System Description and Risk Analysis

Valentyna Pavliv Manuel Meinen Yann Gabbud
Alexandre Délèze

November 26th, 2020

Contents

1 System Characterization

1.1 System Overview

The goal of this project is to permit an employee of the company *iMovies* to request a certificate and a private key to run applications with PKI. He must also be able to manage his personal information and to revoke his certificates.

We base our system on a micro-service architecture. Each service runs on its dedicated server/VM. Figure 1 is the illustration of our architecture.

We have separated the VMs into three different networks. The first one is the DMZ network, which contains the *Web Server*, the *VPN Server*, the *Firewall*, and to which the *Client* connects. The second one is the network between the *Firewall* and the *Core Server*. The last one is the internal network containing the *Firewall*, *MySQL*, *Core*, *CA*, and *Backup* Servers.

The first VM is the *Client*, which is used by the end-user to access the different system functionalities described in subsection 1.2. For this, the user can use Firefox to access the *Web Server* and the command-line to administrate the other VMs via SSH.

We use the *Web Server* as the frontend of our system to serve the different webpages.

The *VPN Server* allows the *Client* to be able to manage the different components of our system.

The *Firewall* blocks all the non-allowed traffic and forwards the authorized one to the internal network and to the shared network with the *Core Server*.

The *Core Server* is in charge of the logic of the system. It receives the requests from the *Web Server*, checks and administrates the session management and the access control, asks the *CA Server* to create new certificates, and push/fetch information to and from the *MySQL Server*.

We use the *CA Server* to generate the new certificates and to revoke them if necessary.

The *MySQL Server* is our database system where we store the personal information of our users and their valid and revoked certificates.

Finally, the *Backup Server* stores a copy of the *MySQL Server*'s databases, the useful configurations and logs of the different servers, and also the user's private keys used to generate the certificates.

1.2 System Functionality

1.2.1 Components

Client The *Client* offers a web-browser (Firefox) to connect to the *Web Server*. It also contains an administrator certificate to access the admin page. Wireguard is installed on this machine to create a tunnel to the *VPN* to be able to manage all VMs via SSH.

Web Server The *Web Server* serves three webpages. The first one is the login page that the user can use his credentials (username and password) or

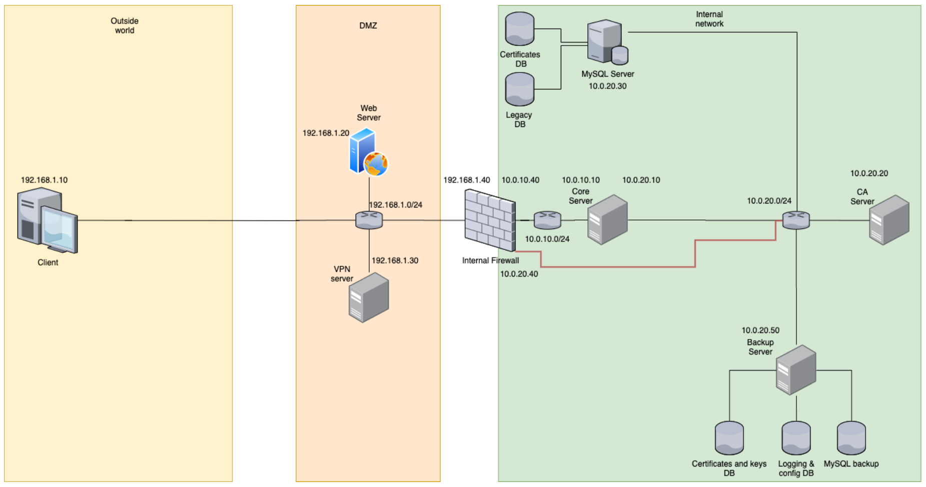

Figure 1: Network architecture (with red lines for the access of the *VPN Server*)

the client-side authentication with his valid certificate. The second page is the home page containing a link to the certificate issuance page, a button to revoke the user's valid certificate, a button to download the certificate revocation list, a button to access the admin panel if the user is authenticated as administrator using the client-side authentication and a log out button.

On the certificate issuance page, the user can modify his private information and requests his PKCS#12 if he has no valid certificate, or otherwise downloads his public certificate. The user can download his PKCS#12 only once and can have only one valid certificate at a time.

On the admin panel, we can observe the actual serial number, the number of issued certificates, and the number of revoked certificates.

It offers the revocation list to all users and does not need to log in to access it.

VPN Server The *VPN Server* runs Wireguard to build a VPN tunnel between the *Client* and the *VPN Server*. It then plays the role of a NAT device for the *Client* to connect to the other components of our system. The *VPN Server* can have SSH access to all other servers and bypass the *Firewall*, allowing the *Client* to have the same privilege as the *VPN Server*.

