NGBAuth - Next Generation Batch Authentication

Zakarias Juto

August 11, 2015

Abstract

This document describes the prototyping of a new solution for the CERN batch authentication of long running jobs. While the job submission requires valid user credentials, these have to be renewed due to long queuing and execution times. Described within is a new system which will guarantee a similar level of security as the old LSFAuth while simplifying the implementation and the overall architecture. The new system is being built on solid, streamlined and tested components (notably OpenSSL) and a priority has been to make it more generic in order to facilitate the evolution of the current system such as for the expected migration from LSF to Condor as backend batch system.

Contents

1 Introduction

This document pertains to the implementation of a replacement for the old LSFAuth system [4] responsible for renewing the Kerberos (and AFS) credentials of long running batch jobs. It contains a quick rundown of the specific requirements and then explains the new implementation. Finally a quick security analysis of how the system could be attacked by a malicious entity and which steps have been taken to protect against this. For more information on the system, you can refer to the project twiki [7] or the source code [11] if you have the correct permissions.

2 The problem

The main issue that this system aims to solve is the fact that user credentials are not valid for a very long time. The long running jobs a user may start may run for much longer than that users credentials are valid. In fact, the job may sit in a queue for so long that when it is finally allowed to start the users credentials have already expired. Thus there needs to be a secure way to automatically renew credentials without any user intervention.

3 Requirements

The primary requirements for the system are:

- The link to AFS must be cut. The old system uses the users AFS token to authenticate for new ticket granting tickets. This token is considered to be of much too high value to keep passing around.
- Remove reliance on legacy technology like ARC which is an old CERN developed protocol for remote execution.

Secondary requirements:

- The system should be able to deal with a dynamic array of batch machines which can leave and join quickly. This means for example to stop using static IP lists which are used at the moment to verify the origin of requests.
- Support standalone deployment via RPM.

4 Implementation

The final implementation that was chosen, the impersonating Super Server solution, will be described in some detail below. The system consists of three major parts and each will be handled in order. The three parts are split between the users submitting jobs, the batch machines running the jobs and a server responsible for distributing new ticket granting tickets. The lynchpins holding this system together is OpenSSL and the Kerberos functionality of AP_REQ messages. OpenSSL handles secure transfer between subsystems through symmetric key encryption with AES-256-CBC and sender verification with signatures. Kerberos however is responsible for the main authentication part. This is done by basically staggering the Kerberos standard way of authenticating a user to a service. In a normal scenario, a user wishing to contact a service would contact the KDC and request a service ticket for that service. Eventually the user would have an AP_REQ message which consists of this service ticket, encrypted with the service key, and an authenticator encrypted by the session key. This would be sent to the service which decrypts and verifies the content before granting access for the requesting user. What NGBAuth does is that it stutter steps this procedure by adding some extra steps between getting the AP_REQ message and presenting it to the server.

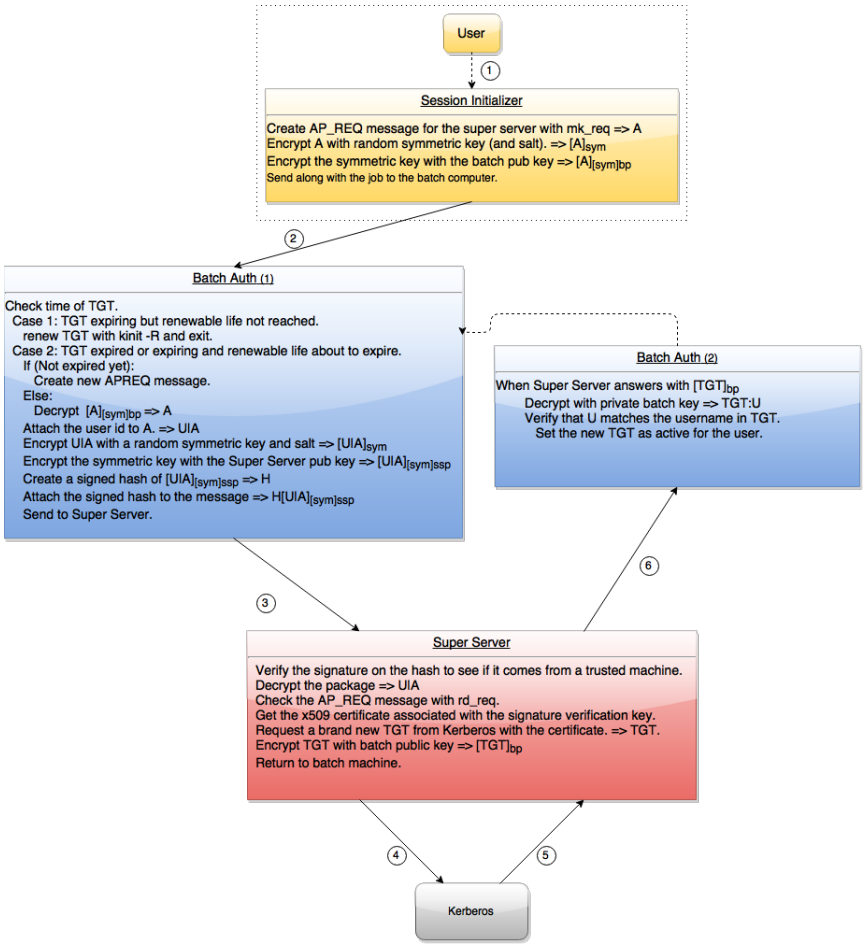

Figure 1: Schematic showing general functionality of system and subsystems. A less compressed version can be found on the twiki [7].

