

DG Home Affairs: Prevention of and Fight against Crime  
ISEC Programme

CENTER OF EXCELLENCE



A CyberCrime Center of Excellence for Training, Research and  
Education in Greece

### **Deliverable D3.4: Report on a Secure Repository for Education Material**

**Abstract:** This document discusses a study on designing and creating a *pilot private repository* for delivering educational material among *law enforcement agencies (LEAs)* in a *safe* and *secure* manner. We focus on setting out the requirements for a secure and sustainable private repository for the Greek Cybercrime Center (GCC).

Contractual Date of Delivery	January 2014
Actual Date of Delivery	January 2014
Deliverable Dissemination Level	Public
Editors	Antonis Krithinakis
QMC Reviewer	Ioannis Iglezakis



---

The Greek Cybercrime Center consortium consists of:

FORTH-ICS	Institute of Computer Science of the Foundation for Research and Technology - Hellas	Coordinator	Greece
SAFENET	Hellenic Self Regulatory Body for Internet Content	Principal Contractor	Greece
KEMEA	Center for Security Studies	Principal Contractor	Greece
AUTH	Aristotle University of Thessaloniki	Principal Contractor	Greece



*With the financial support of the Prevention of and Fight against Crime Programme  
European Commission – Directorate-General Home Affairs*

*This project has been funded with support from the European Commission. This publication  
reflects the views only of the author, and the European Commission cannot be held responsible  
for any use which may be made of the information contained therein.*





## Contents

<b>1</b>	<b>Introduction</b>	<b>9</b>
1.1	Document outline . . . . .	9
<b>2</b>	<b>Digital Repositories</b>	<b>11</b>
2.1	Definition . . . . .	11
2.2	GCC Public Repository . . . . .	11
2.2.1	Open eClass Overview . . . . .	12
2.2.2	Basic Characteristics . . . . .	13
2.2.3	Functionality & Features . . . . .	15
2.3	GCC Private Repository . . . . .	18
<b>3</b>	<b>Threat Model</b>	<b>21</b>
3.1	Terminology . . . . .	21
3.2	Identify Assets . . . . .	22
3.3	Data Loss . . . . .	22
3.3.1	Natural disasters . . . . .	22
3.3.2	Power failure . . . . .	22
3.3.3	Hardware malfunction . . . . .	22
3.3.4	Unintentional deletion . . . . .	22
3.4	Malicious Users . . . . .	23
3.4.1	Social engineering . . . . .	23
3.4.2	Phishing . . . . .	23
3.4.3	Host compromisation . . . . .	23
3.4.4	Server compromisation . . . . .	23
3.4.5	Device theft . . . . .	24

---

<b>4</b>	<b>Securing The Repository</b>	<b>25</b>
4.1	Network Level Security . . . . .	25
4.1.1	Firewalls . . . . .	25
4.1.2	Network Intrusion Detection Systems . . . . .	26
4.1.3	VPN accounts . . . . .	26
4.2	Host Level Security . . . . .	27
4.2.1	Hardware Layer . . . . .	27
4.2.2	Software Layer . . . . .	27
4.3	Application Level Security . . . . .	28
4.3.1	SSL Certificates . . . . .	28
4.3.2	Two-factor authentication . . . . .	29
4.4	Content Level Security . . . . .	30
<b>5</b>	<b>Private Repository Pilot Deployment</b>	<b>33</b>
5.1	Overview . . . . .	33
5.2	Features . . . . .	33
5.3	Installation . . . . .	34
5.4	SSL configuration . . . . .	38
5.5	One Time Passwords configuration . . . . .	40
<b>6</b>	<b>Conclusion</b>	<b>45</b>

## List of Figures

2.1	GCC Public Repository Home page. . . . .	12
2.2	Administrator area. . . . .	14
2.3	Teacher area. . . . .	14
2.4	Student area. . . . .	15
2.5	Lesson access type. . . . .	15
2.6	Agenda section. . . . .	16
2.7	Documents section. . . . .	16
2.8	Announcements section. . . . .	17
2.9	Links section. . . . .	17
2.10	Forum section. . . . .	18
5.1	ownCloud first time configuration. . . . .	35
5.2	ownCloud login page. . . . .	36
5.3	ownCloud default contents. . . . .	36
5.4	ownCloud documents example. . . . .	37
5.5	ownCloud internal PDF viewer. . . . .	37
5.6	ownCloud photo gallery. . . . .	38
5.7	OpenSSL output. . . . .	39
5.8	Google Authenticator on Apple store. . . . .	42
5.9	Google Authenticator start page. . . . .	42
5.10	Tap the plus icon. . . . .	42
5.11	Select the barcode option. . . . .	42
5.12	Server-side configuration for user <i>krithin</i> . . . . .	43
5.13	Google Authenticator password example. . . . .	43
5.14	GCC private repository sign in process. . . . .	44

## LIST OF FIGURES

---

This document is focused on the design of a **secure private repository** which is the main objective of Task WP3.4 of the Greek Cybercrime Center (GCC) project. The fourth task of the *Education and Training Workpackage* (WP3) has the objective to study the ways of securing a repository which would be accessed only by **law enforcement agencies (LEAs)** in a safe and secure manner. The repository could support the secure delivery of educational material and confidential documents.

The repository should be immune to data loss caused by unexpected natural disasters, hardware malfunctions or power outages. Moreover, the network environment and the hosting machine should be appropriately configured to detect and prevent unauthorized access by malicious users.

## 1.1 Document outline

In the following Chapters, we describe our study on designing the GCC private repository. In Chapter 2 (page 11) we initially define the concept of digital repositories, then present information about the GCC public repository and finally list some generic requirements that the private repository should fulfil.

Next, in Chapter 3 (page 21) we define a threat model by describing all possible threats that our repository should deal with in order to keep the material always available and protected against unauthorized access.

In Chapter 4 (page 25) we detail the study of securing the repository in all possible layers. We provide appropriate countermeasures and security controls in *network*, *host*, *application* and *content* level.

In Chapter 5 (page 33) we deploy a beta version of the GCC private repository based on the *ownCloud* [29] platform. We provide technical information of the installation process and the establishment of security mechanisms.

Finally, in Chapter 6 (page 45) we summarize our study.

## 2.1 Definition

A digital repository is a type of information retrieval system which comes with a collection of mechanisms for managing and storing any kind of digital content. The repository should be a safe place to store digital files in order to preserve their integrity and share the knowledge among staff and institutes.

The digital repository should also provide some basic functionality for file importing, exporting, storing and retrieving. The latter presupposes that a search interface, which easily allows resources to be found, is included as well. Digital resources can be stored locally on a single machine, or across multiple machines, in most cases, geographically distributed. In any case, resources should be accessed remotely via computer networks.

Digital repositories have several advantages over physical libraries. Firstly, the availability to individuals is increased because no physical and geographical boundaries exist. Any authorized user can have access to the digital content as long as the repository is accessible via Internet or VPN [46]. Secondly, the digital content is available at any time and multiple users can have access to it simultaneously. Another advantage is that users can easily retrieve any document by searching the entire collection in short time, by using the search interface. Finally, digital repositories encourage the digitization of content which helps in preserving and conserving physical resources.

## 2.2 GCC Public Repository

The third task of the *Education and Training Workpackage* (WP3) of the GCC project focuses on creating a repository to deliver educational material to the *public*. The main objectives of this Workpackage include the conducting of high quality training courses and interdisciplinary University courses, in

the area of cyber crime investigation, targeting both law students as well as computer scientists and engineers. We expect that for each course developed in GCC, the public repository will act as an “*educator’s portal*” where educators, trainees and students will be able to receive education material and support. We expect all Greek students to benefit from this task and also to promote this approach to a European level as well.

During the first year of the GCC project we deployed a couple of open source *learning content management systems* (LCMS) [19] in order to select the most appropriate for the requirements of the project. Among others, the consortium chose the **Open eClass** [27] open source platform.

The main arguments in favour of this selection was the ease of use and the adequate functionality it provides for conducting digital training. Moreover, this platform is offered by the *Greek Academic Network* (GUnet) [14] and is widely used by Greek Universities. A list of the active and official installations of Open eClass in Greece is available online<sup>1</sup>. The web front-end of the GCC public repository can be seen on Figure 2.1.



Figure 2.1: GCC Public Repository Home page.

### 2.2.1 Open eClass Overview

The Open eClass platform has been designed to supplement and enhance the traditional teaching process. It supports the electronic management, storage and presentation of teaching materials, transcending limitations of space and time and thus creates the necessary conditions for a dynamic teaching environment.

