Relying Party / Service

A Relying Party (RP) is the home of the resource that the user is attempting to connect to; most commonly, some server software or a gateway to computing resources (e.g., an OpenSSH or Microsoft Exchange server).

Contents

1. Overview

The RP is essentially the service that the client attempts to connect to. Technically, the RP consists of two separate pieces:

- The Service itself
- The Relying Party Proxy (RP Proxy).

⚠ Note that some people may refer to the RP as being just the service, such that the RP connects to the RP Proxy.

The service is the resource that the user is attempting to connect to; most commonly, some server software or a gateway to computing resources (e.g., an OpenSSH or Microsoft Exchange server).

The RP Proxy is a RADIUS server that connects services to Identity Providers (IdPs) via a trust infrastructure of some kind (typically either a classic hierarchical RADIUS network or a Moonshot-based Trust Router network).

The service, upon receiving a session request from the client, will start the authentication process by speaking to its local RP Proxy. The initial request will typically include a pointer to the home IdP of the user. When using the Trust Router network, the Trust Router enables the RP Proxy to “find” the relevant IdP; the RP Proxy then establishes a secure, direct connection to it.

2. Requirements

2.1. On the Service

2.1.1. Moonshot mechanism

The service must have the Moonshot mechanism installed and configured within the operating system. This will either be:

- The GSS-EAP mechanism configured in the GSS stack.
- *or*, on Windows only, the Moonshot SSP (EAP-SSP) mechanism configured as a SSPI provider.

These mechanisms enable server software (such as an SSH server) to make use of Moonshot as a potential GSS-API/SSPI mechanism for authentication.

2.1.2. Service configured to use GSS

The service needs to be configured to use the GSS-API/SSPI for authentication. How this is done is highly application specific - see the wiki section on configuring servers for instructions for particular server software.

2.1.3. GSS-EAP/EAP-SSP configured to talk to a local RP Proxy

The GSS-EAP / EAP-SSP mechanism needs to be configured to talk to a local RP Proxy so that the service can interface with the Moonshot trust infrastructure and relevant IdP. This will involve configuring the hostname/IP address of the RP Proxy along with keying information of some sort in order that the two can communicate securely.

How-to's for configuring these are available on the wiki section on installing the Moonshot libraries on Linux or the wiki section on configuring the Moonshot SSP, as appropriate.

2.2. On the RP Proxy

2.2.1. RADIUS server with Moonshot capability

The RP Proxy itself is a standard RADIUS server that has been enhanced to include Moonshot functionality. Currently, only FreeRADIUS supports Moonshot.

2.2.2. Configured to talk to the Service

The RADIUS server will need to be configured to talk to the service (i.e., the EAP client). This will typically involve configuring its hostname/IP address along with keying information of some sort.

How-to's for configuring RADIUS clients are available.

2.2.3. Configured to talk to a trust infrastructure

The RP Proxy will need to be configured with an upstream connection to a trust infrastructure of some kind. Exactly what this will be will depend on the trust infrastructure in use, but will likely include information on how to connect to that trust infrastructure (e.g., details for a Trust Router, or details for a RADIUS proxy), along with relevant keying material.

For a detailed RP Proxy deployment guide, see Deploy a Relying Party Proxy under Installation and How-to Guides.

3. How Moonshot is used on the RP.

In general terms, the Moonshot-enabled server receives a request from a Moonshot-enabled client; it negotiates the use of Moonshot as an authentication mechanism with the client; it receives realm information for the credentials the client wishes to authenticate with; it interacts with a trust infrastructure to enable a connection to be opened to the IdP associated with that realm; it participates in setting up a TLS tunnel from the client to the IdP via itself (where it simply passes encrypted traffic back and forth that it cannot read); it receives a "yes" or a "no" from that Identity Provider; and finally, if its own policy decisions also say "yes", it opens up a normal session to the client.

More specifically:

1. The Moonshot-enabled service application receives a connection attempt from a Moonshot-enabled client application using that specific service's own protocols.
2. This connection negotiates GSS/SSP as the authentication method of choice, and GSS-EAP / EAP-SSP as the mechanism to use.
3. The service sends an EAP request to the Client.
4. The service receives an EAP response containing the realm of the credential to be used.
5. The service connects to its RP Proxy over RADIUS or RadSec, and gives this realm to it.
6. The RP Proxy uses that realm information to find the IdP (how this happens exactly depends on the trust infrastructure in use).
7. A secure TLS tunnel is subsequently established from the client to the IdP that flows through the RP (through both the service and the RP Proxy). This tunnel typically has an outer and an inner part. The outer part contains attributes such as the realm of the IdP and these can be seen by all waypoints, but the inner part is fully encrypted between client and IdP and cannot be seen by any of the other components.
8. The RP Proxy receives a RADIUS Access-Accept (yes) or Access-Reject (no). Alongside the yes/no, further information in the form of RADIUS and/or SAML attributes can be present as populated by the Identity Provider.
9. The RP Proxy can add or modify this information as required, and make policy decisions based upon it about whether a client should be allowed access or not. It then forwards the (potentially modified) RADIUS message to the service.
10. The service can further make policy decisions based on the information in the RADIUS message. If it decides that client should be granted access, it will finally allow a session from the client to itself be established.

See the wiki section on the Moonshot protocol flows for a full explanation of what this component does within those flows.