8.8 Secure Proxy

Intent

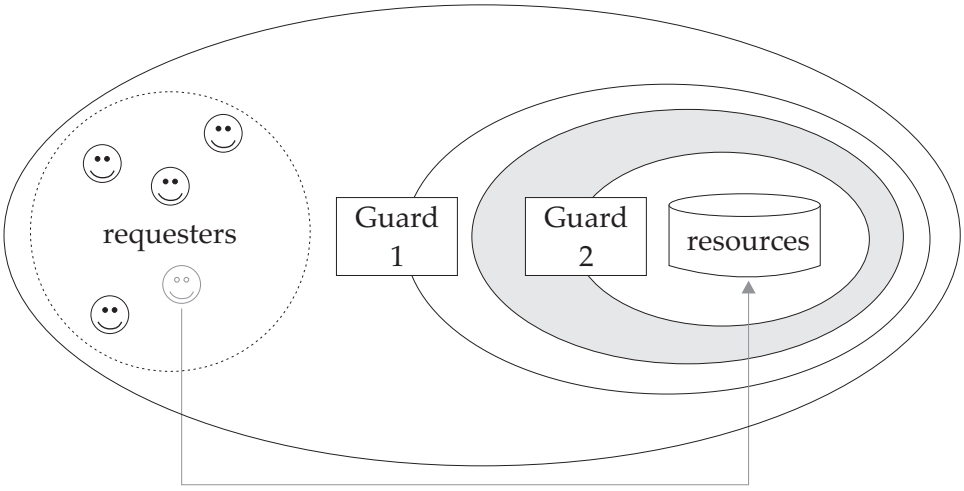
Define the relationship between the guards of two instances of Protected System in the case when one instance is entirely contained within the other.

Also Known As

Defense In Depth, Single Sign-on, Delegation, Security Protocol Encapsulation, Tunneling, Nested Protected Systems

Motivation

Security properties, especially authentication, often do not compose. Nevertheless, information systems are often built by composition. When composition results in one instance of Protected System being encapsulated inside another instance of Protected System, the requirements of both instances of Protected System need to be satisfied before resources inside the inner instance can be accessed.



A number of forces constrain solutions to this problem:

- The user would like to sign on only once.
- Both guards would like to authenticate the user.
- Both guards would like to enforce a policy based on the user's identity.
- The authentication protocol may not authenticate the user to more than one partner.
- The user may not want (or be allowed, or be able) to divulge a password or other authentication data to Guard 1.

Applicability

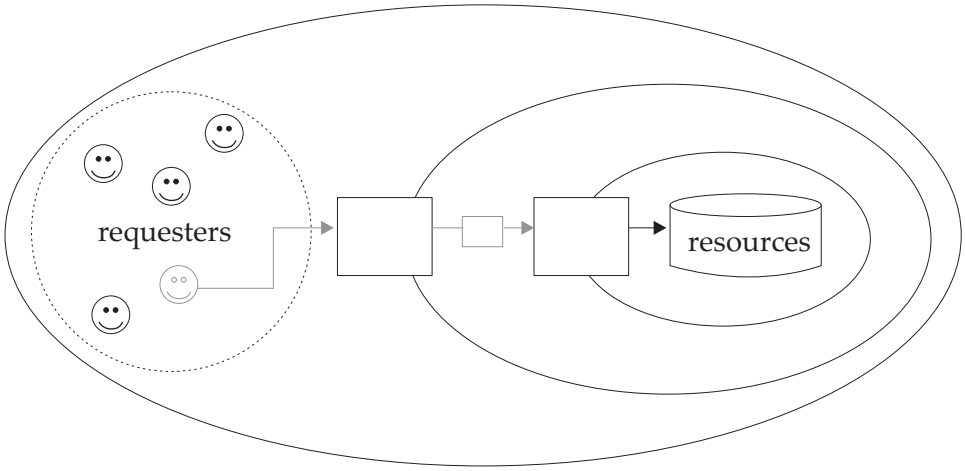
Use Secure Proxy when:

- One instance of Protected System is encapsulated inside another.

