SECURING IBM MAINFRAME BASED WEB SERVICES USING KERBEROS

Gustav Mauer

IONA Technologies Inc.

Gustav Mauer

66 Plettenberg Street

Welgemoed, Bellville, 7530

gustav.mauer@iona.com

ABSTRACT

Web Services offers yet another way to access services in a SOA environment. Mainframes in enterprises often contain valuable applications and data which could provide ideal services for the rest of an organization's IT infrastructure. However, access is needed without compromising the security of the mainframe systems. The OASIS WS-Security specification describes a number of ways to provide authentication information when a web service is called, including a Kerberos profile.

In this paper an overview is provided of the experiences at IONA Technologies in incorporating Kerberos support into the security facilities available for accessing z/OS based mainframe resources as web services. This includes an overview of where Kerberos fits into the overall mainframe security picture, an overview of what the Kerberos support on the mainframe looks like, how Kerberos is used to authenticate web service requests, and IONA's experiences in using the Kerberos facilities on the mainframe.

KEY WORDS

Kerberos, Web services, mainframe integration

SECURING IBM MAINFRAME BASED WEB SERVICES USING KERBEROS

1 BACKGROUND

Kerberos was developed at MIT [4] to provide security for the computer systems on the MIT campus network. This work was done to provide a security facility that would provide secure authentication on an insecure network. It is also possible to develop authorization systems on top of this [5]. Since its development, Kerberos has been incorporated into a number of other security systems.

One of the places where Kerberos was implemented was in the Web Services Security Specification [3] developed by OASIS. This specification includes a token profile for using Kerberos tokens to exchange security information between web service clients and servers communicating over SOAP (Simple Object Access Protocol). Kerberos is an industry verified security infrastructure, and is available on many platforms. It is therefore a natural choice as one of the profiles for exchanging security information.

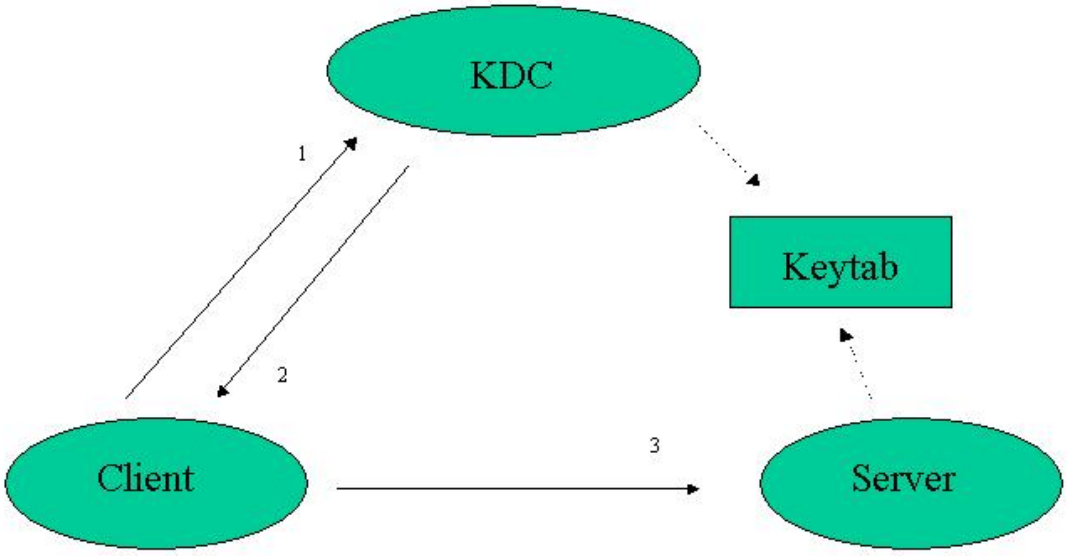
IONA [1] provides Web Services integration products between various platforms, including IBM z/OS based mainframes. Kerberos is available on z/OS [6] and is integrated with the native security services in the z/OS operating system. It therefore provides a good choice as one of the security options offered by products providing mainframe integration for IBM mainframes.

