Review of the External System

Marc Gähwiler Leonhard Helminger Fabian Zeindler

12.12.13

Contents

1 Background

Developers of the external system: *Thomas Knell, Danny Schweizer, Samuel Hitz*

Date of the review: 01.12.2013

2 Completeness in Terms of Functionality

In this section we review the system according to the requirements given in the assignment. We compare each of the requirements stated with the report of group 3 and their implementation.

2.1 Functional requirements

2.1.1 Certificate issuing process

- Use of legacy database, verifying authorized certificate request on basis of this database ✓
- Login with user ID and password ✓
- Display user info from database ✓
- A logged in user can alter his user information ✓
- Issuing certificate on basis of data in database ✓
- Download cerated certificate in PKCS#12 format ✓

2.1.2 Certificate revocation process

- Authentication via user certificate or user ID/password ✓
- Certificate gets revoked ✓
- CRL published ✓

2.1.3 Administrator interface

- Authentication with certificate ✓
- Displays # of issued certificates, # of revoked certificates and current serial number ✓

2.1.4 Key backup

All certificates and corresponding private keys are stored in an archive ✓

2.1.5 System administration and maintenance

- Secure interfaces for remote access ✓
- Automated backup for logging and configuration information ×
  - Only events of the core server are logged
  - Only logs and configurations of the core server are backed up
  - Logs are rotated but not backed up
  - Backup location is on the same host as the original data

2.1.6 Components to be provided

- Web server: user interfaces, certificate requests, certificate delivery etc. ✓
- Core CA: management of user certificates, CA configuration, CA certificates & keys, functionality to issue new certificates, etc. ✓
- MySQL database: legacy database according to provided schema ✓
- Client: Sample client that allows to test the CA's functionality ✓

2.2 Security requirements

- Access control on data (configs, keys, etc.) ✓
- Secrecy and integrity of keys in the key backup $\times$
  - Keys are not encrypted
  - No integrity checks
  - Backup location is on the same host
- Secrecy and integrity with respect to user data $\times$
  - User certificates and the according certificate signing request are left in the /tmp directory
- Access control on all component IT systems ✓

3 Architecture and Security Concepts

3.1 General system architecture

The system consists of two servers, a public facing firewall and the Core Server that houses the CA, a web server that allows users to access the CA and Admin web interfaces, the legacy MySQL database server and the log, key and configuration file backup.
The firewall uses a security in depth approach by only allowing connections on certain ports and forwarding anther set of ports to the Core Server's IP address. All other connection attempts are dropped.
The Core Server again only allows connections on certain ports (in fact the same ports that are forwarded on the firewall).

3.1.1 Critique

We consider the placement of all of the core functionality on one server as non-optimal. According to the system requirements, the web server has to be externally accessible and should therefore be placed in a DMZ that is separated from the CA, any database or storage that carries any sensitive user data and all backup systems.
Especially all backups should be stored on a different server that the rest of the system. There are multiple reasons for this:

- Secrecy and confidentiality requirements of the key backup
- Compromising a server can possibly render all logs stored on this server useless. Therefore it is a good idea to store a copy of all log and configuration files on a location that can not be tampered with in the event of a compromised server
- As soon as the core server suffers any kind of physical harm which requires a complete restoration from a previous backup, the backup has to be stored on a different server

3.2 Risk analysis and security measures

3.2.1 Definition of impact an likelihood

We cannot find fault with the definition if Impact. It is loosely based on the NIST definition but with an emphasis on the reputation of the company which seems to be particularly important when doing investigative journalism.

The definition of likelihood however in our opinion lacks one important ability of the threat source, the capability of the attacker. In the definition of high likelihood it says the motivation of the threat source to exploit the vulnerability is high and/or the vulnerability is easy to exploit. We think it is important to also emphasize the possibility that the capability of the threat source to exploit the vulnerability is especially high in a particular case. E.g. including the possibility that the vulnerability is hard to exploit but the threat source is not only extremely motivated but also extremely capable to exploit the vulnerability.
The same is true for the definition of medium and low likelihood were the possibility that the threat source is not that capable or completely lacks the knowledge/capability to exploit the vulnerability should be included.