Internal Firewall The internal *Firewall* filters the communication between the DMZ, the internal network and the one between the *Core Server* and the *Firewall*. It forwards all SSH traffic between the *VPN Server* and the other VM's inside the internal network. It also forwards the SFTP traffic between the *Backup Server* on one side and the *VPN Server* and the *Web Server* on the other side. Lastly, it allows the HTTPS traffic between the *Core Server* and the *Web Server*.

Core Server The *Core Server* is used to manage the logic of the system. It offers interfaces to the *Web Server* to make the different operations. For the login, it checks either the credentials (username/password) or the certificate that it fetches from the *MySQL Server*'s database. It always uses SHA1 to hash all provided password. Once a user is connected, it creates a new cookie that the user can use to reauthenticate while navigating on the website.

For the certificate issuance, it creates a new pair of public/private keys and creates a CSR that it sends to the *CA Server*. Once it receives the corresponding certificate, it creates a PKCS#12 with the certificate and the private key and forwards it to the *Web Server*. It stores the private key in a temporary folder that the *Backup Server* will read and empty. Finally, it sends the new certificate to the *MySQL Server*.

The *Web Server* can also send a request to modify the personal information and to fetch the valid certificate of a user.

For the process of revocation, it fetches the user's certificate from the *MySQL Server*'s database, sends an order to the *CA Server* to revoke this certificate and asks the *MySQL Server* to mark the certificate as revoked in the database. Then it offers the revocation list to the *Web Server*.

Finally for the admin panel, it fetches the information about the amount of issued and revoked certificates from the *MySQL Server* and from the *CA Server* it fetches the serial number.

CA Server The *CA Server* has three functionalities. The first one is to create a certificate from a certificate signing request (CSR) and to send it back to the *Core Server*. The second is to receive a certificate from the *Core Server* and to revoke it and update its revocation list. The last one is to furnish the serial number and the revocation list to the *Core Server*.

MySQL Server The *MySQL Server* is the database system of our network. It contains the dump database with the personal information of the company's employees (including username and the hash of the password), a table with a correspondence between the employees and if they are CA administrators, and one with the issued certificates with their owners and with the revocation status.

Backup Server The *Backup Server* starts a new backup process every hour via SFTP. It backs up the configuration files and logs of the other components, backs up the private keys contained in the *Core Server* and the *MySQL Server*'s databases.

1.2.2 User Authentication

A user can choose to identify himself via traditional credentials (username and password) or with the client-side certificate. To connect, the *Client* calls *https://webserver/* and then selects a certificate, when Firefox asks to or writes his credentials. Then, the *Web Server* transmits this information to the *Core*

Server. Which then asks the *MySQL Server* to which user the certificate or the hash of the password for a particular username corresponds. If everything goes well, it creates a cookie containing the username, a timestamp, a nonce, a boolean to say if the connection was established using a certificate, and the signature of the hash of the concatenation of the afore-mentioned information. It puts a validity delay of ten minutes and sends the cookie back to the *Web Server*. The user can then use this cookie to reauthenticate itself while navigating inside the website.

Note also that with the client certificate authentication, if a user logs out and comes back to the website, he will automatically reconnect because the login with a certificate does not require any action from the user.

1.2.3 Personal information

Before asking for a new certificate, a user can update his personal information, i.e. last name, first name, e-mail address, and password. Before sending the data to the *Core Server*, the *Web Server* checks that the data is proper and not malicious.

1.2.4 Certificate issuance

To obtain a new certificate, the user must firstly login on the *Web Server*. Then he has to go to the certificate issuance webpage and update his personal information if needed. Then he simply has to click on the download button to obtain his new certificate. If the user does not yet have a certificate, the system sends him a PKCS#12 file containing both the certificate and the private key. However, if he already has a valid certificate, the system sends him only the certificate in a PEM file. A user can have only one valid certificate at a time.

1.2.5 Certificate revocation

If a user wants to or has to revoke his certificate, he must log in on the *Web Server* with his certificate or his credentials and click on certificate revocation on the home page. Once the certificate is revoked, a message confirming the revocation appears onscreen. If the user does not have a certificate, a message informs the *Client* that he does not have a valid certificate.

1.2.6 Revocation list

At anytime, the user can check the certificate revocation list of the system by clicking on the *Revocation list* button of the home page if he is logged in. Once he clicked, the user can download the revocation list. Non-connected users can download the list using the following URL:
https://webserver/revocation_list.

1.2.7 CA Admin Interface

This interface is only accessible by a user who has certificate authority administration rights. The interface shows statistics about the serial number and the number of issued and revoked certificates.

1.2.8 Logout

Once a user has finished what he has to do, he can logout from the website. The logout destroys the cookie.

