Authentication and access control in

Sympa mailing list software

May 2004

Serge Aumont & Olivier Salaün
Comité Réseau des Universités
http://www.cru.fr
Campus de Beaulieu, Rennes
France

1 Introduction

Sympa is a rich open-source mailing list software that is widely used worldwide, primarily by academic institutions but also by private service providers (4 000 known sites). Sympa was designed to interoperate with existing university information systems, mainly to allow the creation of dynamic mailing lists and to avoid duplicated data bases for authentication and authorization.

This paper focuses on the authentication and authorization procedures in Sympa, with a particular emphasis on how it cooperates with single sign-on systems, such as CAS and Shibboleth.

Sympa is able to use existing authentication services and offers many possibilities for customization and thus is easy to administer. This article describes how Sympa manages the authentication (LDAP, X509 user certificates) and authorization procedures, with a particular focus on how it cooperates with single sign-on systems, such as CAS and Shibboleth.

1.1 A hierarchical organization

Sympa was designed to manage multiple mailing lists built on a common base, rather than being only side-by-side mailing lists without any link between them.

The system's architecture puts the virtual mailing list servers under the control of the various listmasters, each of which manages its own set of mailing lists. Like Apache’s virtual host, Sympa can manage numerous mailing list services (called Virtual robots) within a single installation. These virtual robots share the same code and a set of configuration files, including web templates and authorization scenarios, which means that only a limited set of parameters must be customized at the virtual robot level. The same inheritance principles also apply to the mailing lists themselves, thus making their configuration lighter.

A single web interface allows the global management of user preferences such as language and preferred MIME format. Authenticated users have a view on their subscriptions and privileges.

Because Sympa shares out both responsibilities and administrative tasks, roles must be strictly and precisely defined in order to allow mailing lists of up to 100,000 subscribers to be hosted. The following roles have been defined (in descending hierarchical order): the super listmaster, the virtual robot listmaster, the mailing list owner, the mailing list moderator and the mailing list subscriber. The super listmaster is responsible for creating new virtual robots and defines the global default parameters inherited by the virtual robots, while the virtual robot listmaster’s primary task is the activation of the mailing list. List owners manage subscribers (subscriptions requests, bounces) and do list setup (determining access control for each feature). If the size of the mailing list requires it, one of the list owners is given responsibility for delegating administrative tasks to a group of list owners. List moderators (who are often also defined as owners) are responsible for the editorial content of moderated mailing lists.
1.2 Dynamic mailing lists based on LDAP

The standard way to build mailing lists is for members to subscribe to the list, though the list owner of working group mailing lists may sometimes add members in addition. Sympa, on the other hand, is able to dynamically build mailing lists based on relational databases (many supported databases) and LDAP directories. Thus, Sympa can build administrative mailing lists using the institution’s own information system. Such mailing lists can range from small student work groups providing horizontal communication to university employee newsletters providing vertical communication. Information regarding these populations is stored either in a relational database or in the institution’s LDAP directory. This feature is used more and more frequently by universities in France and elsewhere in order to systematically provide a groupware communication tool for each student community.

Sympa fetches members email addresses from an LDAP directory (or a relational database) according to the list configuration. Required parameters include server information and an LDAP filter. Sympa has no prerequisite for the LDAP schema but the directory must contain email addresses. The software adapts to the schema of the institution’s directory. For example, for institutions that use GroupOfUniqueNames in their LDAP schema, Sympa adopts a two-step procedure: first, fetch the selected DNs, and then extract the email address associated to each DN. The resulting entries are cached in Sympa’s own database for performance reasons and are updated using a TTL parameter in a dedicated process. Such a dynamic mailing list opens up many possibilities since it can query multiple LDAP (or SQL) servers and even allow static members to subscribe.

1.3 Interfaces to the mailing list service

The Sympa mail interface is comparable to other well-known mailing list software. Commands (subscribe, review, lists,...) are sent to an email address that is unique for each virtual robot. Each mailing list has a main address for sending messages as well as associated administrative addresses (-request, -owner, -unsubscribe,...).

