System Description and Risk Analysis

Clemens Klopfstein Nicolas Kowenski Jasmin Stadler*
Nathanael Wettstein

December 1, 2022

Contents

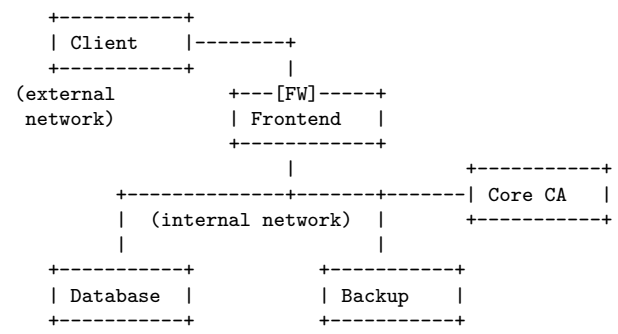
*part of the team until Nov, 16.

1 System Characterization

1.1 System Overview

This document describes the Certificate Authority (CA) system owned and operated by the company *iMovies*. The system provides employees with digital certificates that will be used for secure e-mail communication. The system consists of the following components:

1. Client: Virtual machine (VM) simulating a workstation for users and administrators.
2. Frontend: A dual-homed machine containing a host-based firewall, a web server, a jumphost for administrators and the frontend application.
3. Core CA: A machine holding the CA private keys that generates and manages certificates.
4. Database: A machine serving as a database (DB) server.
5. Backup: A machine collecting and storing backups and logs.



1.2 System Functionality

The system provides an internet-accessible web interface to users with the following features:

- User authentication: Users may authenticate using password or certificates. Admins must authenticate using certificates.
- The application stores user data in a database. Data includes email address, password hash, certificates. The frontend provides a method to modify user data.
- Certificate Issuing Process: A user can request a certificate from the CA and download the certificate, including the corresponding private key, in PKCS#12 format.

- Certificate Revocation Process: A user can revoke their own certificates.
- Certificate Revocation List (CRL): The CA uses a CRL internally, and publishes it to the internet.
- For administration, a web interface to CA admins shows some statistics.

The CA issues two types of certificates: user certificates (for email encryption and signing and user authentication), and server certificates (for server authentication and encryption).

The system performs regular backups of important assets to recover from data corruption and to ensure business continuity. System administration and maintenance can be performed by system administrators with shell access to all hosts. Note that the *secure e-mail communication* mentioned in chapter 1 of the assignment is not in scope of this system.

1.3 Security Design

1.3.1 General principles

The system is designed to implement basic security principles like *Simplicity*, *Compartmentalization* and *Minimum Exposure*. We assume that an adversary may have already breached parts of our network, but must be prevented from elevating privileges or moving laterally. To this end, we require all external and internal entities to authenticate when accessing any resources of the system (*Complete Mediation*). All relevant operations must be logged according to the principle of *Traceability*. All communication, also between internal entities, should be encrypted to achieve *Secrecy*. Key-based authentication (e.g. when using SSH) must be preferred over password-based authentication to maximize entropy of secrets.

1.3.2 Identity and Access Management (IAM)

There are four types of (human) users of the overall system: System administrators (`sysadmin`), CA administrators, CA users and visitors. Their access rights are as follows:

- Sysadmin: An employee of iMovies, Inc., IT department, with full access (sudo) via SSH to all hosts.
- CA administrator: An employee who can monitor CA usage.
- CA users: An employee who can access the web application, update user data, issue and download certificates.
- Visitor: A person with unauthenticated access to the web application.

Users of the frontend application are allowed to edit their user data. However, we do not allow users to modify their email address for two reasons. First,

the email address is used as a unique identifier for users internally. Second, we lack a way of verifying the email addresses, so we must prevent users from impersonating other users.

Linux users are defined per host. This is a static list and is defined in section 1.4. Remote access via SSH is allowed only for Backup and Sysadmin.

Our company only employs one system administrator. To ensure business continuity, there is a second sysadmin account, called vagrant, on the system, the credentials of which are kept in an off-line storage (i.e. a physical safe).

CA users are defined by the `users` DB table. The number of CA users is treated as static, as the application lacks *add user* functionality. Additional users are added by the system administrator according to company policies. Constraint: The initial content of this table is given by the assignment. The scheme and format must not be changed.

CA administrators are defined by the `admin_users` DB table. This list is treated as static. Remark: CA administrators do not have a password, they must use certificate-based authentication.

DB users are allowed to access the *iMovies* database in the MySQL deployment and are defined in a DB table. There are three DB users, each being in the *Least privilege*: the web application that can read and update the `users` and read the `admin_users` table, the backup user that has read-only access to export the database, and the `root` user that owns all tables and whose credentials are stored in a physical safe.

Authentication of CA administrators and CA users is performed by the web server and the web application on the frontend host, based on username and password or based on user certificates. Authorization is then performed by the database host, which holds the information about which user holds which role. In order to have a fail-safe system, the certificate's status is checked by the web server and by the core application through the web application.

1.3.3 Host hardening

To prevent some privilege escalation attacks, we define that every process must run with minimal rights (*Least privilege*). Every file must be minimally accessible (*Least privilege*). Specifically, access to private keys and database must be restricted from all users except for the owner (`chmod 700` for directories and `600` for files). Private keys must be encrypted and must never be stored in plain-text.

Host firewalls are implemented using iptables to reduce the attack surface to a minimum. Specific drop rules are implemented to enable the Sysadmin to detect malicious activity (IE. Attacker pinging hosts). To ensure repeatability, we start with a base image and automate the provisioning using *Vagrant*[1]. We partially adopt hardening guides based on industry best practices[2].

