relates to how easy-to-use and suitable the solution is to its intended audience. As explained throughout, the standard user of the system will be someone with limited technical knowledge, whose only interest is that of utilising the GENeSIS services to analyse social science data. Therefore, the delivered authentication system should be a simple, suitable addition to the current GENeSIS system.

The first item that helps with usability is that of portlet authentication, via a web browser. The original system required the user to open up a Java based client application and enter their login details, alongside SRB information. This result enabled the user to use the separate web based application. The introduction of authentication at the portlet level, allows for the use of a single web application, consisting of both authentication and GENeSIS service utilisation. This enhances usability by reducing something as simple as the number of button clicks. The next item that helps with a user’s satisfaction with the system is the process in retrieving a SARoNGS certificate. The existing system requires the user to have a long-lasting X.509 certificate stored on their local machine to be able to request an NGS account to access the SRB. The unwieldy nature of taking care of certificates could be technically overwhelming for a standard user and so the new system brings an advantage by only generating certificates.
as and when the user requires them. Related to this is the mechanism for getting an NGS account, which is a time-consuming process that can take days or weeks to complete, with the user having to do much of the work to initially get a certificate which can then be used to request an account. With the new SARoNGS implementation, it is just a matter of logging in through Shibboleth then completing the NGS account form. This is undoubtedly a quicker process, increasing the start-up time for a user to begin using the GEneSIS for their benefit. Another help for usability comes from an additional feature that was implemented regarding retrieving files. The original application allowed for one file at a time to be retrieved from the SRB; the new implementation provides a multi-select portlet, allowing for the download of multiple files at once, increasing speed and therefore usability.

The points made above suggest evidence that usability has been increased with regards to GEneSIS and its authentication method. However, it is not without some flaws. There is a problem with SARoNGS when a user uses it to create an NGS account. Currently, SRB which uses the CN (Common Name) of the DN of the NGS account cannot handle the length of this CN. This requires the user to request a different SRB username which is linked to the generated DN. This does add extra complication to the user’s process and could seem confusing. However, as mentioned previously, SARoNGS is an emerging technology and as with all things new, there are bound to be bugs and so in the near future, it is hoped the problem is rectified by the SARoNGS group. The final item that could cause reduced satisfaction is that of the use of the sessions and their link with the login facility. As sessions are utilised to keep state and store the MyProxy values to enable the retrieval of a credential, each time the web browser is closed, the values (state) is lost. Therefore, each new GEneSIS instance will require the user to login. The numerous logins that could occur from a single user could be of annoyance, however, hopefully the process will become quickly familiar.

5.2.3 Security

Security as an evaluation criterion has been explained in the previous section of SARoNGS evaluation. This segment will focus on security within the developed solution, rather than SARoNGS itself.

As mentioned previously, SARoNGS has many security advantages over the existing mechanism of username/password, including the use of short-lived credentials instead of long-term X.509 certificates and the introduction of the ability to provide authorisation as well as authentication as a security feature. However, with regards the security as part of GEneSIS there are other advantages. Firstly, the existing system incorporated a process which required the username and password to be passed in a plain-text
document, which creates obvious security issues. The new implementation ensures that this does not happen anymore, by using credentials. The credentials are the only items that are passed, and they themselves are encrypted. The credential can only be decrypted by the key of the SARoNGS CA, residing on the NGS side. This is an obvious increase in security over the previous mechanism.

There are however security issues existing with the implementation. The web service used allows for the retrieval of a credential, decoupled from the developed application and from SARoNGS itself. This helps increase portability and reusability of the code, helping other projects to utilise the work that has been undertaken. The web service returns the encrypted credential which from a security point of view is fine, however the problem emanates from the input parameters. The parameters include the MyProxy login password value. This value, even though it is only ‘active’ for a short-period of time, could cause security issues if an untrusted third-party gains access to it. The password is sent as clear text in a SOAP message to the service host and therefore could be intercepted. To solve this issue, if time would have permitted, WS-Security could have been utilised which would have encrypted the SOAP message; this is discussed more in future extensions. Another issue that is related to both the SARoNGS mechanism and its utilisation in this project is an issue that was found during testing. When a user logs in to SARoNGS and generated a credential, SARoNGS allows for anybody who has authenticated with their local institution to generate a credential. The security implications are not too severe as the credential cannot be utilised within the context of GENeSIS as they would need an NGS and related SRB account. However, it is conceivable that the credential is used elsewhere to authenticate with a different grid resource. A sensible solution would be to use the VO information of a user to decipher whether they are able to generate a credential in the first instance.

