System Description and Risk Analysis

Marc Gähwiler Leonhard Helminger Fabian Zeindler

21.11.2013

Contents

1 System Characterization

1.1 Introduction

The company iMovies produces film material in the area of investigative journalism. To protect the integrity and the secrecy of email conversations within the company, a certificate based solution should be provided which ensures the confidentiality of these email conversations.
Today the IT infrastructure of the company is divided into an internal network where the employees work with their desktop machines/laptops and a server area in which the a database server with user data, the web and email server as well as central storage for the employees are located. (this is an assumption by us, since there is no information about the status quo and iMovies is a relatively small company.) The status quo of the system is depicted in Figure 1.1.

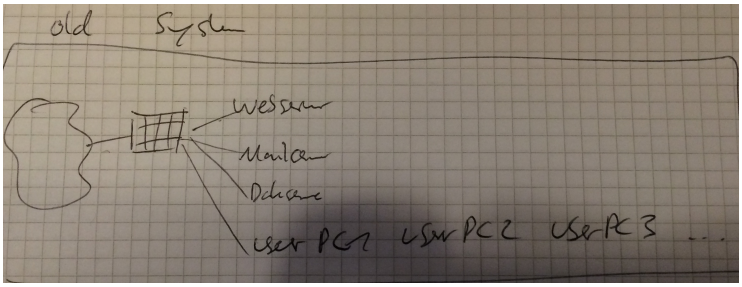

Figure 1.1: Diagram of the status quo.

Our task is to design and implement a certificate authority (CA) that provides the employees with digital certificates that will be used to encrypt and authenticate the previously mentioned email conversations.
The created CA should withstand targeted attacks and integrate into the current system. To authenticate employees so that they can create and use certificates, the existing user database should be used. To guarantee the security of the newly built infrastructure, the old system also has to undergo certain changes. We will introduce a new level of isolation for critical systems and put systems that do not have to be accessed directly by users in a separate zone, so that they can not be reached directly. The design of the system will be evaluated in detail in the following sections. An overview of the proposed system is shown in Figure 1.2.

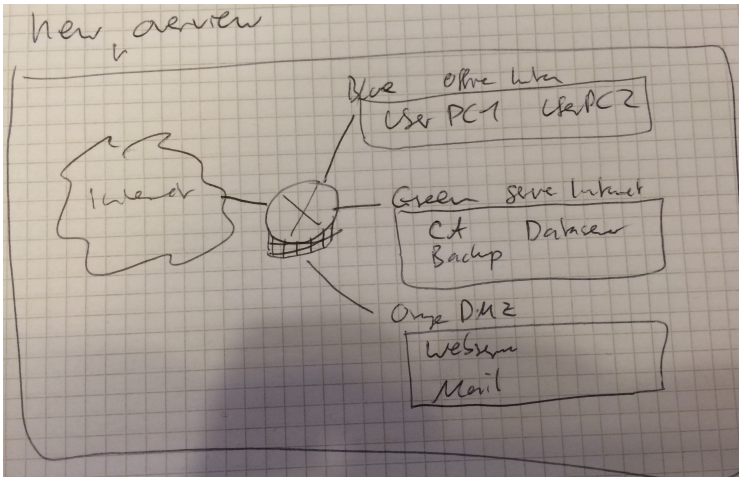

Figure 1.2: Overview of the proposed system.

1.2 System overview

To use the newly developed CA's features, an employee connects to a website, authenticates himself with his credentials that are stored in the preexisting database. Afterwards he then can issue a new certificate carrying the user's data or revoke existing certificates that belong to his account. When at least one certificate is obtained, it is also possible to authenticate with said certificate. Even though a regular user only has to interact with the just described web interface, there is more to the system than that. A separate server is responsible for the process of actually issuing and revoking the certificates in compliance with the information stored in the database. The database could actually remain on a different instance, but for reasons of simplicity and since the company is small, we installed the database on the same server as the CA core which implements the previously described functionality of issuing, verifying and revoking certificates. To prevent loss of data, ensure key recovery and auditability on the system in general, we implement a backup solution as well as a logging system and a key archive. These are all system that do not need regular and close interaction with users or even system-administrators. Because of that, we decided to locate them together on their own, separate backup and archive server.
Since the user only interacts with the web interface which is publicly accessible, this is also the most exposed part of the system. We do not want it in close proximity of the most valuable part of the system, the core CA. We therefore set up a special demilitarized zone (DMZ) in which we place systems and functionality that has to be accessed from the outside. Every other functionality and system is not accessible from the outside and is placed in another zone.
Figure 1.3 shows the Webserver placed in the DMZ, the Core CA as well as the Backup/Archive server in an internal server zone.1

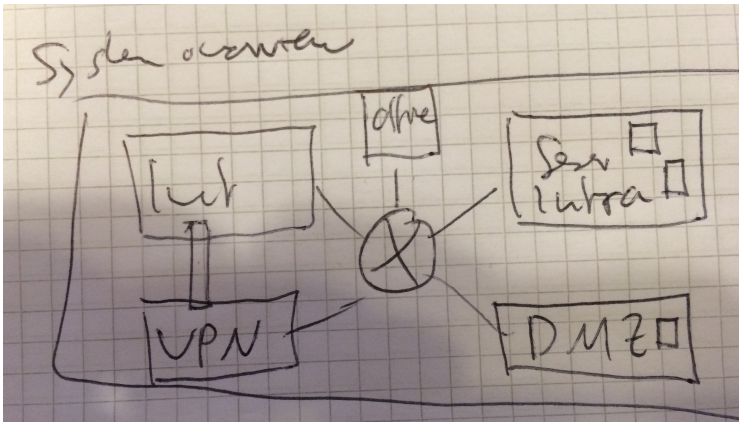

Figure 1.3: More detailed overview of the proposed system.

