European Middleware Initiative

VOMS Core and WMS Security Assessment

EMI DOCUMENT

Document identifier:	**EMI-DOC-SA2-VOMS_WMS_Security_Assessment_v1.0.doc**
Date:	**30/04/2012**
Activity:	**SA2**
Lead Partner:	**CSIC**
Document status:	**Final**
Document link:	**https://cdsweb.cern.ch/record/1277602**

Abstract:
VOMS Core and WMS security vulnerability assessment

1. SECURITY ASSESSMENT OF SOFTWARE PRODUCTS

Products	Release	Discovered Vulnerabilities	Fixed
gLExec 0.8	EMI 1	2	-
Argus 1.2	EMI 1	0	0
VOMS Core 2.0.2	EMI 1	1	0
WMS 3.3.4	EMI 1 Update 11	In progress	-

Currently, the progress status for the components under evaluation is as follows:

- **VOMS Core 2.0.2**: the assessment is completed and one vulnerability was found. We worked on VOMS Core from July, 2011 to January, 2012 (seven months).
- **WMS 3.3.4**: The assessment is currently at the Component Evaluation Step. We started to work on WMS in February, 2012.

Now we show the intermediate artefacts as a result of the three first steps of FPVA, namely the Architectural[1] and Resource[2] diagrams for both VOMS Core and WMS.

[1]**Architectural Analysis**: identify the major structural components of the system, including modules, threads, processes, and hosts. For each of these components, identify the way in which they interact, both with each other and with users. The artefact produced at this stage is a document that diagrams the structure of the system and the interactions amongst the different components and with the end users.

[2]**Resource Identification:** identify the key resources accessed by each component and the operations supported on those resources. Resources include elements such as hosts, files, databases, logs, and devices. For each resource, describe its value as an end target or as an intermediate target. The artefact produced at this stage is an annotation of the architectural diagrams with resource descriptions.