[1] `https://www.vagrantup.com/`

[2] Center for Internet Security (CIS) Benchmarks, `https://www.cisecurity.org/benchmark`

1.3.4 Web services

Web applications are compartmentalized by splitting them into micro-services. Different services communicate over APIs. To avoid most of the *OWASP Top 10*[3] vulnerabilities (*Minimum Exposure*), we define the following requirements for all web application components:

- No JavaScript (to avoid XSS). Use HTML and CSS only.
- No XML parsing (to avoid XEE or XML logic bomb attacks). More generally, no file uploads.
- Use CSRF protection in all forms.
- Sanitize all user input. Specifically, avoid injection attacks by using prepared statements and hard-coded URLs for server-side requests.
- Use up-to-date software packets with no known vulnerabilities.
- Use safe TLS headers including HSTS based on industry recommendations[4]. As all end users are known, we are free to choose "strict" settings to reduce the attack surface without compromising *Usability*.
- Use session handling with cryptographically strong session IDs stored in client-side cookies.
- Disable all debug output.
- Fail gracefully in case internal resources (e.g. database, Core CA) are not available (*Secure and Fail Safe Defaults*).
- Allowed HTTP Status Codes for responses: `200, 301, 302, 304, 400, 401, 404, 500`. All other error codes should be mapped to `500`.
- Use CAPTCHAs[5] in the password login form to achieve some protection against password brute-forcing[6]. Even though simple CAPTCHAs can be solved by software (e.g. using Deep Neural Networks, DNN), we decide against advanced CAPTCHAs[7] that require an external service provider and fingerprint user agents and behavior. This is to avoid a *Single Point of Failure* and to guard user privacy.
- Delay each login attempt by 1 second. This limits the rate of password guessing per connection and provides basic protection against password brute-forcing.

[3] `https://owasp.org/Top10/`

[4] `https://ssl-config.mozilla.org/`

[5] Completely Automated Public Turing test to tell Computers and Humans Apart

[6] Further anti-automation features were not implemented in favor of *Usability*: 1) Count number of failed login attempts per user. Increase delay exponentially after 10, 20, etc. failed attempts. This limits the rate of password guessing for a certain user, even if an attacker uses different connections or origins. However, this opens the possibility of a denial-of-service (DoS) DoS by keeping all application threads busy. 2) Lock account after 20 failed attempts. This enables a simple DoS attack by locking out all users.

[7] Like `https://developers.google.com/recaptcha/`

1.3.5 Backup

The backup server does asymmetric, incremental backups by pulling configuration, set-up scripts and other sensitive files over a secure shell connection using "rsync". This ensures secrecy and integrity of data in transit and prevents a compromised host to (easily) access the backup data. Also, an attacker cannot disable the backup process without going undetected, as every backup event is logged and stored as well.

Every 12 hours, the backup server pulls the specified files and folders, storing them in "daily" folders. This way we can have a historical status of the backed-up information and the possibility of a roll-back. During the time window between two backup events, however, there exists a risk of an attacker hiding their file modifications by e.g. changing configuration or scripts back to the originals before the backup event. Nevertheless, it is not trivial to go undetected because of the logs implementation.

Using syslog-ng we centrally collect relevant logs. Core, Frontend and Database hosts push logs to the backup server where it is being stored on daily folders. This prevents an attacker who may have gained access to one of these hosts to hide traces as logs are being shipped to the backup server in real time.

For making the logs *"tamper proof"* there is a "cronjob" running every 5 minutes hashing the logs and encrypting the hash. Sysadmin can always re-create the log's hash and compare with the decrypted hash for a specific point in time. This way, an attacker can not modify or delete logs without being noticed within the 5 minutes window.

However, logs are transferred in plain text. An attacker who control the network may be able to manipulate (on transit) or "black hole" them. Doing so without leaving any trace via one of the existing hosts must be done in a 5 min window because of the anti-tamper logs measure. Running a new host on the private network to manipulate logs may leave no traces.

To fulfill the requirement *"secrecy and integrity with respect to the private keys in the key backup"*, we transfer the files over SSH with certificate-based authentication. GPG encryption: Once sensitive files like keys and certificates arrive into the backup system they are immediately GPG encrypted, so no plain text sensitive data are stored on the backup server. The System administrator has access to the private key in a secure offline storage device.

The backup server runs a weekly job "rotating" backups and anti-tamper files. This deletes files with access time older than 45 days and anti-tamper files older than 15 days to prevent the disk filling up and a service failure. The system administrator is responsible for archiving the logs in an offline archive for 2 years.

1.3.6 System Maintenance and Patching

System maintenance and (security) patching is performed manually by the sysadmin at regular intervals or immediately if a critical vulnerability (CVSS $>= 8.0$) is announced. The sysadmin will physically connect a laptop containing

a complete mirror of relevant packages (apt, pip, npm, Docker images, etc.) to the server in the data center. This is to minimize exposure of backend systems to the Internet.

Separate environments are used for testing and integration, which are exact copies of the production environment.

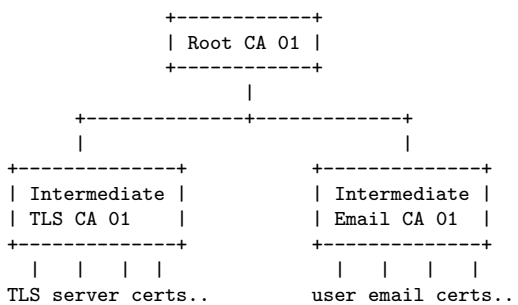
1.3.7 Security Monitoring and Malware Protection

Logging of events (like errors or log-in attempts) is performed on the web server and web application level as well as on sshd. A central collection of logs is implemented to avoid a *Single Point of Failure*.

