CREAM Vulnerability Assessment*

Manuel Brugnoli, Maxime Frydman, Joe Carrión, and Elisa Heymann
Universitat Autònoma de Barcelona
September, 2013

Introduction

The CREAM (Computing Resource Execution And Management) Service is a simple, lightweight service for job management operation at the Computing Element (CE) level [1].

This report presents the results of the vulnerability assessment of CREAM, version 1.14.0, as part of a Quality Control process within EGI InSPIRE. For this task we have used the First Principles Vulnerability Assessment (FPVA) methodology [2] proposed by the University of Wisconsin and Universitat Autònoma de Barcelona Middleware Security and Testing Group.

This report is structured as follows. The first section presents the architectural, resource and privilege analysis of CREAM. The component evaluation of CREAM is discussed in the second section, and the final section provides the results of the vulnerability assessment performed.

Architectural, Resource and Privilege Analysis

In this section, we describe the steps of the FPVA methodology as applied to CREAM. We also show the resulting diagrams for each step.

Architectural Analysis

CREAM is a system designed to manage a CE in a Grid environment. CREAM is mainly composed by an interface based on Web Services, which enables a high degree of interoperability with clients, currently Java and C++ clients are provided, but it is possible to use any language with a Web Service framework.

CREAM is written in Java, and runs as an extension of a Java-Axis servlet inside the Apache Tomcat application server. Requests from users and components come solely through SOAP web services. This offers a single unified interface offering all the features supported by CREAM.

All requests sent to CREAM are first authorized before being processed, when a specific user sends a request the authorization framework will contact VOMS to verify

* This research was funded in part by NATO grant ICS.MD.CLG 984138 and the European Commission (contract number: RI-261323), through the EGI-InSPIRE (Integrated Sustainable Pan-European Infrastructure for Researchers in Europe) project.

whether the user has sufficient privileges to carry out that request. This mechanism serves both for regular users and administrators of the system.

Figure 1 shows the generic architecture of CREAM, this is a high level view showing just the processes and their communication protocols. The architectural diagrams provide the analyst with a great deal of information on the attack surface, attack vector and the general structure of the software.

