Using Kerberos tickets for "true" Single Sign On

This document details the reasoning for, configuration of and experiences from the initial setup of Kerberos tickets for SSO (Single Sign On) using the existing Active Directory setup at Newcastle University. It then discusses the integration of the Kerberos login with Shibboleth 2 and aims to provide advice to system administrators wishing to implement Kerberos SSO with Shibboleth 2 in their institution.

Table of Contents

Introduction

Single Sign On (SSO) for web applications is useful both to users and application developers. Utilizing prior authentication on a system to allow "true" SSO enhances the user experience by allowing for a more personalised web browsing experience.

In this report, the use cases for Kerberos SSO will be discussed along with an outline of the Kerberos protocol. The technical requirements for authenticating via the Apache web server using Kerberos will be discussed as well as integration with the Shibboleth SSO system before a summary of the process, where the advantages of the "true SSO" implementation are reported on.

Purpose

SSO is becoming more and more prevalent in Universities, especially with the Athens switchover taking place across the UK. While logging into multiple web applications/services with a single ID is a time/labour saving feature, many users will just have logged into the operating system itself using the same ID/password and so this feels redundant and adds another step in the process of accessing information on the web.

Personalisation of homepages/CSS/accessibility settings can be automatically applied once the system has identified/authenticated the user. Kerberos provides a means by which, having a logged in to a (client) machine, a user's identity is known on the client machine and can be passed to a Kerberized authenticating server to access any web resource, allowing for a seamless personalised setup.

An example of the SSO use case at Newcastle University is the student portal; this will contain a range of student information tailored to a student's registration details (e.g. module/examination information). The portal page will be launched whenever a student logs into a campus PC; by not requiring a further web-login to access this personalised information it is hoped that the seamless availability of this information will increase readership and be useful to students around campus.

About Kerberos

Kerberos is an authentication protocol which uses key cryptography to provide authentication in client/server scenarios. Strong cryptography is used to prove the identities of both the client and server and, after the identities have been ascertained, all communications are encrypted to provide assurances about the integrity of the data. Kerberos is a platform independent standard so interoperability of diverse OSs (Linux/Windows/Mac) on a single network is possible.

Tickets are used to identify a user who has been authenticated, in order to remove the need for re-authentication. A Key Distribution Centre (KDC) issues a ticket-granting ticket (TGT) when the user first authenticates. After this, for the lifetime of the ticket, the TGT is all that needs to be presented by the client machine to a Kerberized authenticating server, in order to authenticate the user.

Skill Sets Required

Implementing the Kerberos auto sign-on demanded skills in:

- Apache web server configuration
- Apache Tomcat configuration and setup
- Microsoft Active Directory (AD) administration
- Linux system administration (and log monitoring)
- Networking
- For Shibboleth customization, Shibboleth 2.0+ and relevant skills are required

Assumptions

Technical guidance given in the next few sections relates to the following systems configuration (the guide is still applicable to other setups though readers will have to adjust for their setup).

- Clients use Linux (Redhat/CentOS5)
- Krb5 is installed via yum + RPM with default settings
- Apache is installed via RPM with default settings
- The user datastore is Microsoft ActiveDirectory
- The client and server exist on the same network

Prior Configuration

Client Setup

The firewall on the client machine should have the Kerberos ports (`88` and `750`) open. `krb5-devel`, `krb5-libs` and `krb5-workstation` should be installed via `yum` and the `krb5.conf` file in `/etc` should be modified to add a default realm your organisations setup. The example below is a cut down version of the Newcastle setup:

```
[logging]
 default = FILE:///var/log/krb5lib.log
[libdefaults]
 ticket_lifetime = 600
 default_keytab_name = /etc/krb5.keytab
 default_tkt_enctypes = rc4-hmac
 default_tgs_enctypes = rc4-hmac
 default_realm = CAMPUS.NCL.AC.UK
 forwardable = true
 proxiable = true
 [realms]
 CAMPUS.NCL.AC.UK = {
  kdc = kdc.campus.ncl.ac.uk:88
  default_domain = ncl.ac.uk
 }
[domain_realm]
 .ncl.ac.uk = CAMPUS.NCL.AC.UK
[appdefaults]
 kinit = {
  renewable = true
  fowardable = true
 }
```

Apache Setup

In Apache's `httpd.conf` the ServerName alias must be set using a fully qualified domain name (e.g. `website.ncl.ac.uk`) along with the UseCanonicalName directive being set to `On`. This will force the server to correctly identify itself to the KDC when authenticating the user's token.