4.1 Specific technologies

The system is written mainly in Perl for a couple of reasons. First off the old system was, which meant it was easier to cannibalize certain parts of the code. Secondly Perl had a very easy to use interface to Kerberos [5] and finally Perl has the concept of taint mode [6] which makes it easier to ensure that you have sanitized all input and environment variables being used by the system. The system makes heavy use of the MIT implementation of Kerberos 5 [1] for the main functionality as well as standard OpenSSL for encryption and signatures, specifically all symmetric key encryption is done with AES-256-CBC+Salt. This means for example that a machine running any of the three major parts will require a working OpenSSL instance. There is a specific part of the system that is written in c however and that is a homemade kerberos function for extracting the username from an expired AP_REQ message which is not possible by standard Kerberos functionality.

4.2 Session Initializer

This is the part that is run on the client side. In conjunction with submitting a new batch job, this script is also called. This is analogous to how the old system called set_batch_token on job submission [8]. However instead of extracting the user AFS token it creates an AP_REQ message containing a service ticket for the Super Server using the Kerberos mk_req function [2]. Then it encrypts it using a symmetric key which is in turn encrypted with the batch computer public key. A composite of these parts is created and glued to the batch job. The actual transfer of the encoded composite is not handled by the system itself but by the same mechanisms as the old AFS token and session key was transfered previously and eventually Condor will take over responsibility of this transfer.

4.3 NGBatchauth

When a batch job is about to start, and on an appropriate schedule afterwards, say every X minutes, this script is called by the batch machine running the users job. This subsystem actually has three different scenarios in which it can find itself.

Case I: TGT expired.
This is the expected behaviour for a new job arriving at the batch machine after sitting in a queue. In this case the system takes the encrypted composite sent from the user, decrypts the symmetric key using the batch private key. Then decrypts the composite with the symmetric key and extracts the AP_REQ message within. The system creates a composite with the message and the current users username and then encrypts it again with a new symmetric key which is then encrypted by the Super Server public key. The batch system then signs a hash of this composite and sends the encrypted composite and it's signed hash to the Super Server over a standard TCP connection. After which it awaits a response.

Case II: TGT validity time and renewable life above some threshold.
Using the same threshold as the old lsfauth system this means at least 3600 seconds left. In this case the system utilizes the remaining renewable time by performing a local renewal of the TGT using *kinit -R* and exiting.

Case III: TGT validity and renewable life both expiring but still valid.
When the TGT is still valid but below the threshold for performing a local renewal the system on behalf of the user creates a new AP_REQ message using similar mechanisms to the session initializer and sends this to the Super Server in exchange for a new TGT. The value of doing this is that the system reestablishes a more standard way of Kerberos authentication.

If the Super Server approves of the message being sent in case I or III it responds with a new TGT, encrypted by a symmetric key that the batch computer private key can decrypt. The batch machine extracts the TGT and after validating the username on the TGT to that of the current user places it in the users credentials cache.

4.4 Super Server

The Super Server is the final part of the system and the one responsible for actually requesting new ticket granting tickets from the KDC on behalf of the user. This impersonation is done by utilizing a secondary already existing x509 certificate glued to every account that will be running batch jobs. The general procedure the Super Server follows is that it opens and listens to a tcp socket. When it receives a message it checks the signature against a list of [pubkey|certificate] tuples and sets the x509 certificate depending on which public key managed to verify the signature, if any. It then decrypts the message and uses the built in Kerberos function rd_req [3] to try and verify the AP_REQ message hidden within. If it succeeds then this means it came from Case III on the NGBatchauth subsystem and the Super Server can proceed to the impersonation after verifying that the

user on the extracted ticket returned by rd_req matches the user supplied in the composite. If however it fails the server checks why it failed. If the reason was either because of expiration or because of a bad clock skew then the server extracts the username from the ticket with the specific c executable and compares it to the supplied user, relaxing security by allowing these expired tickets. If however rd_req fails with any other error message the message is rejected and the server closes the connection. Once verification is complete the server uses the x509 certificate and the supplied username to impersonate the user in order to make a request for a new ticket granting ticket from the KDC. Once the Super Server has this TGT in hand it encrypts it with a symmetric key, which is in turn encrypted with the batch machine public key. Finally the server sends it back over the same connection from which it received the request composite.

4.5 Future development

There are two major points of future development for the NGBAuth system that are apparent at this stage. First is integration with the upcoming Condor batch job system which replaces the current LSF backend. This should be an ongoing process as that system is developed to ensure that NGBAuth does not go in some radically different direction.