5.2.4 Portability

The development of a credential web service has shown how sections of the authentication can be ported to different applications that requires authentication. The web service developed allows for the generation of a credential stored on a MyProxy server. The credential itself does not need to be coupled with SARoNGS, allowing for any type of credential to be returned from a MyProxy server, creating the possibility for the service to be used by multiple applications

SARoNGS has been demonstrated within the GENeSIS environment; however the authentication procedure is not bound to GENeSIS. The only requirement of the authentication procedure is that the Grid resource can be authenticated against using a certificate. This therefore has the potential for
SARoNGS to be used in its entirety by different systems requiring a method of authenticating with a Grid resource.

### 5.2.5 Summary

In summary, the solution provided has resolved both the requirements and the enhancements. In addition, the solution also provides additional advantages, including:

- A single point of entry for authentication, replacing two separate applications
- A familiarity using a local institution authentication method
- A simple, but advanced form of authentication, without the need to handle long-term certificates
- Passwords do not need to be communicated around in human-readable documents anymore
- An increase in portability with the introduction of a web service

Issues do exist however, including:

- The modern technology of SARoNGS does have challenges that need resolving
- Additional security challenges, including unencrypted SOAP messages

### 5.3 Future Extensions

This final section will take into account the issues raised throughout the testing, iteration evaluation and system evaluation sections to suggest future extensions to the application that, had time and scope permitted, would have been implemented to help improve the delivered solution.

#### 5.3.1 Backup Authentication Method

One issue that has been raised is that of the reliance on a third-party to ensure reliability and availability. The NGS provides the hardware and software for the utilisation of SARoNGS and therefore if an element fails, like the Credential Translation Service, then authentication via SARoNGS will not be possible. A possible solution to this would be to incorporate a backup authentication method that could be used if SARoNGS was unavailable.

There are two different possibilities for the choice of backup implementation, both have advantages and disadvantages. The first choice would be to use the current mechanism of username and password; however, as discussed throughout, this incorporates numerous security and usability disadvantages. To
help alleviate the usability issues, the current username/password interface could be incorporated into GEneSIS as a portlet, creating a more user-friendly platform. This would still leave the obvious security issues however.

Conversely, some of the security issues could be solved by utilising a MyProxy or VOMS only system. This would require the user to retrieve a long-term X.509 certificate from the NGS, and then use it to spawn proxies which could be used to authenticate the user with the SRB. The use of a certificate over username/password helps improve security with the aspect of incorporating authorisation in the form of VO memberships. However, they are less user friendly than username/password and have their own security issues with the user having to ensure that the certificate is stored safely and securely.

5.3.2 Enable Authorisation

Authorisation was demonstrated as part of the project, however without GEneSIS being VOMS-aware, it was not possible to fully incorporate user authorisation alongside authentication. It can therefore be seen that utilising a VOMS server to store a GEneSIS VO would be a sensible future step.

To incorporate authorisation, a VOMS server should be setup, containing a VOMS database. Within the database, a list of users along with their group and roles would be stored, which would need to be maintained by the GEneSIS team. A user could then use the WVOAYF form of SARoNGS to specify the location of the VOMS server to retrieve the Attribute Certificate that would then be embedded within the SARoNGS credential. The GEneSIS application could then utilise the authorisation code written during this project to make decisions based on a user’s particular access right.

5.3.3 Security

Two possible improvements come from issues raised during evaluation. The first item regards the transparency and security issues of the return URL from SARoNGS. This is the URL containing the MyProxy values, especially the password. There is a risk that the password is human-readable and a future enhancement could mitigate this.

There are two possible extensions which have been described earlier and are not mutually exclusive. The first possible improvement would be to incorporate HTTPS rather than the standard HTTP. This could be forced server side to only return the URL to an HTTPS location. However, it would be more convenient to make the change client side and therefore not really upon a third-party. The second extension would be to encrypt the password value in the URL. The unauthorised access of the port, host
and username values would only be seen as minor security risks as they are not meant to be private, however encrypting the real value of the password would be advantageous. To perform this, when the URL is sent back to the client, code in the GENeSIS client could intercept the URL before being displayed to the user. At this stage the value could be encrypted by a passphrase before being passed onto the client. The passphrase itself would be available in the application code behind and therefore hidden from the user. The passphrase could then be used to decrypt the password when it is being sent back to the MyProxy server to retrieve the credential.