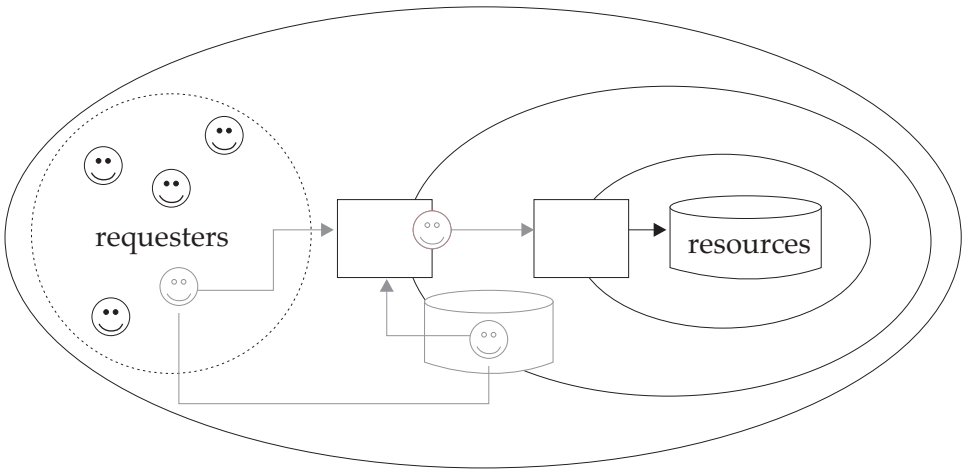
Structure

There are a number of approaches to resolving the forces described in the Motivation section; see Consequences for a description of how each of the Secure Proxy variants below addresses the various forces.

The first variant is the Trusted Proxy; in this variant, the user authenticates to Guard 1. Guard 1 then authorizes the user to access resources owned by Guard 2, and passes the request on to Guard 2 under its own (that is, Guard 1's) identity:

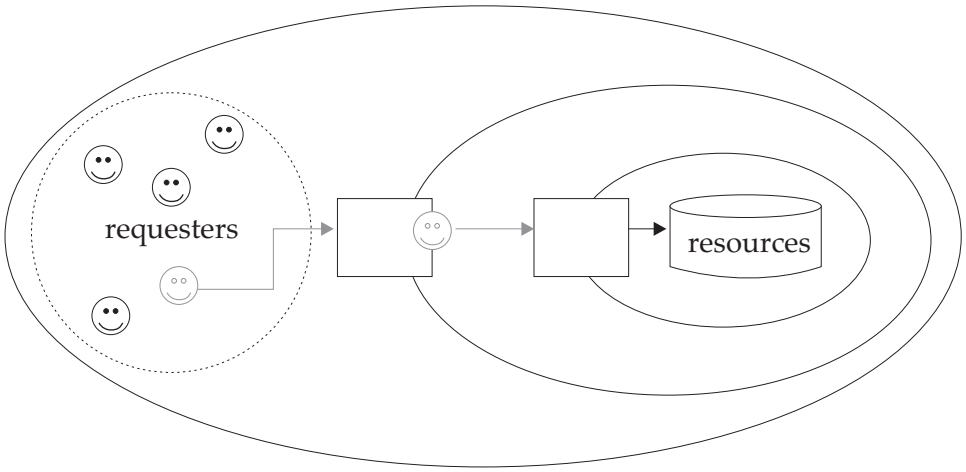


The second variant is the Authenticating Impersonator; in this variant, the user authenticates to Guard 1 and provides to Guard 1 the password (or other secret authentication data) which serves to authenticate the user to Guard 2; Guard 1 uses the user's password to authenticate ''as the user'' to Guard 2. Guard 2 authorizes the user's access to resources.