IONA consequently decided to provide Kerberos as a security option when accessing services on the mainframe via SOAP. This support is based on the OASIS specification to ensure that in interoperates with SOAP products from other web services vendors. Other choices provided by IONA for mainframe SOAP security are options like https, certificates, and Username/passwords. In this paper, however, the focus will be on the experiences associated with providing Kerberos support.

2 OVERVIEW OF KERBEROS

Detailed descriptions of Kerberos are provided on the MIT Kerberos homepage [4] and in the book by Brian Tung [5]. What follows is an overview of the technologies involved.

Figure 1



Kerberos provides a way for servers to authenticate the identity of clients using shared secrets. This is done without actually sharing those secrets. A central authority (server) known as the KDC (key distribution centre) shares secrets with each side and uses that knowledge to introduce one to the other.

For clients, the shared secret that is used to authenticate the user with the KDC is usually the user's password. Kerberos is designed in such a way that this is done without actually transmitting the password on the network, so that the password cannot be snooped. For servers, the secret is a key randomly generated by the KDC, which is then stored into a keytab file which must be made available to the server. Servers cannot use a password, in view of the fact that there is not an actual person available to type in a password — which is why a keytab file is used for them.

A client that needs to access a server first requests a ticket for that server from the KDC ("1" in the diagram). The ticket is requested for a specific server. The KDC constructs a ticket using the server's key, and puts it into a reply to the client ("2") which is encoded with the clients password, so that only the client will be able to read/decode it. The client program receives the reply, decodes it with its password to get the ticket, and sends the ticket as part of his request to the server ("3"). The server can decode the ticket, because it was originally created using the server's key/secret, and the server knows that any data inside the ticket came from the KDC.

The ticket contains the client's identity, but it also includes other directives to the server, such as the duration of the ticket's validity. More information on the format of tickets, and how they are designed to prevent various security attacks, is provided in the references, especially [4] and [5].

Some Kerberos definitions:

- A realm is the scope of the KDC. Realms tend to be named for the TCP/IP subdomain name they cover, such as "amer.iona.com". The convention is that realms are specified in uppercase characters. Each client and server exists in a specific realm. Clients and servers in different realms can interoperate using "cross-realm" authentication.
- Clients and servers are known to the KDC (and each other) by their principal. A Kerberos principal uses the format name/instance@realm. For users, the instance is usually left blank, e.g. gustavmauer@IONA.COM. For servers, the instance is usually the hostname on which they are running, e.g. ftp/bruise.dublin.emea.iona.com@IONA.COM.
- A TGT is a ticket-granting ticket. The diagram above does not include the initial exchange between the client and the KDC to obtain a TGT. This is done in the same fashion as other tickets. This TGT then acts as the "shared secret" between the client and the KDC for further service ticket requests. In this way, the client user only needs to be prompted for a password once, and TGT would then be used after that for further ticket requests to the KDC.
- The TGS is the ticket-granting service. This is the portion of the KDC which issues tickets for clients and servers. The AS is the authentication service, which is used to check tickets. These two services are usually combined into a single KDC server.
- A credentials cache is a list of recently-used tickets, including the TGT, which is kept on each host. Caches are kept on a per user bases. Kerberos provides various utilities for manipulating the cache. If the TGT is removed from the cache, the user will be prompted for a password the next time it wants to use a Kerberized application.

3 OVERVIEW OF THE KERBEROS PROFILE

At the time of writing, the Kerberos profile [3] in the Web Services Security specification is still in draft format. The latest draft is dated 27 July 2004. However, this draft has already been implemented by a number of companies, including IONA and Microsoft — implying that the draft is already a *de facto* standard.

The OASIS specification describes how the SOAP message header should be formatted to pass the Kerberos ticket from the client to the server. The specification does not deal with any aspects on either the client or server sides. For example, the specification does not deal with how a SOAP client application will obtain a ticket, or how a SOAP service/server will authenticate the ticket. Many of these details are platform and language specific, so that it depends on where the implementation of the client or server runs, and which language was used to implement it.

