on budget & time

Project Reference
Graylog LDAP

Owner: Gregory Prays

Date: August 21, 2012

1 Introduction

We are using Graylog2 (http://graylog2.org) and domain network (Active Directory). You need to change the built-in **graylog** authorization system, so that through the web-interface authenticate via LDAP.

1.1 Functionality Routine

1.1.1 About Graylog

The system has "streams" (statistics sections, that can be called an access groups).

Current built-in **graylog** authentication have two types of users: Admin and Reader. Admin has access to all streams, and Reader only to explicitly specified.

Admin configures the user profile to define streams that can be accessed by user with Reader permissions.

1.1.2 About LDAP

All users are stored in LDAP, and users have groups that define the access.

1.2 Project Inside

List of improvements:

1) You need to download source codes of **graylog** web-interface (scripts written in ruby). After the end of modifications, you need to create a patch for the **git** control system. So that we could then apply the changes to our system. E.g. there is no need to send all the source code but just the deviations from the basic scripts.

2) Add ability to specify, in a **graylog** configuration file, the settings for ldap-server, and map groups from LDAP-server to the user types and "streams" (see example).

3) It is necessary, that when authorizing a web-interface into the **graylog** system - LDAP-authentication were used. When authorizing, it is checked whether the user has the access to **graylog** service. If the group does not have access - denial of authorization. And if access exists:

   a) if a user with that username does not exist in the local database, then it is created

b) if a user with that username exists in the local database, then the information on it is updated

User information that must be maintained/updated: user type (Admin or Reader) (see example) and if it's Reader, the "stream" to which it has access (see example).

It is necessary to check the authorization for each user from the LDAP and if he ceases to be to the right access group, to deprive him of the rights, and vice versa - if he is now part of the access group, which previously was not, then update the relevant permissions in **graylog** system.

1.3 Examples

1.3.1 Example 1

In LDAP, there are groups of users: *graylog_admin*, *graylog_stream1, graylog_stream2* and *graylog_stream3*.

In LDAP there are also user **myadmin** (part of the group *graylog_admin*) and the user **myreader** (part of the groups *graylog_stream1, graylog_stream2*).

The **graylog** configuration file will indicate correspondence of our group *graylog_admin* to the Admin user type of **graylog** system.

Also the **graylog** configuration file will indicate correspondence between our groups *graylog_stream1, graylog_stream2 graylog_stream3* and streams in **graylog**: «Stream1", "Stream2" and "Stream3."

Along with **myadmin** user authentication, information that he is in **graylog_admin** group will be taken from LDAP and local user in **graylog** system will be maintained/updated with Admin user type information.

Along with **myreader** user authentication, information that he exist in groups *graylog_stream1, graylog_stream2* will be taken from LDAP and local user in **graylog** system will be maintained/updated with Reader user type information and that he has access to streams: "Stream1", "Stream2" and "Stream3 ".

2 Technical Details

Aspect	Type
Programming Language	Ruby
Technology	Graylog2 (http://graylog2.org)
Prefix	com.onbudgetandtime

2.1 Coding Standards

All code should confirm Ruby best practices like http://oldwiki.rubyonrails.org/rails/pages/CodingStandards.

2.2 Implementation Specifics

Implementation can and should re-use existing modules and methods. It can also use frameworks and libraries as soon as they a free and can be used in commercial software without any charge.

3 Change Management

Described functionality can be change only with an explicit written approval. Such changes should not harm the functionality but make is more usable and clear.

4 Acceptance and Quality

To be accepted application should be:

- Bug-free and well tested
- Not contain glitches and visual issues
- Has well commented and structured source code

5 Warranty

All errors that are discovered in the first 12 (twelve) months of application operation should be fixed at no extra charge.