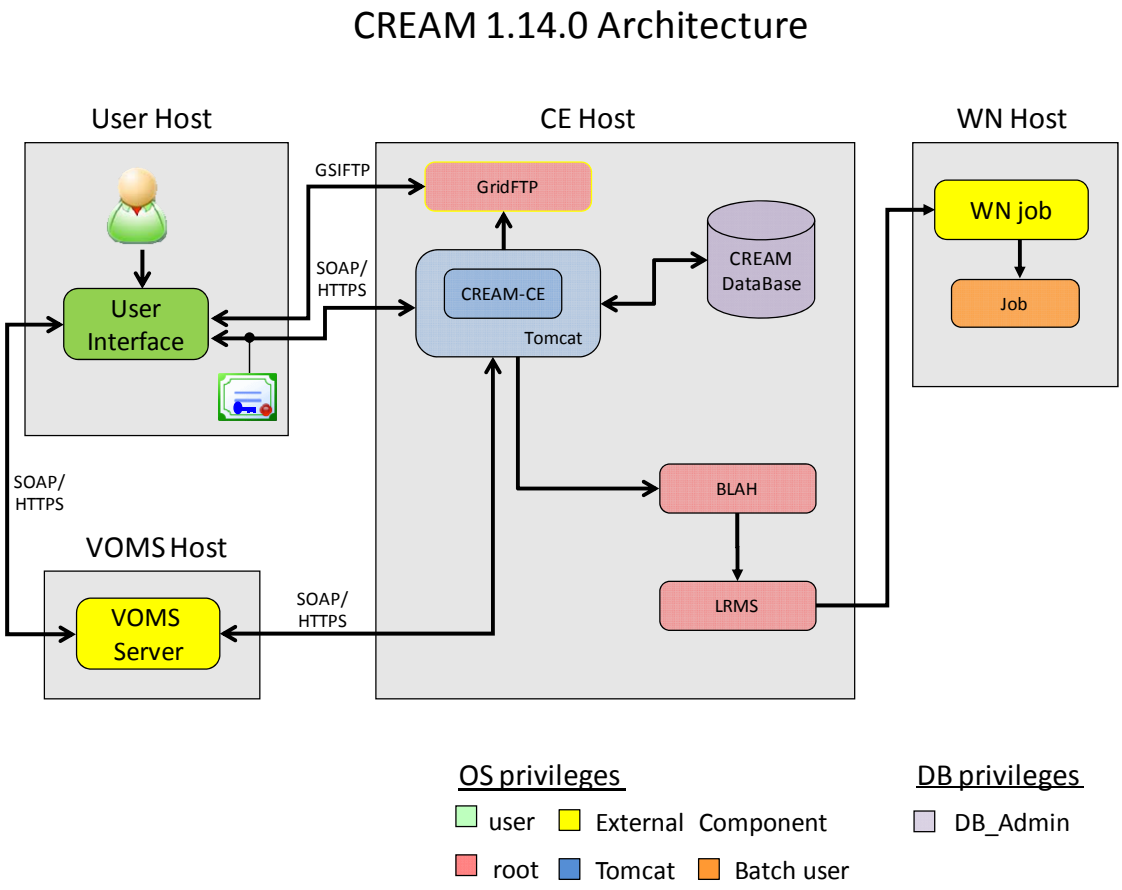

Figure 1. CREAM Architecture Diagram.

The hosts shown are the User, CE, VOMS and the Worker Nodes (WN). The Host details the processes running (these are the components of the middleware), whether a component belongs to CREAM and its privileges. Components that are external to CREAM are not part of the assessment and are not reviewed for vulnerabilities, they can however be used to perform attacks if they can be manipulated to perform malicious actions.

Each communication between hosts is labeled with its protocol. This offers a first insight into attack surfaces, for example the CE communicates with the client using HTTPS encrypted SOAP web services and the GLOBUS FTP protocol. The first is used to offer the functionalities of the CE to the client, while the second is used for the transfer of job files. This gives an initial feeling for possible attacks, particularly communications and API attacks.

Job submission is one of the core services offered by CREAM and involves the interaction of all of its components. This is why the job submission was used to present the CE's interactions as it gives a general idea of how data flows within CREAM. Other services offered by the CE follow the same general pattern of interactions.

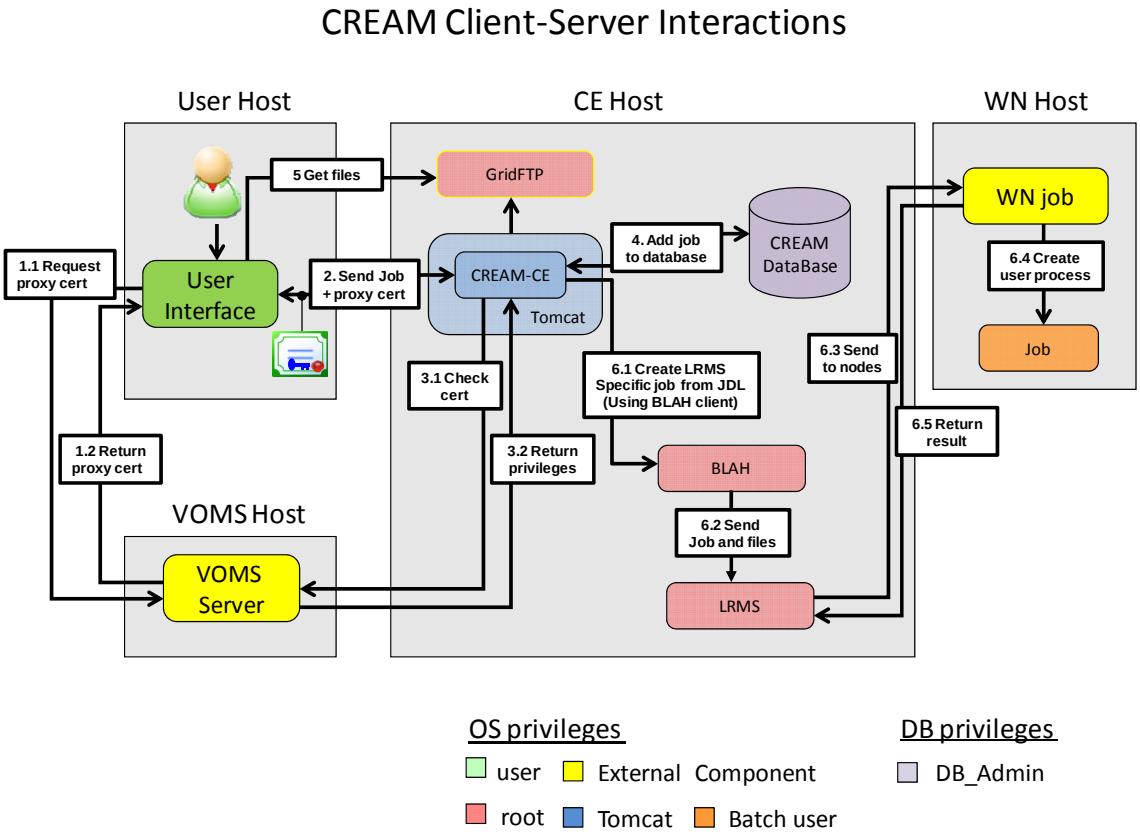

Figure 2. CREAM Client-Server Interactions.