The second major point of future development is to have some way to guarantee the order of operations on the rd_req verification. At this point, no formal guarantee exists that it performs things in any specific order, only a very informal one that can be achieved by parsing through the source code [9][10]. Without a formal guarantee this means for example that any future update of MIT Kerberos could drastically change the order of operations. The way to go about this is likely to create a CERN fork of the rd_req function in which the checks can be guaranteed to be done in a specific order. This fork could be built by extending the c executable for extracting the username from expired messages since this code already does the decoding and decryption of messages. It should thus be possible given some time to make it also extract the other fields of the ticket in order to verify them in a specific order.

5 Security

The actual authentication item of this system that replaces the old AFS token is as mentioned previously the possibly expired AP_REQ message containing a regular Kerberos service ticket. What this thing says when it arrives at the Super Server is that the user either has, or has at one point in time, had a valid credential for accessing this service. For specific connections this document will try to look into two major subcategories of attacks. What can be stolen? And what can be injected? Finally the document will look into potential damage to CERN if one of the three nodes are compromised. The connections and nodes considered are the ones from figure 1. The connections here will use the same numbering scheme but step 1, 4 and 5 will be ignored. The first one because it is internal to the submission machine and the fourth and fifth because they are handled exclusively by Kerberos itself.

5.1 Connections

In short for this part there are two principles. First: If it is being sent over the network, make sure it is encrypted and if origin verification is important, signed. Secondly: As much as reasonably possible, verify any input before using it.

5.1.1 Theft

2. The initial AP_REQ message is transported here. It is however encrypted by the batch computer key pair. It could theoretically be stolen wholesale and attached to another job submission however unless the link is secure. But since the username

on the ticket within is specifically checked it could only be used to request TGTs for that user.

3. On this link an AP_REQ message and a username is sent. The entire thing is encrypted so that only the Super Server can read it. Thus if this is stolen, it could be used to request a new TGT, but only for the actual user inside the encrypted composite since that is the name used to request a new TGT.

6. The valuable TGT is sent here. However without the batch computer private key it is useless.

5.1.2 Injection

2. Presumably a fake AP_REQ message (created by someone who has the right to request service tickets) could be injected here along a bad job. Another possibility is that a stolen AP_REQ message could be attached to a bad job and submitted here but once again, without being able to directly modify the ticket within, the message is bound to a specific user.

3. Without access to one of the batch system private keys the Super Server will reject any input on this link since it will not have the correct signature.

6. Presumably, bad blobs of data, encrypted with the batch computer public key could be injected on this link which could possibly be used to DOS the system.

5.2 Nodes

- If a user machine is compromised, bad jobs could be started using the credentials of users logged in to this system.

- If a batch machine is compromised, the private key of that machine could be used to sign bad blobs of data to send to the Super Server. Also the credentials of running jobs could be stolen.

- If the Super Server is compromised, any user attached to the certificates associated with batch users could be impersonated. A possible additional security measure to mitigate the impact of this that is outside the scope of this system would be to introduce a specific opt-in user group on CERN for people running batch jobs since at the moment there are a lot of people who have this capability but are never even interested in the actual functionality. Which means that even though they are not interested in running batch jobs they could be impersonated. Note that this is true for a compromised batchtok machine under the current LSFAuth system as well.

6 Bibliography

[1] Kerberos 5 MIT Documentation. `http://web.mit.edu/kerberos/krb5-current/doc/index.html`. Accessed: 2015-08-05.

[2] Kerberos 5 MIT mk_req. `http://web.mit.edu/kerberos/krb5-current/doc/appldev/refs/api/krb5_mk_req.html`. Accessed: 2015-08-05.

[3] Kerberos 5 MIT rd_req. `http://web.mit.edu/kerberos/krb5-1.11/doc/appdev/refs/api/krb5_rd_req.html`. Accessed: 2015-08-05.

[4] LSFAuth. `https://twiki.cern.ch/twiki/bin/view/IT/LsfAuth`. Accessed: 2015-08-05.

[5] Perl interface for Kerberos 5. `http://search.cpan.org/~jhorwitz/Krb5/Krb5.pm`. Accessed: 2015-08-05.

[6] Perl Tainting and Sanitizing. http://perldoc.perl.org/perlsec.html. Accessed: 2015-08-05.

[7] Project description on CERN twiki. https://twiki.cern.ch/twiki/bin/view/IT/NextGenerationBatchAuth. Accessed: 2015-08-05.

[8] Schematic of LSFAuth. https://twiki.cern.ch/twiki/pub/IT/LsfAuth/Batchauth2.svg. Accessed: 2015-08-05.

[9] Source code for MIT rd_req. https://github.com/krb5/krb5/blob/50a3c3cbeab32577fba2b21deb72a64015c48ec7/src/lib/krb5/krb/rd_req.c. Accessed: 2015-08-05.

[10] Source code for MIT rd_req_dec. https://github.com/krb5/krb5/blob/50a3c3cbeab32577fba2b21deb72a64015c48ec7/src/lib/krb5/krb/rd_req_dec.c. Accessed: 2015-08-05.

[11] Source code on CERN gitlab. https://gitlab.cern.ch/ngauth/NextGenBatchAuth. Accessed: 2015-08-05.