1.2.9 Backup

To avoid single points of failure, all servers backup relevant data on the *Backup Server*. The data can be logs, certificates, keys, etc. In case of a failure of one of the servers, all necessary data to build back the server is present on the *Backup Server*.

1.2.10 System Administration

All servers can be administrated through SSH by the administrator of the system. To have access to these machines, the administrator has to create a tunnel to the *VPN Server*, and so a physical presence is not needed.

1.3 Security Design

1.3.1 Access control

The system enforces a strict access control policy. Each communication between two servers requires authentication of both parties. For all servers except the *Backup Server*, all certificates, keys, scripts, configuration files, or HTML templates are only readable by the user running the service. However, the administrator has read, write, and execute right on this data. For the *Backup Server*, the backups are only readable by root. Root privileges are needed to change the list of files and folders that will be backed up. The private key, which is used to set up the SFTP connections is only readable by root.

1.3.2 Session management

A session lasts for at most 10 minutes. The session is managed using a cookie. Once a user connects to the system, the *Core Server* generates and signs a cookie containing a timestamp. Each time a user makes a request, the *Core Server* checks the integrity of the cookie and checks that the session is not expired by comparing the timestamp in the cookie with the actual time. If the difference is more than 10 minutes, the user must authenticate again to continue interacting with the system. Once a user logs out, the Web Server terminates the session by destroying his cookie.

1.3.3 Key management

Servers store their private key, their certificate, and the CA root certificate in dedicated folders that are only readable by the user running the system. The signing key of the *CA Server* is stored in */etc/ssl/CA/private* and is only readable by the user running the *CA Server*. The *Client* is responsible for the management of his private keys and certificates.

The private keys created inside the *Core Server* to issue new certificates are stored temporally for one hour in the directory */backup_dir* until the *Backup Server* fetches and erases them. Only the user running the *Core Server* and the `backup_user` can read them.

1.3.4 Data in transit

Both communications between the user and the *Web Server* and communications between internal servers are encrypted. We use TLSv1.3 with only the most secure cipher suites between the *Client* and the reverse proxy and TLSv1.2 for communications between internal servers. These give us a guarantee of confidentiality and integrity for our data in transit. The *VPN Server* uses WireGuard that ensures a high level of security for the data in transit.

Inside the internal network, all unauthorized data-paths are closed. In other words, if the *Web Server* tries to send data to the *MySQL Server*, the *Firewall* will drop the packet because this data-path is not necessary for the system to work. Furthermore, we use two-side authentication to check that a Man-in-the-Middle can not spoof the IP address of a legit component and gives us a guarantee of authentication.

1.3.5 Data at rest

In each server, all certificates, keys, scripts, configuration files, or HTML templates are not encrypted. Only the *MySQL Server*'s database is encrypted.

1.3.6 User inputs

The *MySQL Server* sanitizes the user inputs with prepared statements before saving them in the database to avoid XSS and SQL injection. Furthermore, the *Client* or any other components do not execute any JavaScript. No external data is used in paths to avoid indirect reference. To mitigate the risk of CSRF, the cookies have an expiration delay of ten minutes, and the GET requests do not trigger any actions on the servers.

1.3.7 Software privilege

To avoid that an attacker can read/write/execute anything in the system in the case that he finds a vulnerability in one of our applications, we run all Flask servers with the privilege of the user `ubuntu`. There is an exception for the *CA Server* because it needs root privileges to run some OpenSSL CA commands.

The risk for the *CA Server* is mitigated because only the *Core* and the *VPN* Servers can communicate with it.

1.4 Components

All servers run on Ubuntu server 18.04.5 LTS except the *Client*, which runs on Peppermint OS 10. All servers run an SSH server used for administration and to backup data through SFTP.

1.4.1 Web Server

Applications The *Web Server* runs an Nginx reverse proxy in front of a Flask based backend. The Nginx reverse proxy manages client authentication and HTTPS redirection while the backend process the request of the *Client* and forwards them to the *Core Server*. The *Web Server* is protected by an `iptables` firewall authorizing only the traffic through ports needed to communicate with the *Core Server* and the *Client*. All other ports are blocked.

Interfaces The *Web Server* has one network interface. It is the DMZ one and it allows the clients to reach the *Web Server* on the one hand and, on the other hand, the *Web Server* can communicate with the *Core Server* through the *Firewall*.

The web interface is built with plain HTML and forms with Flask-WTF. There is no CSS or JavaScript. The web interface does not provide any security measures. It only links the *Client* and the backend.

Data records The relevant data that needs to be stored are: the authentication and system logs of the system, the log of the Flask server, the access and error logs of the Nginx reverse proxy, and finally the private key and the certificates used to authenticate communications.