<sup>1</sup>The list of Greek official installations of Open eClass: <http://www.openeclass.org/content/view/19/40/lang,en/>



In addition, Open eClass is available as an open source project. Open source solutions are free to use and have benefits such as visibility of code, customization and adaptation to specific user needs.

The main advantages of Open eClass are:

- **Ease of use.** Open eClass is oriented for users without specialized technical skills. All the necessary functionality for creating a teaching environment and delivering the educational material to students is easily provided by a user-friendly interface.
- **Ease of administration.** Open eClass is very easy to install, configure and update. *PHP* [33] is used as the scripting language and *MySQL* [22] as the database backend. In our installation, Open eClass is hosted on a Linux [41] running machine and is served by the Apache [5] web-hosting server. During the installation process, the administrator completes most of the basic configuration. After completing installation, the administrator can easily create and administer user accounts and courses and monitor the server and database operations.
- **Bilingual support.** The web interface of the Open eClass platform is delivered both in Greek and English language.

### 2.2.2 Basic Characteristics

- **User roles.** Open eClass provides three distinct user roles. The first role is the **Administrator** who has general control over the platform. Administrators basically create and manage user accounts (see Figure 2.2).

The second role is the **Teacher** who is responsible for the creation and administration of electronic courses. He can create an unlimited number of courses, contact student users registered to them, upload educational materials and create discussion forums where course participants can interact (see Figure 2.3).

Finally, the third role is the **Student**. A student can register to courses making an open registration, access educational materials, and participate in working groups, discussion forums and exercises (see Figure 2.4). Student accounts can be created either automatically, by allowing open registration of new users, or by the platform administrators themselves after an on-line request.

## CHAPTER 2. DIGITAL REPOSITORIES

User: Platform Administrator of platform, Logout English

**GREEK CYBERCRIME CENTER**

User portfolio

**Admin Options**

- » Admin Tool

**Basic Options**

- » Courses List
- » Available Manuals
- » Platform Identity
- » Contact

**User Options**

- » Create Course
- » My Calendar
- » Modify my Profile
- » User Statistics

**User portfolio**

My courses (teachers)	Teacher	Administer
» Introduction to Botnet Detection Tools (GCC101)	Greek Cybercrime Center	

**GU net** Copyright ©2003-2011 GUnet POWERED BY OPENeCLASS

Figure 2.2: Administrator area.

User: Νίκος Παπαδόκης, Logout English

**GREEK CYBERCRIME CENTER**

User portfolio

**Basic Options**

- » Courses List
- » Available Manuals
- » Platform Identity
- » Contact

**User Options**

- » Create Course
- » My Calendar
- » Modify my Profile
- » User Statistics

**User portfolio**

My courses (teachers)	Teacher	Administer
» Introduction to Cybercrime (GCC102)	Νίκος Παπαδόκης	

**GU net** Copyright ©2003-2011 GUnet POWERED BY OPENeCLASS

Figure 2.3: Teacher area.

## 2.2. GCC PUBLIC REPOSITORY

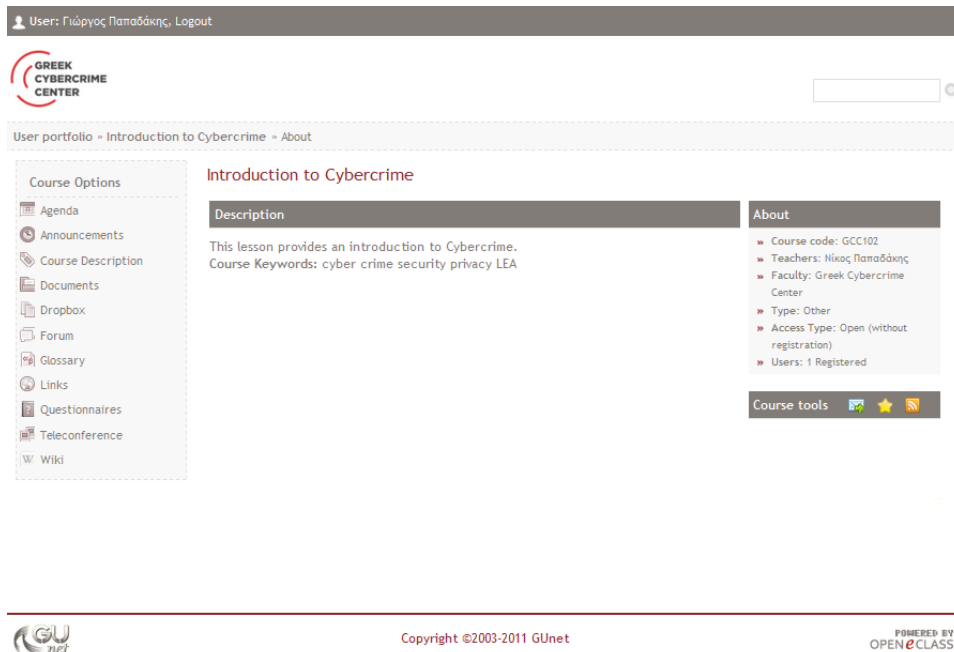


Figure 2.4: Student area.

- **Course categories.** Open eClass also provides distinct course categories. A course can be **Open**, **Restricted** or **Closed**. Open courses are publicly accessible, even by visitors without a user account. Registration based restricted courses are accessible to platform users that are registered to them. This means that these courses are open to everyone having an account. Finally, closed courses are accessible to users only after a course teacher allows access to their account by registering them to the course.

Figure 2.5: Lesson access type.

### 2.2.3 Functionality & Features

The core part of the Open eClass platform is the **eCourse** entity which represents a physical course. Each digital course of Open eClass is comprised

## CHAPTER 2. DIGITAL REPOSITORIES

of learnings tools-components which are maintained by teachers and used by students. The most basic features are:

- **Agenda:** presents important events such as lectures and meetings in a time order.

### Introduction to Cybercrime

Agenda (Inactivation)

		Add an event
Events		Actions
Calendar December 2013		
Today: 16-12-2013 / 22:24		
Thu December 19, 2013 (hour: 13:30)		  
Teleconference for resolving any outstanding issues regarding course slides (Duration: 1 hour) Tele will be streamed through OpenEclass platform.		

Figure 2.6: Agenda section.

- **Documents:** contains all digital material available to students. The platform supports several types of material such as documents, presentations, images and videos. Students have the opportunity to view and download recorded videos of lectures.

### Introduction to Cybercrime

Documents (Inactivation)



		Upload file   Create directory   View available disk space		
Folder: Root directory				
Type	First Name	Size	Date	Actions
	Botnet detection This is a tutorial on How to Detect Bot-infected Machines	4.96 MB	2013-12-16 22:12:54	     

Figure 2.7: Documents section.

- **Announcements:** this section informs students about course conduction and important events. Announcements can be optionally received through emails.
- **Links:** a list of links to external web resources is available.

## 2.2. GCC PUBLIC REPOSITORY

### Introduction to Cybercrime

Information workshop   (Inactivation)

 Add announcement

Announcement	Actions
<p>» <b>Information workshop</b> Monday December 16, 2013</p> <p>The Center of Security Studies (KE.ME.A.) of the Ministry of Public Order and Citizen Protection in collaboration with DG Enterprise and Industry, and the PRAXI network, the National Contact Point for Horizon 2020 Programme, and the support of the General Secretariat of Research and Technology (G.S.R.T.) of the Ministry of Education and Religious Affairs organizes on Wednesday 11 December 2013 (13:30 - 19:00) and Thursday 12 December 2013 (09:00 - 13:30), an information workshop related to:</p> <p>The EU Framework Programme for Research and Innovation on "Secure societies - Protecting freedom and security of Europe and its citizens" and "Space" themes</p> <p>The event will take place in <b>Royal Olympic Hotel</b>, Athanasiou, Diakou 28-34, Athens, Greece.</p>	  

Figure 2.8: Announcements section.

### Introduction to Cybercrime

Links  (Inactivation)

 Add link | Add category

 Links

<p>» <b>Greek Cybercrime Center (GCC)</b></p> <p>The Greek Cybercrime Center (GCC) is part of an emerging coordinated European effort which has the capacity to significantly improve education and research in the newly growing area of cybercrime. As a national project, GCC seamlessly complements transnational projects such as <b>2CENTRE</b> (The Cybercrime Centres of Excellence Network), and <b>B-CENTRE</b>.</p>	 
--	---

Figure 2.9: Links section.