The SOAP header used for passing the ticket is a “wsse:Security” header, which then contains a “wsse:BinarySecurityToken” element. This element, in turn, contains a number of attributes. The most interesting ones are the ValueType which is "wsse:Kerberosv5ST", and the EncodingType which is "xsi:Base64Binary". This indicates to the server that the token that arrived contains a Kerberos version 5 ticket, and that the ticket is binary data which has been encoded in base64 format for transmission. The value of this element is then the Kerberos ticket itself.

Here is an example of a SOAP request which contains a Kerberos ticket:

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope
    xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
>
  <env:Header>
  <wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
    <wsse:BinarySecurityToken ValueType="wsse:Kerberosv5ST"
```

```
EncodingType="xsi:Base64Binary">YIICFwYJKoZIhvcSAQICAQBuggIGMIICAqADAgEFoQMCAQ6iBwMFACAAAA
3qSRNw5I+ICz9fKj845JRUFAtjSp+g07iHDlsMIfcBmyLIEVxUCeQ+PqppFQCDWQ5Rt70FSt9YFLQasMTnwjh3HXbm
gaiKy3ZGQlV2kjPSTb16f+awxhjX0nECbPc6//Y9iaEEVhfUufr7eCUPRGdCJ7Zs8beulu6cvPOotr1EzWkn1DR8yG
  </wsse:Security>
    <wsi:Claim conformsTo="http://ws-i.org/profiles/basic/1.0" xmlns:wsi="http://ws-i.org/
  </env:Header>
  <env:Body>
    <GetCustomerDetails xmlns="http://artixmf.iona.com/Customer">
        <GetCustomerDetails_input>
          <customer_id_record>
               <cr_customer_id>2</cr_customer_id>
          </customer_id_record>
        </GetCustomerDetails_input>
    </GetCustomerDetails>
  </env:Body>
</env:Envelope>
```

This example of a SOAP request invokes on the GetCustomerDetails operation in the Customer sample application provided with Artix on z/OS. The SOAP reply is not shown, in view of the fact that it does not contain anything related to Kerberos.

4 STRUCTURE OF THE IONA KERBEROS IMPLEMENTATION

The product at IONA which provides support for SOAP-based access to the mainframe to access existing and new services provided by mainframe transaction monitors like IMS and CICS is called Artix Advanced for z/OS. This product can enable programs inside CICS and IMS to act both as a server, i.e. provide services in a SOA (Service Oriented Architecture) using SOAP, or as a client, i.e. use services provided by other systems using SOAP.

The SOAP support in Artix for the mainframe is provided as a transport plugin. One aspect of this plugin is to process security information that comes from clients, or to generate security information when the mainframe is acting as a client. It uses a variety of security API's for each of the possible types of security information that the client can transmit.

For example, if the client sends a username and password, then the "login" facility of the mainframe security package (SAF) is used to check whether the userid is known and the password is valid, in exactly the same manner as when a normal user logs on to the mainframe. If the communication is taking place over https, i.e. SSL/TLS is used; the IBM SystemSSL API's on the mainframe is used to handle the mainframe's side of the secure connection. Similarly, if Kerberos is used, then the Kerberos API's [7] available on the mainframe are used.