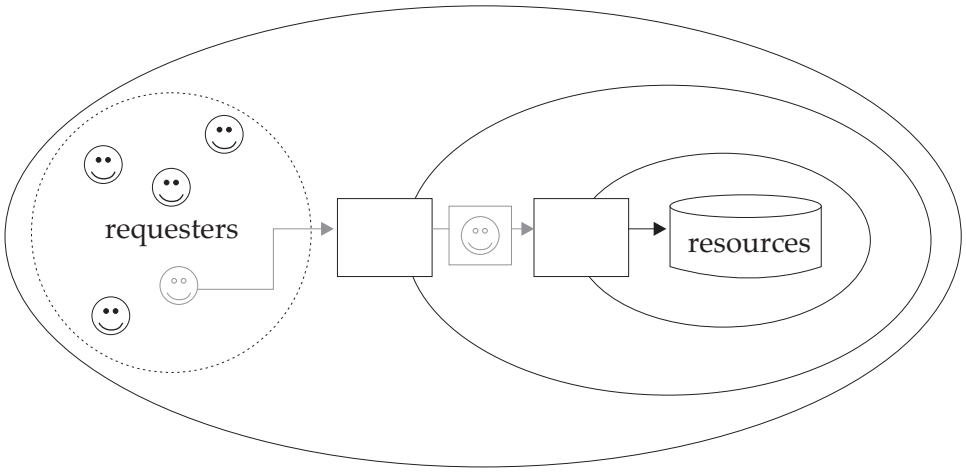


The third variant is the Identity-Asserting Impersonator. This variant is the same as the Authenticating Impersonator, except that Guard 2 ''trusts'' Guard 1 to assert the user's correct

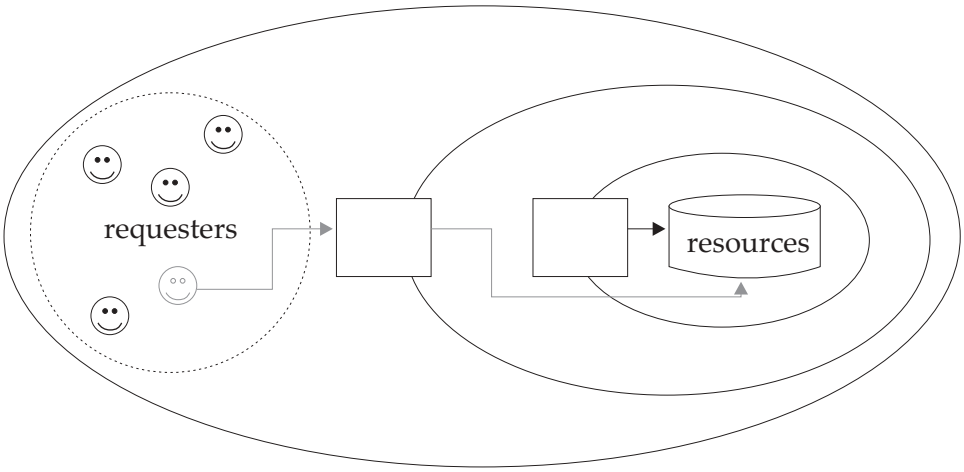
identity and does not require the presentation of a password by Guard 1. This means that Guard 1 does not need the user's password, but it does not eliminate the risk that Guard 1 will impersonate the user in unauthorized transactions with Guard 2.



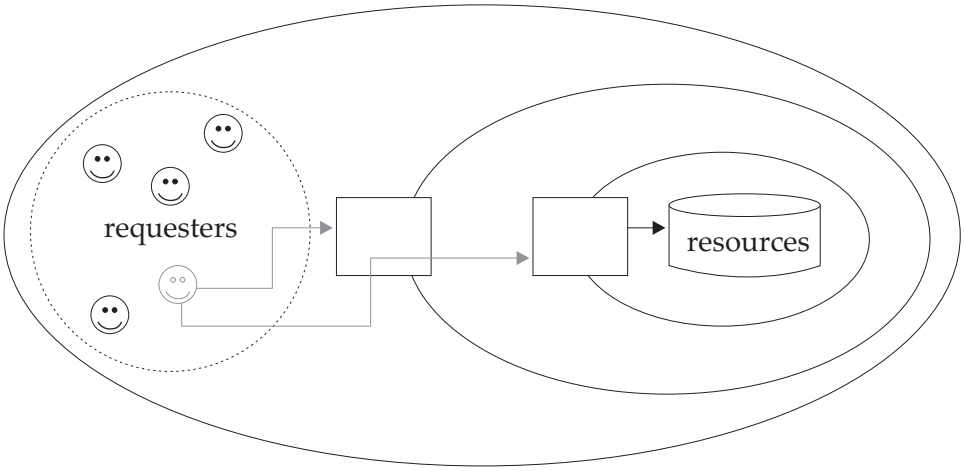
The fourth variant is the Delegate. In this variant, the security protocol shared by the user's system, Guard 1, and Guard 2, supports ''delegation''; that is, the ability of the user to authorize Guard 1 to assume the user's identity for the purposes of a single transaction with Guard 2 (and not with any other party).