- **Forum:** teachers can create forums where students can participate in group conversations about the course.
- **Teleconference:** Open eClass supports real-time video streaming and messaging between teachers and students.
- **Wiki:** students can use wikis in order to collaborate on editing documents. A full history of changes is available to keep track of them.
- **Questionnaires:** which can be used for polls and learning profile surveys



Figure 2.10: Forum section.

## 2.3 GCC Private Repository

The fourth task of the *Education and Training Workpackage* (WP3) of the GCC project has the objective to study the design and the pilot deployment of a secure private repository. This repository can be used to store educational material and private documents accessed only by LEAs in a safe and secure manner.

The private repository, as a digital repository, should accommodate the following generic requirements:

- **Categorization.** The repository should be able to store and categorize any kind of digital material such as text documents, presentations, spreadsheets, images, audio and video files. Moreover, the feature of creating folders is needed in order to better organize files in groups. Finally, built-in viewers for *open document format* (ODF) [26] files and image galleries would be a great improvement to user experience.
- **Searching.** As we discussed in Section 2.1, the repository should provide a user-friendly search interface, giving fast retrieval capability to users. Ideally, the search engine should provide searching both by name and by content.
- **Universal access.** Authorized users should be able to access the digital material via the web independently of their location.
- **Versioning.** An automated versioning system should be included to manage all changes to digital material and keep track of them. Versioning gives the ability to revert a document to a previous revision which is critical for correcting human mistakes.
- **Open source and API enabled.** An open source solution for the repository is required for many reasons. In general, there are many developers and users who are working to improve the security and technical features of the software. Moreover, the visibility of code make

us more confident while at the same time eliminates any undesirable hidden features. Finally, the open source software may be customized and adapted to meet specific user needs. The latter is also possible when an API is available for developing applications and plugins on the platform.

- **Security.** The final and most important requirement is the high level of security we desire to achieve. All previous requirements aimed at the ease of use and the improvement of user's experience. The main task of this deliverable is to secure the private repository against a set of threats which we define in detail in Chapter 3.





In this chapter we describe a *threat model* for the GCC private repository. Threat models can be defined by describing a class of possible attacks in a computer system. The model specifies what needs to be protected and from whom. Firstly, the model lists all valuable assets which worth protection. Then, the model identifies threats and attacks that may misuse the system and compromise the assets. Finally, the security designer should determine countermeasures and safeguards for reducing risks associated with these vulnerabilities.

### 3.1 Terminology

- **Asset.** An asset is any system resource (e.g. hardware, software) which worth protection. It is important for the security designer to optimize or balance the cost of protecting each asset against the cost of losing it.
- **Vulnerability.** A weakness in some aspect or feature of a system which allows an attacker to exploit it. Vulnerabilities might exist in several areas of the network, the host or an application.
- **Threat.** A malicious occurrence that is capable to exploit a vulnerability to breach security and thus cause possible harm.
- **Compromisation.** The act of intruding into or illegally controlling a computer system.
- **Countermeasure.** Any method used to prevent an exploit from successfully occurring and mitigate a computer system compromisation.

## 3.2 Identify Assets

In our case, we aim to protect the **digital material** from unauthorized access and from any violation on the predefined terms of use. In addition, we aim to prevent the **application server** which hosts the repository from being compromised and making unauthorized access on the data. Finally, we should consider cases where **benign users** get defrauded by malicious users.

## 3.3 Data Loss

The first threat we should protect against, is the loss of the digital material.

### 3.3.1 Natural disasters

Catastrophic events, such as earthquakes, floods and fires are likely to happen at any time and any place. In those situations where a natural disaster strikes the physical location of the application server or the storage devices, all data will be lost.

### 3.3.2 Power failure

The sudden outage of the electric power can cause data loss due to hardware damage. Even if hardware remains functional, a power failure could result in data stored in volatile memory not being saved to permanent memory.

### 3.3.3 Hardware malfunction

We consider the case where a hardware malfunction occurs on the server hosting the repository. The most usual example is a hard disk drive failure. This makes the stored information unable to be accessed. Unfortunately, the data restore process is usually ineffective.

### 3.3.4 Unintentional deletion

In this case, an authorized user or an administrator could accidentally delete files from the educational material or from the repository's installation directory tree.

## 3.4 Malicious Users

A malicious user is an individual or group of individuals who exploits vulnerabilities in computer systems. The attacker is usually motivated by profit, protest, or challenge [15].

Common attacks used by malicious users could be dangerous for the security of the repository and therefore for the integrity of our assets.

### 3.4.1 Social engineering

Malicious users, use social engineering tactics to scam benign users into unknowingly releasing personal information and revealing credentials. A malicious user who has successfully gain access to the private repository by using the victim's credentials can go through and even delete material. Generally, every permission on the repository granted to the victim could be maliciously used.

### 3.4.2 Phishing

Any fake website or scam e-mail message that seeks to acquire user credentials by masquerading as a trustworthy entity conducts a *phishing attack* [32]. Fake notifications from banks, e-pay systems and other providers are sent to users in order to encourage them to reply with personal information. Phishing emails sometimes contain links to sites which are infected with malware or look almost identical with the legitimate ones. When the user visits the phishing site and enters his personal credentials, the phisher has just got control on the account. Phishing attacks are an example of social engineering tactics used to deceive security unaware users.

### 3.4.3 Host compromisation

Malicious attackers are armed with exceptional programming skills which use to compromise a host. After a successful intrusion, the attacker can steal credentials and successfully authenticate to the repository.

### 3.4.4 Server compromisation

An attacker can choose to compromise the application server hosting the repository, instead of a benign user's workstation. Generally, servers are administered by professionals who ensure that adequate security measures are used. However, servers often provide multiple services across the local network or via Internet. This feature could allow the malicious user to query some services for vulnerabilities and successfully compromise the machine.

At this point the attacker could have acquired full access to the machine and thus to the digital material, especially when it is stored in plain text.

### **3.4.5 Device theft**

Any portable device such as smartphones, tablets, netbooks and laptops is likely to be stolen and lead to a potential disclosure of the information stored on it. Malicious users can gain access to the private repository material by stealing a computer device with stored passwords and access tokens.

In this chapter we propose appropriate and convenient security countermeasures to mitigate the class of threats described in Chapter 3. We opted to classify the security mechanisms into four main levels which are **network**, **host**, **application** and **content**.

## 4.1 Network Level Security

The network, in which the application server exists, is the first level we need to protect.

### 4.1.1 Firewalls

A standard practice in network security is the use of *Firewalls* [12]. The Firewall is a network security system, usually located at a network gateway server, that controls incoming and outgoing network traffic. Its main activity is to inspect traffic, filter all incoming connections to the network and discard packets which match a particular set of rules. Firewall rules are similar to access control lists (ACLs) [2] applied in filesystems and network elements such as routers and switches. However, modern firewalls operate in a more sophisticated manner, providing a connection state oriented analysis at most of the layers of the TCP/IP protocol stack.

As far as the private repository is concerned, we have created a set of rules for importing into the gateway firewall. These manually operated rules specify the set of IP addresses and network ports that are accepted for communication. This solution provides a robust mitigation against network attacks but sets up an important limitation: benign users should register specific network locations from where the repository will be accessed.

### 4.1.2 Network Intrusion Detection Systems

A *Network Intrusion Detection System* (NIDS) [23] is a detection system which analyses network traffic and attempts to discover signs of malicious activity. NIDS provides real-time network monitoring and when an attack is detected, it stores the event and passively informs the system administrator.

There are mainly two types of such systems, the *signature based* and the *anomaly based* [3] systems. Signature based NIDS such as *Snort* [37], *Suricata* [40] and *Bro* [7] attempt to identify attacks based on signatures in payload of packets. These signatures are known only after the malware is launched and propagated. This limitation makes zero-day attacks undetectable. However, anomaly based NIDS cope better with these kind of attacks as they classify packets based on heuristics or rules and attempt to detect any type of misuse that falls out of normal system operation. This technique has some short-comings, namely a high false positive rate and the ability to be fooled by a correctly delivered attack.

We use the latest stable release 2.9.5.6 of the *Snort* open source NIDS, which combines the benefits of signature and anomaly-based inspection. Our NIDS server is configured with two network cards one for management and the other for stealthy monitoring. All traffic to the private repository pass through Snort for monitoring. When a signature or an anomaly event is detected, the system produces an alert for the administrator and a rule for import into the firewall. That way we combine the packet inspection provided by traditional NIDS with the blocking capabilities of a firewall in order to create a *Network Intrusion Prevention System* (NIPS) [24]. Our NIPS can be also configured to automatically drop malicious packets and reset the connection.