On the other hand, no intrusion detection or prevention (IDS/IDP) system is implemented, neither host-based nor network-based. No functional monitoring is performed. For example, no alerts are generated if a host is down or an unexpected event has happened. No antivirus, sandboxing or other malware protection mechanism is used. No integrity checks or monitoring of important files (like libraries or binaries) on the hosts is performed. No web-application firewall (WAF) is employed. System hardening does not include apparmor or SElinux. Host firewall it is implemented using iptables to reduce the attack surface.

1.3.8 PKI Design and Certificate Authority Policy

Based on best practice, we design our PKI based on a two-tier hierarchy[8]. The root CA is used to create intermediate CAs, which issue leaf certificates. Each intermediate CA issues certificates of a specific type. This architecture allows the root key to be kept offline during operations, as any compromise of the root key is disastrous. It would be very expensive to roll out new root CA certificates to all parties involved. However, in case of a breach or loss of an intermediate CA private key, the company can recover by revoking and issuing new certificates for this intermediate CA using the off-line backup of the private root CA private key. The following illustration shows the two-tier PKI design.



[8] https://techcommunity.microsoft.com/t5/ask-the-directory-services-team/designing-and-implementing-a-pki-part-i-design-and-planning/ba-p/396953

The CA applies the following rules:

- Root certificates are valid for ten (10) years, intermediate certificate for three (3) years, and leaf certificates for one (1) year.
- Key pairs for Root and Intermediate CAs are of type RSA with 4096 bit key length. This is considered to be equivalent to symmetric encryption with over 256 bit key length according to NIST SP 800-57[9] and acceptable for information classified up to TOP SECRET according to NSA[10] and is following the principle of *Generating Secrets*.
- The key pairs for leaf certificates are of type RSA with 2048 bit key length, which is equivalent to 112 bits symmetric encryption. This is considered acceptable by NSA for *"unclassified PKI systems"*, which is applicable to our system. As opposed to the longer key length for intermediate and root CAs, the shorter key length speeds up cryptographic operations significantly.
- Leaf certificates must not be able to issue other certificates (CA flag must be set to false or be absent).

The public/private key pairs for user certifictes and certificate signing requests (CSR) are generated by the core CA. The user can then download the issued certificate. The purpose (`keyUsage`) of user certificates includes `nonRepudiation, digitalSignature, keyEncipherment`. The `extendedKeyUsage` includes `clientAuth` (for logging into the application) and `emailProtection`, according to RFC 5280[11]. The primary identifier (`commonName`) for user certificates is the user's email address. This email address is considered to be unique in the scope of this system. User certificates contain the extension `nsComment` in a slightly non-standard way to include the user-provided `first name` and `last name`. This is to account for the explicit requirement in the assignment stating that the certificate must be *"based on the (possibly corrected) user information"*.

The purpose (`keyUsage`) of server certificates includes `digitalSignature, keyEncipherment`. The `extendedKeyUsage` includes `serverAuth`.

For *simplicity*, certificate transparency is not used. No timestamp server is used. No post-quantum cryptography is being used for key generation or digital signatures, even though such algorithms have recently been announced[12].

1.3.9 Key management

Root CA private keys are stored in an offline backup (bank vault) and can only be accessed by at least two employees of the company, one of which must be an executive (C level). The private key of the Intermediate User CA must be available on the core CA machine to issue certificates. The private key of the

[9] `https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf`
[10] `https://apps.nsa.gov/iaarchive/programs/iad-initiatives/cnsa-suite.cfm`
[11] `https://www.rfc-editor.org/rfc/rfc5280#section-4.2.1.12`
[12] `https://www.nist.gov/news-events/news/2022/07/nist-announces-first-four-quantum-resistant-cryptographic-algo`

Intermediate Server CA may be available on the core CA machine, even though it is not regularly used to issue new server certificates. The private keys of intermediate CAs should be stored on a separate hardware device (e.g. TPM, HSM). In our work, we simulate a *hardware security module* (HSM) by using the storage location `/mnt/hsm`.

Private keys are stored in encrypted form, with a password written to a separate file. Private keys of server certificates are present on the machine where they are being used and are not encrypted. Private keys of user certificates are stored in plain-text on the core CA machine, protected only by file permissions. We favored this approach due to the simplicity and the fact that the key file would need to be stored somewhere near anyway. When a user requests to download a certificate, the PKCS#12 bundle is created with the private key encrypted by a password. This is to prevent the user from saving the private key on their device without encryption. In a full deployment, we would consider using a secret management solution like *Hashicorp Vault*[13], however this is out of scope for this project.

For SSH keys, we use Ed25519 keys generated by the openSSH command `ssh-keygen` (256 bit key length).

Passwords are used to access the frontend application. Passwords must be stored as (strong) password hashes in the database and must never be exposed in plain-text (e.g. inside log files). To ensure the principle of *generating secrets*, a password policy is enforced by the application (at least 8 characters long, of which 1 uppercase, 1 lowercase, 1 number, 1 special character). If a user forgets their password (and cannot log in using the certificate), the password can only be reset by a sysadmin. This is because the scope of the assignment does not provide an alternative method of authenticating users.

1.3.10 Cryptography

A pseudo-random number generator (PRNG) needs a good source of randomness to generate strong key material (e.g. TLS handshake (EC)DHE key shares, RSA keys). If available, use a true random number generator (TRNG) inside an HSM. Use recommended algorithms and key lengths, as indicated by industry standards (see above).