Apache should be setup with `mod_auth_kerb` via `yum`. A configuration (`auth_kerb.conf`) file should be created automatically in the `conf.d` directory of your Apache install, which you should modify to look like:

```
LoadModule auth_kerb_module modules/mod_auth_kerb.so
<Location /protectedURL>
  SSLRequireSSL
  AuthType Kerberos
  AuthName "Test Kerberos Login"
  KrbMethodNegotiate On
  KrbMethodK5Passwd On
  KrbAuthRealms CAMPUS.NCL.AC.UK
  Krb5KeyTab /etc/krb5.conf
  KrbSaveCredentials On
  require valid-user
</Location>
```

As sensitive information is being passed over the network to allow login at this point, we recommend using `https` over SSL only for Kerberized (or any form of) login.

The location protected in the above configuration block should be a directory within the `DocumentRoot` of the Apache server with the following `index.php` file inside it:

```
<?php
  foreach($_SERVER as $key_name => $key_value)
  {
    print $key_name . " = " . $key_value . "<br>";
  }
?>
```

This will list all the headers in the `$_SERVER` array, which should eventually contain `REMOTE_USER` set by the Kerberos ticket in the form `<userID>@default_realm`.

SSO Setup

Keytab Generation

A Keytab is used to verify the identity of a client machine to the KDC. Keytabs may have “principals” (users) assigned to them in the form: HOST or HTTP, in order to restrict the use of the keytab to its initial purpose. For our use case we require the principal `HTTP/server`.

Keytabs are generated on the primary KDC and need to be allowed to populate around any secondary KDCs on the network before use. Below are the commands used to generate a keytab for our client (named “shib2”).

Add the user “shib2” to the AD:

```
C:\temp>net user shib2 password /add /dom
The command completed successfully.
```

Add the HTTP principal to the keytab:

```
C:\temp>ktpass -princ HTTP/shib2.ncl.ac.uk@CAMPUS.NCL.AC.UK -mapuser campus\shib2
        -pass password -out shib2.keytab /crypto rc4-hm ac-nt /kvno 3 -ptype KRB5_NT_SRV_HST
Targeting domain controller: kdc.campus.ncl.ac.uk Successfully mapped host/mothvm3.ncl.ac.uk
to shib2.
Password succesfully set!
```

After this, the keytab file is produced (using the name specified in the `–out` parameter above) and the user in the AD (created in the previous code snippet) will have the service principal HTTP mapped onto their account. The keytab should be transferred to the client machine and referenced in both the Apache `mod_auth_kerb` setup (`/etc/httpd/conf.d/auth_kerb.conf`) and the `/etc/krb5.conf` file.

Network Issues

Kerberos requires that the host name it is authenticating is an A record in the DNS. i.e. the server name must map directly to an IP address and not be an Alias (CNAME) onto a hostname. The Apache error message produced if the DNS setup is wrong in this way simply reads:

```
gss_acquire_cred() failed: Unspecified GSS failure.  Minor code may provide more information
(No principal in keytab matches desired name)
```

This error message does not provide much to work with (assuming the principal name in the keytab is correct); as such, the DNS setup for the client IP is critical to a working Kerberos SSO setup.

Testing Kerberos Login

Checking Keytabs

On the client machine, it is possible to check the principal names held within a Keytab using the command klist –k, on our server this provides the response:

```
[root@mothvm3 conf]# klist –k /etc/krb5.keytab

Keytab name: FILE:/etc/krb5.keytab

KVNO Principal

---- --------------------------------------------------------------------------

   3 HTTP/shib2.ncl.ac.uk@CAMPUS.NCL.AC.UK
```

This proves the Keytab is in a valid format for use for Kerberos login.