### 4.1.3 VPN accounts

A *Virtual Private Network* (VPN) [46] extends a private organisation network across a public network, such as the Internet. It enables hosts to securely communicate over the public network by establishing virtual point-to-point connections. These connections are achieved by the use of tunneling protocols [42] and through security procedures such as encryption.

The most common way to set up a virtual private network is to enable a VPN server on the organization's network and a VPN client software on the user's desktop or laptop computer. This configuration sets up a one-way relationship, where the VPN server authenticates the VPN client attempting the connection. The VPN server should verify that the VPN client has all the appropriate permissions to use the network. Moreover, a mutual authentication scheme can be used where the VPN client also authenticates the VPN server, providing protection against masquerading attacks [38].

We have combined the VPN authentication scheme with the use of the NIDS server and the gateway firewall. We decided to place the VPN server behind the firewall, as it is a more secure approach. Our firewall is configured to allow tunneled data to pass to the VPN server. These packets have their payload encrypted and the firewall is not be able to filter them. However, this is not considered as a security breach because the authentication process of the VPN connection prevents any unauthorized access.

## 4.2 Host Level Security

The second place where security should be applied is the application server hosting the private repository. We should consider how to protect the repository integrity from hardware malfunctions and software level problems and mistakes. In Section 3.3 we discussed several risk factors that could be dangerous for the integrity of the private repository. We describe below the countermeasures we use for these factors.

### 4.2.1 Hardware Layer

We use an *uninterruptible power supply* (UPS) [44] to protect from power outages. UPS provides a constant flow of electricity in the event that a primary power supply becomes unavailable for a short period of time. In addition, to protect against surges (events where voltages increase for a few seconds), we use a special device called a surge protector that absorbs the excess voltage.

Moreover, we use *RAID* [35] technology to protect against hard disk drive failures. RAID is a storage technology which enable a storage unit to provide data redundancy and better performance. RAID helps to avoid data loss in case of a disk failure and provides an effective way for data recovering. In order to set up RAID in our server, we needed a group of hard disk drives and a specific hardware controller.

In order to strengthen more the protection of our data, we use a combination of the RAID technology and regular data backups. An external storage medium, without network access, is used to periodically store snapshots of the server's filesystem.

### 4.2.2 Software Layer

Data loss can also occur by software oriented mistakes such as system crashes and unintentional file deletion. The repository administrator or any authorized user could accidentally corrupt or delete files. To protect against such cases, we have enabled the regular backup strategy we previously discussed.

In addition, as a further enhancement, we opted for a filesystem which supports *journaling* [18]. A journaling filesystem is a fault-resilient filesystem where unsaved data can be restored in the case of a system crash. Such filesystems also ensure that data on the disk has been restored to its pre-crash configuration. We use the *third extended filesystem* (ext3fs) [11] which is the most commonly used journaling filesystem for Linux.

Finally, in order to protect against physical catastrophic events we need a secure backup system located in a different and possibly remote location. There are many online backup systems in existence, but we should take into additional consideration the trust level of the storage provider. A viable solution would be to build and maintain our own backup system, located in a different physical place, and configured with the same security scheme. A more sophisticated solution derives from a distributed parallel filesystem with automated fault detection and recovery such as the *Hadoop Distributed File System* (HDFS) [4]. Currently, our pilot deployment does not provide any protection against natural disasters.

### 4.3 Application Level Security

At the application level, we ensure that the software stack on the server is secure and healthy by performing the following maintenance tasks:

- We apply the latest security fixes and system patches as they become available.
- We keep any locally installed antivirus software up to date.
- We remove any unused software in order to limit potential security issues.
- We assure that the backups are working, before making any changes to the server.
- We have manually created scripts which regularly examine system properties such as filesystem permissions, disk usage, CPU, RAM and network utilization.

However, in addition to the above generic maintenance tasks we have enhanced the security level to the private repository application itself, by using the following mechanisms.

#### 4.3.1 SSL Certificates

The *Secure Sockets Layer* (SSL) [36] protocol help users to protect their data during transfer by providing an encrypted channel for private communication over the Internet.



SSL uses strong encryption, at 128 bits, to encrypt communication between the web browser and the web server. This protocol requires a third party, which is called the *Certificate Authority* (CA) [8], to identify one end or both ends of the transactions. The web server is given an *SSL certificate* [30] which the browser should check if it was previously issued by a trusted authority and if it's still valid and correctly related to the particular website. After that, the two ends interchange a secret key in order to communicate through a secure encrypted channel. SSL certificates consist a robust mitigation against *eavesdropping* and *phishing* attacks.

The web interface of the private repository is configured to be delivered over an SSL encrypted connection. We installed an SSL certificate onto the web server and enabled the communication to be performed over the *HTTPS* protocol, which is the layering of *HTTP* on top of the SSL protocol.

#### 4.3.2 Two-factor authentication

Each user of the private repository has a password protected account for authenticating to the web interface. As we discussed in Subsection 3.4, passwords are likely to be stolen by a malicious user.

Using a *two-factor authentication system* [43] we drastically increase the security of our sign in process. This scheme adds in an additional layer of security by having the user to retrieve an extra secret token from an out of band source. The user should first log in to the private repository using his username and password and then provide the secret token to complete the process. The technology which is widely used for applying two-factor authentication is *One Time Passwords* (OTPs) [25].

An OTP is a kind of password which constantly changes and is valid for only one login session or transaction. A malicious user who has gained user credentials is impossible to log in without having the OTP token. Considering the case where the OTP is also revealed, the malicious user will not be able to use it since it will be expired.

OTPs can be generated in one of two ways, either as *time-synchronized* or *counter-synchronized*. Both approaches require the user to have a special hardware device which generates OTPs. The time-synchronized OTP is generated based on the current time. The password generating device contains an accurate clock that has been synchronized with the clock on the authentication server. The counter-synchronized OTP is generated using a counter which is synchronized between the device and the server. The counter is advanced each time an OTP value is requested of the device.

The OTP scheme can be also applied by using the user mobile device instead of the extra password generating device. This approach keeps costs low because a large customer-base already owns a mobile phone. The OTP can be sent to the user's mobile device via the following ways:

- **Text messaging:** is a ubiquitous communication channel available in nearly all modern mobile devices and through text-to-speech conversion, to all landline telephones. Texting technology can be considered as a low cost delivery method for both users and providers. Moreover, OTP over text messaging can be encrypted using an A5/x [1] standard, but currently is not enough secure.
- **Voice call:** can be used as a backup to text messaging or to ensure real-time delivery of the OTP. The user can answer a call and hit some keys or enter a PIN to session lock the voice network with the Internet, prior to receiving the password.
- **Mobile applications:** can be also used for delivering the OTP. Recently, there is an increase in free OTP applications available for mobile devices offered by OTP providers. For example, *VeriSign* [45], which is one of the biggest OTP providers, offers software for a wide variety of mobile devices, including *iPhone*, *Android*, *Windows Mobile*, *Blackberry* and more. VeriSign OTP services are used by popular online sites including *eBay* and *PayPal*.

Finally, Google recently announced the use of OTPs for accessing all Google accounts<sup>1</sup>. The user has the ability to choose among the ways described above to receive the OTP.

### 4.4 Content Level Security

At this security level, we protect the private repository data itself. Private documents can exist either on the server hosting the repository or stored at the user's personal computer. A malicious user could gain access to both locations and go through the private material. The best way to protect against such activities is the use of data encryption.

All files are automatically encrypted and decrypted at the server-side. Users are able to view and edit their documents without noticing that data encryption and decryption occurs in the background.

In addition, we address cases where a computer device get lost or stolen. This device could have stored private documents in plain text, authentication keys for VPN access or the user account password for the web interface. The use of two-factor authentication mitigates the last two cases but not the first case regarding the plain text data. Therefore users should be given security lessons not to keep repository data stored on their personal computer devices. Additionally, a security aware user should enable extra mechanisms to protect his computer such as a BIOS password or a user account

---

<sup>1</sup>Google 2-step verification :[https://support.google.com/accounts/topic/28786?hl=en&ref\\_topic=3382253](https://support.google.com/accounts/topic/28786?hl=en&ref_topic=3382253)

password on the computer. Moreover, we propose a more sophisticated approach which is the *Full disk encryption* (FDE) [9]. Software or hardware based solutions encrypt everything on the hard disk drive including the operating system. The system has to confirm his identity using the correct credentials. The combination of passwords and FDE can reliably protect data against unauthorized access.