1.4.2 Core Server

Applications The *Core Server* runs a Flask server that processes requests from the *Web Server* and sends requests to the *CA* and *MySQL* Servers. It is protected by an `iptables` firewall authorizing only the ingress and egress traffic with the other servers. The *Core Server* also uses the python library Cryptography to generate CSR.

Interfaces The *Core Server* has two interfaces: one with the *Firewall* that is used for communication with the *Web Server* and one for communication with the internal network.

Data records The relevant data that needs to be stored are: the authentication and system logs of the system, the log of the Flask server, and the private key and the certificates used to authenticate communications.

1.4.3 CA Server

Applications The *CA Server* runs a Flask server that processes requests from the *Core Server*. It uses OpenSSL to generate and revoke certificates. It is protected by an `iptables` firewall authorizing only the ingress and egress traffic with the *Core Server* and the *Backup Server*.

Interfaces The *CA Server* has one interface used to communicate with the internal network.

Data records The relevant data that needs to be stored are: the authentication and system logs of the system, the log of the Flask server, the private key and the certificates used to authenticate communications, the certificate revocation list (CRL), a random seed used to generate the certificates, the root CA signing key, and the OpenSSL configuration file.

1.4.4 Backup Server

Applications The *Backup Server* uses PySFTP to get from all the other servers the data that needs to be backed up. It is protected by an `iptables` firewall, authorizing only the ingress and egress traffic with all the other servers.

Interfaces The *Backup Server* has one interface used to communicate with the internal network.

Data records The relevant data are the authentication and system logs of the system, the private key and the certificates used to authenticate communications, the configuration file, and finally, the lists of all files that need to be backed up for each server. Obviously, we do not back up these data records because they are already on the *Backup Server*.

1.4.5 MySQL Server

Applications The *MySQL Server* runs a Flask server to process requests from the *Core Server* . It runs a *MySQL Server* to store the database. It is protected by an `iptables` firewall authorizing only the ingress and egress traffic with the *Core Server* and the *Backup Server*.

Interfaces The *MySQL Server* has one interface used to communicate with the internal network.

Data records The relevant data that needs to be stored are: the authentication and system logs of the system, the log of the Flask server, the log of the MySQL database and the private key, the certificates used to authenticate communications, and the database itself.

The database contains three tables. The first table contains the users' private information: username, last name, first name, e-mail, and password of type varchar. The second table maps a username to an administrator status, which is represented as a boolean. The last one contains all issued certificates (in varchar 2048) with their owner's username, and their revocation status represented as boolean.

1.4.6 VPN Server

Applications The *VPN Server* runs WireGuard to allow remote connection to the internal network. It makes IP forwarding allowing the administrator to connect via SSH to all system's components. It is protected by an `iptables` firewall authorizing only the ingress and egress traffic with the other servers and ssh connection from the outside of the network through port 22.

Interfaces The *VPN Server* has one interface, which is used to communicate with the internal network through the *Firewall* and so to allow a remote user to use the tunnel to reach each internal component.

Data records The relevant data that needs to be stored are: the authentication and system logs of the system, the private key used to setup Wireguard and the configuration file of WireGuard.

1.4.7 Firewall

Applications The *Firewall* only runs the `iptables` firewall to filter all ingress and egress traffic from the outside and the internal network.

Interfaces The *Firewall* has three interfaces. The first one is open to the world, the second only to the *Core Server*, and the last to the internal network.

Data records The relevant data that needs to be stored are the authentication and system logs of the system.

1.5 Backdoors

In the first backdoor, we set an easy password (*IloveASL*) for the user `ubuntu` on the *VPN Server*. Furthermore, we modified the number of maximum trials per connection to 100'000 and the maximum number of sessions to 100'000 to allow a brute force attack.

The second backdoor is inside the *Core Server*. We run a service on port *11061* which allows the user to make `ls` and `cat` on the whole system as root user. To use this backdoor, you have to make a GET on *10.0.10.10:11061/surprise* with a JSON in the body with a field *greatsecret* containing the `cat` or `ls` command. Furthermore, we opened the corresponding port on the *Firewall* and the *Core Server* and enabled the forward rule on the *Firewall* to allow this

service to be accessible remotely. We can use this backdoor to fetch the user's private keys temporarily stored in `/backup_dir/` before they reach the *Backup Server*. To harden the access, the service is refused randomly two out of three times. Here is an example of an exploit:
curl -X GET -d '"greatsecret":"ls /home/ubuntu"' -H "Content-type: application/json" "http://10.0.10.10:11061/surprise".

