Perforce Single Sign-On via Microsoft Active Directory

This document describes a method for implementing single sign-on (SSO) authentication for a Perforce server, using Microsoft Active Directory as the authentication provider.

Background

Perforce has offered an undocumented SSO mechanism for several years. As of the 2011.1 release, Single Sign-on is fully supported.

Essentially, Perforce's SSO mechanism relies on a client-side program to obtain a credential, and a server-side trigger to verify that credential. When the SSO mechanism is in place, a client must run the p4 login command, but will not be prompted for a password.

Architecture

The basic architecture of this solution is shown in Figure 1.

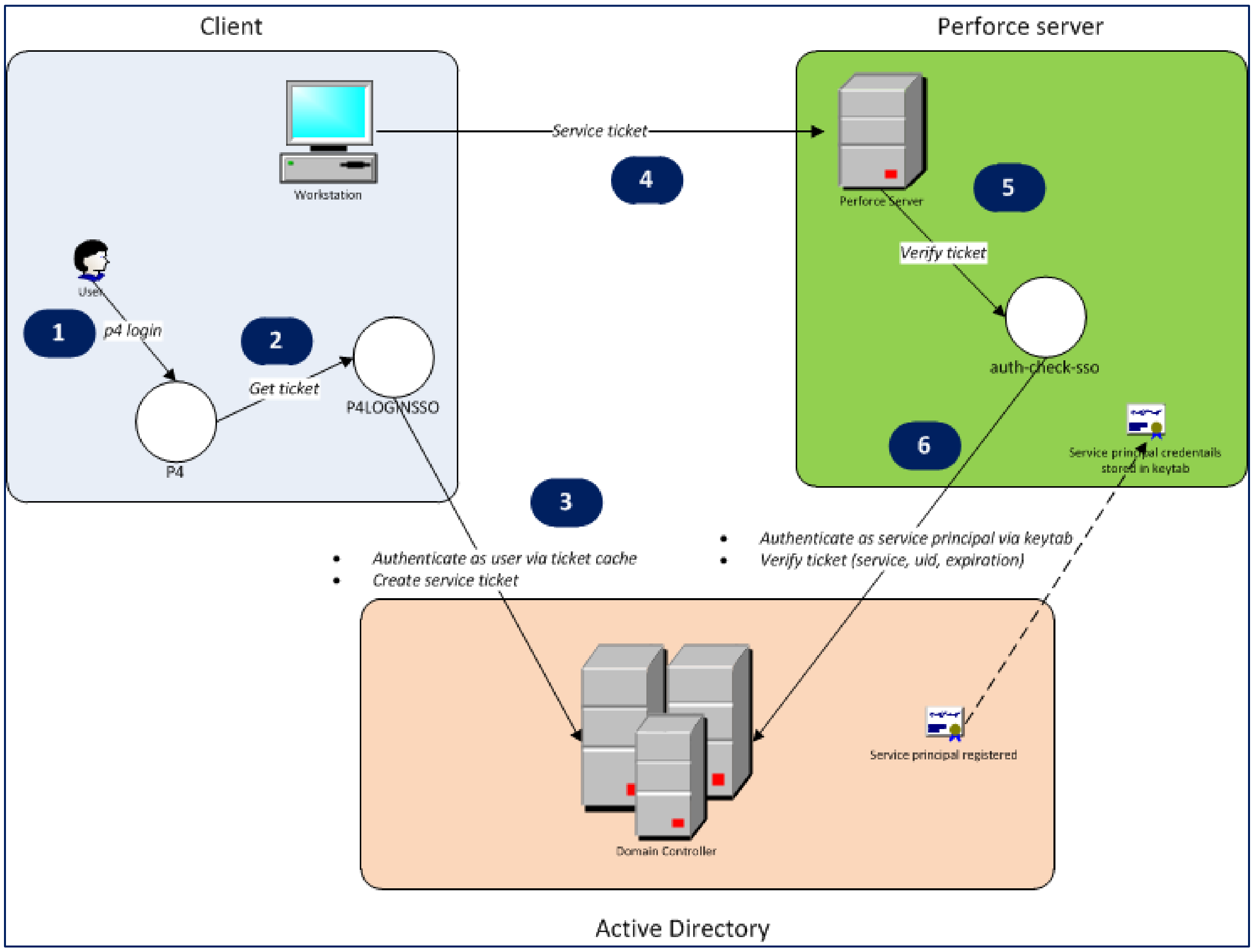

Client
Perforce server
Workstation
Service ticket
4
Perforce Server
5
Verify ticket
User
1
p4 login
2
Get ticket
P4LOGINSSO
P4
auth-check-sso
6
Service principal credentails stored in keytab
3
Authenticate as user via ticket cache
Create service ticket
Authenticate as service principal via keytab
Verify ticket (service, uid, expiration)
Service principal registered
Domain Controller
Active Directory

Figure 1: SSO Implementation architecture

The solution relies on the Kerberos protocol that is part of Active Directory. When this solution is used, a Perforce login request goes through several steps, as shown in Figure 1.

1. The user runs the *p4 login* command, either directly through the *p4* command line client, or through another client program.
2. The client program recognizes the presence of the *P4LOGINSSO* environment variable, and calls the program referenced by *P4LOGINSSO* to obtain a Kerberos ticket-granting ticket.
3. The *P4LOGINSSO* program authenticates to Active Directory as the current user by accessing the Kerberos ticket cache. It then creates a ticket-granting ticket for the Kerberos service used by Perforce.

4. The ticket is returned to the client program and passed to the Perforce server.
5. The Perforce server executes the *auth-check-sso* trigger to verify the ticket.
6. The *auth-check-sso* trigger authenticates itself as the Perforce service principal to Active Directory. It then verifies that the user's ticket is valid.

As the diagram indicates, the Perforce service principal's credentials must be stored on the Perforce server machine in the *keytab* file.