All personal machines of the employees of iMovies are also in their own zone. This is shown in Figure 1.3 but will not be elaborated any further in this report, as the topic of the project is to implement a certificate authority.
To allow system-administrators to access all systems remotely, we provide an VPN that allows every system-administrator to access all of the different network zones from a remote workstation.

We proceed with more detailed descriptions of the functionalities of the system and their implementation with special regard to security.

1.3 Requirements elicitation and system functionality

In this section we list the requirements and the functionalities of the proposed system. We first look at the functional requirements and afterwards give an insight into the non functional requirements. We cover security requirements as a separate unit, since we want to emphasize the importance of them to the certificate authority system.

1.3.1 Functional requirements

We first list the functional requirements we extracted from the assignment we got to implement this system. From this these requirements we conclude general use cases of the system and will also give an insight in possible misuse cases.

User Interface: A simple web interface that allows each user to log in either with his credentials from the legacy MySQL database, or one of his previously generated certificate and private key combinations. Once logged in the user can view his information (last name, first name and email address), change his password and update his information (last name, first name and email address). Additionally it is possible for the user to request the system to issue a new certificate (based on his possibly

changed credentials) and download the certificate with the newly generated private key in PKCS#12 format.

Administration Interface: A simple web interface where CA administrators can consult the current CA state after a login process which requires the CA administrators to authenticate themselves with their certificate. This includes the number of issued certificates, the number of revoked certificates and the current serial number.

Certificate Issuing and Revocation: The CA offers an interface that allows other systems to

- Generate new public/private key pairs.
- Generate a certificate that ties a public key to a user's credentials (first and last names and his email address) and sign this certificate with the CA's key.
- Revoke a previously signed certificate.
- Get a list that contains all revoked certificates (Certificate revocation list).

Key Backup: To prevent the loss of any information, that was encrypted with an issued certificate, every issued certificate and the according private key are archived.

System Administration and Maintenance: Each server is remotely accessible for the system administrator. In addition to this, all the configuration files of all machines that are part of the systems as well as the log entries that were created are stored on the backup server.

1.3.1.1 Use cases

Figure 1.4 shows a graphical representation of the use cases. Additionally we will go over every use case individually and offer a short description for it.

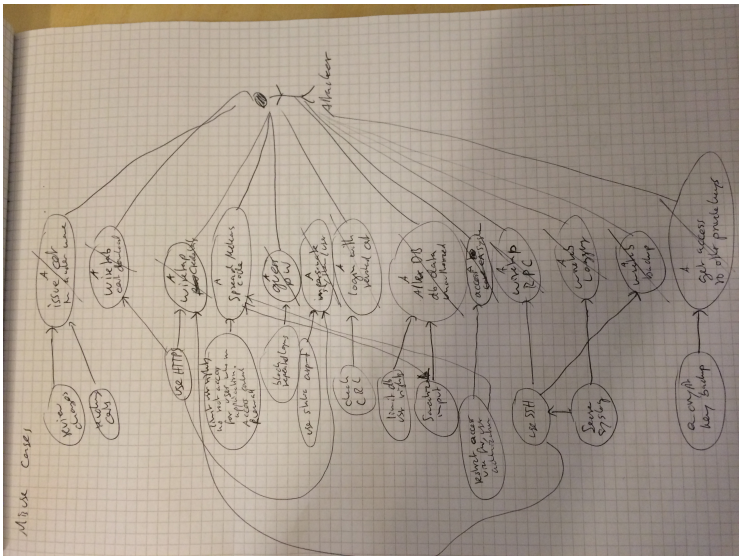

Figure 1.4: Use cases of the proposed system.

We first identify the participating roles:

- User (The employee who uses the web interface to view/change user data, issue, verify, download, revoke certificates)
- CA Administrator (Administrator who has access to certain CA information)
- System Administrator (Administrator who can access all systems remotely)
- Website/Webinterface (Displays information to users and CA Administrators)
- Core CA (Issues, stores, revokes, verifies certificates)
- Database (Stores user information)
- Backup/Archive (Stores certificates, private keys, logging information, backed up configurations)

We now list the use cases of these roles:

Web login User logs onto the webserver with his credentials or a certificate.

With certificate Login by presenting a certificate.

With credentials Login by presenting user name and password.

Verify credentials Presented credentials get verified against the information stored in the database (RPC call to CA Core).

Validate certificate The presented certificate gets verified. CA Core ensures that it is a proper certificate and not yet revoked.

Show user information The user information (not including the user's password) stored in the database is displayed to the user (RPC call to the database).

Change user data The user changes any of the information stored (not including his username) in the database.

Review user data changes If a user changes information other than the password, the changes have to be reviewed by the CA Administrator before writing them to the database and all the users certificates get revoked if the change is accepted.