Like most mailing list servers, Sympa’s web interface provides equivalents for commands available on the mail interface (archives, subscriptions, member reviews,...). Other features are only available via the web interface (document repository, list creation and setup, template customization,...). All web pages are defined by templates that have been internationalized (14 languages) and can be customized by the listmasters. User authentication is either password- or certificate-based, and privileges for each operation are evaluated by an authorization engine (described in section 4).

In the past, external applications would access Sympa’s database directly, thus bypassing Sympa’s authentication and authorization procedures. However, a SOAP server has recently been added to the Sympa package. This server provides an API for integrating simple mailing list services within another application, including a list
subscription form in a PHP page, for example. Instead of allowing direct access to Sympa internal data, the Sympa’s SOAP server is a kind of object oriented API. It will guarantee that the calling program respects Sympa authentication and authorization procedures. The new SOAP interface makes it easy to develop a Sympa Uportal [6] channel, as has been done by the French ESUP project [5], which uses CAS authentication.

2 User authentication in Sympa

Sympa needs to authenticate users (listmaster, owners, moderators, subscribers) for both its mail and web interfaces in order to apply the appropriate access controls to subsequent requests (authorization process). Like most web applications, Sympa started out providing its own authentication service based on internal user passwords and email challenges. However, several other authentication backends have since been added. These backends are defined in an ordered configuration file that allows each of them to be restricted to a subset of users, and are applied using a regular expression appended to the user's email address.

It is crucial to provide various methods for authentication in a mailing list server because, unlike a webmail service, a mailing list server is not restricted to local users. Users from other universities, or from outside of the university community, will certainly use the mailing list service, and Sympa must be able to authenticate them. The backends make this authentication possible.

2.1 Basic authentication

For the mail interface, basic authentication is based on From: message header field. This authentication method is used by all mailing list servers, but it is very easy to spoof the sender email address, as viruses and spams have shown. A better level of authentication security relies on a mail challenge password. For example, when necessary, Sympa may require a confirmation from the mail sender in the form of a random confirmation number.

On the web interface, a password is required for authentication. Like many other web applications, Sympa sends this password to the user by mail at first use (or when the password has been forgotten). For both the email challenge and the web password authentication methods, Sympa assumes that the user mailbox is reasonably confidential, and therefore both methods provide the same basic level of authentication security.

Of course, the security level of authentications based on SMTP header fields is extremely low, which is why authentication technique is one of the crucial parameters in the user privileges evaluation. For sending a message to some lists, the SMTP From: header field is good enough; for other more sensible lists, it is not sufficient. Sympa allows the required authentication level to be set individually for each service of a given list (subscription, message distribution, subscriber review, list administration …).
2.2 X509 user certificates

Sympa’s basic authentication method has been extended to support x509 user certificates in order to provide stronger authentication both for accessing through the web and for sending mail. When needed, the listmaster can configure Sympa to require either the HTTPS or S/MIME authentication method for specific lists or operations.

2.2.1 HTTPS user authentication

HTTPS is not only used to encrypt CGI data, but also uses SSL V3 to authenticate end users who have an X509 personal certificate. This method provides stronger authentication and may be more user friendly. However, this method depends on user certificate deployment and cannot be generalized at this time.

The Sympa HTTPS interface uses the mod_ssl features to inherit user authentication from Apache. For this reason, Apache must be configured with a trusted CA (Certificate Authority) bundle and a reliable CRL (Certificate Revocation List), according to the certificate validation policy for this application.

2.2.2 S/MIME user authentication

S/MIME signatures are used for strong email sender authentication, both for messages on the list and for commands sent to the robot. (PGP is under development.) The S/MIME encryption distribution process is also supported. However, the description of this feature is not within the scope of this article. For more information, please refer to “S/MIME and Sympa mailing lists manager” [7]

The Sympa email interface is designed to use the OpenSSL command to verify the email signature and the certificate validity. The listmaster should configure the trusted CA bundle to share the same mod_ssl configuration as the trusted CA. Sympa checks if the message sender and the message signer are the same. But it will not use the smime authentication level (see section 3) if the subject header field is used to send a command to the robot (S/MIME standards do not include message headers when checking message integrity).

Unfortunately, “openssl smime” and “openssl verify” can not yet implement validation with CRL or OCSP (Online Certificate Status Protocol) servers, so Sympa is not able to provide this service currently.

