Table of Contents

Contents

Introduction

This document describes the realization of Canonical Use Case 4, Interactive Login, using the XSEDE XUAS architectural components. See http://hdl.handle.net/2142/46550 for the use cases. It is assumed that the reader has already read and is familiar with the XSEDE Architecture Level 3 Decomposition (L3D), in particular §3 (Access Layer). The authors suggest that this document be open or on hand when reading this document.

Structure of this Document

This document comprises two sections. §2 reviews the interactive login use case and §3 describes how the XSEDE components are used to implement the use case from §2.

Canonical Use Case 4

Canonical use case 4 is “Interactive login.” The description is “An XSEDE user establishes an interactive login session on an SP-operated login service.”
The use case starts with a number of assumptions, specifically:

1. The User is aware of the existence of and name of a specific XSEDE SP-operated login

service.

2. The User is authorized to use the SP-operated login service.
3. The User has access to the Internet.
4. The User has authenticated to XSEDE per UCCAN 6.
5. The User has prepared his/her local workstation to support remote graphical applications (e.g., running an X server).

The use case is simple: the user needs to gain access to a login shell on a specific SP-operated login service, and needs to do so using his/her XSEDE identity (as opposed to a separate local username and password on the login server). As explained by the quality attributes, the user must be able to accomplish this task using either a standard web browser or a standard SSHv2 client. In addition to obtaining access to the login shell, the user may also need to be able to run programs on the login host that are, in turn, able to open graphical interfaces on the user's local display.

Table 1 lists the quality attributes for the use case. There are no variations.

Table 1: Interactive login use case quality attributes

UCCAN 4.0	Interactive login
QAS-CAN4.a	The implementation should never require that an unencrypted password be transmitted over a network.
QAS-CAN4.b	The user must be able to complete this use case using a standard web browser.
QAS-CAN4.c	The user must be able to complete this use case using a standard SSHv2 client.
QAS-CAN4.d	If the login shell session is lost for some reason, the user must be able to reattach to the session after a reconnection.

Architectural Response

Assume that

1. The User is aware of the existence of and name of a specific XSEDE SP-operated login service.
2. The User is authorized to use the SP-operated login service.
3. The User has access to the Internet.
4. The User has authenticated to XSEDE per UCCAN 6.

5. The User has prepared his/her local workstation to support remote graphical applications (e.g., running an X server).
6. The SP has installed and configured the GSI-enabled SSH service (GSI-OpenSSH's sshd or an interoperable equivalent) as described in [L3D §8.1.3].

Access via XSEDE User Portal (XUP)

Access to XSEDE login services via web browser is supported via the XUP, making use of XSEDE's common identification and authentication services and the GSI-enabled SSH services on each login server. When a user logs into the XUP, the XUP uses XSEDE's Kerberos services [L3D §7.1.2.1] to authenticate the user. The XUP then uses XSEDE's MyProxy service [L3D §7.1.2.2] to obtain a short-lived X.509 proxy credential for the user, which it stores on the XUP servers. The XUP interface allows the user to list XSEDE HPC and HTC services that he/she is authorized to use. Clicking a link next to one of these services downloads a Java applet GSI-SSH client to the user's local browser environment. (NOTE: The browser must support Java to use this applet.) The Java GSI-SSH client runs in the user's browser and obtains a delegated proxy credential [L3D §7.1.2] for the user from the XUP web server. It then uses that credential to authenticate to the selected login service via SSHv2 with X.509 authentication [L3D §7.3.5] and presents the user with an interactive shell on the login server.

Access via SSHv2 Client

Access to XSEDE login services via SSHv2 client is supported via the XSEDE login service [L3D §3.2.6]. If the user is using a graphical SSH interface, he/she connects to login.xsede.org and authenticates using his or her XSEDE userid and password. The login.xsede.org SSH service uses XSEDE's MyProxy service [L3D §7.1.2.2] to authenticate the user and obtain a short-lived X.509 proxy credential, which it stores locally. It then presents the user with a restricted login shell that allows the user to ssh to any of XSEDE's SP-operated login services, at which point the ssh client and service perform X.509 authentication using the proxy credential stored previously. The user does not need to provide his/her userid and password again.

If the user is using a command line SSH interface, this procedure can be further simplified using the following syntax.

ssh -l*userid* login.xsede.org ssh *login-server-hostname*

This approach performs the same protocol and authentication transactions described above, but accomplishes it for the user in a single step.

Quality of Service Attributes

CAN4.a - No unencrypted passwords over the network

The architectural response above offers two ways to establish a shell: (1) via a Web browser and the XUP, and (2) via an SSHv2 client and the XSEDE login service.

When using a Web browser, the user authenticates to the XUP using a secure HTTP connection (https) and his or her password is encrypted via the TLS session key. The authentication mechanisms between the XUP and the XSEDE Kerberos service and MyProxy service (covered in UCCAN 6) are fully encrypted and do not involve passing a user's password (or private key) over the network. (See the architectural response to UCCAN 6 for details. References for the rest of the authentication protocols mentioned below are provided above in the architectural response.) The Java applet GSI-SSH client provided by the XUP is downloaded to the user's local system and runs there. The X.509 proxy delegation transaction between the GSI-SSH client and the XUP web server (to obtain a short-term proxy certificate) is fully encrypted and does not involve passing a user's password (or private key) over the network. Finally, the X.509 authentication mechanism between the GSI-SSH client the SP-operated login services is fully encrypted and does not involve passing a user's password (or private key) over the network.

When using an SSHv2 client, the user authenticates to the XSEDE login service using a secure SSH connection and his or her password is encrypted via the TLS session key. Subsequent authentication transactions (a) between the XSEDE login service and the XSEDE MyProxy service and (b) between the XSEDE login service and the SP-operated login services are fully encrypted and do not involve passing a user's password (or private key) over the network. References to these protocol transactions are provided above in the architectural response.

CAN4.b - Standard web browser

The architectural response details how to accomplish this use case via a standard web browser.

CAN4.c - SSHv2 client

The architectural response details how to accomplish this use case via an SSHv2 client.

CAN4.d - Reattach session

Our response does not provide an architectural feature to accomplish this. We believe that a feature built into the system architecture would be prohibitively expensive and is not necessary to obtain the intended user benefit.

We note that most interactive Unix systems, such as those offered by XSEDE SPs, include a tool

called “screen” that offers this functionality. To enable reattaching the session, the user first uses the screen command to establish a persistent terminal session. If the session is disrupted for some reason, the user can re-connect to the login service as described above and then use the screen command again to connect to the previously established terminal session.

Put another way, we propose that session reattachment should be a feature provided by the interactive login services themselves, as opposed to the system that connects the user to the login services.