Request certificate A user sends a request for a new certificate (RPC call to CA Core).

Issue certificate The CA Core issues a new certificate for the user who requested it. The certificate and the private key get archived.

Download certificate The user can download its newly generated certificate and his private key.

Backup keys After creating a new certificate, the CA Core stores the new certificate and the according private key on the backup/archive server.

Revoke certificate A user or the CA Administrator set a certificate for revocation. (RPC call to CA Core).

Generate CRL When revoking a certificate, the CA Core generates a new or updates an existing CRL.

List CA information When a CA Administrator is logged in, his web interface displays the number of issued certificates, the number of revoked certificates and the current serial number.

System login The System Administrator remotely logs into one of the running system and can get root access.

Logging Each system automatically sends logging information of the system and the interactions with the system to the backup/archive server.

Run backups The backup server periodically runs backups to collect the configurations of the other systems.

Logout CA administrator and users that are logged in, can log out, which causes a new login prompt when visiting the site again.

1.3.1.2 Misuse cases

Figure 1.5 shows the misuse cases and also additional use cases that prevent this misuse cases in a graphical manner. We also list each misuse case individually with a short description.

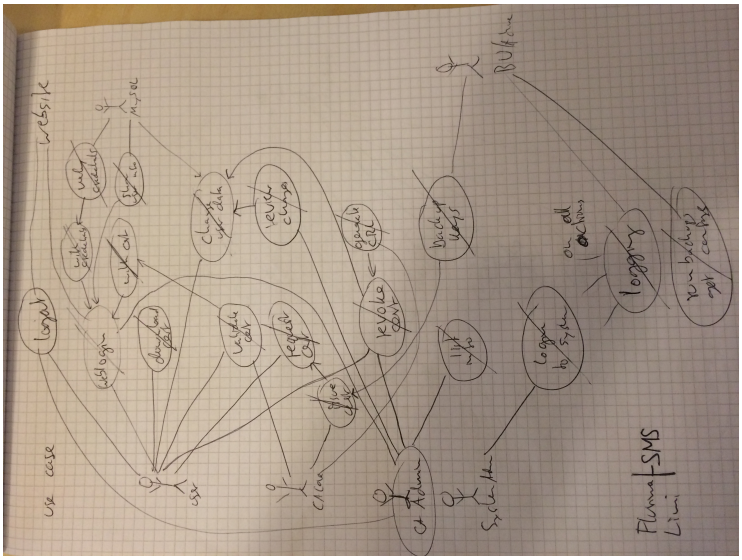

Figure 1.5: Misuse cases and additional use cases of the proposed system.

We first identify the participating roles:

- Threat agent (The attacker which can be an outsider as well as an misbehaving employee)

We now list the misuse cases of these roles as well as the preventing use cases:

Issue certificate in another name The attacker is able to alter his information and then issue a certificate with this information. This gets prevented by the CA Administrator reviewing changes other than to the password and revoking all its issued certificates.

Wiretap certificate download The attacker can obtain a copy of the certificate/private key pair of a user. This is prevented with enforcing HTTPS when connecting to the webservice.

Wiretap user credentials The username and password get obtained by the attacker during transmission. This is prevented with enforcing HTTPS when connecting to the webservice. For system administrator credentials, use SSH and VPN to connect to remote systems.

Guess password With multiple guesses, the attacker can bruteforce the password. To prevent this, limit the number of times a password for a given user can be tried.

Impersonate system/user An attacker can spooth and poison communication mechanisms to impersonate a certain role. Use HTTPS and SSH to prevent this. In the infrastructure controlled by iMovies, use static ARP tables to prevent ARP poisoning.

Login with revoked certificate The attacker may get access to the system with presenting an invalid certificate. To prevent this, always check the CRL.

Alter database date unauthorized An attacker alters data in the database that does not belong to him. To prevent this, one can sanitize the input, review all changes to the database by the CA Administrator and restrict the database user to only the functionality he needs.

Access systems The attacker can log in to the server systems. This is prevented by enforcing access control, only allowing certain traffic through the firewall and onto the system.

Access other private keys The stored private keys in the archive get obtained by the attacker. To prevent this, the private keys are stored encrypted.

Wiretap RPC An attacker obtains information transmitted between the webserver and the CA Core. To prevent this, the RPC traffic is tunneled via SSH.

Wiretap logging The logging information from the servers and applications sent to the backup server gets obtained by the attacker. To prevent this, secure syslog is used which uses an SSH tunnel.

Wiretap backup The attacker obtains system and configuration information by intercepting the backup transmission. SSH is used for backup to prevent this.

Running malicious code An attacker is able to run malicious code on one of the systems. To prevent this, the users executing this systems have limited rights, strong access controls are implemented and input is sanitized.

1.3.2 Non functional requirements

The following list specifies non functional requirements gathered during the analysis.

- For the user database, the legacy MySQL database has to be used.
- The schema of the legacy database can not be modified
- New certificates and their private key are issued in the PKCS#12 format.
- Users with a valid certificate can authenticate themselves over SSL/TLS to the website using the certificate.
- THERE IS MORE, BUT IT IS LATE...

1.3.3 Security requirements

Since the system is developed to improve the general security of the communication between employees, we treat security requirements in a separate section. The following list is compiled from the initial assignment as well as from issues that arise from the misuse case diagram.