Implementation

This solution uses Java programs to retrieve and verify a Kerberos service ticket for the Perforce service, using the techniques described at http://www.javaactivedirectory.com. The source code is available, along with an Ant script for building and packaging.

Deploying the solution

The details of deploying this solution will almost certainly vary depending on your environment. Therefore, the instructions below are for reference only.

These instructions reference files described in Appendix 2: File manifest.

Test environment details

These instructions worked in a testing environment consisting of:

- A 32-bit Windows Server 2003 machine serving as the Active Directory domain controller (see Appendix 1: Domain Controller setup notes).
- A 2010.2 Perforce server running on the Windows Server 2003 machine.
- One client workstation running 32-bit Windows XP and the Perforce 2010.2 command line client.
- One client workstation running 64-bit Ubuntu 11 and the Perforce 2010.2 command line client. This workstation used *winbind* to join the Active Directory domain, as described in https://help.ubuntu.com/community/ActiveDirectoryWinbindHowto.

The following client programs were tested:

- The Perforce command line client, *p4*, version 2010.2
- P4V, version 2010.2
- P4Eclipse, version 2010.1
- P4SCC, version 2010.2

Of these four clients, *p4,* P4V, and P4Eclipse let a user authenticate without any password prompt. P4SCC gave a password prompt but authenticated successfully if the password field was left empty.

Client

- If using Windows, make sure that Windows will allow use of ticket-granting tickets. (See http://www.javaactivedirectory.com/?page_id=93.)

- On each client workstation, put the files *login.conf*, *sso-client.jar*, and *sso-client.[bat|sh]* into a directory accessible to the user.
- Modify the *login.conf* file, keeping only the *SignedOnUserLoginContext* section.
- Customize the driver script (*sso-client.bat* for Windows, *sso-client.sh* for non-Windows). The general format is:
  "<path to JDK 6 java>" -jar <path to sso-client.jar> <path to SSO log file> <overwrite SSO log file during each run? true|false> <domain/realm> <domain controller> <path to login.conf> "<fully qualified service principal>"
- Set the *P4LOGINSSO* environment variable to point to *sso-client.bat*.