This API used for Kerberos on z/OS is a version of the GSS-API (Generic Security Services Application Programming Interface) [8] which has been modified slightly to make it suitable for use on the mainframe. Most GSS-API based programs would work on the mainframe without change. However, a couple of minor additions have been made to the API to make use of additional services available on the mainframe. For example, it is possible for a program that has already verified itself with the mainframe security system to obtain tickets without having to supply a password again specifically for the Kerberos TGT exchange.

When the mainframe Kerberos support was initially investigated at IONA in 2004, the first step was to configure Kerberos on the mainframe [6] and to set up a KDC on the mainframe. Following this, the demos provided in /usr/lpp/skrb/examples as part of the operating system install were studied to see how the GSS-API is used to integrate with the security services on the mainframe.

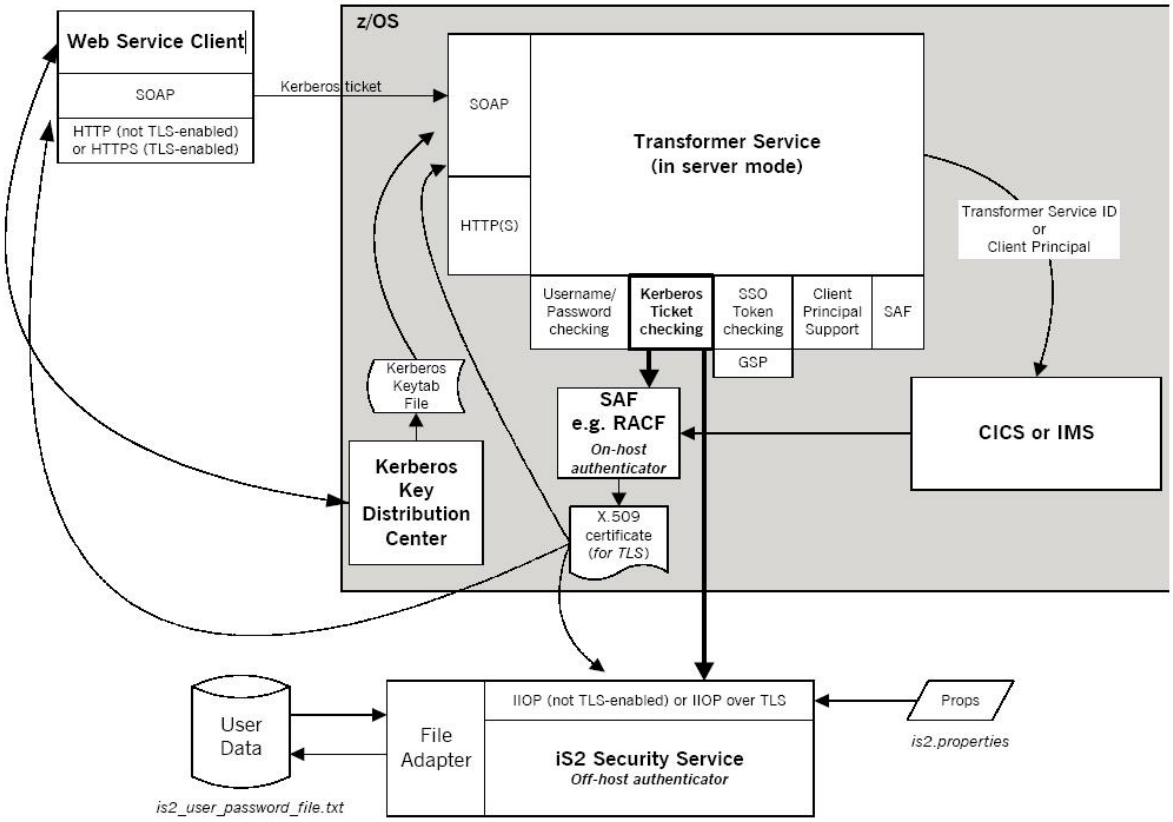
The next step was to investigate other products that provide support for Kerberos tickets in SOAP headers to see how they conform to the standard set out by OASIS, and to ensure that the

standard published matches the actual implementations done by various companies by that stage. The WSSE header support in the SOAP transport plugin was then extended to provide support for Kerberos tickets in addition to the existing security options. The final step was to test the Kerberos support with various other products to ensure that they interoperate.

Some of the issues we faced during this process were:

- Configuring Kerberos on the mainframe was not easy the first time round, since we had to figure out from scratch how it worked. The fact that we work as a distributed team also added to the communication problems on what needed to be done and how. Setting it up on other partitions since then has been a lot easier.
- One funny bit is when using Kerberos utilities like “kinit”, one has to type the password in uppercase on non-mainframe platforms, since mainframe passwords are always stored in uppercase when running the KDC on the mainframe. This tripped up everybody in the team a couple of times.
- The mainframe team also had to figure out how to install and use Kerberos on Solaris and Windows, since there was no general Kerberos infrastructure available in IONA at the time when we did the development. We normally use these platforms for development, so needed Kerberos working there for both development and testing.
- The de facto implementations for Kerberos security in Web Services deviate slightly from the spec in that pure tickets are not used, but rather tokens with extra integrity support. This meant that the GSS API gss_init_sec_context() had to be used to build and verify the token, rather than the simpler Kerberos API’s.
- It took us a while to figure out how to handle keytab files, since the mainframe implementation did not have “kadmin” like other platforms, but rather used its own “keytab” utility.
- The Java JAAS Kerberos support did not like the keytab files generated on the mainframe, since they contained Triple-DES entries, so we had to manually remove those.

Figure 2



The logical structure of the Artix Advanced for z/OS Transformer server [2] is shown in figure 2. This server is the component in Artix Advanced for z/OS which provided Web Service integration between SOAP based client programs in the CICS and IMS transaction systems on the mainframe. This figure shows that the client and server both communicate with the KDC to obtain tickets, and that the client then sends the ticket with the SOAP request, which is, in turn, authenticated by the server.