- Access control to CA Core functionality and data.
- Secrecy and integrity of private keys in the archive.
- Integrity, non-repudiation and accountability of log files on the backup server.
- Secrecy and integrity of the user data stored in the database.
- Access control on all systems.
- Confidentiality, integrity and authenticity with regards to key transport.
- Secrecy and integrity of CA Core data during processing and transport.
- THERE IS MORE, BUT IT IS LATE...

1.4 Components and Subsystems

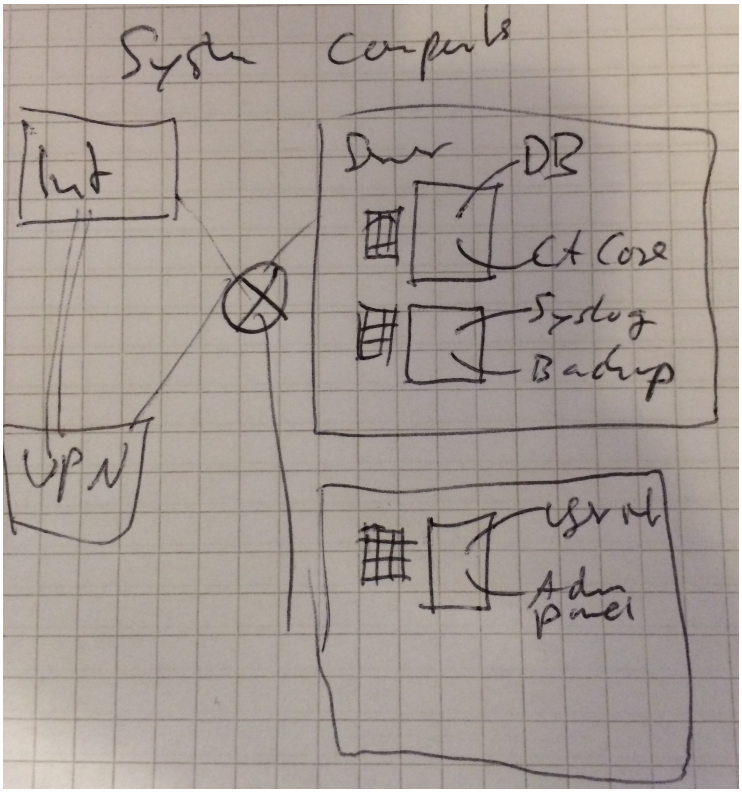

Figure 1.6: Components of the proposed system.

1.4.1 Platforms

The components (as depicted in Figure 1.6) are as follows:

Main Firewall A software firewall based on Linux (IPCop) that divides the iMovies network into the zones DMZ, internal server network, office network and the internet. The firewall functions as a VPN endpoint for remote administrative access and restricts traffic between the subdivided networks according to a security policy described later.

Web Server A Linux machine running Debian 7.0. It hosts the website for the user web interface and the administrative panel. The web server is located within the DMZ, a local iptables firewall allows only HTTPS connections to the outside and SSH connections to the firewall for administration, to the CA Core for secure RPC calls and to the backup server for secure backup and syslog.

CA Core Server A Linux machine running Debian 7.0. It hosts the logic for issuing, verifying and revoking certificates as well as the legacy MyQSL database. The server

is located within the internal server network, a local iptables firewall allows only SSH connections to the firewall for administration, to the web server for secure RPC calls and to the backup server for secure backup, syslog and key backup.

Backup/Archive Server A Linux machine running Debian 7.0. It hosts a syslog server, the logic for periodic backups and storage for encrypted key backup. The server is located within the internal server network, a local iptables firewall allows only SSH connections to the firewall for administration, to the web server and CA Core server for secure backup, syslog and key backup.

Offsite Backup Backups of all platforms that is stored on high volume and long living storage media (for example tapes). The media are stored in an highly secure off-site location (for example a bank vault). This is not implemented in the project but would be highly advisable in practice.

1.4.2 Applications

User Web Interface Web application, developed in python running on nginx on the web server that allows the user to issue, verify and revoke certificates as well as view and alter user data. The website interacts with the CA Core via RPC calls.

Administration Panel Web application, developed in python running on nginx on the web server that allows the CA Administrator to review user changes and display information about the CAs status. The website interacts with the CA Core via RPC calls.

Legacy DB MySQL database with the legacy schema. Running on the CA Core server.

CA Core Application using the OpenSSL library that provides basic interfaces to create new key pairs, sign existing key pairs and revoke certificates. Running on the CA-Core server, interacting with the web server via RPC calls and backing up private keys on the archive server via SSH.

CA Core Storage A database that is used by the CA Core to store certain data relevant to the CA Core application. Running on the CA Core server.

Archive Storage for encrypted public keys and certificates on the Backup/Archive server

Backup A script that collects and stores multiple backups from all other servers. This includes configuration files and other important data. It runs on the Backup/Archive server.

Syslog A syslog server that receives logging data via SSH tunnel from the other servers, running on the Backup/Archive server.

1.4.3 Data

User Information Basic information according to the schema of the legacy database. This includes the users username, his first and last name, his email address and a hash of his password. This information is stored in the legacy database.