The definition of risk follows the general notion and seems appropriate with regard to the definition of impact and likelihood.

3.2.2 Stakeholders, assets and threat sources

The important stakeholders are listed but we think one could include the investigated party since it is also mentioned as potentially malicious third party in the description of other stakeholders.

All important physical assets were considered. However, the client machines have two states associated, normal operating mode or defective which ignores the state were a machine could be functioning but be compromised. All other physical assets are missing the possible associated states.

Logical assets describe the software running on the physical assets. Again also, no states were associated with the assets and the iptables on the core server were not mentioned. Regarding logical assets we miss the information assets which are not specified. The user database is listed which could be understood as the user data asset but other assets like certificates or private keys are not specified although they are integral parts of the CA system.

The listing of threat sources is complete for the context of the iMovies CA system. However it would have been nice to state why other threat sources were not considered.

3.2.3 Risks and countermeasures

In the listing of risks, only risks that where not threated with countermeasures during the implementation and risks to physical assets are listed which we consider insufficient, as the risk analysis should state all risks and describe the proposed as well as the implemented countermeasures. Risks to information assets like private keys, communication or certificates are missing since they are not listed as assets or because apparently the authors did not include any threats to the assets that they think they dealt with during the implementation.

The evaluation of risks to the physical assets seems to be complete. For the rest, several things came up that we consider problematic. Three times the use of anti virus software is mentioned, but in the implementation we could not find any (document and vms). Also many of the proposed countermeasures are formulated very generic. The use of “good software” was suggested seven times without including a clear definition of the term and what the group considers as “good software”.
Furthermore for example in risk number 17, the proposed countermeasure is having strong protection mechanisms for all critical infrastructure, where the means of strong protection and a more detailed description of critical infrastructure are not outlined in any way. Risk number 21 where the countermeasure is to make sure the database is secured with appropriate access control mechanisms is another example.

The same is true for the proposed countermeasures in the risk acceptance section where state-of-the-art security measures are proposed, again without any kind of explanation what is considered as state-of-the art. Other countermeasures as for example suggesting external security audits or mandatory security seminars seem to be valid countermeasures. Overall, the expected impact is not described for any of the listed threats.

4 Black-box Testing

To perform a back-box analysis of the whole system, we loosely followed the OWASP Testing Guide v3 ($https : //www.owasp.org/index.php/OWASP_Testing_Guide_v3_Table_of_Contents$). In particular we left out all sections that could not be applied to the system or did not result in interesting data and we added a few points that are not mentioned in the testing guide that caters to the testing of web applications.

4.1 Tools Used

- Nmap
- Firefox (Addons: ...)
- sqlmap
- slowhttptest

4.2 Open Ports

4.2.1 Tools Used

To scan all systems for open ports we used nmap (...)

4.2.2 Discoveries

- Firewall: TCP port 22, no open UDP ports
- Core: TCP ports 22, 442 and 4433, no open UDP ports

4.3 General Entry Points

4.3.1 Tools Used

To detect what services are listening on the open ports we used nmap again.

4.3.2 Discoveries

On both servers an SSH daemon (Version string "OpenSSH 5.9p1 Debian 5ubuntu 1.1") listens for SSHv2 connections on the port 22.
Additionally on the core server an Apache HTTPD is listening on the ports 443 and 4433.

4.4 Application Entry Points

4.4.1 Tools Used

Firefox with the Firefox Developer Console to record all requests sent to the server while using the web panels

4.4.2 Discoveries

We did not recognize anything out of the order (requests to get data are performed with an HTTP GET request, request that update data use the POST verb). We used this information in later stages.

4.5 Error Codes

4.5.1 Tools Used

We used Firefox with the tamper data add on.

4.5.2 Discoveries

The standard Apache error pages leak the Apache version the core system is using.

4.6 SSL/TLS

4.6.1 Tools Used

TODO!

4.6.2 Discoveries

4.7 Application Configuration

4.7.1 Tools Used

Firefox with the tamper data add on

4.8 Discoveries

The server leaks the detailed version of the Apache HTTPD ("Apache/2.2.22 (Ubuntu)") and the used version of the PHP interpreter ("PHP 5.5.5") in the response HTTP headers.