Server

- If using Windows, make sure that Windows will allow use of ticket-granting tickets. (See http://www.javaactivedirectory.com/?page_id=93.)
- On the Perforce server machine, put the files *login.conf*, *sso-server.jar*, and *sso-server.[bat|sh]* into a directory accessible to the account running the Perforce server.
- Customize the *ServicePrincipalLoginContext* section of *login.conf*. The general format is:
  com.sun.security.auth.module.Krb5LoginModule required useKeyTab=true doNotPrompt=true principal="<fully qualified service principal>" useKeyTab=true doNotPrompt=true storeKey=true keyTab="<path to keytab>";
  See Appendix 1: Domain Controller setup notes for a description of how to generate the *keytab* file.
- Customize the *sso-server.bat* driver script. The general format is:
  "<path to JDK 6 java>" -jar <path to sso-client.jar> <path to SSO log file> <overwrite SSO log file during each run? true|false> <Perforce user ID passed as trigger argument> <domain/realm> <domain controller> <path to login.conf> "<fully qualified service principal>"
- Install the trigger:
  - Run p4 triggers.
  - Add an entry similar to: *sso auth-check-sso auth "<path to sso-server.bat> %user%"*
  - Restart the Perforce server.

Usage

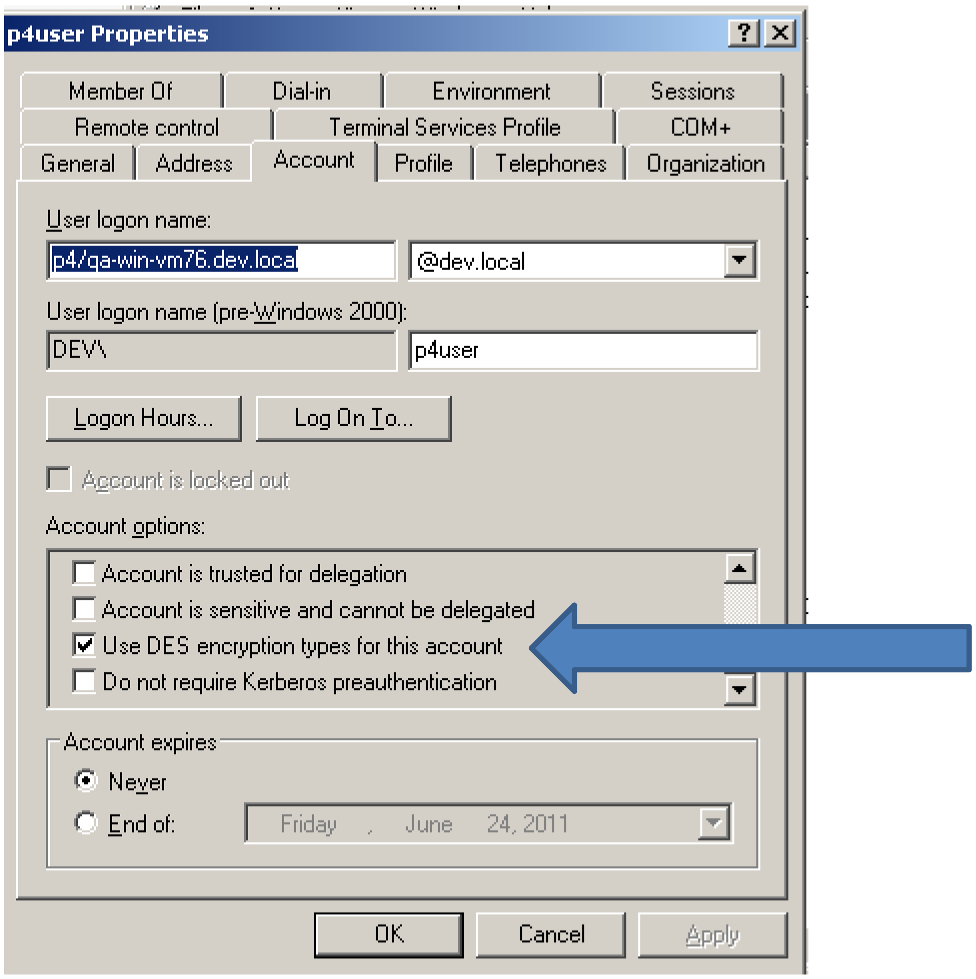
Once the SSO solution is in place, a Perforce user will not have to supply a password when logging in, although command line users will need to run *p4 login*.