Key Pairs Consist of a private key and the according public key that the CA Core generates on request. They are stored permanently in the archive and can be downloaded by the user. It is important, that the CA Core destroys his record of the private key as soon as possible.

Certificates A certificate that is signed by the CA Core. It is also stored in the archive and additionally also in the CA Storage (to allow certificates to be revoked).

Certificate Revocation List A list of certificates, that have been revoked by the CA Core. Stored on the CA Core server.

Local Users Credentials, that are used for system administration and communication between the components of the system. Individual to each system.

Configuration Files Configuration files define the configuration of a given system. They are in place at the system in question and backed up to the Backup/Archive server.

Log Files Log files generated by systems and applications. Stored on the Backup/Archive server.

Master Key Master private key to decrypt the encrypted certificate/private key pairs that were generated on request. Stored in multiple high security offline locations (for example a vault in a Swiss bank).

1.4.4 Interfaces

Web Interface Different for user and CA Administrator but in general it offers a way of authentication via credentials or certificate. Information flows from the outside through this interface via secure RPC calls into the CA Core and vice versa in which case the information gets displayed.

Legacy DB Interface Offers the functionality to ask for, alter and insert information to and from the database.

CA Core Interface Gets accessed via a RPC call to offer functionality to issue, revoke or verify certificates.

Backup/Archive Interface Implements syslog functionality to provide a syslog endpoint as well as storage for key and configuration backup.

1.5 Implementation

This section describes the implementation of our hardware and software with an emphasis on the security measures installed.

Main Firewall The Main Firewall is a software firewall running the IPCop Linux. It functions as a firewall as well as a router and divides the network into the sections DMZ, internal server network, office network (which is not implemented here), internet and VPN network. The routes to the known hardware of the iMovies infrastructure are statically assigned as well as the ARP tables to prevent poisoning. The firewall maintains a VPN using OpenVPN to allow remote access to the VPN network. Remote access to the firewall is possible via SSH out of the VPN network or via webinterface from address 10.10.10.33 from within the internal server network.
The firewall implements several rules on how traffic is allowed to flow between those networks. By default all traffic is denied and silently dropped at the firewall. The following list shows the allowed exceptions:

Source	Protocol	Destination	Description
Webserver	HTTPS (443)	Internet	Allow secure webtraffic from the webserver
Internet	HTTPS (443)	Webserver	Allow secure webtraffic to the webserver
10.10.10.33	IPCop HTTPS (8023)	MainFirewall	Allow the use of the webinterface
OpenVPN network	IPCop SSH (8022)	MainFirewall	Allow remote administrative access
BackupServer	IPCop SSH (8022)	MainFirewall	Allow backup
OpenVPN network	SSH (22)	Backupserver	Allow remote administrative access
OpenVPN network	SSH (22)	CACore	Allow remote administrative access
OpenVPN network	SSH (22)	Webserver	Allow remote administrative access
Backupserver	SSH (22)	Webserver	Allow backup and tunneled syslog
Webserver	SSH (22)	CACore	Allow tunneled RPC calls
CACore	SSH (22)	Webserver	Allow tunneled RPC calls

Servers in general We deploy three servers, the web server, the CA Core server and the Backup/Archive server. All servers are running debian in version 7.0. They have three users, root, an admin and an operational user. The operational user is not in the sudoers list and cannot get root access and should therefore be used to run the applications. We deactivated su and only allow sudo. Per default we deny all

access via PAM (the Pluggable Authentication Modules). We ensure that the root user can only login locally and disable remote login. Failed login attempts get logged and automatically transmitted to the syslog server located on the Backup/Archive server. To prevent poisoning, we fix the ARP tables with the MAC addresses of all interacting systems.
SSH is used for various interactions. We only allow the admin user and the operational user to have SSH access. With the help of sshguard we introduce a exponential time delay after three failed login attempts by implementing temporary rules to the local iptables.
Each server has a local firewall based on iptables which blocks all connections by default. The exceptions are specific to the server and listed in their detailed implementing description.
On the servers, the number of running processes is kept to a minimum, especially processes with network capabilities. With the help of the debian harden package, all unnecessary packages like telnet were removed. The services running on every system are sshd and rsyslogd. Some systems run additional services which are specified later.

Webserver The basic setup is as described in **servers in general**. As the webserver needs additional functionality for running the websites and interacting with the CA Core, some more details have to be provided.
The HTTP daemon used is nginx which was modified to only accept HTTPS requests, meaning redirecting HTTP requests to HTTPS.
We modified the nginx configuration file in such a way, that the HTTPD does not leak it's version string in any of the error pages or in the Server header of HTTP responses. Additionally we used several configuration options to reduce the size of different buffers that nginx uses internally for client connections to reduce the risk of buffer overflows.
To reduce the effectiveness of small distributed denial of service attacks (DDOS attacks) we reduced the default timeouts to prevent single clients to freeze the HTTPD by keeping alive a huge number of simultaneous connections and not sending any data. This is a trade-off because client on a very slow connection wont be able to access the webserver. This does not concern us because our use case does not involve connections from very remote places via very slow connections (like satellite uplinks or similar connections). Additional methods to prevent or reduce the effectiveness of larger DDOS attacks include hardware solutions and third party services like Cloudflare or Dragonara. These solutions were not evaluated because they are not in the scope of this report.
To detect potential attacks we use the built in logging facilities of nginx that enable us to store certain information (like the IP address, the HTTP verb or the URL) of every HTTP requests that hits the server. This would enable us to use an intrusion detection system to detect any kind of abnormal activity. At the moment we are not using such a system.
The two applications, the user web interface and the admin panel were developed in