2.3 LDAP authentication

Most listmasters ask for a mailing list service to be well integrated within their information system, beyond HTML customization. Thus, the first thing that must be done is to make the service’s authentication process conform to institution policy. Most universities don’t yet use a Single Sign-On solution, but they do use an LDAP directory as the reference for all password authentication. Sympa can validate user identifiers and passwords against this LDAP directory, prior to formulating the LDAP request that will fetch the user email address associated with the DN.

The LDAP authentication process involves the following three-step algorithm:
1. Search for the DN given the userid
2. Bind with the DN and password to validate the password (using LDAP over SSL if needed)
3. Fetch the email address associated with the DN entry

2.4 Sympa and CAS single sign-on

French universities are currently deploying single sign-on services to solve the multiple authentication problem of web portals. This problem has surfaced in the ESUP project [5], an open-source Virtual campus that has been developed by a consortium of French Universities and that may be generalized at the national level. We’ve been working on integrating CAS, the single sign-on software chosen by the ESUP consortium, into Sympa’s web interface.

CAS [3] (Central Authentication Service), a single sign-on web developed by Yale University, draws heavily on Kerberos concepts, using tickets and allowing the delegation of user authentication. Like most single sign-on
web systems, CAS uses HTTP redirections quite a lot. A brief description of CAS authentication is given below:

1. An HTTP Client needs to be authenticated, and the application does not recognize any cookie or session id.
2. The application server redirects the client to the CAS server.
3. One of two things happens:
   a. The CAS server needs to authenticate the client, but no authentication cookie is found, so the user fills out an authentication form.
   b. The CAS server recognizes the user session via a cookie
4. The CAS server provides a ticket and redirects the client back to the original application.
5. The application receiving the CAS ticket connects to the CAS server to validate the ticket and retrieve the user id from CAS server.
6. The user is authenticated, and the application maintains a session using its own mechanism.

These redirections apply the magic of single sign-on, transparently verifying that the user was previously authenticated by the CAS server and skipping the login step. Unless the application supports multiple CAS servers (or it uses CAS authentication only for a subset of users), introduction of CAS has low impact on the application.

2.4.1 Authenticating against multiple CAS servers

When Sympa is configured to use more than one CAS server, any redirection to a CAS server is a dead end for a user that can't login on that particular server. Thus CAS provides a way to perform non-blocking login redirections by only returning the user login status and not requesting a login action. This feature allows Sympa to test whether or not a user has already been authenticated with different servers. If none of the CAS servers has an active login session with that user, the user will be redirected back to the Sympa login banner. The Sympa login banner proposes an email/password form along with a menu listing all the supported CAS servers, thereby acting like a Shibboleth WAYF (Where Are You From).

The redirection technique has one major drawback however: if the CAS server is unreachable, the user is never brought back to Sympa. While this risk is acceptable for institutions using their own CAS server, the average risk is too high for virtual organizations that rely on several CAS servers because the whole service is interrupted if only one CAS server is down. Sympa allows the listmaster to configure each CAS server in order to disable automatic redirection.

Two Sympa optimizations are planned to reduce the risks related to CAS redirections:

- Sympa will adjust the redirection order using heuristics based on the user IP address in order to determine which CAS server is capable of authenticating the user. This optimization primarily addresses performance issues, but it would be a rather tedious task for the listmaster to maintain the set of IP addresses.
- Sympa will regularly ping each CAS server and disable unreachable servers until they come back online.

2.4.2 CAS logout

When using its native authentication, Sympa proposes a logout button that will remove its session cookie and safely close the current login session. For users who have logged in on a CAS server, the logout button should provide the same feature using a redirection to the appropriate CAS logout URL. However, some administrators don't want any target application to provide a "global logout" function, believing that the portal itself should propose such a feature. Sympa keeps track of which backend was used to authenticate the user's session cookies, and then adapts the logout behavior to that backend.