In this case the client application would use the Kerberos APIs to generate a ticket (mainly using krb5_sendauth(), or something similar, depending on the platform and implementation language), and then send it to the transformer server in the SOAP header. The Artix for z/OS transformer server will extract the ticket from the SOAP header and authenticate it using the Kerberos API's (mainly using krb5_recvauth()). This authentication will then take place against whatever KDC server has been configured for the mainframe. This could be a KDC running on the mainframe or it might be a KDC running on another host in the network. If the authentication passes, the request will be extracted from the SOAP body, passed to IMS/CICS to be processed, and the reply from IMS/CICS returned to the client. The IONA Artix mainframe transformer server also provides the option to use Kerberos tickets supplied by the IONA security service, if it proves necessary to use this instead of a KDC.

5 BENEFITS OF USING KERBEROS TO SECURE MAINFRAME WEB SERVICES

- A very strong industry standard form of authentication is used.
- Kerberos is available on many platforms and has been extensively tested, which makes it less likely that security would be compromised than would be the case in a home-grown system.

- The user only needs to sign in once to use many different services that have been Kerberized.
- This authentication information is exchanged between the client and server in a standard format, which ensures that clients and servers from various vendors can interoperate.
- Kerberos support is integrated into the mainframe security services (SAF), which makes it easy to set up, and also enables it seamlessly to integrate with the authorization checks done by the mainframe security system (RACF, etc.).
- By using the Kerberos support provided in various Web Services products like Artix and .Net, the developer can use Kerberos without having to know much about the details of how Kerberos support works. For example, on the mainframe in Artix, Kerberos support is provided with a couple of simple configuration options, and no coding is needed. This means that robust security can be provided with relatively little effort.

Kerberos is often seen as complex, but it is possible to hide most of the complexity from the users, so that advanced security facilities can be provided to critical systems — like mainframes — without creating too big a burden on the people who have to implement the system.

6 REFERENCES

1. IONA Technologies Internet homepage, http://www.iona.com
2. IONA, Artix 3.0 for z/OS: Administrator's Guide, 2005, http://www.iona.com/support/docs/artix/mainframe/3.0/admin_guide/index.html
3. OASIS, Web Services Security: Kerberos Token Profile 1.0, 2004, http://www.oasis-open.org
4. MIT, Kerberos: The Network Authentication Protocol, Kerberos Home Page, http://web.mit.edu/kerberos/www/
5. Brian Tung, Kerberos: A Network Authentication System, Addison Wesley Longman, Reading, 1999
6. IBM, SecureWay® Security Server: Network Authentication Service: Administration, 2001 http://www.ibm.com
7. IBM, SecureWay® Security Server: Network Authentication Service: Programming, 2001 http://www.ibm.com
8. Sun Microsystems, GSS-API Programming Guide, 2000 http://docs.sun.com

7 ACKNOWLEDGEMENTS

I would like to thank Ben North for the input provided for this paper.