1.6 Additional Material

1.6.1 REST interface for the Client

Path	Need to be logged	Response
https://webserver/	NO	Home page to connect
https://webserver/	YES	Main page with the different possible operation
https://webserver/account	YES	Get/Update personal information and download new certificate
https://webserver/ca_admin	YES	Information about the serial number and amount of issued and revoked certificates
https://webserver/revocation_list	NO	Revocation list of the *CA Server* in a .crl file

1.6.2 CA Administrator

We added a new user to the *MySQL Server*'s database to play the role of a CA Administrator. His PKCS#12 is loaded in the *Client*'s Firefox, and his credentials are `admin:KK38O!M=HiCC2Og9mS_gFgC`.

2 Risk Analysis and Security Measures

2.1 Assets

2.1.1 Physical Assets

In general, all machines under the control of the company must be protected from physical destruction and theft. Therefore, they should be kept in a locked room having the usual security and safety measures in place (smoke detectors, fire extinguishers, ventilation/cooling systems, surveillance cameras, motion detectors, etc.). Furthermore, only authorized personnel should be allowed access to the server room. Physical assets with stricter requirements are listed below.

Backup Server The *Backup Server* should be kept in a separate room, such that in case of a fire, flooding, or any other natural hazard, only either the *Backup Server* or all the other machines are affected but not both.

Client Computer It is the responsibility of the *Client* to make sure that his computer is not compromised such that the private key, as well as his password, are kept confidential.

2.1.2 Logical Assets

Software

Web Server We need to restrict access to the *Web Server*. External users should only be able to communicate through ports 443 and 80. The administrator should only communicate through port 22. The *Core Server* should only communicate through port 10443. All other ports must be closed. The administrator has read and write access to the file system (keys, configuration files, etc.). The user running the web service has only read access.

Core Server This server must implement strong access control and firewall. It communicates with all services inside the internal network and is, therefore, a single point of failure. We need to set up firewall rules on it, as well as limit the access to the file system. We will restrict the write access to the different keys to the root user and the read access to the user running the service.

VPN Server The directory containing the SSH public key must be owned by root with write access granted only to root. Furthermore, the Server should be updated.

CA Server We need to restrict the access by using the on-device firewall such that only connections to the *Core Server* and the *Backup Server* are allowed.

Firewall Rules Only an authenticated administrator can change the *Firewall* rules.

Backup Server Root privileges are necessary to start a new backup-process or to change the configuration of the *Backup Server* (like the list of files that are backed up from the remote machines).

MySQL Server Root privileges are necessary to open access to the database and change its content. Only the *VPN* and *Backup* Servers (as well as `localhost`) are allowed to communicate with it over HTTPS or SFTP, which is enforced with an on-device firewall.

Information

Keys and Certificates

- SSH Keys: Each server needs to store the SSH key of the administrator such that he can configure or manage them remotely.
- HTTPS Certificates and Keys: Each server possesses a key and an according certificate that it uses to authenticate itself with the other servers and encrypt its communications. A server only accepts communications from someone who can show a valid certificate. These certificates and keys are used only for internal communication. Therefore, we can use our root certificate to issue and sign certificates for each server. The user running the service on the server has read access on the key and the certificate, and the administrator has read and write access.
- Signing Key of the *CA Server*: We need a private key to sign the certificates that are issued. This key should be carefully protected because anyone in possession of it can issue his certificates. Therefore, the user running the service on the server has read access to the key and the certificate, and the administrator has read and write access.

Private Keys of Users Once a private key is created, and the *CA Server* has delivered a corresponding certificate, the *Core Server* sends the certificate and the private key back to the *Web Server* via HTTPS, and additionally, a backup of the private key gets transmitted via SFTP to the *Backup Server.* The private key is written into a folder, whose content is erased once the content has been backed up successfully. The private key needs to be kept confidential.

User Certificates We require that the certificates are issued only for strong keys, which means that they consist of at least 2048 bits. Furthermore, the user must be authenticated to issue a certificate.

Session Cookies Sessions of users are managed using session cookies. To obtain a session cookie from the *Core Server*, the *Client* needs to successfully authenticate itself using either a valid combination of a username and a password or a valid certificate. It is the client's responsibility to store the cookie in a way such that confidentiality are guaranteed to avoid session hijacking. While in transit from the *Core Server* to the *Client*, the cookie's confidentiality and integrity is protected by sending it over HTTPS. To make it impossible for an attacker to forge a session cookie, it is signed by the *Core Server* when created and only accepted as an authentication method if the signature can be verified. For security reasons, a session cookie expires after ten minutes.

Certificate Revocation List (CRL) The information on whether a certificate is revoked or still valid is maintained in two places. There is a classical CRL on the *CA Server*, where only the *Core Server* can add revoked certificates to the CRL. On the other hand, whether a certificate is revoked or not is also indicated in the certificate database on the *MySQL Server*. Also, in that case, only the *Core Server* is allowed to update the corresponding field. The *Core Server* only revokes a certificate if an authenticated user wishes to revoke it.