## Private Repository Pilot Deployment

In this Chapter we describe a pilot deployment of the GCC private repository based on the *ownCloud* file hosting framework [29]. At the time of writing of this document *ownCloud* was at the stable version of 5.0.13.

The GCC consortium chose the *ownCloud* framework to deploy a beta version of the private repository because its features fit our requirements (see Section 2.3) and is considered as one of the best cloud storage alternatives to *DropBox* [10] and other file hosting services.

### 5.1 Overview

*ownCloud* is a free open source software for data synchronization, file sharing, and remote storage of documents. It is written in the *PHP* [33] and *JavaScript* [17] scripting languages and supports several database management systems, including *SQLite* [39], *MariaDB* [20], *MySQL* [22], *Oracle Database* [28], and *PostgreSQL* [34].



### 5.2 Features

**Universal file access** is provided through a web interface or *WebDAV* [47] which is an extension to the *HTTP* protocol. Users can also access their files from mobile devices via mobile applications for *iOS* and *Android* and from desktop clients available for PCs running *Windows*, *Mac OS*, or *Linux*.

**Categorization** and folder management is better implemented and redesigned in the last version by providing a navigation bar on the left and

a better focus on the content of directories at the center. In addition, ownCloud supports viewers for ODF formatted documents and includes a powerful search engine with a by name and by content search capability.

**Versioning** is supported for files and automatically keeps revisions per day until a maximum quota is reached. Then can be configured to expire old versions. Also, a thrash bin is implemented to keep deleted files and provide a file restore feature. The combination of those two features, mitigates the data loss risk by an unintentional file deletion.

**Backups** on ownCloud platform are easily performed by retaining a copy of the configuration folder, the data folder and the database. To restore an ownCloud instance, the administrator simply copies these three elements to their correspondent location. For user account migration an easy export and import feature is provided as well.

Finally, ownCloud is *extendable* via a simple but powerful **public API** for developing applications and plugins on the platform. Additionally, existing applications can be downloaded and automatically installed via the ownCloud **application store**.

## 5.3 Installation

We opt to install the ownCloud server version 5.0.13 on a Linux running machine with *Ubuntu Server 12.04.3 LTS* installed. We use a ready-to-deploy package for Ubuntu distribution provided by the ownCloud website<sup>1</sup>. We perform the following steps:

Download the ownCloud release key and add it to apt.

```
$ wget http://download.opensuse.org/repositories/\
isv:ownCloud:community/xUbuntu_12.04/Release.key

$ apt-key add - < Release.key
```

Add ownCloud repositories, update and install ownCloud. Then fix file properties.

```
$ echo 'deb http://download.opensuse.org/\
repositories/isv:ownCloud:community/xUbuntu_12.04/ \
/' >> /etc/apt/sources.list.d/owncloud.list
$ apt-get update
$ apt-get install owncloud
$ cd /var/www/
$ chown -R www-data:www-data owncloud
```

---

<sup>1</sup><http://software.opensuse.org/download/package?project=isv:ownCloud:community&package=owncloud>

At this time, visit the ownCloud web interface in order to create the administrator account and configure the database backend and the data folder.

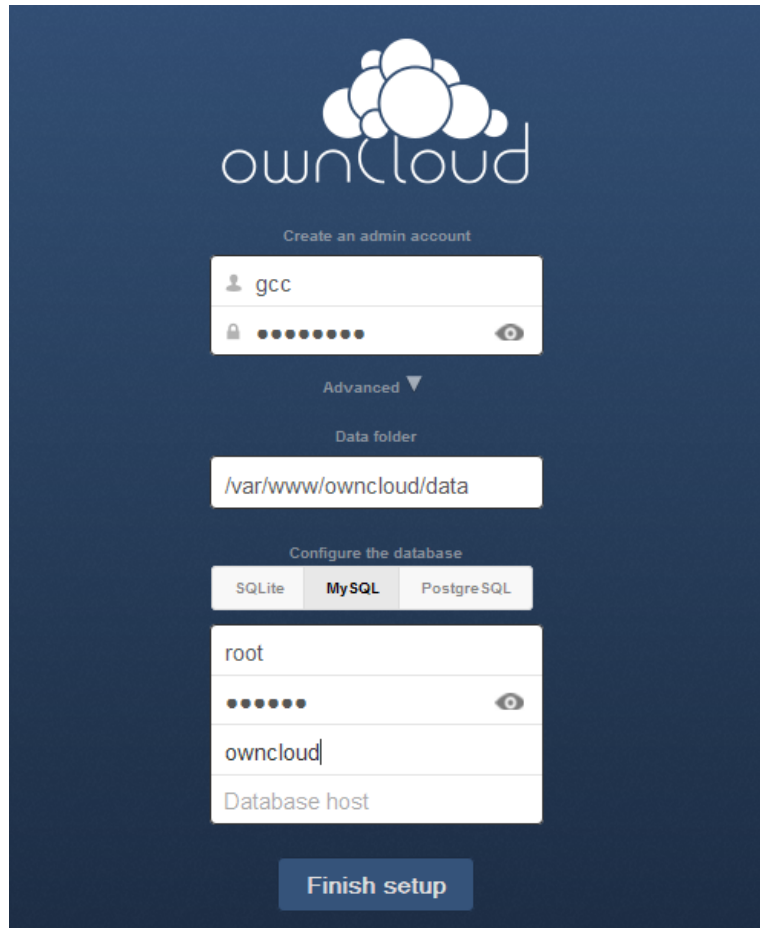
The image shows the ownCloud first-time configuration web interface. It has a dark blue background with the ownCloud logo at the top. Below the logo, there's a section titled "Create an admin account" with a form containing a username field (filled with "gcc"), a password field (filled with dots), and a "Show/Hide" eye icon. Below this is an "Advanced" dropdown menu. Underneath is a "Data folder" section with a text input field containing "/var/www/owncloud/data". The next section is "Configure the database", which has three tabs: "SQLite", "MySQL" (which is selected), and "PostgreSQL". Below the tabs are four input fields: "root" (username), a password field (filled with dots), "owncloud" (database name), and "Database host" (empty). At the bottom is a blue "Finish setup" button.

Figure 5.1: ownCloud first time configuration.

Then, replace the default ownCloud logo with the GCC logo. The repository is now ready for use (see Figure 5.2 and Figure 5.3).

Finally, we upload some files for testing. We upload the GCC presentation given at the CyNC 2013 (Cybercrime Network Conference) in both *.pptx* and *.pdf* format (see Figure 5.4) and some photos from the conference (see Figure 5.6). The ownCloud provides an internal PDF viewer (see Figure 5.5), using *pdf.js* [31] by the Mozilla Foundation.

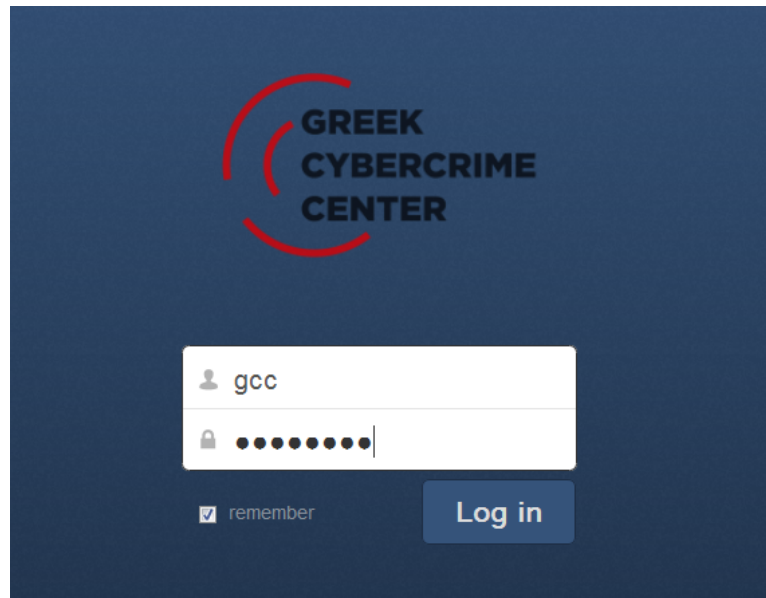


Figure 5.2: ownCloud login page.