1.3.11 Integrity and Availability

Data on disk is protected against physical failures (e.g. disk failures, bit flips by cosmic rays) by using low-level error-correction codes on disk controllers. Redundant storage (e.g. RAID10) is used to ensure availability of data. Moreover, the chosen file-system `ext4` features some error detection mechanisms[14].

Backups are periodically (once a week) stored physically on-site (off-line) and a copy is stored in a separate location (once every 3 months). To secure

[13] `https://www.hashicorp.com/products/vault`
[14] `https://www.kernel.org/doc/html/latest/filesystems/ext4/overview.html`

against physical attacks, all data at rest is secured by disk encryption. However, memory on servers is not encrypted, as is common practice.

1.3.12 User Education and Awareness

Human users pose a certain risk to any system. End users must be made aware of their duties and potential problems. This is achieved by regular training and awareness campaigns, as well as informational notes on the frontend website. Examples include: *"Please do not click on links in suspicious phishing emails. Use a strong password. Use a password manager. Do not share your password with anyone. Back-up your private key on an offline device. Never store your private key unencrypted."*

1.4 Components

This section lists the hosts with their properties and key software components. We only use IPv4 for simplicity. No DHCP or DNS is used. Host names are defined in `/etc/hosts` for better *usability* and to allow the use of server TLS certificates.

1.4.1 Client [client.imovies.ch]

This host runs Linux Ubuntu Desktop OS with a graphical user interface (GUI). The default web browser is Firefox. This host resides in the external network and has the IPv4 address 192.168.57.100/24. Only one user with log-in enabled: vagrant. There is no hardening or other security configuration on this device because it is outside the scope of this assignment.

1.4.2 Frontend [imovies.ch]

This host runs the same OS as the client for simplicity of development. However, frontend does not offer a GUI to minimize exposure. As a firewall, we use iptables. As a management interface, we use SSH (openSSH software). The web application is written in Python using the framework Flask. Gunicorn is used as WSGI server and Nginx as web server. This host is dual-homed, i.e. it has two network interfaces. The IPv4 address on the external network is 192.168.57.101/24. The IPv4 address on the internal network is 192.168.56.101/24. Notable users include root, web application user, sysadmin. This host also serves as a jumphost for admins to access other hosts via SSH. Based on the principle of *compartmentalization*, this machine is almost stateless, i.e. no persistent data is stored here except for configurations and logging information.

1.4.3 Database [database.imovies.ch]

This host runs the same OS, firewall, management interface, web application framework, WSGI and web servers as frontend. This host resides in the internal

network and has the IPv4 address 192.168.56.103/24. Notable users include root, database user (which also runs the web application), sysadmin. The frontend application authenticates itself at every request with an API Token. The database content is dumped to a file in regular intervals to be collected by the backup process. Data is initialized as mentioned in 1.3.2. One reason to implement the "Database" as a micro-service was to be able to easily migrate to a different database management system once the legacy database will be deprecated.

1.4.4 Core CA [core.imovies.ch]

This host runs the same OS, firewall, management interface, web application framework, WSGI and web servers as frontend. This host resides in the internal network and has the IPv4 address 192.168.56.102/24. Notable users are root, CA user, sysadmin, Backup user. The frontend application authenticates like described in database. This host contains standard files for CA management (config, index, certificates, private keys, root CA cert, CRL, etc.). A simulated HSM stores the intermediate CA's private keys.

1.4.5 Backup [backup.imovies.ch]

This host runs the same OS, firewall, management interface as frontend. The main backup software are rsync, Syslog-ng, GPG encryption. This host resides in the internal network and has the IPv4 address 192.168.56.104/24. Notable users include root, Backup user, sysadmin. The backup process authenticates to other hosts using SSH key and pulls the backup files. Syslog-ng receives logs from other hosts and store them locally, implementing an anti-tamper technique and encrypting sensitive data. See section 1.3.5 for more details.

1.5 Backdoors

"Left-over" user (simple) We have an undocumented "left-over" user called `admin` (password: `admin`) in the frontend machine and on the backend machine that is accessible over ssh. This user has no root privileges but has unrestricted shell access to the system. An attacker may guess this common username-password pair and try to elevate their privileges or to move laterally.

How to exploit: Open a shell on the client, then `ssh -J admin@imovies.ch admin@backup.imovies.ch` to directly connect to the backup server. On there, an unencrypted dump of the database can be found which violates the *secrecy* requirement of the user data.

How to mitigate: Remove `admin` user from the systems.

Webshell (medium) A webshell is a method of gaining persistence and easy access to a target system by the attacker. A call to a specific URL will make the application run commands on the OS and return the result. Webshells are

widely used as of 2022, with many examples being available[15]. An attacker may perform lateral movement and upload malicious code. This backdoor is not easily detectable in blackbox testing without access to source code because the URL may be guessed. Detectable by blue-team e.g. through reviewing source code or through frequency analysis of HTTP access logs.

In our case, we place a webshell on the frontend server through which the host can be accessed. We use an innocent-looking URL and parameter name. No authentication is performed, but could easily be added (e.g. by checking for a certain parameter, password or `user-agent` string).

How to exploit: On client, clone and install repository PyShell[16], then run `python pyshell.py -p cert_id https://imovies.ch/list_certs GET`.

How to mitigate: Remove the backdoor from code[17].

