# PCSF Features for Kerberos and SSL

Following are the enhancements that I have done to PCSF in order to make sure it runs in Kerberos and SSL mode.

I am making two basic assumptions about the reader:-

* Reader understands the basic design of PCSF and its usage in vanilla mode
* Reader is familiar with the custom Curl changes required for Kerberos support.

## Changes to Protocol parameters:-

Following parameters (along with their appropriate getters and setters are added as a part of existing parameter structure):-

1. Keytab location.

2. Credential cache location (pointer to file based credential cache) Please do not confuse this use case with SSO.

3. SSL certificate location (i.e pem format truststore)

4. SPN (This is the target SPN to be connected to)

5. Delegated credential (a misnomer. It actually means any arbitrary gss credential being passed).

6. Security flags (for both Kerberos and SSL)

NOTE : The user name parameter is used as principal. So it’s mandatory that it be set in non SSO case.

NOTE: All changes are present in pcsfprtprm.h

## Procedure following while deciding priorities in Kerberos

1. First the credential cache location is checked. If it is absent of any error occurs while reading it (i.e its corrupted or outdated), no error is raised and continues to 2. Otherwise it constructs a gsscredential from the credential cache and passed to curl

2. Secondly the keytab parameter is checked. If it’s found then gss credential is created from it and passed to curl. Otherwise user/pass Is checked. If that is found then gss creds created from it and passed to curl. If none of these work then curl defaults to SSO.

NOTE: In PC Clients curl SSPI is used. Under that scenario only SSO holds meaning.

NOTE: After PCSF ticket optimization story, if optimization is turned on – then the credentials are cached and reused in used in succession. The optimizer holds correct across theads.

NOTE: PCSF does not provide features to write to credential cache file, only reads are allowed.

NOTE: All changes are present in pcsfprtcl.cpp

## Procedure following while deciding priorities in SSL:-

1. Firstly if user has explicitly passed a certificate location – that is considered path of truststore. This is highly discouraged as its not correct as per PCSF ssl design.

2. INFA\_TRUSTSTORE env var is accessed. This points to the directory of truststore. The file expected is $(INFA\_TRUSTSTORE/infa\_truststore.pem.

3. INFA\_HOME is accessed. If found then the file expected is INFA\_HOME/services/shared/security/infa\_truststore.pem. If not found the file expected is INFA\_HOME/clients/shared/security/ infa\_truststore.pem.

4. If none of the above work, then and exception is thrown.

NOTE: All changes are present in pcsfprtcl.cpp

## Debug callback handler for curl:-

1. Unlike KAM, PCSF does not have the luxury of getting curl error messages in the form of exceptions. The reason for this is Curl is a C project which only returns error codes. However it does provide callback handler functionality to get debug information. In order to set it in curl following code is used:-

curl\_easy\_setopt(m\_curl, CURLOPT\_VERBOSE, 1);

curl\_easy\_setopt(m\_curl, CURLOPT\_DEBUGFUNCTION, kerberos\_logging\_callback);

curl\_easy\_setopt(m\_curl, CURLOPT\_DEBUGDATA, &m\_kerbLog);

Here, verbose option enabled debug functions to be activated inside curl. CURLOPT\_DEBUGFUNCTION is a handle to a method whose job is to filter out a subset of these logs and after certain modifications store it to the data structure passed by CURLOPT\_DEBUGDATA.

NOTE: As of the time of writing this, any debug message which starts with KRB5\_DEBUG and KRB5\_INFO are parsed in debug data. I included these messages inside curl as a part of my custom changes to it. Afterwards these messages can be wrapped around exceptions are thrown to caller layers.

NOTE: All changes are present in pcsfprtcl.cpp

## Rewind callback for curl:-

While sending non trivial messages in the POST request body CURLOPT\_READFUNCTION callback handler is defined. It’s a handle to a method whose job is to read information from a user defined data structure pointed to by CURLOPT\_READDATA. This is based on the assumption that the servlet understands this input and can respond in kind. The response is handled by CURLOPT\_WRITEFUNCTION and then written into a user defined data structure pointed to by CURLOPT\_WRITEDATA.

This scenario is sufficient for a single exchange (client connects once and server responds) which was the case in vanilla PC mode. However in case of Kerberos mode the negotiate HTTP mode is followed (in which the client tries to communicate, server responds – needs auth via 401 error code, client sends its message along with auth header and then server responds with the intended response). The first time client sends request the read callback reads the message and moves its pointer within the message to its end. The second time client tries resending nothing is sent (as it mistakenly believes the beginning of the message to be the pointer’s location which is by this time the end of the message). In order to counter this problem – curl introduced a rewinding mechanism CURLOPT\_IOCTLFUNCTION which is a handle to a function which deals with putting the pointer back to its original location.

While suboptimal, I was not able to find a way to send the request only in the second half. It is still an open question how to make it optimized so that this rewinder is not needed at all.

NOTE: All changes are present in pcsfprtcl.cpp

## PCSF intelligence: a way to detect security state automatically.

The following logic is used to determine the security state before making a PCSF call. Note that this logic is a part of name service lookup and does not apply to cases in which url is passed as an argument.

1. If either Kerberos or ssl state is set, PCSF assumes both are set and they are correct. The reason is Client either knows complete information about security state of domain or none at all.

2. If neither flags are set, Client queries domains.infa file (pointed to by INFA\_DOMAINS\_FILE env var) if the security state is present or not. If yes then that is returned. The assumption is that domains.infa file in client is an accurate description on the domain side. If the client believes this to be false, there exists APIs to ascertain the correct security state and update the domains.infa.

3. If domains.infa is unreachable or it dosent have security flags fields in domain entry – a domain state detector call is made to the portals object (a set of nodes – hostname:port passed by user). If it succeeds the domains.infa is updated with the values. All the nodes in the list are queried and when any of them returns the state info, it updates the domains.infa and returns the state to PCSF. This is based on the fact all nodes on the domain will have identical security config.

4. If even nodes are unreachable, PCSF sticks to default values of security state (false for both Kerberos and ssl at time of writing) and proceeds to connect.

NOTE: All changes are present in pcsfsvclkp.cpp