Figure 2 presents all the steps involved in the submission of a job on the CE. This diagram provides the analyst with information over which component accesses the data sent to the CE and how that data is used. Where the overview diagram gave information about possible attack surfaces the interaction diagram gives information on attack vectors. Knowing where specific input sent to the CE will be used and what actions are triggered guides the analyst in the assessment of a component to evaluate if it is vulnerable to an attack.

The steps involved in the submission of a job are the following:

- **Certificate Acquisition:** To submit jobs on the CE a proxy certificate is required. This certificate is generated by the VOMS component and serves to authorize the user on the CE. Proxy certificates are a temporary form of authentication, by default they last 24 hours, and are used to mitigate the security risk of sending the real user certificate to the CE. If the CE is compromised or the proxy certificate is intercepted it is only of very limited value due to its short-lived nature.

- **Certificate Delegation and Job submission from the client to the CE:** The certificate and the job description are sent to the CE. The CE will store the certificate associated to a delegation identifier, this identifier can then be used to send multiple jobs using the same certificate. The information over the job submitted to the CE is only the JDL file that describes the various parameters of the job like the location of the executable, the location of the data and the files that have to be retrieved as output.

- **User Authentication and Authorization:** The CE will immediately contact VOMS before storing any of the information over the jobs to confirm the user identity and establish whether the user has the privileges required to create jobs. The authentication is done using the user real certificate signing the data with his private key on the client side. The CE will then verify that this data is correct by comparing it to the information sent by VOMS. If everything checks out the job creation will proceed.

- **Database Storage:** When the user has been authenticated the CE will then store the information over the job and the proxy certificate inside its database and call GLOBUS to launch file transfers.

- **File Transfer:** GLOBUS is required to transfer the executable and input files from the client to the CE. The authentication of GLOBUS uses the certificate previously delegated and then transfers the requested les for storage on the CE. The files will reside on the CE until the job is sent to the worker nodes.

- **Job Submission from the CE to the Resources**: When the CE is notified that there are free resources on the worker nodes the JDL is sent to BLAH. BLAH will translate the generic commands described in the JDL file into commands specific to the LRMS. BLAH will also generate a wrapper executable that will be executed on the worker nodes. The purpose of the wrapper is to launch the executable specified in the JDL with all the parameters requested by the user. This deals with site specific configurations like the location of the input data for an executable and authentication on storage resources, it can be modified if needed by the administrator to adapt to the specifics architectural details of a CE.

  The submission of the files from the CE to the worker nodes is left to the LRMS. When all the files have been transferred to the resources the LRMS will launch the wrapper execution. When the execution is finished the LRMS will transfer the results back to the CE so that the user can retrieve the results.

Also, the interactions diagram shows the order in which tasks are handled and the responsibilities of each component. It also details the general steps of the job submission process. The whole process is not a single user action, for example step 1 which acquires the proxy certificate is handled by the VOMS client while step 2 is carried out by the CREAM client and are separate actions by the user.

For example, in the second step of the submission the user sends the certificate and the JDL to the CE, some of this information is then stored inside the database. This indicates the possibility of SQL injection attacks from these inputs.

The last diagram that is part of the architectural analysis is the component diagram as shown in Figure 3. This diagram is not always included in the analysis but is useful when analyzing software which runs within a virtual environment like java's Tomcat used by CREAM. The purpose of the component diagram is to show the various packages and their responsibilities in treating requests.