Psychic signatures (hard) Inspired by the vulnerability *Java Psychic Signatures* (CVE-2022-21449[18]), we adapt the concept to RSA signatures. We compiled Nginx and openSSL to accept all certificates with all-zero signatures as valid. An attacker may use manipulated certificates to gain unauthorized user or admin access to the system.

How to exploit: Based on a valid, known certificate, craft new certificate with attacker-controlled private key and modify fields to taste. Add an all-zero signature and use client-side TLS to log in. You might need to use `curl` as browsers like Firefox validate the certificate before inserting it into the storage.

How to mitigate: Fix the source code in openSSL[19], then recompile Nginx with openSSL. Or: Remove and re-install Nginx and openSSL packages from official Ubuntu repository.

1.6 Constraints given by Assignment

The delivered database contains passwords in the form of unsalted SHA-1 hashes. This is against best practices, as SHA-1 is considered broken. Passwords should be stored as salted hash using a hash function designed with space and time requirements, like bcrypt or PBKDF2. This measure helps protect against an adversary that runs an offline password recovery attack against a database of leaked passwords. To illustrate this problem, we use publicly available databases[20] to recover the passwords (i.e. pre-images) by searching for the hash value of e.g. `SHA1(Astrid)`. Although we wanted to change the hash function, we were explicitly told[21] that this is not allowed.

[15] `https://www.cisa.gov/uscert/ncas/alerts/TA15-314A`
[16] `https://github.com/JoelGMSec/PyShell`
[17] Affected file: `source/frontend/app/app.py`, line 559
[18] `https://security.berkeley.edu/news/psychic-signatures-vulnerability-java-cve-2022-21449`
[19] Affected file: `crypto/asn1/a_verify.c`, line 169
[20] like `https://crackstation.net/`
[21] By Christoph Sprenger during the consultation session on the 03.11.2022

2 Risk Analysis and Security Measures

2.1 Assets

Physical Assets

Servers: The servers along with their software are the main components of the system. They are all dedicated servers in a data center of Equinix in a dedicated rack. The data center operator manages the hardware and replaces defective parts. Additionally, the data center operator performs the access control and keeps logs.

Internal Network: The internal network is a local area network in the server rack where the servers are located and connects the servers of the CA. The management is performed by the system administrators.

Internet Connectivity: The connection from the router (not modeled) that is placed in the server rack to the internet over an ISP's network. It is an essential component as it is the only connection to the clients. The availability is of key importance to the functionality of the CA and to let partners check the validity of certificates via CRL. iMovies has a service contract with an ISP guaranteeing 99.9% availability at a Bandwidth of 1 Gbit/s.

Hardware Security Module (HSM): nShield Connect HSM is used to store passwords, private keys and to perform cryptographic operations. Integrity and secrecy are the main security properties.

Software

Server Operating System: The operating system for all servers is the server version of Ubuntu 20.04 LTS. The management and maintenance is performed by the system administrator in regular intervals. The main security property is integrity, so that no malware is present on the system.

Web Server: Nginx web servers that serve the frontend, core and database applications. They are maintained and configured by the system administrator. The main security properties are the availability of the server and the integrity of the software.

Frontend Application: A Flask web application that serves the website and communicates with the core and database application over a narrow API. It is managed by the system administrator.

Backend Applications: Flask web applications that serve different specialized functionalities to the frontend. It is managed by the system administrator and the states are equivalent to the server OS. It either interfaces with a MySQL database or with OpenSSL. The integrity of these services is important, as is the basis for the CA.

Backup Scripts: A set of bash scripts using rsync[22] to pull relevant data off the servers. The scripts are scheduled using cron[23] jobs. All stored data is GPG-encrypted. Its security property is integrity to ensure the backup process followed.

MySQL Database: The MySQL Server database contains the user list. It is a legacy system, and its security properties are secrecy and integrity.

Information

Root CA Private Key: Used to generate the certificates for Intermediate CAs. It is stored offline on a storage device inside a bank vault. It must remain secret and accessible.

Intermediate CA Private Keys : These are the private keys that are used to sign the CSRs of user and server certificates. They must remain secret and accessible.

User Private Keys: The private keys are generated by the Core CA and provided to the requesting user of *iMovies*. They must remain secret and accessible.

Certificates: Certificates bind a public key to an *iMovies* entity. They have no special security requirements, as they are intended to be publicly verifiable.

User Data: The user data is of high importance to *iMovies*. The main focus is on the user password that can be used to log in. The desired properties are integrity and secrecy.

Backup Data: All data that is gathered for the backup. This includes mainly logs, private keys and user data.

Administrator credentials: The administrator credentials are used to manage the servers. This includes username and password for local system access and key pairs for remote access over SSH.

Database Administrator password: This password is used to initialize and manage the database on the server. It is stored in the bank vault.

Backup Private Key: The GPG private key that is used to decrypt the backed data. It is stored offline on a storage device inside a bank vault.

People

***iMovies* Employee:** Most *iMovies* employees are journalists focusing on investigative reporting. They are the principal users of the CA.

[22] `https://man7.org/linux/man-pages/man1/rsync.1.html`
[23] `https://man7.org/linux/man-pages/man1/crontab.1p.html`

CA Administrator: The CA Administrator is a special role of an *iMovies* employee that has access to the CA Administrator interface.

System Administrator: The system administrator is responsible for managing the server's software and the internal network. The system administrator has access to all data on the system and must be trusted.

Intangible Goods

Company Reputation: The credibility of the company with the general public and customers is the basis for its market success.

User confidence: The trust the users of the CA have towards the system. This is influenced by the usability of the entire CA system.