2.4.3 Obtaining user attributes

CAS only handles user authentication; it does not distribute user attributes. Thus, the CAS authentication provides a user identifier, but not the user email address that Sympa requires. To remedy this problem, Sympa assumes that the user attributes, including the email address, are stored in an LDAP directory. Along with the CAS server definition, the Sympa configuration file also includes a description of an LDAP query for fetching the user email address associated with each user-id. The LDAP server and filters are defined in the same way that they are for LDAP-based authentication.
2.4.4 CAS authentication across Uportal

Sympa can be queried through an Uportal [6] channel via its SOAP interface, described briefly earlier. Authentication on the SOAP interface requires an email address and a password, or a CAS Proxy Ticket. In the latter case, the Uportal channel acts as a CAS proxy and will request a Proxy Ticket for the Sympa service; the Sympa authentication library is able to validate such CAS tickets. Since SOAP is based on HTTP, Sympa uses HTTP cookies to maintain authentication sessions with the user.

3 Managing access control with “authorization scenarios”

Since Sympa proposes three interfaces (mail, web and SOAP) for most of its services, the access control rules which are applied to the same function via the different interfaces must be equivalent. Thus, the authorization process and the way that authorization rules are expressed must be common to each function. To reach this goal, Sympa implements its own access control language, “authorization scenarios”.

When a service is requested, it is mapped by Sympa to the corresponding authorization scenario and evaluated by the authorization engine. A service request is associated to a context (user attributes and roles, list configuration, environmental variables, authentication method, mail headers and body…) that will be used by the authorization engine. Depending on the context, the scenario will describe which action(s) must be performed to achieve the authorization process.

The scenario file contains sequentially evaluated rules and is composed of three elements:
- a condition,
- an authentication method, and
- an action.

If the service request context matches a rule condition and satisfies its authentication method, then the corresponding action is returned by the authorization engine to the main program. The action might be
- a request for a stronger authentication process,
- a validation process (validation by list moderator, list owner or listmaster)
- the execution of the requested service, or
- a rejection.

Here is an example of a sophisticated authorization scenario for a moderated mailing list:

| title.us Moderate list for non subscribers or multipart messages |
| title.fr Modération pour les non abonnés ou messages multipart |
| is_editor([list->name],[sender]) | smtp,md5 -> do_it #rule 1 |
| match([header->Content-type],/multipart/) | smtp -> editorkey #rule 2 |
| is_subscriber([list->name],[sender]) | smtp,md5 -> do_it #rule 3 |
| true() | smtp -> request_auth #rule 4 |
| true() | md5 -> editorkey #rule 5 |

For non-members of a list, Sympa will evaluate the scenario above three times:
- first, rule (4) is matched and requests that the message sender be confirmed (request_auth)
- confirmation is then identified by the md5 authentication method in rule 5, and the message is forwarded to the moderator (editorkey)
- finally, if the message is validated by the moderator, rule 1 matched and the message is distributed to list subscribers (do_it)

Sympa defines a limited set of condition operators (is_subscriber, is_owner, is_listmaster, equal, match, …) and actions (do_it, reject, request_auth,…). Condition arguments can be variables (noted [var]), such as user attributes, list configurations and environment variables.
Sympa comes with a set of 100 scenario files that can be overloaded according to Sympa hierarchical organization, and of course, listmasters can write new scenarios. Due to the rich context that is passed to the authorization engine, the behavior of Sympa services can be extended to meet the listmaster's needs very easily without requiring any changes in the code. From the list owner’s point of view, however, a scenario is just a list of possible values (scenario titles) for one list parameter; list owners are not allowed to create or edit existing scenarios.

In the future, the authorization scenarios will be extended to provide more parameters to scenarios, and Sympa’s web admin interface will also be extended to include a scenario editor, so that list owners can write their own scenarios based on a few standard rules.

4 Sympa and Shibboleth

Shibboleth [4] is a software developed by Internet2 that aims to interconnect local authentication systems with distant web resources, thus allowing sophisticated access control management. Shibboleth has three main components:

- The Origin package, installed in the user’s home organization, is a front-end to the local authentication system and user attribute database.
- The Target package, installed in front of a web resource that requires Shibboleth access control, will communicate with the Origin to perform user authentication and user attribute negotiation.
- The WAYF (Where Are You From) is the central component, shared by multiple organizations. It is in charge of guiding users to their Home organization Shibboleth component (Origin) in order to be authenticated.