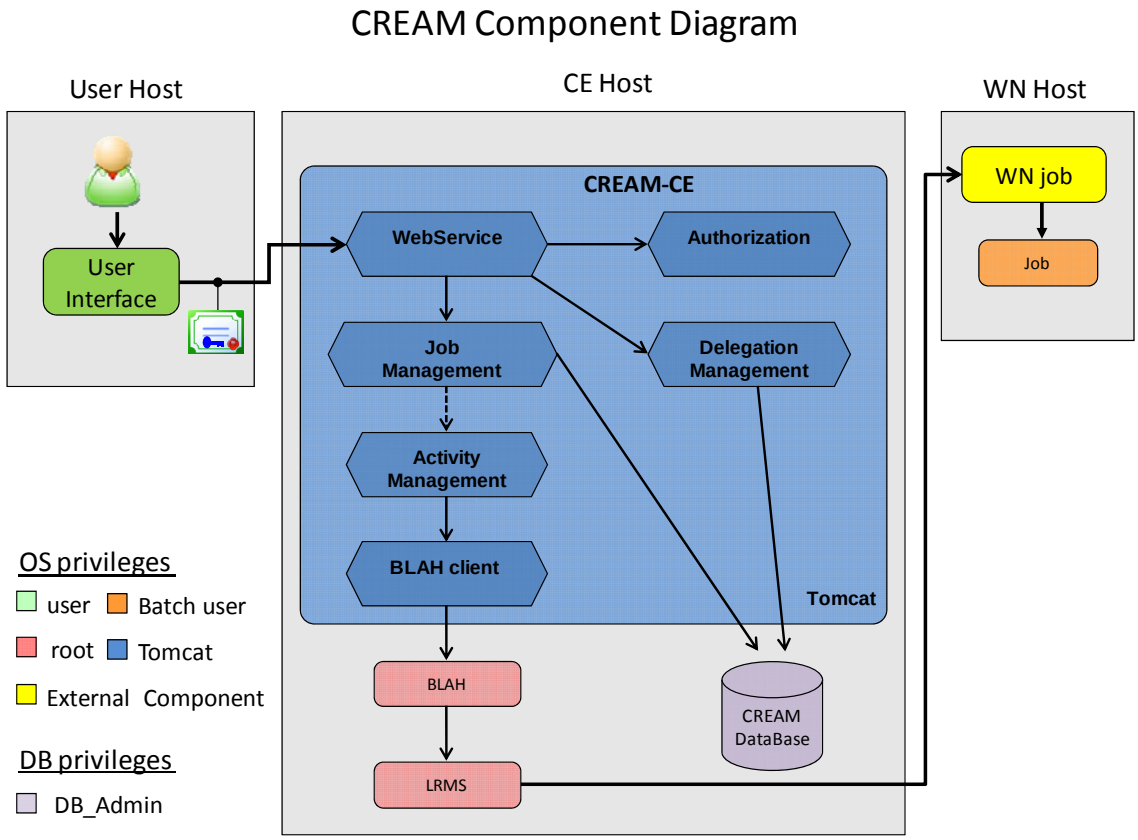

Figure 3. CREAM Component Diagram.

The component diagram provides a great deal of information over the structure of the code, this gives the analyst a precise idea of where each section of code is located and what type of information is passed on from one package to the other. This information is useful when the code is reviewed and serves as a road map when specific functionalities are assessed to locate the relevant sections of code.

The described components are the following:

- **WebService:** The WebService package is the front end of the CREAM application, it offers the web services of the two underlying packages that represent the core functionalities of the CE. It is also the only direct means of communication with the CE for users, administrators and schedulers alike.

  It offers SOAP web services over the encrypted HTTPS protocol, each web service takes a number of arguments depending on the precise service and additionally a proxy certificate for authentication.

- **Authorization:** The authorization package is the first one called after a request is sent to the CE. It serves to authentify the request by communicating with VOMS to ensure that the identity submitted is legitimate and that the right privileges are met to execute a specific command. If it is successful then the CE will transmit the request further to the appropriate package while if it fails the operation is terminated and the user notified.

  This package also facilitates further operations for the CE, since all request have obligatorily passed through authentication and were carried on only if the requirements were met no further verification is required in other packages.

- **Job Management:** Job management is the first package containing the core functionalities of the CE, it deals with all the commands that interact with jobs and includes job creation, job cancellation and job output retrieval. The package is called by the web service package when job related commands are called after the authorization process. The package includes all the logic for in memory and database storage for the tables that are related to jobs. Instructions that will be transmitted to the LRMS are stored inside the database before creating a new activity which will invoke the BLAH client. Also included in this package are the operations for automated job cancellation due to certificate expiration.

- **Delegation Management:** Delegation Management is the second package containing the core functionalities of the CE, it handles all the operations involving proxy certificates like proxy delegation and renewal. The package is called by the web service package when certificate operations are used.