The PHP configuration allows errors to be displayed to the user. For example a missing post field results in an error on the top of the page, describing which field was missing, in which file the error occurred and on which line the erroneous statement is.

In addition to the login page at /index.html, there seems to be an old login page at index.php which contains the output of the phpinfo() function that leaks a huge amount of information about the HTTPD and the PHP interpreter.

4.9 File Extensions

4.9.1 Tools Used

Firefox and curl

4.9.2 Discoveries

We did not discover anything out of the ordinary.

4.10 Application Administration Interfaces

4.10.1 Tools Used

Firefox

4.10.2 Discoveries

Even though the admin panel is publicly accessible on port 4433 any connection without a valid SSL client certificate. We consider this not as a vulnerability.
We did not find anything of interest besides this.

4.11 HTTP Methods

4.11.1 Tools Used

curl

4.11.2 Discoveries

The Apache HTTP responds to an OPTION request with the list of the allowed methods. These are HEAD, GET, POST and OPTION. All other request methods result in a “501 method not allowed response”.

4.12 Credential Transport

4.12.1 Tools Used

Firefox with the built-in developer console

4.12.2 Discoveries

Because bot web interfaces are only accessible by SSL, no data is transferred in clear text.

4.13 User Enumeration

4.13.1 Tools Used

Firefox

4.13.2 Discoveries

The application gives no feedback about the correctness of a username, it seems to only check the combination of username/password against the database. There is no built-in rate limiting, so a brute force attack could be viable against known or guessable usernames (especially because they seem to be based on the combination of the first and last name of the employees) and weak passwords.

4.14 Authentication Bypassing

4.14.1 Tools Used

Firefox, curl

4.14.2 Discoveries

We could not bypass neither the user password authentication, nor the SSL client certificate authentication that is also used in the administration panel.

4.15 Logout and Cache Handling

4.15.1 Tools Used

Firefox

4.15.2 Discoveries

On logout the session seems to be destroyed and is not usable anymore.

4.16 Cookies

4.16.1 Tools Used

Firefox with the built-in developer tools

4.16.2 Discoveries

Only the session-id is stored in a cookie. No additional data is saved on the client.

4.17 Session Managment/Fixation

4.17.1 Tools Used

Firefox with the built-in developer tools

4.17.2 Discoveries

The session-id saved in a cookie is not renewed after a successful login. This opens the door for potential session fixation vulnerabilities.

4.18 Cross Site Request Forgery

4.18.1 Tools Used

Firefox with the built-in developer tools

4.18.2 Discoveries

Only the certificate creation page uses a CSRF token, all other forms are vulnerable for CSRF attacks. This is used in one of the backdoors.

4.19 Path Traversal

4.19.1 Tools Used

Firefox

4.19.2 Discoveries

No path traversal attack vectors were found.

4.20 Cross Site Scripting

4.20.1 Tools Used

Firefox with the tamper data add on

4.20.2 Discoveries

The first name of a user is not sanitized correctly. This allows an attacker controlling the first name field of a user to inject arbitrary data into the "user.php" page. All other values seem to be sanitized correctly.

4.21 SQL Injection

4.21.1 Tools Used

Firefox with the built-in developer tools and sqlmap

4.21.2 Discoveries

No SQL injection attack vectors were found.

4.22 Denial Of Service

4.22.1 Tools Used

slowhttptest

4.22.2 Discoveries

We could not find any obvious attack vector for known denial of service vulnerabilities in the Apache HTTPD or PHP. The Apache HTTP has correctly set a keep-alive request timeout, that prevents the usage of known slow http attacks like slowloris.

5 White-box Testing

For the white-box analysis we used the knowledge we gained from the book and the experience we gained while planning, developing and describing our own system.

5.1 Static Analysis

We used a static PHP source code analyzer called RIPS. We did not find any vulnerability this way.