Backups All backups are transmitted via SFTP and are therefore encrypted in transit. On the *Backup Server*, the backups are only readable by a user with root privileges. On the remote machines, the `backup_user` needs to have read access to the files that need to be backed up.

Database Tables Database tables are handled by the *MySQL Server*'s database on the *MySQL Server*. Only the *Core Server* or a user with root privileges is allowed to deal with this data, otherwise, it is stored encrypted. If the *Core Server* requests some data, it is sent with HTTPS.

Connectivity

Connectivity in the Internal Network Connectivity between servers in the internal network is of the utmost importance. If the communication to one server becomes impossible, the entire system does not work as it should. For example, if the communication to the *Backup Server* becomes impossible, the backup process would fail and, therefore in case of a crash of a different server, data could be permanently lost. Therefore, the internal network is shielded by a *Firewall* on a dedicated machine.

Connectivity to the Internet If connectivity to the internet is lost, employees could no longer request certificates and since they are used to secure the e-mail communication of the company, e-mail communication would become impossible (or at least insecure).

2.1.3 Persons

Administrators Administrators can remotely access any machine via SSH by connecting to a *VPN Server* . Since they are allowed to have root access on any machine, they need to be extremely trustworthy.

Users Employees, which are the users of this system, are responsible for keeping their passwords and their privates key confidential. Otherwise, an attacker could impersonate them or eavesdrop on their e-mail communication.

2.1.4 Intangible Assets

Security of Employees Communication The e-mail communication of the employees needs to be authentic as well as confidential since a breach of authenticity or confidentiality could lead to financial damage for the company.

Availability of the Company's E-Mail Service If the CA service becomes unavailable, secure e-mail communication could be suppressed by an attacker. This could lead to the loss of customers or any other kind of financial damage.

2.2 Threat Sources

2.2.1 Competition

A competing company could do corporate espionage to get a heads up on a new story and to get a competitive advantage. Or they could try to compromise the operationality to gain new customers, which switch to the competition after being disappointed with the service they receive from iMovies.

2.2.2 Normal Cybercriminals

Cybercriminals with monetary interests could mount Ransomware attacks, conduct extortion on employees, or try to steal corporate secrets and sell them to the competition. Furthermore, they could commit a Fake President Fraud (FPF) by impersonating iMovie's CEO and trick the financial department into authorizing transactions to the attackers under the order of the CEO.

2.2.3 Employees

Employees that are unhappy with their working conditions might steal confidential data and sell it to the competition to hurt their own company. Or if they are corruptible, they might also conduct attacks against their own company on the behalf of any other threat source. Due to their physical access to the company's premises, employees are a very attractive collaborator for any kind of attack against the company. Furthermore, employees might also endanger the security of the company's infrastructure due to a lack of knowledge. For example, they might insert a USB stick they found on the street into a computer that has access to the internal network without any bad intentions.

2.2.4 Opposers of Investigative Journalism

If an investigative report exposes some injustice like corruption or any other illegal activity, then the exposed person, country, organization, or company might want to figure out the identity of the source to punish or discredit them. Furthermore, they might also try to prevent iMovies to ever release the report, which would expose them, or they could try to delete the proofs iMovies possesses that prove their guilt.

2.2.5 Script Kiddies

Since the system is connected to the Internet, script kiddies could want to try out the new scripts they found online for the sake of fun or recognition of their friends.

2.2.6 Governmental Agencies

Some governments might not agree with the content iMovies is publishing. Therefore, their Intelligence Agencies could want to spy on the company or try to find ways to turn iMovies silent in times of political tensions.

2.2.7 Nature

The physical assets are, of course, vulnerable to physical impacts caused by natural events. Server rooms can be flooded, they can catch fire, lightning can strike the building, earthquakes can let the building collapse, or dust might harm the hardware in the long run.

2.2.8 Malware

Malware that is being spread randomly over the Internet can by chance also infect servers of iMovies. The main concern is undirected malware since it is even a threat source in the absence of any concrete adversary of the company iMovies.

2.2.9 ETH Students

To get a good grade for the Applied Security Laboratory, some students may try to break the system.

2.3 Risks Definitions

The risk is defined to compare different events that may cause damage. More formally, the risk depends on the impact of an event and of the likelihood that

Likelihood	
Likelihood	Description
High	Attacker's goal can be achieved without breaking any defence mechanisms.
Medium	Attacker's goal can be achieved by breaking defence mechanisms with publicly known techniques.
Low	Attacker's goal can be achieved only by exploiting publicly unknown vulnerabilities.

Table 1: Definition of Likelihood Levels

Impact	
Impact	Description
High	The system gets completely put out of service or data normally accessible only by super users, e.g. passwords, is leaked or manipulated.
Medium	The system gets limited or some confidential data (e.g. username) is leaked or manipulated.
Low	Internal data gets leaked or the system experiences abnormal high usage.

