Kerberos and Single Sign-On with HTTP

Joe Orton

Red Hat
Overview

- Introduction
- The Problem
- Current Solutions
- Future Solutions
- Conclusion
Introduction

- WebDAV: common complaint of poor support for authentication in HTTP
- Basic: not good enough
- Digest: not widely available
  - Cannot integrate with other authentication systems
- Kerberos:
  - Large deployments for Unix shops
  - Active Directory
The Problem

• How to integrate HTTP servers into a Kerberos infrastructure?
• Single Sign-On: reducing the number of times people enter passwords
• Ideal: user authentication happens exactly once per “session”
Problem Scope

- Covering intranet-, enterprise- organisation-wide HTTP authentication
- Out of scope: SSO for “The Web”
- In scope? Authentication to HTTP proxy servers
  - Useful for organisations where Web access must pass through an HTTP proxy
  - Strong authentication needed for policy enforcement
Authentication Sessions

- “Session” defined from initial user authentication
- Sessions should be universal to achieve the goal of “Single Sign-On”
- User should never have to authenticate:
  - to any individual server
  - to use any particular service (protocol)
- How to terminate a session?
One-Slide-Guide to Kerberos

- Shared secret keys, a trusted third-party (KDC), and symmetric key encryption
  – KDC = Key Distribution Centre; trusted by all

- KDC authenticates user, gives out “TGT”

- Using TGT, client obtains “ticket” from KDC encrypted with service's secret key

- Client can prove user identity to a service

- Mutual authentication: service authenticated to client
What makes HTTP different?

• Traditional Internet protocols (e.g. SMTP, IMAP, ...) all support Kerberos authentication forever

• Why is HTTP different?
Strong authentication is not much use without message integrity, and probably also confidentiality.

Integrity/confidentiality = transport layer.

HTTP authentication is independent of the transport layer; unlike SMTP, IMAP, ...

Many approaches to improving HTTP authentication don't understand this.
Current Solutions

- Stanford WebAuth: forms and cookies
  - Similar solution: Pubcookie
- Using HTTP “Basic” authentication with Kerberos
- HTTP “Negotiate” authentication
Stanford WebAuth

- Based on forms and cookies
- Token-passing via browser redirects between web server and “WebKDC”
- Kerberos credentials passed to WebKDC via HTML form
- WebKDC authenticates as user to KDC
WebAuth protocol

User Agent

GET /private

Web Server

302 Redirect
Location: http://webkdc.example.com/...
WebAuth protocol 2

User Agent

GET /authenticate-me

WebKDC

200 OK

<html>...
<form>...</form>...
WebAuth protocol 3

User Agent

POST /authenticate-me

WebKDC

302 Redirect
Location:
http://origin.example.com/...
Set-Cookie: blah

KDC
WebAuth analysis

- “Application level” solution
- Cookies + HTML ≠ HTTP authentication
- Requires a complete web browser
  - Doesn't work with automated agents, WebDAV
- Credentials over the wire at HTTP level
  - Kerberos designed to avoid doing this
  - No mutual authentication
  - Requires SSL to be secure
WebAuth analysis 2

- Training users to enter Kerberos credentials into web forms is Very Bad™ - phishing
- Session scope: within one web browser but then covers all servers
- Cannot authenticate to HTTP proxies
- Session termination? Flush cookies
Kerberos via Basic Auth

- Use standard HTTP Basic authentication
- Client sends Kerberos credentials as normal Basic auth credentials
- Web server authenticates as user directly to KDC
  - Custom server code needed
  - e.g. mod_auth_kerb
Kerberos via Basic on the wire

GET /secret/ HTTP/1.1
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Basic realm="Blah"

GET /secret/ HTTP/1.1
Authorization: Basic QWxuIHNlc2FZQ==
HTTP/1.1 200 OK
Kerberos via Basic, analysis