  The package includes database operations to store and retrieve certificates, when a certificate is delegated it is stored inside the database to be used when new jobs are created by associating the job with its corresponding proxy certificate.

- **Activity Management:** The activity management package handles a pool of Java threads used to communicate with processes that are external to CREAM like BLAH. When a job management command requires interaction with BLAH a new activity is created, this activity runs in its own thread and will create a new BLAH client.

  This mechanism is used so that the request of a client can be directly answered while operations that can incur delays are processed independently. Since the BLAH component is external to CREAM and therefore information over its state is not always accurate the package also handles multiple retries of a command until it is successfully processed.

- **BLAH Client:** The BLAH client is a package used by the activity management package to communicate with BLAH. BLAH is an external process independent of CREAM and all the logic to interface with it is held within this package. Commands to be transmitted to BLAH are stored within the database and then passed on to the BLAH client which passes this information on to the actual BLAH process that will handle the conversion of the generic command to one that is specific to the LRMS.

Resource Analysis

Resource diagrams focus on identifying all the resources used by a specific component and how they are accessed. This includes configuration files, logs and databases. Resources are an important part of vulnerability assessment and gaining access to them is often what leads to successful attacks.

The Figure 4 focuses on the resources of the CE. As seen previously the CE runs within the Tomcat environment, the resource shown are all those accessed by the Tomcat process relating to CREAM during the operations of the CE.

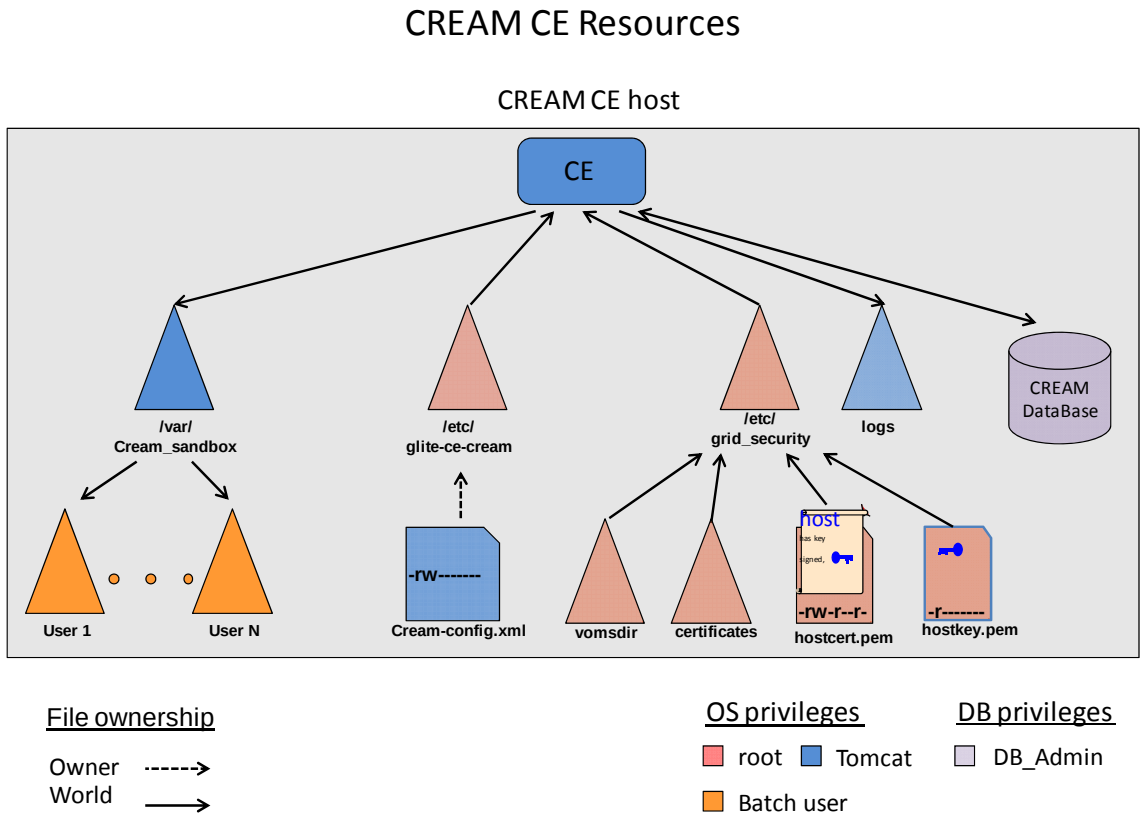

Figure 4. CREAM CE Resources Diagram.