Appendix 1: Domain Controller setup notes

This section is not a comprehensive description of how to configure an Active Directory domain controller. Rather, it describes particular steps that may be of interest.

As a general approach, the steps described at at http://d3planet.com/rtfb/2009/08/26/building-an-active-directory-domain-controller-for-development-using-vmware-workstation-pt-2/ were used to configure the domain controller for a test environment.

- The domain controller required a recent version of Windows Support Tools. For Windows Server 2003, the Support Tools are available from http://www.microsoft.com/downloads/details.aspx?FamilyID=96a35011-fd83-419d-939b-9a772ea2df90.
- The domain controller required a patch for a Kerberos encryption bug (Microsoft KB#833708).
- When configuring the service principal account, the *Use DES encryption* option must be selected immediately after creating the account. Otherwise, the password must be reset after this option is selected.


p4user Properties

User logon name: p4/qa-win-vm76.dev.local @dev.local

User logon name (pre-Windows 2000): DEV\ p4user

Account is locked out [ ]

Account options:
- [ ] Account is trusted for delegation
- [ ] Account is sensitive and cannot be delegated
- [✓] Use DES encryption types for this account
- [ ] Do not require Kerberos preauthentication

Account expires: [✓ Never] End of: Friday , June 24, 2011

Figure 2: DES encryption option

- When registering the service principal account on the Perforce server machine, the *keytab* file was created using *ktpass*. For example, in a testing environment, using *p4user* as the service principal account for the *p4* service, in a domain called *DEV.LOCAL*:

*ktpass -princ p4/domain-controller.dev.local@DEV.LOCAL -ptype KRB5_NT_PRINCIPAL -crypto DES-CBC-CRC -mapuser "p4user" -pass * -out c:\keytab -kvno 0*

The *keytab* file is then used on the Perforce server, and is referenced in *login.conf*.

- The *p4user* account was linked to the *p4* service using *setspn*:
  setspn -a p4/domain-controller.dev.local@DEV.LOCAL p4user

Appendix 2: File manifest

Apache-license.txt	Apache Software Foundation license
Java\KerbClient	Java source code and JAR file for client side program
Java\KerbServer	Java source code and JAR file for server side trigger
Java\login.conf	*Login.conf* file used on both client and server
Java\sso-client.bat	Batch file used to run the client Java program as the *P4LOGINSSO* program on Windows
Java\sso-client.sh	Shell script used to run the client Java program as the *P4LOGINSSO* program on non-Windows platforms
Java\sso-server.bat	Batch file used as *the auth-check-sso* trigger
Java\build.xml	Ant script used for building and packaging the Java code. Depends on log4j and commons-codec Apache libraries.

Appendix 3: FAQ

Question	Answer
Why write this solution in Java?	Java has well-structured and documented Kerberos libraries. The same Java programs can be used on multiple platforms.
How much can Perforce support me if I have problems implementing this solution in my environment?	Perforce technical support can provide assistance with Perforce's SSO mechanism. Unofficial assistance may be available for the Java programs. Detailed assistance with Active Directory problems may be outside the scope of Perforce's ability to help.
What happens if a client doesn't set the *P4LOGINSSO* variable?	They will fall back to regular authentication. You may wish to provide an *auth-check* trigger so that they can still provide their Active Directory password.
What happens if Active Directory is down?	A client can unset *P4LOGINSSO* and fall back to regular password authentication. If you need to remove the *auth-check-sso* trigger on the server and are locked out of Perforce, consult http://kb.perforce.com/article/838/regaining-super-user-access-to-your-perforce-database and http://kb.perforce.com/article/834/p4d-xf-flag.