Table 2: Definition of Impact Levels

this event happens. Thus, the formula we will use to compute the risk is as follows:

$Risk(event) = Impact(event) \times Likelihood(event)$

The difficulty of this formula is that we actually cannot compute the impact or the likelihood - it is too difficult to quantify them. What we will do instead is to approximate them with the following values: *low, medium, high*, where *high* $>$ *medium* $>$ *low.* The likelihood and impact levels are defined in tables 1 and 2.

	Impact		
Likelihood	Low	Medium	High
High	Low	Medium	High
Medium	Low	Medium	Medium
Low	Low	Low	Low

Table 3: Risk level according to Likelihood and Impact

2.4 Risk Evaluation

2.4.1 *Evaluation Asset Backup Server*

No.	Threat	Countermeasure(s)	L	I	Risk
1	Malicious user: access the stored data.	- Data is accessible only for super users.	*Low*	*Medium*	*Low*
2	Malicious user: install malware by code injection.	- *Backup Server* does not execute received data.	*Low*	*Medium*	*Low*
3	Malicious user: use software vulnerabilities to take control over the server.	- Use only software without known vulnerabilities.	*Low*	*High*	*Low*

2.4.2 *Evaluation Asset Client Computer*

No.	Threat	Countermeasure(s)	L	I	Risk
4	Malicious user: install malware on client's computer in order to impersonate the client.		*Medium*	*Medium*	*Medium*
5	Malicious user: could steal the computer with open session in order to impersonate the client.		*Medium*	*Medium*	*Medium*
6	Malicious user: could steal a cookie during the 10min it is valid.		*Medium*	*Medium*	*Medium*

2.4.3 *Evaluation Asset Web Server*

No.	Threat	Countermeasure(s)	L	I	Risk
7	Malicious user: code injection in the login form, leading to a switch from data context to code context.	- Sanitize the inputs of the user.	*Low*	*Medium*	*Low*
8	Malicious user: XSS attack by injecting malicious script into the viewed web page.	- Sanitize the inputs of the user.	*Low*	*Low*	*Low*
9	Malicious user: insecure direct object reference attack, by manipulating direct object reference in the URL to access unauthorized objects (for instance keys).	- Indirect object reference must be avoided. - If it is not possible, a hashed reference should be used.	*Low*	*Medium*	*Low*
10	Malicious user: invalidated redirects and forwards attack, by making the web application redirect request to a URL contained within an untrusted input (for instance a malicious certificate).	- No untrusted input should be accepted. - Sanitize the inputs of the user.	*Medium*	*Low*	*Low*
11	Malicious user: insufficient transport layer protection attack, consisting of man in the middle between the *Web Server* and the *Core Server*.	- Strong authentication and encryption must be enforced between the two servers.	*Low*	*Medium*	*Low*

2.4.4 *Evaluation Asset Core Server*

No.	Threat	Countermeasure(s)	L	I	Risk
12	Malicious user: spamming the *Core Server* to make it inaccessible for other.	- The traffic is blocked by the *Firewall*.	*Low*	*Medium*	*Low*
13	Malicious user: try to brute force users' passwords to modify/access the data.	- Limit number of tries.	*Low*	*Medium*	*Low*
14	Malicious user: try to bypass the authentication process and create an illegitimate certificate.	- Use HTTPS communication with an authenticated client. - Setup *Firewall* to allow only the *Web Server* to ask a new certificate. - Update frequently the HTTP server.	*Low*	*Medium*	*Low*
15	Misconfiguration vulnerability: leak user's private key during the generation process.	- Store the keys in a temporary directory and delete it when they are backuped. - Only use HTTPS and SFTP to transmit the private keys. - Enforce the Access Control.	*Low*	*High*	*Low*

2.4.5 *Evaluation Asset VPN Server*

No.	Threat	Countermeasure(s)	L	I	Risk
16	Malicious user: connect without authorization and get the IP address of the *VPN Server*.	- Use Wireguard and strong key.	*Low*	*Medium*	*Low*

2.4.6 *Evaluation Asset CA Server*

No.	Threat	Countermeasure(s)	L	I	Risk
17	Malicious user: issue fake certificate.	- Only *Core Server* can issue a certificate, block all the other requests. - Use HTTPS.	*Low*	*High*	*Low*

2.4.7 *Evaluation Asset Firewall*

No.	Threat	Countermeasure(s)	L	I	Risk
18	Malicious user: change the *Firewall* rules in order to expose machines in internal network to the internet the way they were not supposed.	- Only root user can change the rules.	*Low*	*Medium*	*Low*
19	Malicious user: spam the service in order to make it inaccessible for other users (DDoS attack).	- Accept only legitimate requests.	*Low*	*Medium*	*Low*