Checking Ticket Flags

From the command line on the client machine the `kinit` command launches a Kerberos session (given a keytab with the `–t` flag and a user ID to authenticate). You will then be challenged for the user's password. Once this has been entered successfully, control is passed back to the shell with no apparent change – in the background, a Kerberos ticket has been created in `/tmp`. Running `klist –f` (the `–f` option shows the ticket flags) will describe the ticket. Ticket flags can be deciphered using `man klist`; from the example below FPIA stands for F Forwardable, P Proxiable, I Initial, A preAuthenticated. `kdestroy` removes all existing Kerberos tickets from the machine (if this command is not run, the ticket will exist until the expiry date is reached). After the tickets have been expired, running klist again will show an empty list.

Running these commands at the command line looks like this:

```
[root@shib2 ~]# kinit -t /etc/krb5.keytab <userID>
Password for <userID>@CAMPUS.NCL.AC.UK: ******
[root@shib2 ~]# klist -f
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: <userID>@CAMPUS.NCL.AC.UK

Valid starting     Expires            Service principal
12/22/08 11:07:56  12/22/08 11:17:56  krbtgt/CAMPUS.NCL.AC.UK@CAMPUS.NCL.AC.UK
Flags: FPIA

Kerberos 4 ticket cache: /tmp/tkt0
klist: You have no tickets cached
[root@shib2 ~]# kdestroy
[root@shib2 ~]# klist
klist: No credentials cache found (ticket cache FILE:/tmp/krb5cc_0)


Kerberos 4 ticket cache: /tmp/tkt0
klist: You have no tickets cached
```

Browser Customisation

Internet Explorer will automatically negotiate Kerberos login with servers on the same network and so no customisation is generally required. Other browsers require small customisations in order to accept Kerberos login. However, if IEs security is set to "High" you may need to add the login servers to the configuration. Other browsers require this to be done in any case.

In Firefox, for example, browse to "`about:config`" and set `network.negotiate-auth.trusted-uris` to be the fully qualified name of your Kerberos client machine (or alternatively, the domain you wish to trust Kerberos login from – in our case `.ncl.ac.uk`). There are several guides online with instructions on modifying the setup of other browsers (Safari etc.) for use with Kerberos login.

Testing mod_auth_kerb

Given the prior Apache setup (installing `mod_auth_kerb` and setting Kerberos authentication configuration on a location as noted previously) and browser setup (allowing Kerberos login from the client server/domain – mentioned above), checking `mod_auth_kerb` should simply be a matter of restarting the Apache server, browsing to the protected location and scrolling down the index page, created earlier, to find the `REMOTE_USER` server header set.

It should be noted that a fully qualified domain name (FQDN) should be used, as negotiation will not take place with a partial domain name – even if the server later redirects this to use the FQDN.

If you are challenged for authentication or else simply no `REMOTE_USER` header is set, after ensuring that the browser/Apache server are configured correctly you should check your SSL error logs on the Apache server and refer to online documentation to find a suitable work around. As mentioned in the "Network Issues" section, one problem we came across at Newcastle was due to the CNAMEing of service addresses onto an IP with a different name in DNS. The error message here was misleading so it has been included in this document to remove this problem for other institutions.