Another security issue is that of the MyProxy password being sent in the SOAP request to the web service that responds with an encrypted credential. The way to solve this would be to utilise WS-Security [12], which would allow for parts of the SOAP message to be encrypted, i.e., just encrypt the MyProxy password. Specific SOAP headers should then be incorporated into the message to outline the properties of who is allowed to decrypt the message, ensuring only the service host can decrypt the password.

5.4 Conclusion

The solution has shown how SARoNGS can be used for user authentication as an alternative and enhancement to username and password. The solution also shows a use of SARoNGS within an e-Social Science environment and how it can be utilised for increased security and usability.

The project itself has utilised the extensive background research to collate a selection of possible mechanisms, which were then evaluated to select the optimum option for implementing a successful solution. The outlined iterative methodology then helped to incrementally deliver the authentication solution, which met all requirements and specified enhancements to allow for successful incorporation into GENeSIS.
Bibliography


[33] NGS. Who we are and what we do. http://www.ngs.ac.uk/aboutUs [Accessed 21-Feb-2010].


Chapter 6

Appendix A - Personal Reflection

Overall, I believe the project was extremely successful and as a result it was enjoyable to complete. In the first instance of discussing what the project would entail, I was unsure of the project suitability as my knowledge of computer security and authentication was limited. Security is no longer taught as a module in the School of Computing and therefore my exposure to security was limited to parts of modules and self-taught knowledge. However, after completing the module I have found the topic of security interesting, especially with its constant development in attempting to stay one-step ahead.

One thing I found difficult to start off with was getting my head around the different methodologies and technologies available. There are so many implementations, many of which do similar things, and so it was confusing at times to grasp concepts. Things started ‘sinking-in’ once I got programming and adding a practical aspect to the theoretical work. The more practical aspect seemed to match my way of working and helped with understanding. My advice to any future students researching security would be to start almost from day one, just experimenting with technologies to gain a grasp, more quickly.

Once the programming had got underway, the enthusiasm and drive increased as there was a tangible benefit to the research that had been done. It was possible to see how the work being done would help a large project like GENeSIS, and help improve it for the present and for the future. The reward at the end was being able to see what I had developed being utilised within GENeSIS, replacing and enhancing the existing authentication mechanism. Another moment of great enjoyment was once I realised that the work I was doing was completely unique. A problem that had to be solved by the National Grid Service showed that no-one else has attempted such a project, and that provided a point of satisfaction that the mechanism being developed was somewhat pioneering.
One main issue found during the project was the reliance of third-parties. The use of SARoNGS required the reliance on NGS tools and support. There were instances where my developed application would simply not work because of a fault at the NGS side, and therefore there was a delay in implementing until the problem had been solved. Advice to future students would be use third-parties frugally, and attempt to gain a contact immediately as the project starts, so you have a point of contact throughout the project.

Other issues did occur during the project that future students could learn from. Firstly, the original schedule was somewhat ambitious with regards to implementation times. As can be seen by the schedule, compared to the mid-project report, the timings for the implementations are different. The first iteration took longer than expected because of the initial start-up of setting up the environment and getting all the APIs and tools necessary to perform the coding. Also, iterations one and two overlapped as there was work in two that required the completion of one, so whilst I was waiting for issues to be resolved I began the work on iteration two. The extended time for two also meant less time was spent on iteration three, however this is not a huge issue and development is in full-flow by then, and there were no issues with hardware at that stage. My advice would be to not treat the schedule as the be-all and end-all, just as a guideline and also maybe assign a day or two in week 2 or 3 to set up the programming environment.

Finally, the last piece of advice regards the report writing. I have found it a struggle in the last weeks to finish the writing of the project as I underestimated the time necessary to compile all my work. I originally wrote the work in Word and the transferred across to LaTeX, which reading back now sounds
rather silly. It was a difficult putting the work into LaTeX and so I would advise students to do all work in LaTeX; it may take slightly longer but the rewards will be worth it. Also, I would advise that sections of the project are not written in blocks over time, as it is then difficult to collate the work together and make the writing flow from section to section.
Appendix B - Test Results

7.1 End of First Iteration Testing

To test the software delivered after iteration one, the following test sequences were performed. The result of the test sequence would allow for a conclusion to be drawn on whether the iteration’s aim ‘programmatically authenticate a user to the SRB using SARoNGS’ has been successful.