2. VOMS CORE 2.0.2

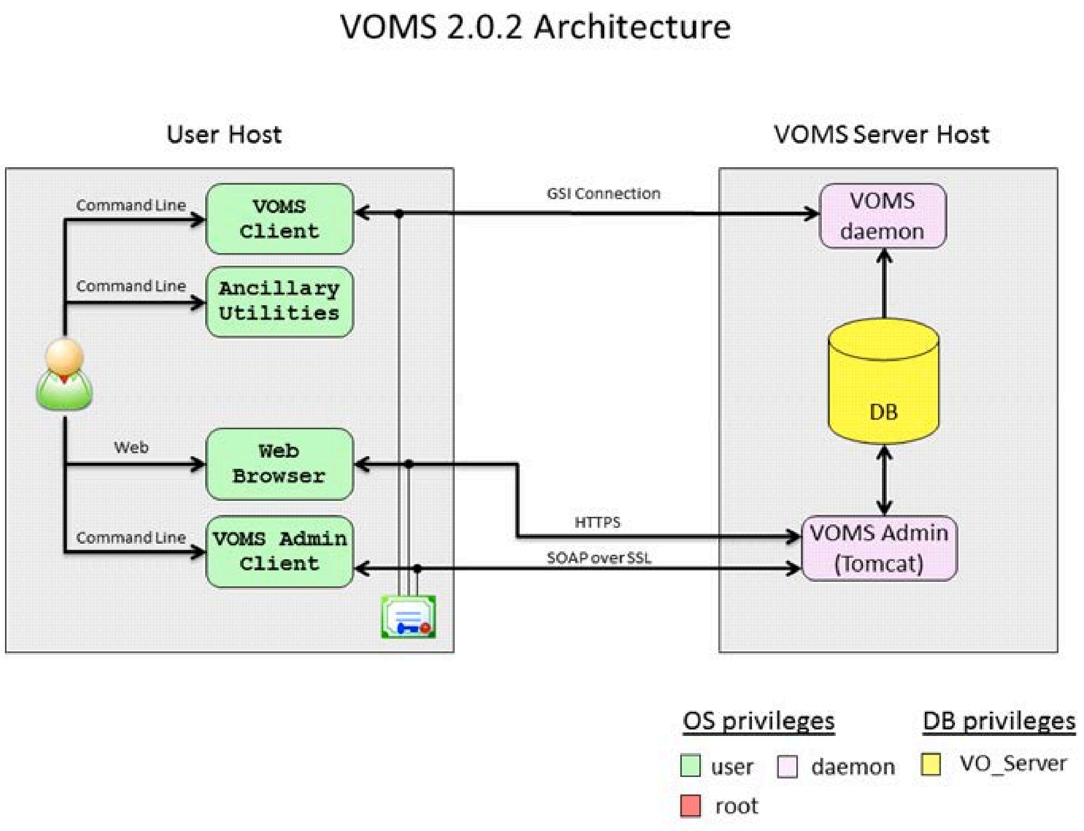

Figure 1: VOMS Architecture Diagram

Figure 1shows the generic architecture of VOMS. VOMS is divided in two main components, VOMS Core and VOMS Admin.

To connect and authenticate to the VOMS daemon, users must have a valid end entity certificate that is signed by a trusted CA, and has not been revoked. This certificate is used to sign a time limited proxy certificate that is used for authentication to the VOMS daemon. To gain access to the VOMS daemon services the client software uses the authentication proxy certificate to negotiate a SSL connection using mutual authentication between the client and the VOMS daemon.

On the other hand, the VOMS Admin component consists of a VOMS Admin server, which is a Java application that runs under Tomcat; a web user interface, which is used by users for daily administration tasks; and a SOAP interface for remote clients, called *voms-admin* that can perform most of the main actions using a SOAP interface to VOMS Admin.

The VOMS database stores the membership of the virtual organization, their roles and attributes, and data only used by VOMS Admin. The VOMS daemon only reads the database, while VOMS Admin component both reads and writes the database.

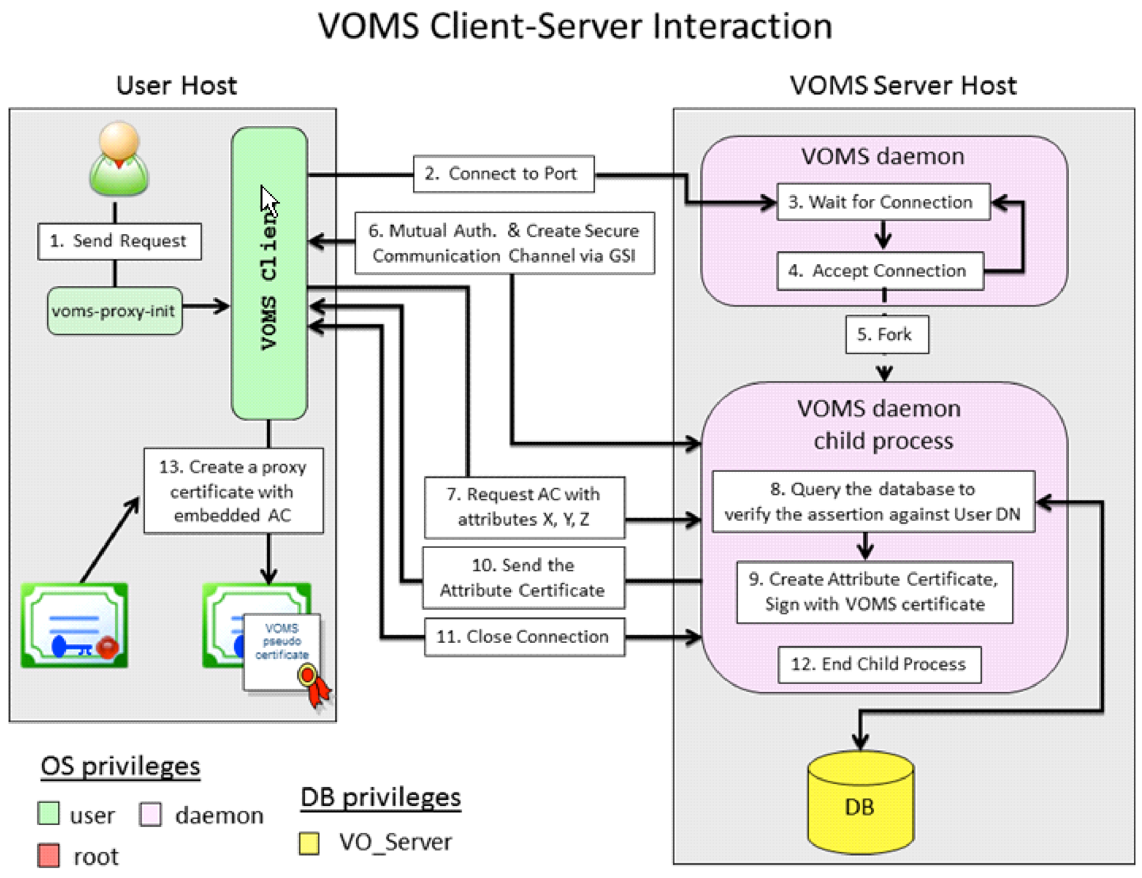
VOMS Client-Server Interaction
User Host
VOMS Server Host
1. Send Request
voms-proxy-init
VOMS Client
2. Connect to Port
6. Mutual Auth. & Create Secure Communication Channel via GSI
VOMS daemon
3. Wait for Connection
4. Accept Connection
5. Fork
VOMS daemon child process
13. Create a proxy certificate with embedded AC
7. Request AC with attributes X, Y, Z
8. Query the database to verify the assertion against User DN
10. Send the Attribute Certificate
9. Create Attribute Certificate, Sign with VOMS certificate
11. Close Connection
12. End Child Process
VOMS pseudo certificate
DB
OS privileges
user
daemon
root
DB privileges
VO_Server