2.4.8 *Evaluation Asset MySQL Server*

No.	Threat	Countermeasure(s)	L	I	Risk
20	Malicious user: code injection in order to manipulate the data.	- Sanitize inputs.	*Low*	*High*	*Low*
21	Malicious user: try to connect to the server in order to get data.	- Accept connections only from *Core* and *Backup* Servers.	*Low*	*High*	*Low*
22	Malicious user: steal data from server.	- Data is encrypted and only readable by root user, so to have plaintext data the malicious user has to get root privileges.	*Low*	*High*	*Low*

2.4.9 *Evaluation Asset Private Keys of Users*

No.	Threat	Countermeasure(s)	L	I	Risk
23	Malicious user: impersonate a user using the private key.	- Revoke a certificate once a private key gets leaked.	*Medium*	*Low*	*Low*

2.4.10 *Evaluation Asset of Keys and Certificates*

No.	Threat	Countermeasure(s)	L	I	Risk
24	Configuration vulnerability: leaking of certificates private keys, if the user finds the directory where it is stored on *Core/Backup* Servers.	- Transmit the keys via SFTP internally. - Only root user have read and write access control in the *Backup Server*. - Recovery procedure need human action.	*Low*	*High*	*Low*
25	Issue fake certificates for not trusted users.	- Authenticate the user with username/password or with a challenge which implied an old certificate. - Setup an HTTPS connections between the *CA* and the *Core* Servers with a client authentication to avoid that another user on the internal network can ask to issue a certificate. - Access control of the root signing key on the *CA Server* with read access only for root.	*Low*	*Medium*	*Low*

2.4.11 *Evaluation Asset of Backups*

No.	Threat	Countermeasure(s)	L	I	Risk
26	Malicious user: steal the backup of the certificates and keys.	- Transmit the backups over SFTP. - Restrict read access to the admin.	*Low*	*High*	*Low*
27	Malicious user: steal the backup of the logs and the config files.	- Transmit the backups over SFTP. - Restrict read access to the admin.	*Low*	*Medium*	*Low*
28	Malicious user: steal the backup of the *MySQL Server*'s database containing the user's passwords.	- Transmit the backups over SFTP. - Restrict read access to the admin.	*Low*	*High*	*Low*
29	Nature: a fire can start in the *Backup Server* room or in the room with other servers.	- Put more fire extinguisher and smoke detector. - Keep important data on external disk in order to be able to recover it.	*Low*	*High*	*Low*

2.4.12 *Evaluation Asset Connection*

No.	Threat	Countermeasure(s)	L	I	Risk
30	Malicious user: cut the cables inside the building (internal network).	- Put a guard. - Lock the server room and restrain its access to administrator.	*Low*	*High*	*Low*
31	Malicious user: cut the cables outside the building (external network/Internet).		*High*	*High*	*High*

2.4.13 *Evaluation Asset Session Cookies*

No.	Threat	Countermeasure(s)	L	I	Risk
32	Malicious user: forge a cookie.	- Sign it.	*Low*	*High*	*Low*
33	Malicious user: steal a cookie.	- Let it expire after 10min.	*Low*	*High*	*Low*

2.4.14 Accepted High Risk

No.	Threat	Countermeasure(s)	L	I	Risk
34	Malicious user: cut the cables outside the building (external network/Internet).		*High*	*High*	*High*
35	Students: could find the backdoors.		*High*	*High*	*High*

2.4.15 Detailed Description of Selected Countermeasures

To prevent the most common attacks, it is important to use software without known vulnerabilities and to prevent the user from accessing any resources or perform any action we do not want him to. Concerning the software, we use secure protocols and pay particular attention to access control. Regarding the user's possible actions, we sanitize inputs, especially to prevent SQL command injection attacks. Finally, we only use strong passwords for the VMs' users.

2.4.16 Risk Acceptance

There are potential measures that could be taken to reduce accepted risks. We cite them in the following table.

No. of threat	Proposed additional countermeasure including expected impact
4	2FA - Implement a password check with additional information check (like an SMS), which decreases the risk because the malicious user cannot impersonate using only the password.
4	Confirm any data changes by another media (like e-mail, etc), which decreases the risk because the malware will have to bypass the second media's security measures as well.
5	Face ID check - The malicious user will have to bypass this security measure that decreases the risk.
6	Remove the cookie implementation - Replace it with a physical asset and always using a two-side authentication (for example, a USB stick containing private keys).
6	Give a continuous education to iMovie's employees about the most common attacks through Social Engineering and give them tools to avoid falling for such attacks and raise their awareness.
31	Put guards outside.
34	Hide the backdoors better.