Authenticate Valid User
This sequence will attempt to test that a valid user can authenticate themselves programmatically and store an SRB file locally. A valid user is seen as a client of the GENeSIS system who has sufficient rights to login via Shibboleth through their local institution. They must also have an NGS account, created through the SARoNGS NGS account application process, which provides the user with an SRB account, which can be used to access files for use within GENeSIS.
<table>
<thead>
<tr>
<th>Step</th>
<th>Task</th>
<th>Expected Outcome</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Login via SARoNGS to manually retrieve MyProxy values and input values into getProxyCred() method</td>
<td>Successful login via SARoNGS. Values retrieved and inputted</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>Use getProxyCred() to retrieve GSSCredential</td>
<td>Instance of GSSCredential created</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>GSSCredential used to authenticate with SRB</td>
<td>SRBFileSystem object successfully created</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Copy file from SRB to local file system</td>
<td>File stored locally</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Prevent Invalid User from Authenticating #1**

In this scenario, the user can authenticate with Shibboleth, but doesn’t have an NGS account

<table>
<thead>
<tr>
<th>Step</th>
<th>Task</th>
<th>Expected Outcome</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Login via SARoNGS</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>a</td>
<td>Shibboleth using local institution authentication</td>
<td>Successful login taken to WVOAYF page</td>
<td>Yes</td>
</tr>
<tr>
<td>b</td>
<td>Complete SARoNGS login from WVOAYF page</td>
<td>Unable to create credential and therefore no MyProxy values</td>
<td>No</td>
</tr>
</tbody>
</table>

Step b was unsuccessful as SARoNGS allows for the creation and upload of a credential, irrespective of whether the user has an NGS account. As step b was unsuccessful, extra steps were added:

<table>
<thead>
<tr>
<th>Step</th>
<th>Task</th>
<th>Expected Outcome</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Use getProxyCred() to retrieve GSS-Credential</td>
<td>Instance of GSSCredential created</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>GSSCredential used to authenticate with SRB</td>
<td>Unable to authenticate due to DN having no mapping to NGS account</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Prevent Invalid User from Authenticating #2**

In this scenario, the user can’t authenticate with Shibboleth and doesn’t have an NGS account

<table>
<thead>
<tr>
<th>Step</th>
<th>Task</th>
<th>Expected Outcome</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Login via SARoNGS</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>a</td>
<td>Shibboleth login using local institution authentication</td>
<td>Unable to login</td>
<td>Yes</td>
</tr>
</tbody>
</table>
7.2 End of Second Iteration Testing

The testing for iteration two is split into two test processes, one is to use the user interface to retrieve a file from the SRB and the second is to ensure the time-to-live functionality. The outcome of the testing will allow for the realisation of whether the iteration’s aim was realised.

Retrieve SRB File Dynamically via user interface

This test ensures that the portlets create a suitable interface for a user to login and download an SRB file.

<table>
<thead>
<tr>
<th>Step</th>
<th>Task</th>
<th>Expected Outcome</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Use Login Portlet to login via SARoNGS</td>
<td>Login is successful and the user is redirected back to portlet page. Login Portlet is updated with the button removed and a success message displayed</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>Grab Get File portlet and add to working area</td>
<td>List box displays all SRB files</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>Select multiple items</td>
<td>Multiple items can be selected</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Download selected items</td>
<td>Items downloaded to local system</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Dealing with Time-to-Live

Test that the values in the session persist for the length of the session.

<table>
<thead>
<tr>
<th>Step</th>
<th>Task</th>
<th>Expected Outcome</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Open instance of Liferay</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>Use Login Portlet to login via SARoNGS</td>
<td>Login is successful and the user is redirected back to portlet page. Login Portlet is updated with the button removed and a success message displayed</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>Open a second browser window</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>Point the browser to the location of the original instance of Liferay</td>
<td>New browser window will display Login Portlet with successful message displayed and no Login button</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Test that the values expire when the session expires.
<table>
<thead>
<tr>
<th>Step</th>
<th>Task</th>
<th>Expected Outcome</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Open instance of Liferay</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>Use Login Portlet to login via SARoNGS</td>
<td>Login is successful and the user is redirected back to portlet page. Login Portlet is updated with the button removed and a success message displayed</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>Close the Liferay instance (browser)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>Open a new instance of Liferay</td>
<td>Login Portlet will revert back to displaying the Login button</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Test that expired credentials cannot be used, which, instead of waiting for a credential to expire, will force the credential to expire by creating a new one