5.2 Configuration Analysis

5.2.1 Firewall

As expected after the black-box analysis, the firewall consisted of a simple iptables configuration that accepts TCP connections on port 22 and forwards TCP connections on the IP of the core server to the core server. Additionaly there exists a rule to forward all traffic comming from the core server to any outgoing server on the port 1234. We could not find any kind of backup solution as it is be required in the project assignment.

5.2.2 Core

Like the firewall, also the core server uses a simple iptables based solution to only accept TCP connections on the ports 22, 443 and 4433.

The Apache HTTPD serves the html and PHP pages from /var/www/admin_interface for the admin web panel and /var/www/user_interface for the user panel. As we already suspected in the black-box analysis, the keep-alive value was set to five seconds, which prevents the usage of slow HTTP attacks. Besides this we did not find any interesting changes to the configuration files.

The PHP interpreter's configuration file that is used by the Apache module mod_php enables user visible error logging with the display_errors directive. This is generally considered a bad idea for production systems. We did not find any other points of interest in this file.

We could not find a solution for the backup of log files, they are just rotated by the default ubuntu logrotate script. Some configuration files (namely all files and directories in /etc/apache2/, /etc/ssl and /etc/ca-setup) are copied daily via a cronjob into /home/sysadmin/backup. Old backups are therefore overwritten daily, which renders the whole backup nearly useless.

5.3 General System Analysis

5.3.1 Firewall

We did not find anything of interest here.

5.3.2 Core

There is a very suspicous script in /usr/bin/rshell and /usr/bin/X11/rshell that seems to be some kind of reverse shell that tries to connect to a user defined ip and port that are given by the arguments of the call.

Generally the system uses the standard Ubuntu package management. The only exception for this is the PHP interpreter, which was compiled from the source code. This made us suspicious and together with the source code analysis lead us to a backdoor.

As seen in the Apache configuration, all html and PHP files are in the previously mentioned directories. Additionally we found a strange file called batman.jpg which seems to be there for no purpose. The image has been modified with GD, which is a graphics manipulation library. This could be a sign for some automated processing to steganographically hide data in an image. We tried to find the source image and compare the two versions, but were not able to get any kind of information because of the lossyness of the JPEG compression format.

5.4 Code Analysis

5.4.1 Core

The only files of interest here are the PHP scripts that make up the user and admin panels. Here are the points of interest that we found:

- Even though all queries are written by hand and no ORM is used, all queries use the mysqli PHP module to access the legacy MySQL database via prepared statements. In conjunction with correctly written SQL statements, the mysqli offers a secure way to perform database queries without the risk of SQL injections.
- Behind the scene, each request that requires interaction with the CA itself, executes the openssl binary via the exec() and shell_exec() PHP functions. This made us suspicious, because there is no reason to switch between those two functions, as they nearly do the same and only differ slightly in the way they return the output of the executed command.
- All parameters passed to the exec() and shell_exec() functions are sanitized with escapeshellarg() function. This prevents a potential adversary to execute own code in the shell.
- As suspected, the first name of a user is not sanitized in user.php which results in a XSS vulnerability.
- Additionally the user's password is not checked when a user updates his data with the form in the user interface.
- To prevent any kind of race condition while creating or revoking certificates, the system uses file based locks to prevent simultaneous access to all files that the openssl binary uses.
- Again as suspected, the system includes a index.php file, that executes the phpinfo() command.
- The basic PHP session system is used to keep track of a users login state and his csrf token.
- Surprisin

6 Conclusion

In this section we will list the countermeasures where the implementation differs from the description in the documentation. Additionally we cater to implementations we encountered that we consider security risks or bugs. At the end we summarize some of our testing of the system that proved to be not successful in terms of breaking the system or finding vulnerabilities.