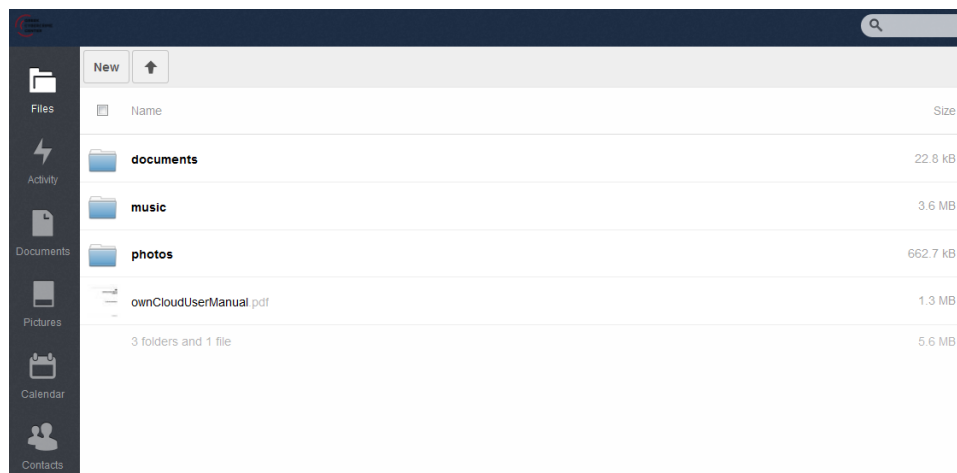


Figure 5.3: ownCloud default contents.



### 5.3. INSTALLATION

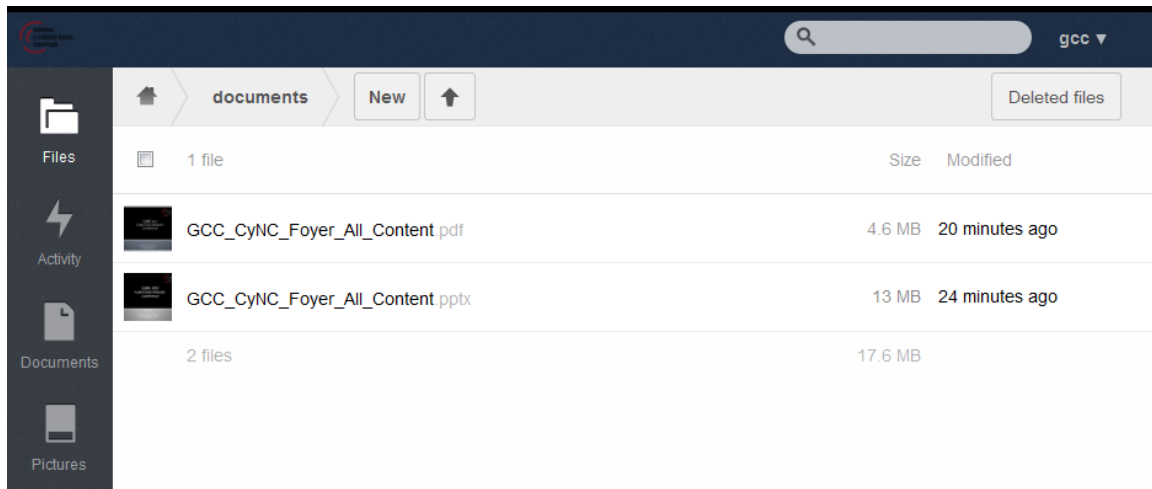


Figure 5.4: ownCloud documents example.

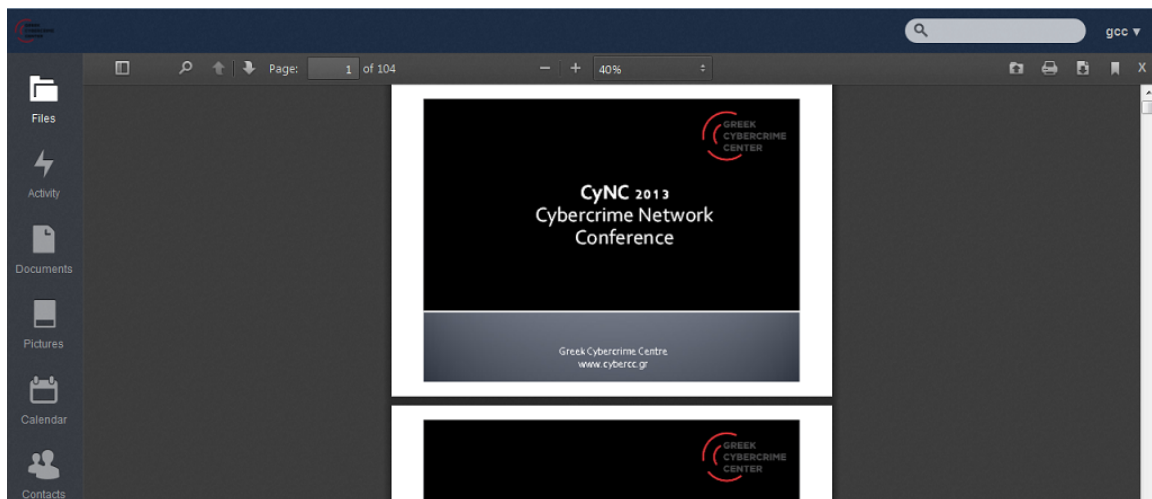


Figure 5.5: ownCloud internal PDF viewer.

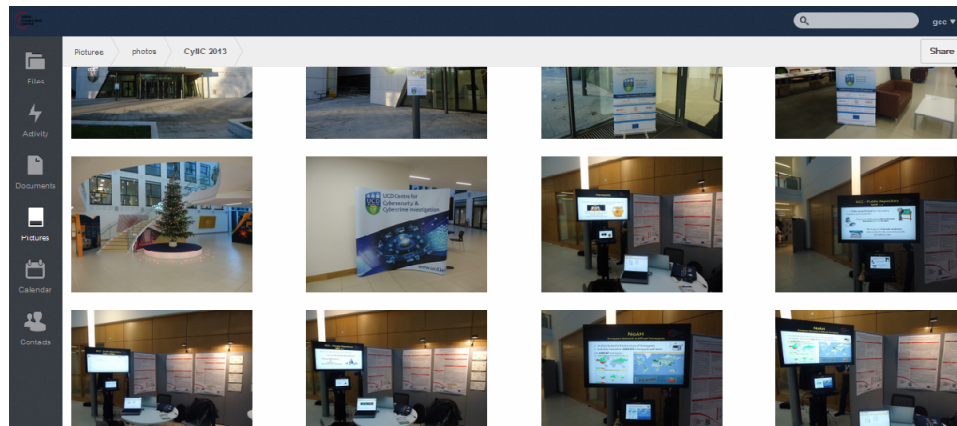


Figure 5.6: ownCloud photo gallery.

## 5.4 SSL configuration

In this section we configure the ownCloud web interface to be delivered over an SSL connection. The Apache configuration files will be placed in `/etc/apache2/sites-available/`.

Enable the following Apache modules and restart Apache.

```
$ a2enmod rewrite
$ a2enmod headers
$ a2enmod ssl
$ service apache2 restart
```

Create some directories and files that we will need.

```
$ mkdir /etc/apache2/ssl/
$ chmod 700 /etc/apache2/ssl/
$ touch /etc/apache2/sites-available/owncloud.conf
```

*File: /etc/apache2/sites-available/owncloud.conf*

```
#####
# https
#####
<IfModule mod_ssl.c>
    NameVirtualHost xxx:443
    <VirtualHost xxx:443>
        ServerName xxx
        DocumentRoot /var/www/owncloud/
        ServerAdmin krithin@ics.forth.gr
        SSLEngine on
```

```

        SSLCertificateFile /etc/apache2/ssl\
        /owncloud.pem
    </VirtualHost>
</IfModule>

```

Generate the SSL certificate.

```

$ openssl req @$ -new -x509 -days 3650 -nodes -out \
/etc/apache2/ssl/owncloud.pem -keyout /etc/apache2/\
ssl/owncloud.pem

```

Output:

```

Generating a 1024 bit RSA private key
.....+++++
...+++++
writing new private key to '/etc/apache2/ssl/owncloud.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GR]:
State or Province Name (full name) [Some-State]:Heraklion
Locality Name (eg, city) []:Heraklion
Organization Name (eg, company) [Internet Widgits Pty Ltd]:FORTH-ICS
Organizational Unit Name (eg, section) []:DCS-LAB
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:krithin@ics.forth.gr

```

Figure 5.7: OpenSSL output.