2.2 Threat Sources

The following threat sources were identified:

Nature: The data center is located in Zurich, Switzerland. Although this location has a low risk for natural disasters, events like earthquakes or power outages may occur that disturb the data center's operations.

Benign User: The employee does not intend to harm the company, but might act in negligence due to time pressure in his investigations.

System Administrator: A system administrator might try to leak confidential information to the public or other parties. They can sabotage the system with their access privileges and knowledge. They are motivated by personal factors, monetary compensation or the hope of fame.

Script Kiddie: Since the service is connected to the internet, script kiddies can test their tools and look for reactions from the service. They are motivated by thrill, short-term fame and curiosity.

Party Under Investigation: Parties under investigation form the primary threat source, as they are interested to detect under-cover journalist or whistle-blowers and disrupt investigations against them.

Skilled Hacker: Skilled hackers might be hired by competing organizations, government agencies or parties under investigation. They are mainly monetarily motivated and make use of advanced knowledge, as well as their ability to write custom tools to exploit vulnerabilities.

Government Agency: Government agencies are interested in monitoring and manipulate investigations to be able to influence public opinion and protect international relations. They have a large budget and highly skilled teams that have access to *zero-day exploits*.

2.3 Risks Definitions

We provide risk definitions based on the lecture book[24].

Likelihood	
Likelihood	Description
High (H)	The threat source is highly motivated and sufficiently capable of exploiting a given vulnerability in order to disrupt or modify the intended system functionality. The controls to prevent the vulnerability from being exploited are ineffective.
Medium (M)	The threat source is motivated and capable of exploiting a given vulnerability in order to disrupt or modify the intended system functionality, but controls are in place that may impede a successful exploit of the vulnerability.
Low (L)	The threat source lacks motivation or capabilities to exploit a given vulnerability in order to disrupt or modify the intended system functionality. Another possibility that results in a low likelihood is the case where controls are in place that prevent or at least significantly impede the vulnerability from being exercised.

Impact	
Impact	Description
High (H)	The event (1) may result in a highly costly loss of major tangible assets or resources; (2) may significantly affect *iMovies* investigations; or (3) may result in serious damage to human assets of *iMovies*.
Medium (M)	The event (1) may result in a costly loss of tangible assets or resources; (2) may affect *iMovies* investigations; or (3) may result in damage to human assets of *iMovies*.
Low (L)	The event (1) may result in a loss of some tangible assets or resources or (2) may slightly affect *iMovies* investigations.

[24] `https://ethz.ch/content/dam/ethz/special-interest/infk/inst-infsec/information-security-group-dam/education/as2022/seclab/asl-book-as2022.pdf`

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

2.4 Risk Evaluation

2.4.1 *Evaluation Asset: Servers*

No.	Threat	Countermeasure(s)	L	I	Risk
PAS1	A natural disaster (e.g. hurricane, earthquake, heat wave) damages server components, cooling facilities, electricity supply, etc.	Maintain system backups offsite so that the data center can be quickly changed and services reinstated.	*L*	*H*	*L*
PAS2	A malicious system administrator or a government agent infiltrates the data center and sabotages servers.	Maintain system backups offsite so that services can be quickly reinstated on a new set of servers. Inspect the access logs of the data center regularly. Restrict knowledge of server location to system administrators and a selection of executives.	*L*	*H*	*L*
PAS3	Physical failure of hardware components.	Maintain system backups offsite so that services can be quickly reinstated on a new set of servers. Server storage on SSDs in a RAID 10 configuration. Use redundant power supplies. Ensure that a service team is available quickly.	*L*	*M*	*L*

2.4.2 *Evaluation Asset: Internal Network*

No.	Threat	Countermeasure(s)	L	I	Risk
PAIN1	Physical failure of network components (caused by nature or unwitting people).	Use redundant network cables and switches such that the failure of a single cable or switch does not impact the network functionality. Ensure that an emergency team is 24/7 available to exchange damaged hardware. Use commodity hardware that is widely available, and keep spare parts in the server rack for fast replacements.	*L*	*M*	*L*
PAIN2	System administrator misconfigures network switches.	Use an exact duplicate of network configuration in integration environment. Automate and test all configuration.	*L*	*L*	*L*
PAIN3	A malicious system administrator or a government agent infiltrates the data center, cuts the network cables.	Use monitoring software to check the network connectivity. Inspect the access logs of the data center regularly. Restrict knowledge of server location to system administrators and a selection of executives.	*L*	*H*	*L*

2.4.3 *Evaluation Asset: Internet Connectivity*

No.	Threat	Countermeasure(s)	L	I	Risk
PAIC1	Physical damage to ISP links (caused by nature or humans).	Use redundant links to the ISP.	*L*	*H*	*L*
PAIC2	A ISP system administrator misconfigures the BGP protocol and thus revokes the server's public IP from the internet.	Use multiple ISPs. Use SCION to be able to choose network paths.	*L*	*M*	*L*

2.4.4 *Evaluation Asset: Hardware Security Module (HSM)*

No.	Threat	Countermeasure(s)	L	I	Risk
PAPPF1	A privileged insider accesses the HSM.	Access to the server rack is restricted to *iMovies* executives and the system administrator. HSM has physical tamper-protection that prevents data from being extracted.	*L*	*H*	*L*
PAPPF2	Physical damages to HSM	Keep a copy of the HSM's content in a bank vault.	*L*	*H*	*L*