Figure 2: VOMS Client-Server Interaction Diagram

All interactions between the VOMS server and a VOMS client follow a pattern shown in Figure 2. To begin the client-server interaction, the client process makes a connection to the VOMS daemon. When the connection is made, the VOMS server uses a forking server model to handle all requests, so a forked copy of the VOMS daemon is created that handles the interaction with this connection. Next, the VOMS daemon checks the user authentication certificate DN for authorization, and that the requested attributes are valid. If authorized, the VOMS daemon returns an attribute certificate binding the user DN with the requested attributes, and the client creates a new proxy certificate with the returned attribute certificate embedded. After servicing the request, the forked VOMS daemon exits.

The above sequence of steps may be repeated any number of times. The forked server model allows concurrent execution of request, and isolates requests from each other.

The main resource that the VOMS daemon manages is the VOMS database, which contains all the information about users in a VO with their roles and attributes. The database also contains data used by VOMS Admin. VOMS can use an Oracle or MySQL database. In our assessment we used a MySQL database.

The VOMS daemon also uses other files such as the TRUSTED_CA directory that contains files describing all the trusted Certificate Authorities (CA) including the CA certificates and signing policy files of the CAs trusted by the Grid Security Infrastructure (GSI). The */etc/grid-security* directory contains the X.509 host certificate and private key. The CONFIG_DIR VO_NAME directory contains the configuration files of VOMS, including a file with the database user and password. Finally, the log files are used to keep track of the activity performed by VOMS Core.