The fifth variant is the Authorizing Proxy. In this variant, Guard 1 authenticates and authorizes the user, and Guard 2 simply ''steps out of the way'' and allows all requests originating from Guard 1 to pass through.



The sixth and last variant is the Login Tunnel. In this case, Guard 1 authenticates the user and then permits traffic to flow through to Guard 2 unimpeded. Guard 2 then authenticates the user for a second time and authorizes access to resources.



Participants

- User requests access to resources.
- Protected System 1 guards access to its own resources but also encapsulates Protected System 2.
- Guard 1 enforces Protected System 1's policy and authenticates users.
- Protected System 2 guards access to resources the user wishes to use.
- Guard 2 enforces Protected System 2's policy and authenticates users.

Collaborations

See Structure.

Consequences

No solution to the Secure Proxy problem is ideal; each of the variants described imposes a tradeoff between desired properties. The table below summarizes the tradeoffs.

	passwd to guard1	userid to guard2	guard2 authn	guard2 authz	sso	delegation protocol
ideal	no	yes	user	user	yes	no
trusted proxy	no	***no***	***guard1***	***guard1***	yes	no
authn impers	***yes***	yes	user	user	yes	no
id-assert impers	no	yes	***no***	user	yes	no
delegate	no	yes	user	user	yes	***yes***
authz proxy	no	***no***	***no***	***no***	yes	no
login tunnel	no	yes	user	user	***no***	no

The table is read as follows: the column labels describe desirable properties of a solution to the Secure Proxy problem.

- *passwd to guard1* means that the user must disclose a password or other secret authentication data to *guard1*, thus creating a risk that *guard1* will later impersonate the user in an unauthorized transaction.
- *userid to guard2* means that the user's Subject Descriptor information is provided to *guard2*, so that *guard2* can use it to make policy decisions.
- *guard2 authn* means that *guard2* authenticates a particular party — either the user, *guard1*, or no one at all.
- *guard2 authz* means that *guard2*'s authorization policy is based on Subject Descriptor information for a particular party — either the user, *guard1*, or no one at all.
- *sso* means that the user only needs to engage in one authentication dialog.
- *delegation protocol* means that the user's system, *guard1*, and *guard2* all have access to the same security protocol, and that protocol implements delegation functionality.

The first row of the table describes the ''ideal'' solution to the Secure Proxy problem; this solution meets all of the user's usability and risk mitigation requirements, meets all of *guard1*'s policy requirements, meets all of *guard2*'s policy requirements, and does not require ubiquitous implementation of complicated, inefficient, and difficult-to-manage delegation protocols.

Each subsequent row of the table describes the characteristics of one of the variant solutions to the problem described in Structure. Undesirable characteristics are highlighted in ***Bold-Italic*** font.

Implementation

Future Writer's Workshops will identify appropriate text for this section.

Known Uses

- Identity-Asserting Impersonator: IBM SNA LU6.2 ‘‘Already Verified’’
- Delegate: OSF DCE Kerberos
- Login Tunnel: Many VPN servers

Related Patterns

- Protected System [TG_SDP]
- Policy Enforcement Point [TG_SDP]
- Secure Communication [TG_SDP]
- Trusted Proxy [NAI]
- Layered Security [Romanowsky]