2.4.5 *Evaluation Asset: Server Operating System*

No.	Threat	Countermeasure(s)	L	I	Risk
LASOS1	A Government agency uses a zero-day exploit to gain root access and install a root kit.	Regularly inspect access logs. Perform awareness trainings with the system administrators.	*M*	*H*	*M*
LASOS2	A skilled hacker exploits a recently found vulnerability in the operating system to gain root access.	Regularly inspect access logs. Update OS and other packages frequently (at least weekly).	*M*	*M*	*M*
LASOS3	A system administrator misconfigures the OS such that unauthorized users gain access to the system.	Perform no manual configurations and use scripts that were tested and validated to configure the OS. Inspect the system logs regularly.	*M*	*M*	*M*
LASOS4	A skilled hacker exploits a weakness in the firewall and gains access to the internal network.	Encrypt the traffic in the internal network. A notable exception is unecrypted syslog traffic.	*M*	*M*	*M*

2.4.6 *Evaluation Asset: Web Server*

No.	Threat	Countermeasure(s)	L	I	Risk
LAWS1	A skilled hacker uses an exploit in the SSL module to validate fake certificates for user authentication.	Perform regular maintenance and update to the latest version.	*L*	*M*	*L*
LAWS2	A system administrator misconfigures the web server, allowing unauthorized access.	Perform configuration checks and tests before deployment. Monitor the logs.	*L*	*M*	*L*
LAWS3	A skilled hacker manipulates the time source used by the web server to accept expired certificates.	Use multiple authenticated timeservers over NTS. Use a secondary time source (GPS receiver).	*L*	*L*	*L*

2.4.7 *Evaluation Asset: Frontend Application*

No.	Threat	Countermeasure(s)	L	I	Risk
LAFA1	A script kiddie brute forces credentials to log into the system.	Add a CAPTCHA to the login to increase the effort of the attack using off-the-shelve components. Add an artificial delay to make login requests take longer.	*M*	*M*	*M*
LAFA2	A skilled hacker modifies the served website to enable a CSRF attack.	Use randomized CSRF tokens to protect against CSRF attacks.	*M*	*L*	*L*
LAFA3	A skilled hacker performs an XSS to obtain the session cookie from an authorized user.	Perform thorough input validation. Do not use any JavaScript to minimize exposure.	*M*	*L*	*L*
LAFA4	A script kiddie sends malformed input data to the service in hope to crash it.	Enforce strict formatting rules for user input and validate it with commonly used libraries. Default to reject the input.	*H*	*L*	*L*

2.4.8 *Evaluation Asset: Backend Applications*

No.	Threat	Countermeasure(s)	L	I	Risk
LABA1	A script kiddie or skilled hacker performs an SQL injection attack to dump or modify the database.	Enforce the usage of prepared statements. Make the service only available through a narrow API that performs input sanitation.	*H*	*L*	*L*
LABA2	A script kiddie constantly requests the CRL to reduce the service availability.	Provision the service with enough resources.	*M*	*M*	*M*
LABA3	Benign user (developer) unwittingly inserts a security vulnerability into the production code.	Use static code analysis to check for errors. Test code before push to production with integration tests.	*H*	*M*	*M*

2.4.9 *Evaluation Asset: Backup Scripts*

No.	Threat	Countermeasure(s)	L	I	Risk
LABS1	A script kiddie performs a DoS attempt, generating large log files that fill up available space for backup.	The system administrator regularly deletes backup-ed logs. Provision application with enough resources.	*H*	*L*	*L*

2.4.10 *Evaluation Asset: MySQL Database*

No.	Threat	Countermeasure(s)	L	I	Risk
LAMD1	A skilled hacker uses an exploit or misconfiguration of the database server to gain administrative privileges.	Update the software regularly. Place the server in a private network that is not directly reachable from the internet.	*M*	*L*	*L*

2.4.11 *Evaluation Asset: Root CA Private Key*

No.	Threat	Countermeasure(s)	L	I	Risk
IARCPK1	Skilled hacker or Government agency gains access to the asset	Store the asset on a password protected storage device in a bank vault.	*L*	*M*	*L*
IARCPK2	A system administrator hands over the asset to a malicious party.	Document the usage of the asset and enforce that at least two authorized people are present. Perform regular background checks on the system administrators.	*L*	*M*	*L*

2.4.12 *Evaluation Asset: Intermediate CA Private Keys*

No.	Threat	Countermeasure(s)	L	I	Risk
IAICPK1	A skilled hacker gains access to the private keys.	Save intermediate CA private keys on an HSM.	*L*	*H*	*L*

2.4.13 *Evaluation Asset: User Private Keys*

No.	Threat	Countermeasure(s)	L	I	Risk
IAUPK1	A benign user makes his private key available to a third party by accident or negligence.	Regular training and awareness campaigns for users are performed. In case of a detected breach, users can revoke their certificates.	*M*	*M*	*M*
IAUPK2	A skilled hacker obtains the private key through social engineering or spear fishing.	Users are trained regularly and can revoke their certificates. A revoked certificate cannot be re-activated.	*H*	*M*	*M*
IAUPK3	A involved party bribes an employee to hand over a copy of the private key.	Regularly perform background checks on the employees. Revoke certificates of suspicious employees.	*M*	*M*	*M*
IAUPK4	A skilled hacker gains access to the private user keys that are stored unencrypted on the core server.	Restrict access to a minimum set of users on the server. Save data at rest using disk encryption.	*M*	*M*	*M*

2.4.14 *Evaluation Asset: Certificates*

No.	Threat	Countermeasure(s)	L	I	Risk
IAUCE1	A government agency can forge a certificate using huge computational resources or insider knowledge.	Use industry standard algorithms for certificate generation. Have a short lifespan for certificates.	*L*	*M*	*L*