The CE uses relatively few resources and relies heavily on the database to store content with the exception of the CREAM configuration, logs, certificates and job files.

- **CE Configuration:** The CE configuration file holds all the information necessary to launch the CE, it holds information on the location and credentials of the database, information over the VOMS server and other configuration settings required to launch the CE. Only the CE process reads this file, this is reflected in its permissions by being owned and only accessible by the TOMCAT user.

- **Job Sandbox:** The job sandbox contains all the files submitted by the user for a job and after execution the outputs of those jobs. Each user of the VO has his own folder containing the data from all the jobs he has submitted. On the CE each user is

assigned a CE batch user from a pool of users created for that purpose, these users also exist on all the worker nodes and serve for execution.

Each sandbox is owned by its corresponding batch user, which was assigned to a VO user.

- **Logs:** There are a number of log files on the CE that describe the operations that were executed. These files contain some information about the jobs submitted but do not hold sensitive information. Gaining access to them would be of little value to an attacker. These are low-value assets.

- **Host Certificates:** The CE holds a number of public certificates to authenticate the hosts it communicates with and its own public and private certificate. These certificates are used to guarantee the identity of hosts, when a client connects to the CE it will use the public certificate of the CE (which is publicly available) to ensure the data was signed using the private certificate (which is only available to the CE).

  Gaining read access to the public certificates holds no value to an attacker as these are meant to be shared. However being able to modify the public keys could allow certain attacks like host replacement. The private key of the CE however is an extremely valuable asset, if this key is compromised then attackers could replace the CE with a custom host without alerting the clients. This makes the public certificates low-value assets while the private certificate of the CE is a high-value asset.

- **The Database:** The CREAM-CE uses the database for most of its operations. The database is used to hold the state of the CE holding information like job information and proxy certificates but also for communication between the components of the CE like commands that will be sent to BLAH. The database has a single user for both reading and writing, if an attacker was to gain access to the database he would gain control over most aspects of the CE. This makes the database a high value-asset.

The second resource diagram produced, Figure 5, focuses on the resources of the client.

When analyzing the client resources the objective is twofold, the client can both be the target and the source of an attack. First, similarly to the CE, understanding what resources on a client can be compromised and the type of damage that would result highlights the most valuable resource to target when attacking the client. Second a number of the resources located on the client will be transmitted to the CE and are potential vectors of attack.

When assessing the client's security there are some specific use cases taken into account. If the client is totally isolated, having only one user that has total control over his machine and there no specific services running on the client host that would allow access to his resources then there isn't much that can be done with this information. However if there are situations where clients coexist on a single host or where access to the resources is possible then this information becomes relevant.