Modify the `/var/www/owncloud/.htaccess` file to permit only HTTPS access. We use the `mod_rewrite` module to rewrite non HTTPS urls.

File: `/var/www/owncloud/.htaccess`

```

...
...
<IfModule mod_rewrite.c>
    RewriteEngine on
    ...
    ...
    RewriteCond %{HTTPS} off
    RewriteRule (.*?) https://%{HTTP_HOST}%{REQUEST_URI}\
    }
</IfModule>
...
...

```

Finally, load the new configuration and restart Apache web server.

```

$ a2ensite owncloud.conf
$ service apache2 restart

```

## 5.5 One Time Passwords configuration

In this section, we provide information on how to import two-factor authentication, using one-time passwords, into the sign in process of the private repository web interface.

The application store of ownCloud provides third-party software solutions for enabling one-time passwords. However, we opted to follow a more generic solution based on the web server and not on the ownCloud platform itself. We used *mod\_authn\_otp* [6], which is an Apache module for two-factor authentication using one-time passwords generated either via the *HOTP/OATH* [16] algorithm defined in *RFC 4226* or the *MOTP* [21] algorithm.

To install the *mod\_authn\_otp* Apache module download the file and extract it in a directory of your choice. Then, compile, install and load the module into Apache, using the *LoadModule* directive as shown above.

```
$ apt-get install apache2-prefork-dev
$ wget http://mod-authn-otp.googlecode.com/files/\
mod_authn_otp-1.1.6.tar.gz
$ tar zxvf mod_authn_otp-1.1.6.tar.gz
$ cd mod_authn_otp-1.1.6/
$ ./configure
$ make
$ make install
$ echo "LoadModule authn_otp_module /usr/lib/\
apache2/modules/mod_authn_otp.so" >> /etc/apache2/\
apache2.conf
```

Then, update the *owncloud.conf* configuration file to include information for the OTP authentication.

*File: /etc/apache2/sites-available/owncloud.conf*

```
#####
# https
#####

<IfModule mod_ssl.c>

    NameVirtualHost xxx:443

    <VirtualHost xxx:443>

        ServerName xxx
```

## 5.5. ONE TIME PASSWORDS CONFIGURATION

```
DocumentRoot /var/www
SSLEngine on
SSLCertificateFile /etc/apache2/ssl/owncloud.pem

<Directory /var/www/owncloud/>
  AuthType basic
  AuthName "Sign in using your One Time Password \
  !"
  AuthBasicProvider OTP
  OTPAuthUsersFile /etc/apache2/OTP_users
  OTPAuthLogoutOnIPChange on
  OTPAuthMaxOTPFailure 4
  OTPAuthMaxLinger 1800 #browse for 1800 seconds
  Require user krithin
</Directory>

</VirtualHost>

</IfModule>
```

The OTP user configuration file (*/etc/apache2/OTP\_users*) should be manually created. This file contains the secret information about users, PINs, and token keys on the server. Each line defines one user combined with a token. We define the file location via the *OTPAuthUsersFile* directive. Apache updates the file at every authentication attempt, thus we should only edit it when the server is stopped (or no authentication is occurring). More information on the user configuration file can be found online<sup>2</sup>.

Regarding the client software we use the *Google Authenticator* [13], a solution provided by Google for implementing OTP security tokens in mobile apps. Google Authenticator is extremely reliable and well supported and it can be used for every site as well as for Google accounts.



We test Google Authenticator on an Apple *iPhone4* with iOS 7.0.4. To find and install the application on Apple store, type "google authenticator" in Apple store search bar (see Figure 5.8).

After the installation is complete, tap the plus icon to add a new entry. The application supports both automatic setup via a QR code or manual setup by selecting time-based or counter-based code generation.

<sup>2</sup>:<http://code.google.com/p/mod-authn-otp/wiki/UsersFile>

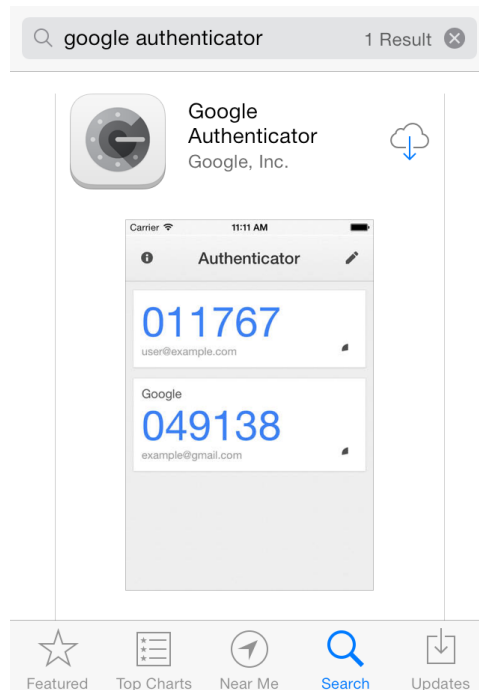


Figure 5.8: Google Authenticator on Apple store.

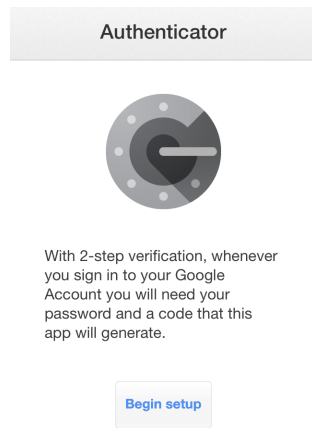


Figure 5.9: Google Au-

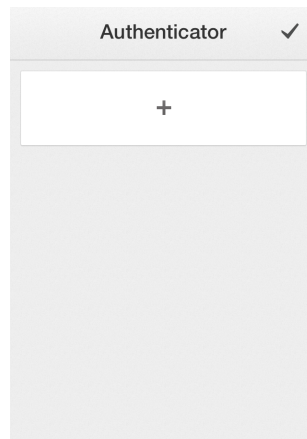


Figure 5.10: Tap the plus icon.

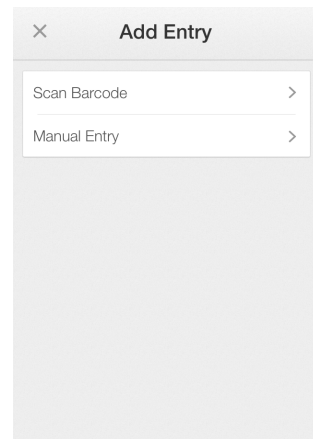


Figure 5.11: Select the barcode option.

When selecting the first option, there are many configuration generators, for Google Authenticator, online<sup>3</sup>, where the user fills in the website name and a user name and gets the configuration. The output includes the server

<sup>3</sup>:[https://www.cnysupport.com/index.php/free-stuff/using-google-authenticator-with-apache-mod\\_authn\\_otp](https://www.cnysupport.com/index.php/free-stuff/using-google-authenticator-with-apache-mod_authn_otp)

## 5.5. ONE TIME PASSWORDS CONFIGURATION

---

side information for the user configuration file (`/etc/apache2/OTP.users`) and a QR code for scanning by the mobile application.

### QR Code Generator Demo for Google Authenticator

Add this line to the `mod_auth_opt` users file on your server

```
HOTP/T30 krithin - [REDACTED]
```

Figure 5.12: Server-side configuration for user *krithin*.

When the application is properly configured via the QR code, it shows password keys. These passwords are randomly generated and time expiring.

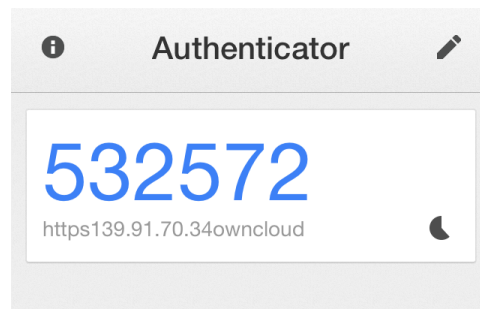


Figure 5.13: Google Authenticator password example.

Two-factor authentication is now enabled. The web interface of the private repository, first asks the user to authenticate using the username and the OTP provided by the Google Authenticator. Then, if the OTP token is correct, the user is asked to provide the username and the password of the ownCloud account. Finally, if the account credentials are also correct, the user gets authenticated to the GCC private repository.