2.4.15 *Evaluation Asset: User Data*

No.	Threat	Countermeasure(s)	L	I	Risk
IAAUC1	A benign user chooses a weak password.	Enforce a password policy specifying password properties and disallow updates with weak passwords.	*H*	*L*	*L*
IAAUC2	A skilled hacker gains access to the user data stored in the database.	Separate the database from the internet. Specify a narrow API to communicate with the database.	*M*	*M*	*M*

2.4.16 *Evaluation Asset: Backup Data*

No.	Threat	Countermeasure(s)	L	I	Risk
IABD1	Skilled hacker or government agency gains access to the backed-up data	Encrypt the backup. Store the key for decryption offline.	*M*	*L*	*L*

2.4.17 *Evaluation Asset: Administrator Credentials*

No.	Threat	Countermeasure(s)	L	I	Risk
IAAC1	A party under investigation bribes the system administrator to leak his SSH keys.	SSH keys are regularly changed. Access is monitored.	*M*	*H*	*M*

2.4.18 *Evaluation Asset: Database Administrator Password*

No.	Threat	Countermeasure(s)	L	I	Risk
IADPC1	A skilled hacker with access to the database server uses to database administrator password to obtain and modify all user data.	Store the password offline on an HSM in a bank vault.	*L*	*H*	*L*

2.4.19 *Evaluation Asset: Backup Private Key*

No.	Threat	Countermeasure(s)	L	I	Risk
IABPK1	A skilled hacker with access to the backup server uses the backup GPG private key to read the plaintext data.	Store the GPG private offline on an HSM in a bank vault.	*L*	*H*	*L*

2.4.20 *Evaluation Asset: iMovies Employee*

No.	Threat	Countermeasure(s)	L	I	Risk
PAIEMP1	A skilled hacker employs social engineering techniques to make an employee download malicious software to their end device, thus, gaining access to the user credentials.	Regularly perform mandatory awareness trainings.	*H*	*M*	*M*

2.4.21 *Evaluation Asset: CA Administrator*

No.	Threat	Countermeasure(s)	L	I	Risk
PACA1	A party under investigation bribes the CA Administrator to reveal the number of active certificates to deduce the number of active journalists.	Force the CA Administrator to sign a non-disclosure agreement. Regularly perform background checks.	*M*	*L*	*L*

2.4.22 *Evaluation Asset: System Administrator*

No.	Threat	Countermeasure(s)	L	I	Risk
PACSA1	A government agency exerts pressure onto the system administrator to hand over user data.	Perform regular background checks on key employees to detect potential susceptibility for pressure.	*L*	*H*	*L*
PACSA2	A party under investigation bribes a system administrator to leak data of *iMovies* employees that are part of the investigation.	Perform a background check on the system administrator during the hiring process and check for moral integrity.	*H*	*H*	*H*

2.4.23 *Evaluation Asset: Company Reputation*

No.	Threat	Countermeasure(s)	L	I	Risk
TGACR1	A skilled hacker publishes data obtained from an attack to discredit the company.	Employ all technical and operational countermeasures described in this chapter to avoid this situation. Have a communication strategy available to respond quickly.	*L*	*H*	*L*

2.4.24 *Evaluation Asset: User Confidence*

No.	Threat	Countermeasure(s)	L	I	Risk
IGAUC1	Script kiddies damage the user's confidence by making the service unavailable due to DDoS attacks.	Over-provision server resources.	*H*	*M*	*M*
IGAUC2	Insider leaks personal data of users to the public.	Immediately revoke access credentials after an employee leaves. Perform background checks on the system administrators.	*M*	*H*	*M*

2.4.25 Risk Acceptance

We describe further countermeasures that could be implemented against all risks that are medium or high.

No. of threat	Proposed additional countermeasure including expected impact
LASOS1	Hire contractors in regular intervals to perform vulnerability scans and penetration tests to find security vulnerabilities. Using the resulting reports, undetected configuration flaws can be fixed.
LASOS2	Deploy an IDS and IPS to find anomalies automatically. Employ malware protection (antivirus, sandboxing).
LASOS3	Add a web application firewall (WAF) to block previously unknown attacks.
LASOS4	Encrypt all traffic in the internal network.
LAFA1	Block users after 10 unsuccessful login attempts and email them with a reactivation link to unblock the account.
LABA2	On frontend, cache CRL response from core. Only request new CRL every 5 minutes or if a certificate has been revoked.
LABA3	Use an automated pipeline with unit-, integration- and end-to-end tests.
IAUPK1	Provide to users an HSM or Smart Card where the certificates and private keys are stored securely. Provide an application that directly seals the certificate-key pair into the Smart Card such that it is not in the user's possession.
IAUPK2	Same as IAUPK1.
IAUPK3	Same as IAUPK1.
IAUPK4	Encrypt user private keys on an HSM or password-protect them on the server with a password that is protected by an HSM.
IAAUC2	Use table level encryption to protect the data at rest. Use a state-of-the-art hash function and salts instead of SHA1 to store the passwords.
IAAC1	Use Smart Cards to securely store the private keys. The absence of a Smart Card can easily be detected, and the duplication requires highly skilled resources.
PAIEMP1	Add an emergency hotline where employees can revoke certificates and get assistance after being a victim of an attack.
PACSA2	Pay high salaries to make staff less susceptible to bribes. Rotate key personnel strategically to make social engineering harder. Perform surveillance on own employees (morally questionable).
IGAUC1	Use Cloud providers such as Cloudflare to assist in the reduction of severeness of DDoS attacks.
IGAUC2	Same as PACSA2.