python. Both applications were implemented using a python web micro-framework called Flask. This open source solution provides us with the tools to rapidly develop simple and secure web applications. WRITE THIS YOU LAZY BASTARD! XSS AND CSRF SHIT!!!
To prevent cross-site request forgery (CSRF) attacks we are using the CRSF token provided by the flask framework and the user inputs are sanitized in flask automatically to render cross-side script (XSS) attacks impossible.
Data between the web applications and the CA-Core backend is exchanged with a python RPC library called Pyro4. Pyro4 does not support any kind of encryption of the communication. Because of that we tunnel all RPC over a SSH connection to ensure the secrecy and integrity of all calls.

Like all other physicial machines, the webserver has a local iptables firewall which silently drops all packets by default and implements the following exceptions:

Source	Protocol	Destination	Description
Backupserver	SSH (22)	Webserver	Allow backup
OpenVPN network	SSH (22)	Webserver	Allow remote administrative access
Internet	HTTPS (443)	Webserver	Allow secure web traffic
CACore	RPC (4444)	Webserver	Allow tunneled RPC call
Webserver	HTTPS (443)	Internet	Allow secure web traffic
Webserver	RPC (4444)	CACore	Allow tunneled RPC call
Webserver	syslog (10514)	Backupserver	Allow tunneled syslog
CACore	SSH (22)	Webserver	Allow tunneled RPC call
Webserver	SSH (22)	Backupserver	Allow tunneled syslog

CA Core In addition to the basic setup described in **servers in general**, the CA Core server houses the legacy MySQL database and the application that provides the functionality of issuing, verifying and revoking certificates.
The MySQL database grants access to the table iMovies.users via the database user dbuser. This user only has limited rights, namely he can INSERT, SELECT and UPDATE only. There is no need for encrypting SQL traffic, as the database is on the same host as its consumer.
The application which implements the certificate authority (CA) functionality is written in python and uses the pyOpenSSL module wich is a python wrapper for the openssl library. It is used to create, revoke and sign certificates. In addition to that, another openssl wrapping library called m2crypto is used to verify if a given certificate is valid because pyOpenSSL does not provide this functionality.
To query the legacy MySQL database the CA application uses a python object relational mapper called SQLAlchemy. This enables us to query the database without writing our own SQL statements. In addition to that SQLAlchemy automatically escapes user input in all statements which serves as protection against SQL injection attacks. Internally SQLAlchemy uses MySQLdb, a python interface to MySQL databases.

As previously mentioned in the **Webserver** item, the CA application offers a RPC endpoint that is implemented using the Pyro4 middleware.
To ensure the secrecy of the generated private keys, the application discards the keys after it sent them to the web application that requested the creation.
The application allows normal users to log in with a user/password combination that is stored in the legacy database or a previously generated and not revoked private key/certificate pair. A successful login starts a new session that is attached to the user that signed in. Without a valid session, the CA application does not allow to create new certificates. A session becomes invalid after 30 minutes without any action. CA administrators can only authenticate themselves with a special CA admin certificate.
After generating a new private key and creating a certificate for the according public key, the CA application stores the private key and the certificate as an encrypted file locally. Once every hour all those key backup files are transferred to the Archive via a secure SSH connection. They can only be decrypted with a RSA private key that is stored in multiple high security offline locations (for example a vault in a Swiss bank) and can not be accessed by a system administrator.
The CA application implements fine grained logging using three different log levels called debug, info and error. Messages logged on the debug level are discarded immediately on any production system. Info and error messages are sent to the Backup using the syslog daemon.

Source	Protocol	Destination	Description
Backupserver	SSH (22)	CACore	Allow backup
OpenVPN network	SSH (22)	CACore	Allow remote administrative access
Webserver	RPC (4444)	CACore	Allow tunneled RPC call
CACore	RPC (4444)	Webserver	Allow tunneled RPC call
CACore	SSH (22)	Backupserver	Allow key backup
CAcore	syslog (10514)	Backupserver	Allow tunneled syslog
Webserver	SSH (22)	CACore	Allow tunneled RPC call
CACore	SSH (22)	Webserver	Allow tunneled RPC call

Archive/Backup server The functionality this server implements in addition to the basic setup described in **servers in general** is storage for the backed ups private keys, a script that backs up the configurations of the servers and a rsyslog server that collects the syslog data sent by the other machines and applications. Backup, syslog and key backup is done directly via SSH or via an SSH tunnel.
The local iptables firewall that closes all ports by default implements the following exceptions:

Source	Protocol	Destination	Description
CACore	SSH (22)	Backupserver	Allow key backup
OpenVPN network	SSH (22)	Backupserver	Allow remote administrative access
Backupserver	SSH (22)	CACore	Allow backup
Backupserver	SSH (22)	MainFirewall	Allow backup
Backupserver	SSH (22)	Webserver	Allow backup
Webserver	syslog (10514)	Backupserver	Allow tunneled syslog
CAcore	syslog (10514)	Backupserver	Allow tunneled syslog

1.6 Operation

During operation of the system several actions can be performed to keep the system secure and detect misuse or an attack early on.