Tracking Headers

There is an extension available for Firefox to allow the headers of HTTP traffic to be monitored on the client machine (the plugin is called “Live HTTP headers” and is available from https://addons.mozilla.org/en-US/firefox/addon/3829 at time of writing). Once installed, this extension can be used to monitor the authentication messages sent between the client and authenticating server. An example of successful authentication is shown below. The long authorisation header is a sign of successful Kerberos login. In failure modes there will be a much shorter authorisation header indicating that the SPNEGO negotiate protocol is sending an NTLM ticket rather than a Kerberos ticket.

```
GET /idp/Authn/RemoteUser HTTP/1.1
Authorization: Negotiate
YIIFTAYGKwYBBQUCoIIFQDCCBTygJDAiBgkqhkiC9xIBAgIGCSqGSIb3EgECAgYKKwYBBAGCNwICCqKCBRIEggUOYIIFCg
YJKoZIhvcSAQICAQBuggT5MIIE9aADAgEFoQMCAQ6iBwMFACAAAACjggQdYYIEGTCCBBWgAwIBBaESGxBDQU1QVVMuTkNM
LkFDLlVLoiIwIKADAgECoRkwFxsESFRUUBsPc2hpYjIubmNsLmFjLnVro4ID1DCCA9CgAwIBF6EDAgEDooIDwgSCA75Zur
v2COIV6K4/mzNTZABDTHSmejpaXgGTaUmnYCthcysHrt/UIVG1h3UG0IxSPXDMpTiXNcvhCOd1034p0FX7Iy07p2+du+1w
GlCKK8yWqFEZPHjCfZjN/kBJHLZOB/rt4/NZw7fcsANij3TBYXpRFoztdyJhh22immoIglrvLLY9KD5HxG9qABNfkyNRtv
o21jHSWhwnw+JQxHML8zAIWBa/Z3AbiHzKnSW2R27g/z7oenbqoI/4GjFpbNweM95JkV4Rlq/1ZZ1pVtDwRtQMPm1PkTzm
eMmTt1SsZCbKQmuUdjssO37732fJkbuBCvcXNQYoNHDx8QZPRHANKxm2q6WE+GDfIlnTUB4Hn3BE6b5i1Yzi6m8qmY/AyB
rZlLATqrOJfxTejMclpGutVkgUO7rPqWRLghrEL5G1FKMJAbYJWvRfRRQvmvW8oIupLWA3YZp7x4RTdoQLXJmXc+3W8XZw
Jk6GQpyHlkVZeHt3QfglIRMGguY/1JkPZeiulQKi8xq+fKgLzAhwBGWI57IVR5VG4RJHpRKq+8XvKnF+cKPqTNodVw0CTb
wZIsSn2xJllUMNeh88fOULVVEdIYyaFpDrxik/yF+VRcLynxIxj/td+81J36MgbGbgkkMdhbAxlWSS+mDSZ51ccZOIzmHH
D4dICcM1fWzs4BA/OtN44aI5WoTLVQZLJggjqxjw9WH+eQ0ADHs+IyrudSoYxJtLEUuMuX35dEdDxdIhVAJ70m4UcLuljt
XUI4o2gTdsU6T8SO7AixAXtGnpGmevHZZMuWyqzLfqhtlWKHBUEkLhl2ZvU0//GLAGvlNavPDFNSTtoUpocqhHAmj19MZ5
KfX/F6Fsjus0nEX8ORRKxRK7DTGbzNiOXVkBF2TadYZZLuOb0Co7djGvpfrzA0BKZZsictTf+w84DDeEiOlQW5QglQLLvx
1LDKIu2TKh+2wTGTyxkv0HUDclb3JRJWQ2ta2ZIvsiLlXA302tpgclpcRJiqll/35rwEazRhapwq4L5Lhxex3vgnJRfdql
bVZQJWfv4KtGp/q8ZpkWju30KNsqIrLgwJP8QVsyXLK2SLD9RVadDRCQHk5ht9Vs1r8USrTvOyIx4cDHAoDFU2M6GYr0WN
00itag+5WolFGavaWQ2Twd+S/X9C35aVm/c/cEipTTfZbdRIC+8FeW1WevbdyQWLOsjdWEoSth2LcUydz3bv/6cbZL2j3P
MBXNQxQzxSE69rdWaFD35/DDhZggCN0S1t6GcI+R/N/gRcxY8SOvpIG+MIG7oAMFAReigbMEgbD1djgrlybprwJpJqUbYp
QysYNiRoliQTih1Abe3o8GeGCl8SQ3RnGQLo7ufiGi6ATXpK6yUxYRZHYENuip3uW/0Rt9+SS6ZBI3Lt03lL5lsoAOq/je
oemeH58+AfcNwgYAshQHt7dFlEdBk6CrgiayAxcYYRtK+5pcWKm8MGW9GwWo9KIn15TH3JGKq8jI34nGYFGYZiIXbNP1-
wc5TZFOxe6MKpfXCnDzfiwEqIrQfQ==
```

Integration with Shibboleth SSO technology

Shibboleth2 has different login handlers for dealing with a) username/password auth or b) `REMOTE_USER` authentication. Combining this with Apache’s location protection (above) and making a few tweaks to Tomcat’s server.xml (Shibboleth is a Java application that performs well inside the Tomcat container) provides an excellent starting point for true SSO in an institution.