- Simple to set up
- Works with any HTTP client
  - Including automated clients, WebDAV
- Again, sending credentials over the wire defeats the point of using Kerberos
  - Requires SSL to secure credentials
  - No mutual authentication
- Can authenticate to proxies, but insecurely – cleartext only to proxy
Kerberos via Basic, analysis 2

- Session scope: one web browser, one server
- Training users to enter credentials into HTTP authentication dialogs is also Very Bad™ (maybe only Quite Bad™, but still not Good™)
- Session termination: flush cached credentials within browser
The “Negotiate” Scheme

- New HTTP authentication scheme (kind of)
- Written by Microsoft; I-D published 2001
- Became “Informational” RFC 4559 in 2006
- Uses GSSAPI token exchange, wraps Kerberos protocol over the wire
- Custom server, client extension
Negotiate: Protocol trace

1. GET /secret/ HTTP/1.1
2. HTTP/1.1 401 Unauthorized
   WWW-Authenticate: Negotiate [token]
3. GET /secret/ HTTP/1.1
   Authorization: Negotiate Y......Q==
   [goto 2, or...]
   HTTP/1.1 200 OK
Implementing Negotiate

- Supported at HTTP protocol level; works with WebDAV etc.
- Implemented by Firefox, MSIE
  - also Curl, elinks, neon (hence, e.g. Subversion)
- No Kerberos credentials on the wire!
  - requires SSL to secure the connection
- Works with proxies
  - but not securely
Negotiate analysis – the bad

• Even the name is bad
• Per-connection authentication!
  – ...assumes all subsequent requests on given TCP connection are authenticated
• Arbitrarily breaks RFC2617 grammar
  – “WWW-Authenticate: Negotiate b64blob”,
  – Should be ... “Negotiate token=b64blob”
• Abuses RFC2617 headers
  – “WWW-Authenticate” in a 2xx response
Negotiate analysis – the good

- Real Single Sign-On!
- "Kerberized" HTTP
  - No credentials over the wire
  - Mutual authentication
- Session scoped to all servers, all services
- Session termination dictated by system-wide Kerberos login session
mod_auth_kerb

- Module for Apache httpd 1.3/2.x
- Maintained by Daniel Kouril, BSDy license
- Version 5.0 released August 2006, first non-beta release
  - Latest is v5.3, November 2006
- Supports both Negotiate and Kerberos-over-Basic authentication
mod_auth_kerb Configuration

- Obtain a service key from the KDC
- Name, for example: HTTP/www.example.com@EXAMPLE.COM
- Service key in keytab file – check permissions!
- Load module and add access control configuration, either httpd.conf or .htaccess
Access control Configuration

<Location /private>
  AuthType Kerberos
  AuthName "Kerberos Login"
  KrbMethodNegotiate On
  KrbMethodK5Passwd Off
  KrbAuthRealms EXAMPLE.COM
  Krb5KeyTab /etc/httpd/conf/keytab
  require valid-user
  SSLRequireSSL
</Location>
Client configuration

- Firefox:

  ![Image of Firefox about:config settings]

- MSIE should work automatically within the “Intranet zone”
Conclusion

- Strong authentication as an HTTP authentication scheme alone is not enough
- “Negotiate” is a practical if flawed solution for Kerberos Single Sign-On with HTTP
- But **must** be used over SSL
Future Solutions

- Redesign Negotiate, without the warts?
- RFC2712: TLS with Kerberos ciphersuites
  - Implemented in OpenSSL; no deployment
  - Not GSSAPI-based = Bad
- draft-santesson-tls-gssapi: TLS with GSSAPI authentication exchange
  - GSSAPI = Good, but breaks TLS state machine?
- A “GSSAPI Transport Layer” for HTTP?
Resources

- http://webauth.stanford.edu/
- http://www.pubcookie.org/
- http://modauthkerb.sourceforge.net/
- These slides:
  http://people.apache.org/~jorton/ac08eu/
• Any questions?