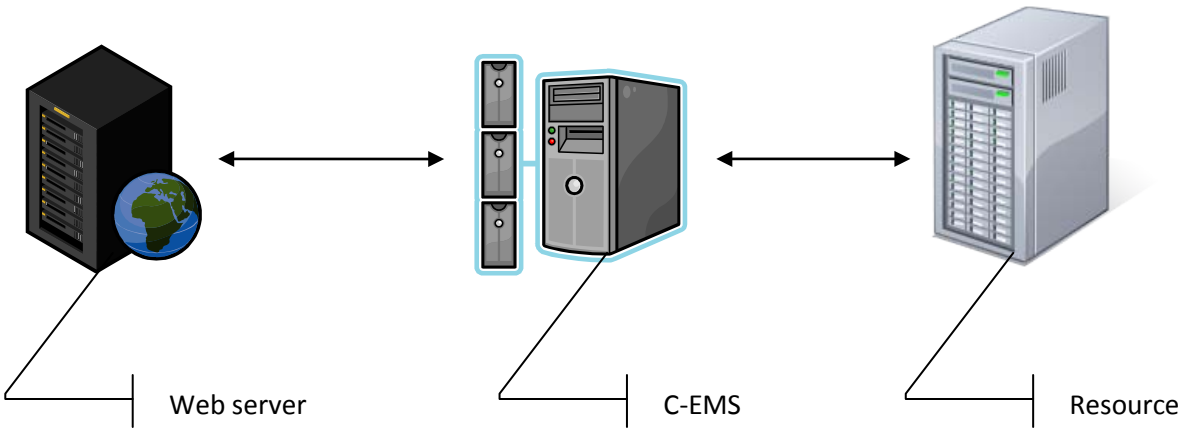
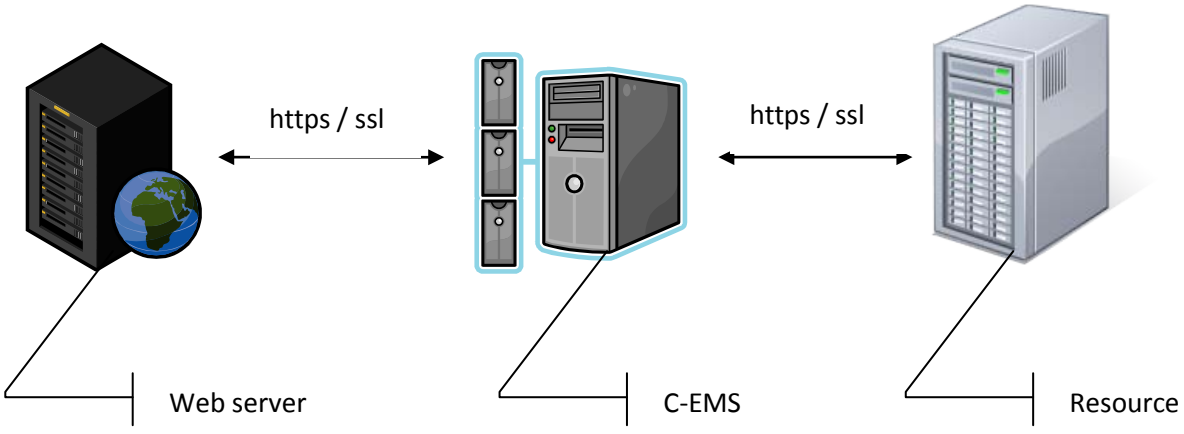
C-EMS Security

Convertigo introduces an intermediate server (C-EMS) between a Web server and a resource (mainframe application, Web site, Web service, RDBMS, etc.):



HTTP / SSL

C-EMS basically relies on HTTP/SSL for security:

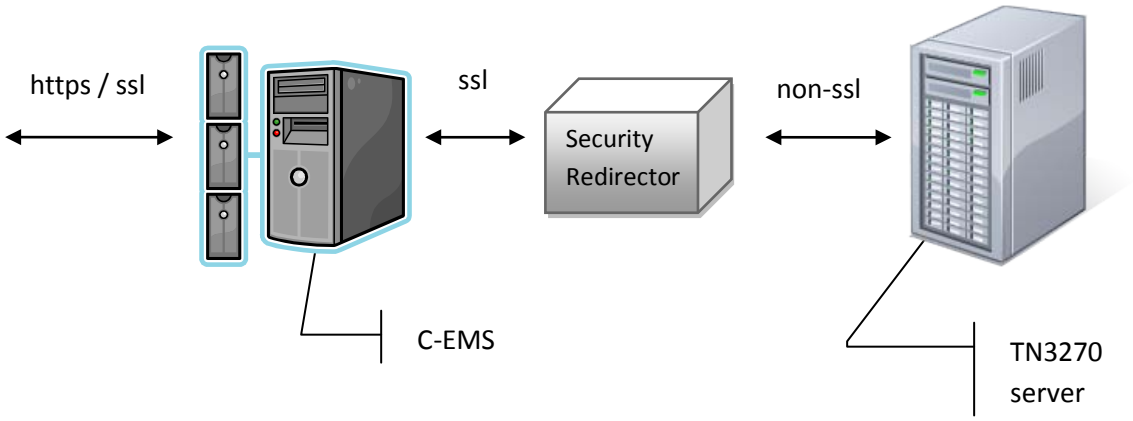


Mainframe access

In the particular case of an access to a mainframe resource, C-EMS uses 128-bit, 168-bit Triple DES, 40-bit or 56-bit DES SSL encryption. In that case, C-EMS can also manage both client and server certificates when they are required.

If the TN3270 server does not support the secured SSL protocol; Convertigo can provide an optional autonomous component (Security Redirector) which secures and encrypts all connections to a non-SSL ready TN3270 server. This component acts as a port mapper, redirecting any incoming SSL-based traffic to non-SSL plain TCP-IP on the other side.

Once installed, the Security Redirector will hide mainframe identity to all users.



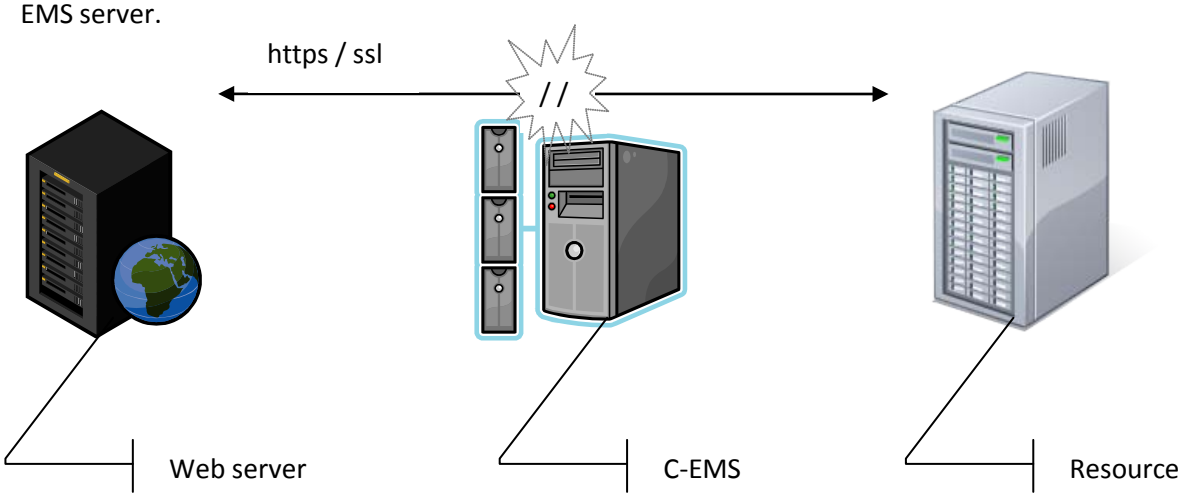
Local password

Sometimes, a system login is used instead of end-user credentials to connect to a resource. It is a generic system account that is shared by all connections to access the resource.

In that case, the login / password information will have to be stored somewhere within C-EMS. It is a recommended practice to avoid hardcoded user login and password defined in Convertigo projects (source code). This information should typically be stored in an encrypted file (see section C-EMS local information hereinafter) that will be used securely by C-EMS. Convertigo programmers can use DESKey, a built-in triple DES based cryptography component, to store and access encrypted credentials.

C-EMS server security

However, C-EMS has to intercept, read, parse and manipulate data flows during its normal execution. This means that the end-to-end SSL connection is interrupted somewhere within the C-EMS server.



A malicious person (user, employee…) could try hacking some of this information by redirection from C-EMS internal resources (memory, Java objects…) to external or internal devices.