<table>
<thead>
<tr>
<th>Step</th>
<th>Task</th>
<th>Expected Outcome</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Open instance of Liferay</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>Use Login Portlet to login via SARoNGS</td>
<td>Login is successful and the user is redirected back to portlet page. Login Portlet is updated with the button removed and a success message displayed</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>Open a new browser window and point the URL to the NGS SARoNGS login page</td>
<td>SARoNGS login page will display</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Perform the standard, manual SARoNGS login</td>
<td>A credential will be created and uploaded, overwriting the original</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>Return to instance of Liferay and drag Get File portlet in</td>
<td>SRB files will not show as the original credential has expired A message should be displayed to the user to explain this</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Re-use the Login Portlet to login via SARoNGS</td>
<td>User able to login through the portlet</td>
<td>No</td>
</tr>
</tbody>
</table>

In step 6 the SRB files were not shown, and a message was displayed as in step 5, however as the session values still persisted, the user cannot use the Login Portlet again (without closing down the instance of Liferay). To resolve this issue, the code behind the Get File portlet was altered to catch
MyProxyException’s, which would allow for the capture of problems due to expired credentials. At the point of an error being caught, the MyProxy values are removed from the session. A message is the displayed to the user asking them to refresh their browser; this will force the Login Portlet to check for MyProxy values in the session, at which point it will see that there are none and therefore display the 'Login’ button.

### 7.3 End of Third Iteration Testing

<table>
<thead>
<tr>
<th>Step</th>
<th>Task</th>
<th>Expected Outcome</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Open instance of Liferay</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>Use Login Portlet to login via SARoNGS</td>
<td>Login is successful and the user is redirected back to portlet page. Login Portlet is updated with the button removed and a success message displayed</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>Drag the WebServiceAuth portlet into the working area</td>
<td>WebServiceAuth portlet is displayed, showing the 'Authenticate’ button</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Click 'Authenticate’</td>
<td>Web Service is called and returns a successful message if authentication has been successful</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**User Authorisation Test**

The testing performed in this section looks at user authorisation and is performed using business-logic, no user interface. This is because when GENeSIS transitions to utilising different roles and rights, there will be no specific separate user interface portlet, it will however require the extra functionality in the code-behind, which is what this deliverable is demonstrating.

The intention of this test is to ensure the application can deal with different credentials, in terms of those that contain VOMS information and those that do not.
### Task 1
Login via SARoNGS to manually retrieve MyProxy values. Select no VO at the WVOAYF page

Successful login via SARoNGS and MyProxy values retrieved

Yes

### Task 2
Add MyProxy values as the arguments of the call to the web service

Username, password, host and port added as arguments

Yes

### Task 3
Run application

Application will return access denied, no VOMS certificate attached

Yes

This second set of tests look at extracting VOMS information and dealing with different VO and roles.

For these tests, it is assumed that only a member of the Leeds.ac.uk group within the UK Federation can have access to the SRB File System.

### Step 1
Login via SARoNGS to manually retrieve MyProxy values. Select Leeds.ac.uk group, role is 'NULL'

Successful login via SARoNGS and MyProxy values retrieved

Yes

### Step 2
Add MyProxy values as the arguments of the call to the web service

Username, password, host and port added as arguments

Yes

### Step 3
Run application

Application will return access denied, role of null is not sufficient to gain access

Yes

Repeat test, changing VO role to 'Member'

### Step 1
Login via SARoNGS to manually retrieve MyProxy values. Select Leeds.ac.uk group, role is 'Member'

Successful login via SARoNGS and MyProxy values retrieved

Yes

### Step 2
Add MyProxy values as the arguments of the call to the web service

Username, password, host and port added as arguments

Yes

### Step 3
Run application

Application will return success message, role of Member is sufficient to gain access

Yes
Chapter 8

Appendix C - Diagrams

Figure 8.1: GENeSIS User Interface Interaction

NB. It is assumed that the .genesis files already exist on the SRB before the process is initiated. The creation and storage of the .genesis files is outside the scope of the project

a) The user opens the desktop tool, specifying the SRB details to allow them access to the .genesis files held on the SRB
b) A user-specified .genesis file is retrieved from the SRB and sent to the user
c) The user opens the web application and uploads the file. The user then selects an action to perform on the data
d) The result of the action is displayed
Figure 8.2: SARoNGS Process. Where Are You From? (WAYF)

Figure 8.3: SARoNGS Process. Local Authentication
Figure 8.4: SARoNGS Process. SARoNGS Account Registration

Figure 8.5: SARoNGS Process. Which VO Are You From? (WVOAYF)
Figure 8.6: SARoNGS Process. Credential Upload

Figure 8.7: Application. SARoNGS Login Process

Figure 8.8: Web Service. Hosting test page

Figure 8.9: Web Service. Client code to call web service