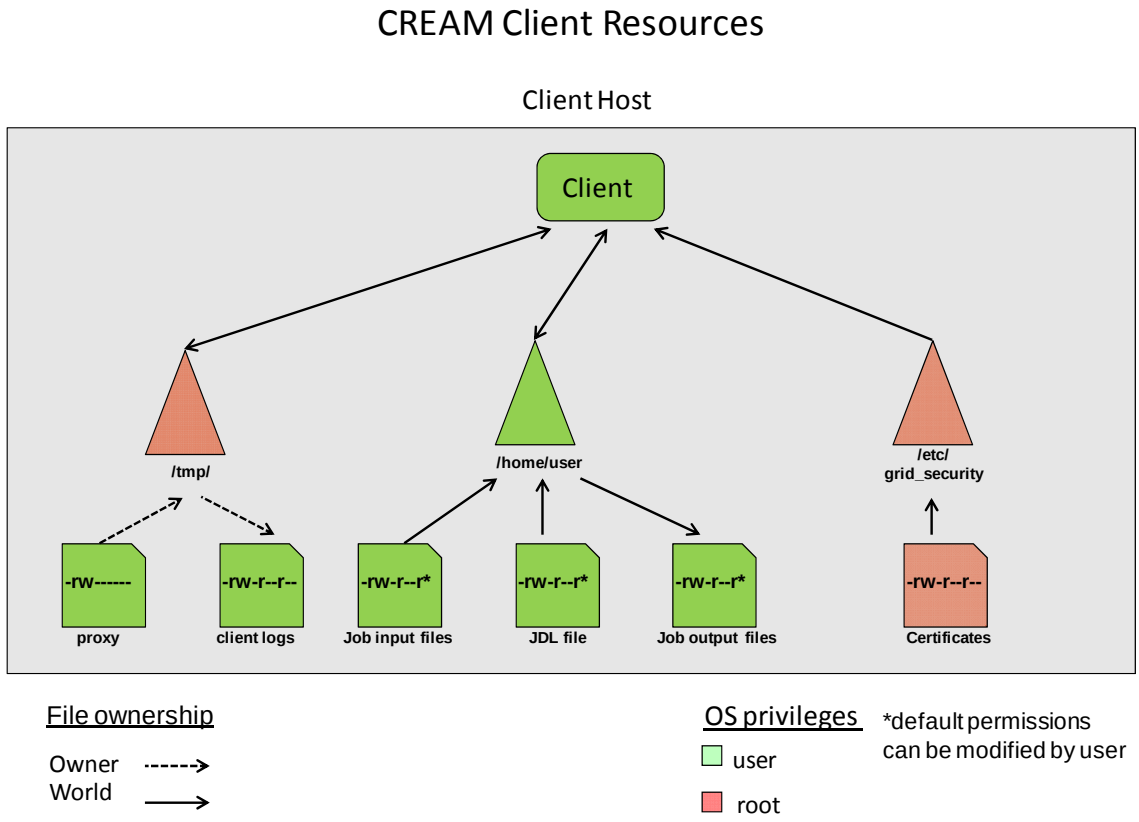

Figure 5. CREAM CE Resources Diagram.

The CREAM client allows interactions with the CE, it does so by implementing simple commands that call one or multiple of the web services offered by the CE. The client is less complex than the CE but still uses a number of files that are required for its operations.

The client has resources specific to the user like his certificate, job files and logs while certificates that identify the various hosts of the Grid are shared across a client host.

- **Proxy Certificate:** The proxy certificate is required to authenticate with the CE, without it no commands can be executed. This certificate produced by the VOMS client is a requirement on the CREAM client host. The proxy is stored on the client host in the temporary folder (/tmp), it is owned and can only be read by the user who has created it.

- **Logs:** Similarly to the logs contained on the CE, the client logs contain information over the transactions between the client and the CE. They hold no data of particular value to an attacker and make them low-value assets.

- **User Files:** To submit jobs on the CE the user will have to place all the files and executables related to a job on the client and the output of jobs will be stored on the client when the user retrieves them. All these files are by default stored in the user home directory with the permissions of that user. These files are outside of the control of CREAM; they fall under the responsibility of the user and are not considered during the assessment. They are however relevant to the attacker as a vector of attack as these files will be submitted to the CE and could potentially be used to exploit the CE.

- **Host Certificates:** Similarly to the CE the client holds a number of certificates to authenticate the hosts that he will communicate with. These include the public keys for VOMS, the CE and any other hosts configured in the client. They are stored in a special folder on the client owned by root but readable by all users. The client hosts does not require its own certificate as authentication is based on the certificates of the users.

Privilege Analysis

The privilege analysis focuses on the user permissions that execute running processes and resource permissions. This information is gathered last and used to enrich the architectural and resource diagrams. The diagrams presented previously are the final versions and already include this information.

First the privilege analysis adds a very important layer of information to the analysis, when elaborating attacks on a particular component knowing which assets can be accessed and in what way is critical. Where the architecture and resource diagrams highlights from which component and to what end an attack can be elaborated the privileges limit the possibilities by clearly defining whether it would be possible to interact with the resource.

Second it also serves to spot erroneous permissions which are a common type of vulnerability. The latest statistics on vulnerabilities by OWASP for 2013 [3] list security configuration as the fifth most common type of vulnerability, these are simple errors that can lead to severe data exposure but are easily rooted out through the privilege analysis.

Component Analysis

The vulnerabilities found were presented in the form of reports sent to the developers, these reports are presented here in a redacted format meant to respect the confidentiality that binds the assessors and the software producer.

Following are five vulnerability reports describing the findings produced during the assessment, each of these vulnerabilities are currently undergoing fixing.

- **Vulnerability Report 1: The Client Proxy Certificates Attack**

The first vulnerability report sent to the developers concerns the CREAM client. When a client host is shared amongst multiple users, certain condition allows a malicious user to get complete control over jobs submitted by other user by tampering with certificate files. When successful this allows a malicious user to force the execution of other user's jobs under the malicious user identity.

This vulnerability is based on the assumption by the developers that attackers would attempt to steal certificate files but in this case an attacker gains unauthorized access by sharing his certificate with users.