This implies the C-EMS server must be deployed and monitored according to common J2E security policies and best practices:

- Limit the number of system accounts declared on the server to administration accounts.
- Restrain the access rights of these accounts to the strict minimum.
- Reduce the rights (read and write) on directories and files to the absolute required
- Disable all unnecessary protocols, ports, etc.
- All other core security patterns and corporate security tools and policies apply here.

C-EMS local information

C-EMS does not store any unencrypted information locally but in LOG files, when they are enabled. LOG files should be normally configured to only keep relevant high-level logging information required for regular monitoring.

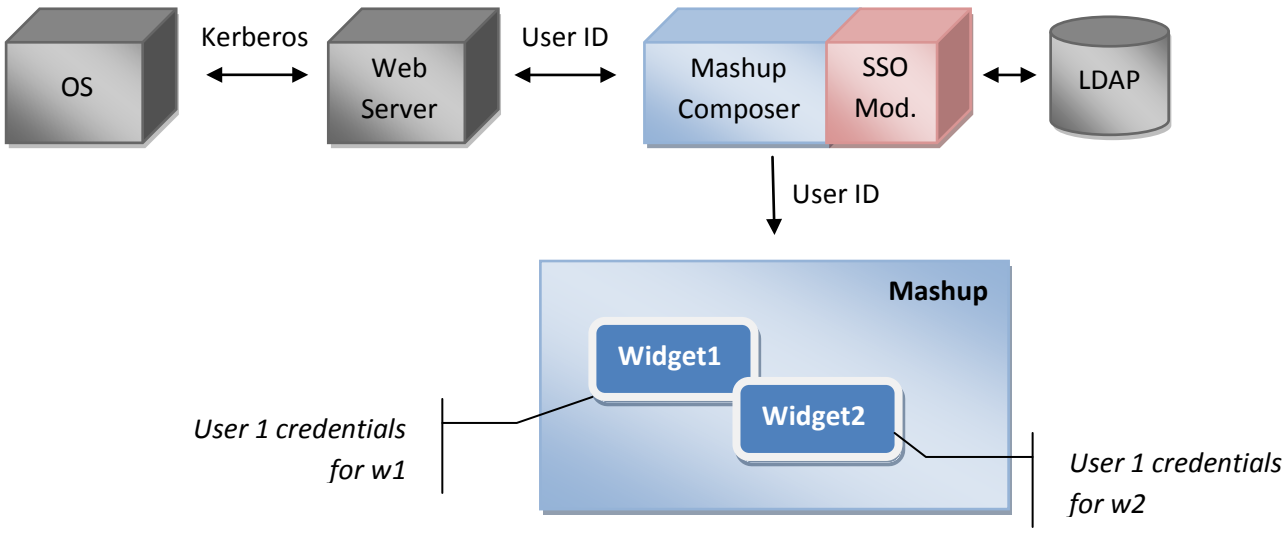
When more detailed logging is required (mostly for debugging or performance analysis) this should be limited in time and a LOG cleaning process must be put in place.

Access to Convertigo Log files should be strictly restricted to system administrator (using hosting operating system security policies) or Convertigo administrator (using Web console administrative credentials).

Single Sign-On

Single Sign-on (SSO) is important to mashups, as they give access to heterogeneous data sources and applications. Convertigo Mashup Composer supports widget personalization. This includes ability to store user credentials within the widget. This allows users to connect to their mashups at once and then navigate between the widgets without additional connection.

Convertigo Mashup Composer can also retrieve user credentials through system based authentication mechanisms (like Kerberos, for instance). In that case, a specific SSO module must be coded into the project to check the user rights against an enterprise directory (LDAP server, active directory…) and automatically synchronize the user credentials in the local Convertigo Mashup Composer directory.



NB: in the case the Mashup Composer used is from a different vendor (such as Adobe Mosaic, IBM Mashup Center for instance) and C-EMS is mainly used as a mashup enablement layer the same mechanisms should be put in place from that tool.

Conclusion

The customer target software architecture must integrate C-EMS in such a way it does not change the security policies already in place. This is made easy because C-EMS is compliant with standard J2E security mechanisms and policies.

As any other J2E component, C-EMS must be deployed and monitored with security concerns in mind. When properly managed within a global security management system, C-EMS will guarantee a secured execution in production.