Internet2 has put a request to integrate Shibboleth with Sympa for their own use. The process of integration has given us a better understanding of the Shibboleth architecture and a clearer vision of how it could be used by French universities.

Within the Shibboleth architecture, Sympa’s web interface can act as a protected resource on the Target side. Sympa redirects the user to the WAYF that will delegate user authentication at the appropriate Origin site, and in turn, Sympa receives the user identity and attributes (Student, Teacher …) from Shibboleth Origin site allowing it to grant the user appropriate application privileges.

The changes that were made to Sympa to allow Shibboleth authentication should adapt to any other single sign-on software — as long as that software is able to protect a single URL (not the whole site) and assuming that the user attributes are passed to Sympa as HTTP header fields.

4.1 Shibboleth authentication

A single Sympa server may authenticate users by means of various authentication methods (internal passwords, LDAP passwords, X509 client certificates, CAS), depending on the user population. In addition, many Sympa features (including web archives) are also accessible to unauthenticated users. Therefore, unlike static web pages, Sympa’s web interface can not be located behind a single authentication point but rather needs to trigger authentication when, and only when needed.

Sympa delegates Shibboleth authentication to the web server (Apache) via the ShibRM module (RM = Resource Manager). This module acts like Apache’s mod_auth module to restrict access to a web URL for a set of authenticated users. To allow a triggered authentication, only a Sympa-dedicated URL (/sso_login) is protected by ShibRM. Users who select “Shibboleth Inqueue authentication service” from Sympa’s login banner are redirected to the /sso_login URL that will guide them to their home organization’s login banner. Once authenticated, users are brought back to Sympa and an authentication HTTP cookie is set.
4.2 User attributes

Shibboleth management of user attributes in both Origin and Target packages is quite sophisticated, allowing various data sources to be used; the exchanged attributes (including user ID and email addresses) can be filtered on both sides.

The user identity and attributes are carried from the ShibRM to Sympa via HTTP header fields. Sympa requires that at least the user email address be set because this is the user identifier for the mailing list manager. Any additional user attributes provided will be stored in Sympa’s cache for later use until the user logs out. When it gets the user’s email address, Sympa considers the user authenticated and sets a Sympa session cookie.

During the login session, Shibboleth user attributes (obtained from the Origin site) are available in web/mail templates and in authorization scenarios (via a variable structure usable according to scenario conditions). In this way, the web interface can include additional user information and user privileges can be tailored (extended or restricted) according to the information provided by the user's home organization. Below is an example of a scenario defining the privilege for subscribing to a mailing list based on a Shibboleth attribute.

<table>
<thead>
<tr>
<th>title.us Subscription restricted to members of Rennes University</th>
</tr>
</thead>
<tbody>
<tr>
<td>match([{user_attributes-&gt;HTTP_SHIB_ORIGIN_SITE},/Rennes/]) md5 -&gt; do_it</td>
</tr>
<tr>
<td>true() smtp,md5 -&gt; reject</td>
</tr>
</tbody>
</table>

4.3 Changes in the Sympa user interface

If Shibboleth servers have been defined in auth.conf, a menu is added to the Sympa login banner with the list of defined Shibboleth (or CAS) servers to which the user can be redirected. Users who select a Shibboleth server are not directly redirected to the corresponding WAYF, but rather to a Shibboleth-protected *sso_login* page that will redirect the user to the WAYF.

The logout function is unchanged, and closes only the Sympa login session. It neither performs a logout at the user’s home organization nor at the local level of the ShibRM.

5 Conclusion

As described above, Sympa can be considered to interoperate with complex information systems. The authentication architecture, which is an important concern for interoperability, has been made highly configurable. The modular design of it has facilitated the integration of different major authentication technologies, including LDAP, single sign-on and X509 certificates. In the near future we will considering support for SAML authentication on the Sympa SOAP server.

We also have to take into account the fact that information systems are moving toward dedicated attribute delivery services. This means Sympa has itself to improve its own attribute management. Between the authentication and authorization layers Sympa’s attribute management process will be extended to integrate the use of different attributes sources. The scenario engine would thus benefit from these new user attributes.

6 Références