Full details about this were reported in the vulnerability report CREAM-2013-0001.

- **Vulnerability Report 2: SQL Injections and Denial of Service Attacks**

The second vulnerability report concerns direct SQL injections from the client on the CE. A total of 17 vulnerable SQL queries were found on the CE, these findings were grouped in one report that provides examples for a selected few. These examples included listing the jobs of all users on the CE, DoS attack on the CE that requires a reboot to resume operations and a DoS that prevented any operations and persisted through reboots.

Full details about this were reported in the vulnerability report CREAM-2013-0002.

- **Vulnerability Report 3: Arbitrary Job Cancellation Through Indirect Injection**

The third vulnerability report submitted showed how a user could cancel any and all jobs on the CE regardless of his privileges. This attack allowed all jobs to be canceled at once or to target specific jobs and users. The report included a complete program that would automate this attack and use information obtained in the vulnerabilities shown in report 2 to get full control over which jobs to be cancel.

Full details about this were reported in the vulnerability report CREAM-2013-0003.

- **Vulnerability Report 4: Access to the CREAM databases and throughout the information stored there access to: output and input files of other users, complete list of Job ID, delegations, and JDL files**

The fourth vulnerability report sent to the developers concerns a malicious user can gain access to all the information contained inside the CREAM database for which they do not have permission, including certificates created for delegations, job ID, JDL files and job files of all users. This is performed by submitting a malicious job register command to the CREAM CE through a custom CREAM Client created using the CREAM API.

Full details about this were reported in the vulnerability report CREAM-2013-0004.

- **Vulnerability Report 5: Reducing Availability of the CREAM CE Through Denial of Service Attacks**

The fifth vulnerability report submitted showed how a malicious user can submit a job capable of submitting false status notifications directly to CREAM CE. These false notifications consume operating system resources, such as memory, and other limited resources in the operating system causing the CREAM CE to deny client requests.

This vulnerability is possible because the CREAM CE receives job status notification by the LRMS daemon. However, CREAM CE has no established limit on the maximum number of simultaneous connections from the LRMS daemon. This lack of control can result in the reduced availability of the CREAM CE.

Full details about this were reported in the vulnerability report CREAM-2013-0005.

Results and Recommendations

After completing the analysis described in this document we have discovered several security problems in CREAM 1.14.0, these security problems were reported to the developers on five reports.

In general the code was of good quality and numerous security measures are in place to prevent attacks. The consistent use of prepared statements and authorization, the use of standard Grid toolkits like GLOBUS and proper use of users and privileges on the Linux operating system made the system robust and more difficult to attack.

Some of the features that made CREAM a robust system are the following:

- The attack surface in CREAM is very small. It is limited to the component that receives requests from the clients.
- Local administration permissions were secure and no way was found to escalate a user to an administrator on the system.
- Each web service request passes through the authentication framework before being handled, the authentication mechanism that works in combination with the VOMS server is well implemented and prevented any unauthorized access to services and could not be bypassed.
- Encryption between the client and the CE is well implemented and cannot easily be broken, this prevents eavesdropping on the information sent. Host spoofing techniques are thwarted by the authentication mechanisms using host certificates and without access to these certificates are not possible.
- Certificate in the client hosts are well isolated between users and host certificates cannot be modified.
- CREAM is written in Java. This prevents possible vulnerabilities that we can find in other languages (e.g. C and C++), such as those related to the memory allocation and management.

References

[1] Cristina Aiftimiei, Paolo Andreetto, Sara Bertocco, Simone Dalla Fina, Alvise Dorigo, Eric Frizziero, Alessio Gianelle, Moreno Marzolla, Mirco Mazzucato, Massimo Sgaravatto, Sergio Traldi, and Luigi Zangrando. *Design and implementation of the gLite CREAM job management service*, Future Generation Computer Systems, Vol. 26, Issue 4, April 2010, Pages 654-667.

[2] James A. Kupsch, Barton P. Miller, Eduardo César, and Elisa Heymann. *First Principles Vulnerability Assessment*, 2010 ACM Cloud Computing Security Workshop (CCSW), Chicago, IL, October 2010. URL http://www.cs.wisc.edu/mist/papers/ccsw12sp-kupsch.pdf

[3] OWASP. Owasp list of top ten most common vulnerabilties in 2013. last checked july 2013. URL https://www.owasp.org/index.php/Top_10_2013-Top_10.