

Windows Eudora Kerberos 5 Architecture

February 24, 2006

1 Introduction

Windows Eudora offers Kerberos 5 authentication support for the POP, IMAP and SMTP protocols.

2 Additional Required Software

Eudora requires two additional pieces of software to successfully perform Kerberos authentication: ticket generating software and a Kerberos DLL.

2.1 Ticket Generating

Eudora does not currently provide the ability to generate Kerberos tickets, instead a user must use a third party product (such as Leash) to generate a ticket before attempting any connection which requires Kerberos authentication.

2.2 Kerberos DLL

Eudora's code is designed to interact with a GSSAPI DLL that would presumably be offered by the site requiring Kerberos 5 authentication. The Eudora GSSAPI code calls into the DLL to do the actual work. The default name of the library that Eudora looks for is "gssapi32.dll" but may be modified by changing the IDS_GSSAPI_LIBNAME resource. *It perhaps would make sense for this setting to be an INI setting rather than just a resource to allow a site to change it more easily but to date we have had no requests for this feature.*

3 The Eudora Code

The class CGssapi in the gssapi.cpp file inside the QCUtills project implements the generic GSSAPI negotiation.

3.1 Communicating with the GSSAPI DLL

The CGssapi class uses function pointers into the GSSAPI DLL to do the actual work of authenticating and generating result codes from the authentication process. These

function pointers are initialized in `CGssapi::Initialize()` and are relatively straightforward when viewed in the context of the code.

3.2 External Access to CGssapi

The only method from this class that is used outside of this module (apart from the constructor and destructor, obviously) is the `Authenticate()` method. This method performs the entire authentication process and returns a code to indicate the result of the authentication. As noted above, if the user does not have a valid ticket at the time Eudora calls into `Authenticate()` this function will fail and the appropriate error code will be returned.

3.3 Subclasses of CGssapi

The `CGssapi` class is an abstract base class and each protocol which uses GSSAPI must provide a subclass of `CGssapi` (for example, POP provides a class `CPOPGssapi`). This subclass must override the following protocol specific methods:

```
virtual CString    GetServiceName() = 0;
virtual CString    GetDLLName() = 0;
virtual void       *GetChallenge(unsigned long *lLength) = 0;
virtual BOOL       SendResponse(char *szResp, unsigned long
lSize) = 0;
virtual void       AddLastErrorString(const char *szError) = 0;
```

3.4 Authentication Functions

Each protocol also provides a function which encapsulates the entire authentication process. This function creates an instance of the appropriate `CGssapi` subclass and calls its `Authenticate()` method. The following code shows how the `POPGssapiAuthenticator()` works:

```
long POPGssapiAuthenticator(void *challenger,
                             void *responder,
                             void *s,
                             char *szUser,
                             CString szHost)
{
    // Sanity checks removed for clarity...

    // Instantiate a CGssapi to do all the work.
    CPOPGssapi gssAuth(challenger, responder, s, szHost, szUser);

    return gssAuth.Authenticate();
}
```