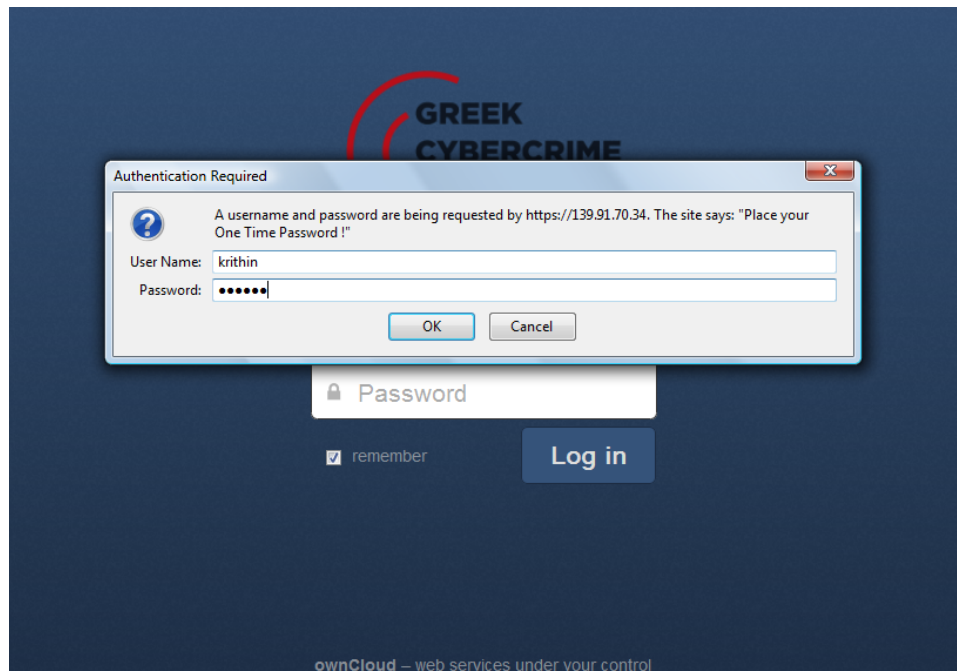


Figure 5.14: GCC private repository sign in process.



In this document we discussed the GCC private repository which could be used to support the secure delivery of education material to LEAs. We initially provided a description of digital repositories and how modern learning content management systems (LCMS) handle all aspects of the teaching process. We also described the GCC public repository and its purposes in the context of the training activities of the GCC project.

We mainly focused on setting out the requirements for a secure private repository for GCC. Initially, we defined a threat model, which indicates the set of possible attacks to the repository and lists certain circumstances where data loss may occur. Then, we proposed and deployed security countermeasures and safeguards to protect against these threats. We opted to classify our security mechanisms into four main levels to achieve a good layered architecture.

Finally, we provided technical information on how we deployed the GCC pilot private repository based on the ownCloud open source project and how we applied security mechanisms such as SSL encryption and two-factor authentication using OTPs.



## Bibliography

- [1] A5/1 security issues. <http://en.wikipedia.org/wiki/A5/1#Security>.
- [2] Access control lists. [http://en.wikipedia.org/wiki/Access\\_control\\_list](http://en.wikipedia.org/wiki/Access_control_list).
- [3] Anomaly-based intrusion detection systems. [http://en.wikipedia.org/wiki/Anomaly-based\\_intrusion\\_detection\\_system](http://en.wikipedia.org/wiki/Anomaly-based_intrusion_detection_system).
- [4] Apache Hadoop Distributed File System. <https://hadoop.apache.org/>.
- [5] Apache HTTP server project. <http://httpd.apache.org/>.
- [6] Apache2 OTP module. <http://code.google.com/p/mod-authn-otp/>.
- [7] Bro Network Security Monitor. <http://www.bro.org/>.
- [8] Certificate Authority. [http://en.wikipedia.org/wiki/Certificate\\_authority](http://en.wikipedia.org/wiki/Certificate_authority).
- [9] Disk Encryption. [http://en.wikipedia.org/wiki/Disk\\_encryption](http://en.wikipedia.org/wiki/Disk_encryption).
- [10] DropBox file hosting service. <https://www.dropbox.com/>.
- [11] ext3 filesystem. <http://en.wikipedia.org/wiki/Ext3>.
- [12] Firewall. [http://en.wikipedia.org/wiki/Firewall\\_%28computing%29](http://en.wikipedia.org/wiki/Firewall_%28computing%29).
- [13] Google Authenticator. [http://en.wikipedia.org/wiki/Google\\_Authenticator](http://en.wikipedia.org/wiki/Google_Authenticator).
- [14] Greek Universities Network. <http://www.gunet.gr/>.
- [15] Hackers. [http://en.wikipedia.org/wiki/The\\_Hacker\\_Crackdown](http://en.wikipedia.org/wiki/The_Hacker_Crackdown).
- [16] HMAC-based One-time Password Algorithm. [http://en.wikipedia.org/wiki/HMAC-based\\_One-time\\_Password\\_Algorithm](http://en.wikipedia.org/wiki/HMAC-based_One-time_Password_Algorithm).
- [17] JavaScript programming language. <http://en.wikipedia.org/wiki/JavaScript>.
- [18] Journaling file system. [http://en.wikipedia.org/wiki/Journaling\\_file\\_system](http://en.wikipedia.org/wiki/Journaling_file_system).
- [19] Learning Content Management Systems. [http://en.wikipedia.org/wiki/Learning\\_content\\_management\\_system](http://en.wikipedia.org/wiki/Learning_content_management_system).
- [20] MariaDB. <https://mariadb.org/>.
- [21] Mobile One Time Passwords. <http://motp.sourceforge.net/>.
- [22] MySQL Open Source Database. <http://www.mysql.com/>.
- [23] Network intrusion detection systems. <http://www.combofix.org/what-it-is-network-intrusion-detection-system.php>.

## BIBLIOGRAPHY

---

- [24] Network Intrusion Prevention System (NIPS). [http://en.wikipedia.org/wiki/Intrusion\\_prevention\\_system](http://en.wikipedia.org/wiki/Intrusion_prevention_system).
- [25] One Time Passwords. [http://en.wikipedia.org/wiki/One-time\\_password](http://en.wikipedia.org/wiki/One-time_password).
- [26] Open Document Format (ODF). <http://en.wikipedia.org/wiki/OpenDocument>.
- [27] Open eClass platform. <http://www.openeclass.org/>.
- [28] Oracle Database. <http://www.oracle.com/us/products/database/overview/index.html>.
- [29] own Cloud framework. <http://owncloud.org/>.
- [30] Public Key Certificate. [http://en.wikipedia.org/wiki/Public\\_key\\_certificate](http://en.wikipedia.org/wiki/Public_key_certificate).
- [31] PDF.js. <http://en.wikipedia.org/wiki/PDF.js>.
- [32] Phishing Attacks. <http://en.wikipedia.org/wiki/Phishing>.
- [33] PHP. <http://php.net/>.
- [34] PostgreSQL database. <http://www.postgresql.org/>.
- [35] Redundant Array of Independent Disks (RAID). <http://en.wikipedia.org/wiki/RAID>.
- [36] Secure Sockets Layer. [http://en.wikipedia.org/wiki/Transport\\_Layer\\_Security](http://en.wikipedia.org/wiki/Transport_Layer_Security).
- [37] Snort. <http://www.snort.org/>.
- [38] Spoofing attacks. [http://en.wikipedia.org/wiki/Spoofing\\_attack](http://en.wikipedia.org/wiki/Spoofing_attack).
- [39] SQLite database. <http://www.sqlite.org/>.
- [40] Suricata. <http://suricata-ids.org/>.
- [41] The linux kernel archive. <http://www.kernel.org/>.
- [42] Tunneling protocols. [http://en.wikipedia.org/wiki/Tunneling\\_protocols](http://en.wikipedia.org/wiki/Tunneling_protocols).
- [43] Two-step verification. [http://en.wikipedia.org/wiki/Two-factor\\_authentication](http://en.wikipedia.org/wiki/Two-factor_authentication).
- [44] UPS. [http://en.wikipedia.org/wiki/Uninterruptible\\_power\\_supply](http://en.wikipedia.org/wiki/Uninterruptible_power_supply).
- [45] VeriSign. <http://www.verisigninc.com/>.
- [46] Virtual private networks. [http://en.wikipedia.org/wiki/Virtual\\_private\\_network](http://en.wikipedia.org/wiki/Virtual_private_network).
- [47] Web Distributed Authoring and Versioning (WebDAV). <http://en.wikipedia.org/wiki/WebDAV>.