- The system offers every puser the possibility to change his information exactly according to the very vague (and in our eyes insufficient regarding security and confidentiality requirements) description in the project assigment. We consider the fact, that a user can change all its information without an audit by an administrator or similar, a security risk. The user could change his information completely and thus impersonate every other user of the system. In our opinion, it is the responsibility of a CA system to ensure the validity of the user information. In addition to this, old certificates are not revoked once a user changes his user information, which also poses a security risk, as an attacker could change the user information of a compromised user to something arbitrary, create a new certificate that carries the new user data and change the data back again so the user has a harder time to recognize any changes.
- In the documentation it is stated, that for a change of the user name the user password is needed. There is a password field, but the input to this field is not verified.
- When describing system administration and maintenance the documentation states that Apache access and error logs are daily backed up using logrotate. Logrotate does not back up data, it just rotates the logs (but does not mover or copy them to another location). According to the configuration logs are rotated every week and stored for a year. The same is true for MySQL logs which are rotated daily and stored for a week.
- Backups in general are done with a daily cron job. Configurations are taken from /etc/ and backed up to /home/sysadmin/backup on the same server. Logs and data are not backed up. It could be pure coincidence, that the ca system stores all created keys in a subdirectory of /etc/.
- As specified in the documentation under system administration the backup is done with a daily cron job. The cron job invokes rsync which copies configurations (but no logging like stated in the documentation). This copy procedure overwrites old backup files which is usually not desired.
- In addition to the above, no configuration or logging data from the firewall is backed up. This is a violation of the requirements in the project assignment.
- The documentation also lists public key authorization for ssh as a security measurement against unauthorized access, but there are no known keys or public key functionality in place. However sshd is configured to accept public key authentication, but it also allows remote access for root which we consider unnecessary.
- Key backup as described in the documentation is not a real backup as described earlier in. In addition the keys are protected solely via linux access control but not encrypted or protected by other means.
- In addition to that the certificates and corresponding private keys (.crt, .crs, .p12) are stored in /tmp/ during creation and not deleted afterwards.
- When trying to get information about the system, we could obtain the version of SSH, the operating system, via index.php detailed information about the used PHP, the version of the Apache webserver, etc.

7 Backdoors

7.1 CSRF combined with XSS

As previously mentioned, the form that updates a user's information does not use a CSRF token, even though the general functionality is provided by the system. Combined with the fact, that the database value of the first name is not sanitized properly and that the user's password is not checked on updates of his information, this gives an attacker an easy possibility to gain access to a user's account.

A possible way to compromise a user is to get a user to submit a form that the attacker controls that posts it's data to the CA user interface. Because the form does not require the knowledge of the user's password, any data can be submitted and therefore written to the database. Afterwards the user is redirected to the user page, where the previously submitted form data is displayed without sanitizing it. This enables the attacker to inject his own code into the user's main page.

7.2 Batman.jpg

As mentioned above in the white-box analysis, this publicly available image could hide any kind of information. We were not able to get any kind of prove that this is the case. Still, we would strongly suggest to remove this file from the production servers.

7.3 PHP/exec() backdoor

After analyzing the self compiled binaries and especially having a look at the shell_exec() and exec() functions, we came to the conclusion, that there is a hidden backdoor in the exec() function. We think that exec() function splits the string it gets on each semicolon character and then only executes the second slice. This provides an attacker with a way to execute any code even if all input arguments are sanitized with escapeshellarg(). Additionally the firewall forwards all connections from the core server to any external ip on port 1234.

This can be used by changing the first name of a user to ;rshell 10.1.0.2 1234; and creating a new certificate. This results in the server opening a new shell and binding it to a socket connection to the provided ip and port. Combined with the CSRF and XSS vulnerability above, this enables an attacker to compromise the whole system.

8 Comparison

Comparing the both systems, the first thing that we noticed was the lack of separation in the reviewed system. They also have a designated firewall which filters traffic but they have combined all functionality (webserver, CA core and even backup) on one single server where our system has four different zones, the internet, the network where employees hate their workstations, the DMZ which hosts the webserver and is the only machine accessible from outside by non administrators, as well as the intranet which hosts the CA functionality on one server an the backup and key archive on another server. The access control to all those networks is handled by the firewall.

For remote access our system provides we use VPN where the reviewed systems just forwards the SSH ports which both seems to work.

Another major difference is the handling of user data change. Where in the reviewed system users can change data as they like and thus potentially impersonate other people, our system implements a review of user data changes other than the password to ensure the authenticity of a certificate.

WRITE SOMETHING ABOUT THAT: Are there any remarkable highlights in your system or the external system?