Configuration

`mod_proxy_ajp` should be used to map the URL you have specified for your IDP installation (/idp by default) over to Tomcat - `ProxyPass /idp ajp://localhost:8009/idp`

Apache should then be configured to protect the location `/idp/Authn/RemoteUser` using Kerberos login (using a location block as shown above). The Shibboleth `handler.xml` file (in `<idpHome>/conf`) should be set to use the “`RemoteUser`” login handler (as `REMOTE_USER` is the server header set by Kerberos login). Tomcat’s `server.xml` should contain the following extract:

```
<!-- Define a Connector on port 8443 for SSL comms -->
<Connector port="8443"
           maxHttpHeaderSize="8192"
           maxSpareThreads="75"
           scheme="https"
           secure="true"
           SSLEnabled="true"
           sslProtocol="TLS"
           clientAuth="want"
           keystoreFile=YOUR JKS FILE
           keystorePass=PASSWORD
           truststoreFile=YOUR JKS FILE
           truststorePass=PASSWORD
           truststoreAlgorithm="DelegateToApplication"
           protocol="org.apache.coyote.http11.Http11Protocol" />

    <!-- Define an AJP 1.3 Connector on port 8009 -->
    <!-- Connector port="8009" protocol="AJP/1.3" redirectPort="8443" / -->
    <Connector address="127.0.0.1"
           port="8009"
           redirectPort="8443"
           protocol="AJP/1.3"
           enableLookups="false"
           tomcatAuthentication="false" />
```

Setting "`tomcatAuthentication`" in the `8009` connector and setting `clientAuth="want"` in the `8443` connector will force Tomcat to look for the `REMOTE_USER` set from somewhere else (in this case, our Kerberos module provides it).

Testing

Initial testing can be carried out using the TestShib website at http://www.testshib.org/testshib-two/index.jsp

Having registered your Shib2 IDP with this service, browse to https://sp.testshib.org/ and enter your IDP's domain name. If everything is correct, after a short delay, you should be redirected to the TestShib results page with, amongst other things, the EPPN header listed on the page (note that as TestShib cannot be configured to accept custom defined attributes, only the standard set of attributes (`eduPersonPrincipalName`, `eduPersonAffiliation` etc.) can be picked up by the TestShib SP).

Once proof of concept has been made, the IDP should be registered with a federation and further testing/productionalisation should be carried out using registered SPs within your institution.

Summary

Setting up Kerberos SSO requires a diverse skill set in systems administration as well as knowledge of the Shibboleth IDP. Having said this, the actual deployment is straight forward if approached in a linear order: register the IDP in the AD, generate a Keytab for the IDP, setup Apache/Tomcat/Shibboleth on the IDP and test with various SPs inside and outside of your institution.

The benefits of true SSO, such as portal login/student homepages etc. are obvious. The uptake of Shibboleth SSO systems in Universities, along with the features provided by Shibboleth 2.x for handling login, make Kerberos SSO achievable in institutions across the UK (given the appropriate supporting infrastructure).

Further Work

We intend to look into a failover system whereby, when a Kerberos ticket cannot be found on the client machine (when a student is accessing a resource remotely for example) the login server will redirect to the standard institutional login page, rather than a “grey-box” Kerberos login prompt. This style of login is both disconcerting to the user and stores the password entered in the `$_SERVER` header variable making it unviable for production use. Central Authentication Service (CAS) is a login handler that allows for drop-through authentication such as this and, as such, we intend to investigate linking this with Kerberos + Shibboleth SSO and will report on this in the coming months.