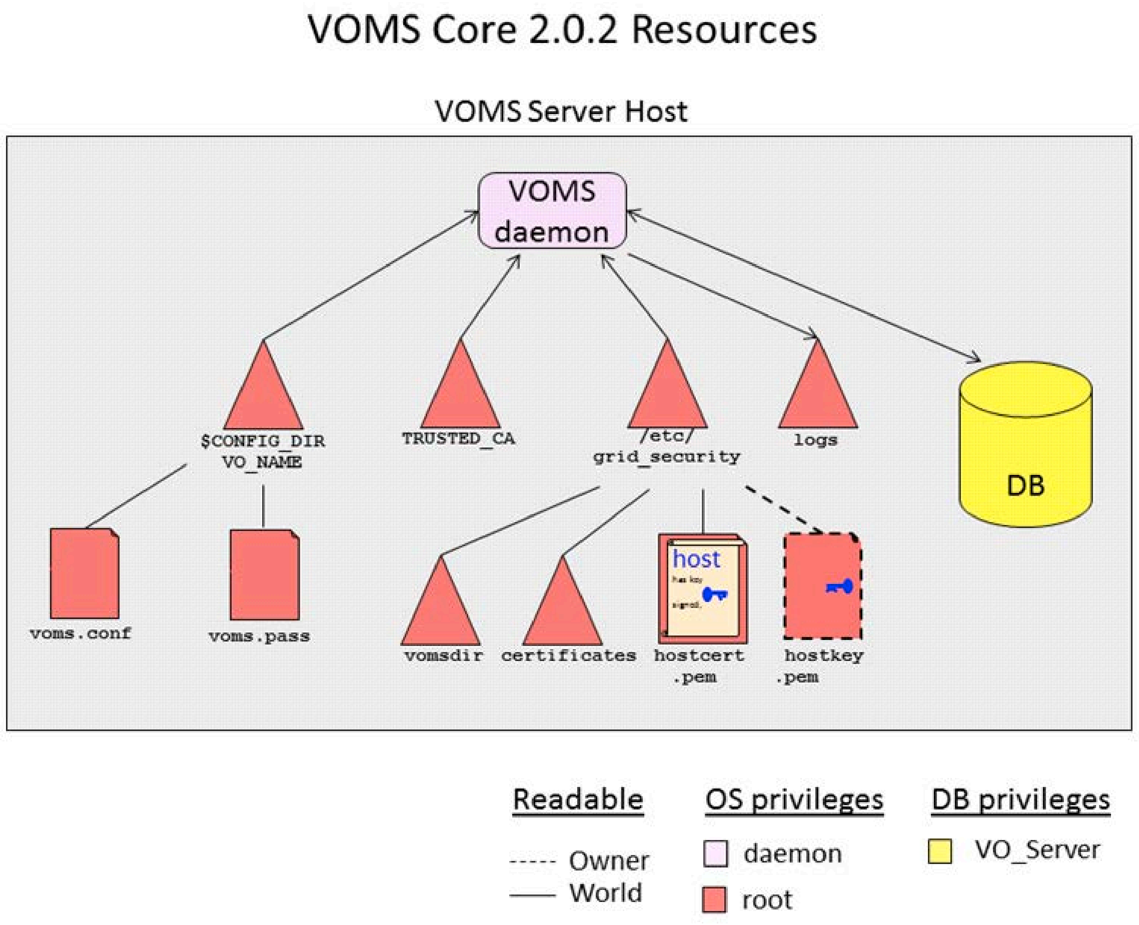

Figure 3: VOMS Core Resources Diagram (VOMS Client)

On the client side, as shown in Figure 3, the main resource that the VOMS Client manages is the user certificate/key, called respectively *usercert* and *userkey*. Both PKCS12 and PEM formatted credentials are acceptable.

3. WMS 3.3.4

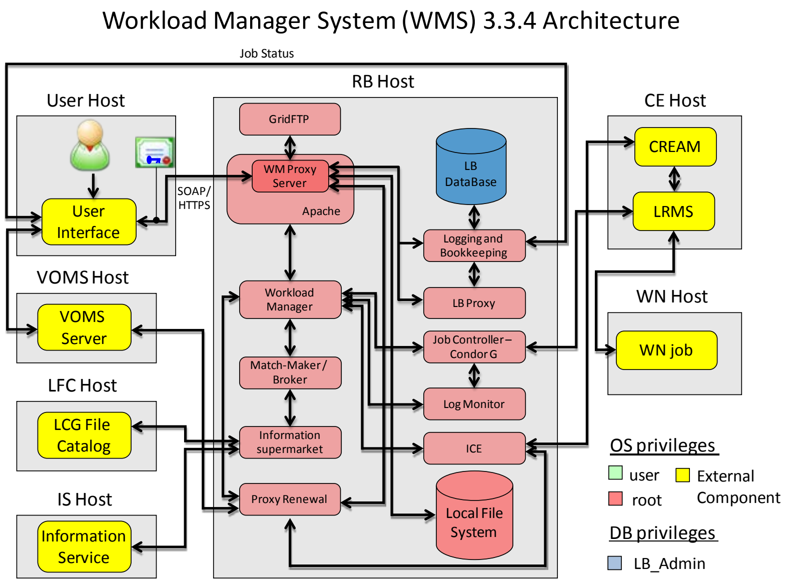

Figure 4: WMS Architecture Diagram

As shown in Figure 4, the Workload Management System (WMS) is a collection of components responsible for the distribution and management of tasks across grid resources.

The WMS basically receives requests of job execution from a client, finds the required appropriate resources, then dispatches and follows the jobs until completion, handling failure whenever possible. The core component of the WMS is the Workload Manager (WM), whose purpose is to accept and satisfy requests for job management coming from its clients.

The main resources that the WMS manages are the user certificate/key, the SandBox directory and the log files.