Update Perform weekly security updates on all production machines. To perform the updates the whole system is shut down for a small period to guarantee that the update can not have any side effects.

Logging Evaluate the logs and look for anomalies unintended patterns.

Audit Make sure that the information in the logs can be linked to a specific user and action for accountability.

Backup Check that the backups work and also check if playing back a backed up configuration would work.

Detection Try to detect misuse with monitoring programs. Like monitoring overall network traffic which should be fairly low under normal usage or run intrusion detection systems. The installed libraries harden-nids for network intrusion detection and harden-surveillance for network surveillance from the harden package can help with that task.

1.7 System documentation

This section is in some sense a summary of the above with some additional information about the systems like usernames, passwords, IP addresses, virtual network names etc. Use this section for administrative purposes of the system.

1.7.1 Network diagram

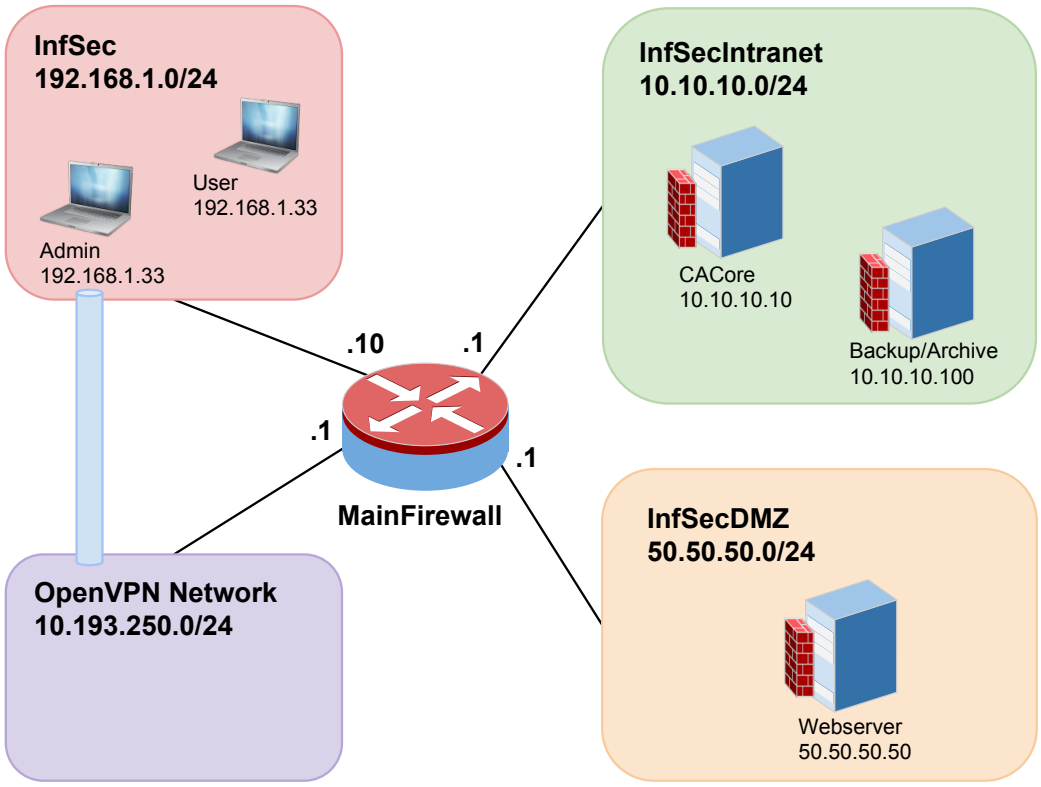

Figure 1.7: Network Diagram

1.7.2 Machines

All servers are equipped with at least one NIC corresponding to the network they are in as shown in Figure 1.7 and a NAT NIC that can be used for internet access (git, apt-get, etc.).

1.7.2.1 MainFirewall

Accounts & Passwords

admin: wT7nDB7A7d7V **root:** 5hmAMWxN6uVa

Installed software

IPCop

User for ssh access

Accesable only on 10.10.10.1 with port 8022 from 10.10.10.33 (user in InfSecIntranet) or from OpenVPN-Network.
ssh -p 8022 admin@10.10.10.1

Additional information

There is a webinterface on https://10.10.10.1:8023. Only accessable from 10.10.10.33 (user in InfSecIntranet).

1.7.2.2 Webserver

Accounts & Passwords

serveruser: 3FaVLt9RNxLu **operuser:** KLs3PbjoXu9m

root: cLepMVRq8wDQ

operuser cannot use sudo/su and should therefore be used to run the application.

Installed software

Debian, iptables, nginx, python, flask, openSSL, openSSH

Running services (lsof -i)

sshd, nginx

User for ssh access

ssh {operuser, serveruser}@50.50.50.50

additional information

SSL signing key for HTTPS: 9klTRxBQcAnM

1.7.2.3 CoreCA

Accounts & Passwords

causer: 9BxkXM5fLLL8 **operuser:** PrCgs5TLqW4f

root: 8kSeddphG6Ac

operuser cannot use sudo/su and should therefore be used to run the application.

Installed software

Debian, iptables, python, mySQL, openSSL, openSSH

Running services (lsof -i)

sshd, mysql (only localhost)

User for ssh access

ssh {operuser, causer}@10.10.10.10

additional information

MySQL root: Cm7NsWBhf52C
MySQL dbuser: Q8mxLsBwTLJi
dbuser is only allowed to INSERT, SELECT, UPDATE the table user.iMovies, thus should be used for connecting to the database.

1.7.2.4 Backup/Archive Server

Accounts & Passwords

archiveuser: 4uMtrPMLxShw **operuser:** QT5wbxjCN8gG

root: gaBWUt5EH8vU

operuser cannot use sudo/su and should therefore be used to run the application.

Installed software

Debian, iptables, syslog, openSSH

Running services (lsof -i)

sshd

User for ssh access

ssh {operuser, archiveuser}@10.10.10.100

1.7.2.5 User

Accounts & Passwords

alice: alice

additional information

When running in netwok InfSec, VPN is possible (VPN start script on Desktop):

PKCS12 PW: StgmE58sadQu

When running in netwok InfSecIntranet, firewall web access on https://10.10.10.1:8023

1.7.3 Firewall rules

1.7.3.1 MainFirewall

All connections are closed by default. The following list shows the allowed exceptions:

Source	Protocol	Destination
Webserver	HTTPS (443)	InfSec
InfSec	HTTPS (443)	Webserver
10.10.10.33	IPCop HTTPS (8023)	MainFirewall
OpenVPN network	IPCop SSH (8022)	MainFirewall
10.10.10.33	IPCop SSH (8022)	MainFirewall
BackupServer	IPCop SSH (8022)	MainFirewall
OpenVPN network	SSH (22)	Backupserver
OpenVPN network	SSH (22)	CACore
OpenVPN network	SSH (22)	Webserver
Webserver	RPC (4444)	CACore
CACore	RPC (4444)	Webserver
Backupserver	SSH (22)	Webserver
Webserver	SSH (22)	CACore
CACore	SSH (22)	Webserver
Webserver	syslog (10514)	Backupserver

1.7.3.2 Webserver

All connections are closed by default. The following list shows the allowed exceptions:

Source	Protocol	Destination
Backupserver	SSH (22)	Webserver
OpenVPN network	SSH (22)	Webserver
InfSec	HTTPS (443)	Webserver
CACore	RPC (4444)	Webserver
Webserver	HTTPS (443)	
Webserver	RPC (4444)	
Webserver	syslog (10514)	Backupserver
CACore	SSH (22)	Webserver
Webserver	SSH (22)	

1.7.3.3 CoreCA

All connections are closed by default. The following list shows the allowed exceptions:

Source	Protocol	Destination
Backupserver	SSH (22)	CACore
OpenVPN network	SSH (22)	CACore
Webserver	RPC (4444)	CACore
CACore	RPC (4444)	Webserver
CACore	SSH (22)	Backupserver
CAcore	syslog (10514)	Backupserver
Webserver	SSH (22)	CACore
CACore	SSH (22)	Webserver

1.7.3.4 Backup/Archive Server

All connections are closed by default. The following list shows the allowed exceptions:

Source	Protocol	Destination
CACore	SSH (22)	Backupserver
OpenVPN network	SSH (22)	Backupserver
Backupserver	SSH (22)	CACore
Backupserver	SSH (22)	MainFirewall
Backupserver	SSH (22)	Webserver
Webserver	syslog (10514)	Backupserver
CAcore	syslog (10514)	Backupserver

1.8 Backdoors

Describe the implemented backdoors. **Do not add this section to the version of your report that is handed over to the team that reviews your system!**

2 Risk Analysis and Security Measures

2.1 Information Assets

Describe the relevant assets and their required security properties. For example, data objects, access restrictions, configurations, etc.

2.2 Threat Sources

Name and describe potential threat sources.

2.3 Risks and Countermeasures

List all potential threats and the corresponding countermeasures. Estimate the risk based on the information about the threat, the threat sources and the corresponding countermeasure. For this purpose, use the following three tables.

Impact	
Impact	Description
High	...
Medium	...
Low	...

Likelihood	
Likelihood	Description
High	...
Medium	...
Low	...

Risk Level			
Likelihood	**Impact**		
	Low	Medium	High
High	Low	Medium	High
Medium	Low	Medium	Medium
Low	Low	Low	Low

2.3.1 *Evaluation Asset X*

Evaluate the likelihood, impact and the resulting risk, after implementation of the corresponding countermeasures.

No.	Threat	Implemented/planned countermeasure(s)	L	I	Risk
1	...	...	*Low*	*Low*	*Low*
2	...	...	*Medium*	*High*	*Medium*

2.3.2 *Evaluation Asset y*

No.	Threat	Implemented/planned countermeasure(s)	L	I	Risk
1	...	...	*Low*	*Low*	*Low*
2	...	...	*Medium*	*High*	*Medium*

2.3.3 Detailed Description of Selected Countermeasures

Optionally explain the details of the countermeasures mentioned above.

2.3.4 Risk Acceptance

List all medium and high risks, according to the evaluation above. For each risk, propose additional countermeasures that could be implemented to further reduce the risks.

No. of threat	Proposed countermeasure including expected impact
...	...
...	...