Securing IceGrid

IceGrid's registry and node services expose multiple network endpoints that a malicious client could use to gain access to IceGrid functionality and interfere with deployed applications. This presents a significant security risk in network environments that are exposed to untrusted clients. For example, a malicious client could connect to a node and use IceGrid's internal interfaces to deploy and run its own server executable.

This page describes the steps you can take to secure your IceGrid application.

On this page:

IceGrid Security Overview

Using a firewall is one way to prevent unauthorized use of IceGrid's facilities. Another solution is to use IceSSL: you can generate SSL certificates for each component and configure them to trust and accept connections only from other authorized components. The remainder of this section discusses the IceSSL solution but also provides useful information for those interested in securing IceGrid with a firewall.

To restrict access using IceSSL, we need to establish trust relationships between IceGrid registry replicas, nodes, and deployed servers. IceSSL allows us to do this using configuration properties. The trust relationships are based on the information contained in SSL certificates.

There are several possible strategies for generating certificates. At a minimum you will need the following:

- one certificate for all of the registries
- one certificate for all of the nodes
- one certificate for all of the servers managed by IceGrid

The certificates that you generate for registries and nodes should be protected with a password to ensure that only privileged users can start these services. However, we do not recommend using a password to protect the certificate for deployed servers because it would need to be specified in clear text in each server's configuration (servers that are activated by IceGrid must not prompt for a password). Furthermore, this password might appear in multiple places, such as an XML descriptor file, the IceGrid registry database, and property files generated by IceGrid nodes. The complexity involved in protecting access to every file that contains a clear text password could be overwhelming. Instead, we recommend that you protect access to the server certificate using file system permissions.

Depending on your organization and the roles of each person that uses IceGrid, you may decide to create additional certificates. For example, you might create a unique certificate for each IceGrid node instance if you deploy nodes on end-user machines and wish to configure the IceGrid registry to authorize connections only from the nodes of trusted users.

You can use the `iceca` script to establish a certificate authority and generate certificates. The sections that follow describe the interactions between the registry, node, and servers, and show how to configure IceSSL to restrict access to trusted peers. For the purposes of this discussion, we assume that the SSL certificates use the common names shown below:

- `IceGrid Registry`
- `IceGrid Node`
- `Server`

The Ice distribution includes a C++ example that demonstrates how to configure a secure IceGrid deployment in the `demo/IceGrid/secure` subdirectory. This example includes a script to generate certificates for a registry, a node, a Glacier2 router, and a server. For more information, see the `README` file provided with the example.

Understanding the Registry Endpoints

The IceGrid registry has three mandatory endpoints representing the client, server, and internal endpoints. The registry also has two optional endpoints (the session manager and administrative session manager endpoints) that are only useful when accessing IceGrid via Glacier2.

Client Endpoint

The registry client endpoint is used by Ice applications that create client sessions in order to use the resource allocation facility. It is also used by administrative clients that create sessions for managing the registry. Finally, the client endpoint is used by Ice applications that use the `IceGrid::Query` interface or resolve indirect proxies via the IceGrid locator.

Two distinct permission verifiers authorize the creation of client sessions and administrative sessions. The remaining functionality available via the client endpoint, such as resolving objects and object adapters using the `IceGrid::Query` interface or the Ice locator mechanism, is accessible to any client that is able to connect to the client endpoint.

It is safe to use an insecure transport for the client endpoint if it is only being used for locator queries. However, you should use a secure transport if you have enabled client and administrative sessions (by configuring the appropriate permission verifiers). Creating a session over an insecure transport poses a security risk because the user name and password are sent in clear text over the network.

If you include secure and insecure transports in the registry's client endpoints, you should ensure that applications that need to authenticate with IceGrid permission verifiers use a secure transport.

It is not necessary to restrict SSL access to the client endpoints (using the property `IceSSL.TrustOnly.Server.IceGrid.Registry.Client`) as long as you use client and administrative permission verifiers for authentication. This property is only useful for restricting access to client and administrative sessions when using null permission verifiers. Note however that if both client and administrative sessions are enabled, you will only be able to restrict access to one set of clients since you cannot distinguish clients that create client sessions from clients that create administrative sessions.

Server Endpoint

Ice servers use the registry's server endpoint to register their object adapter endpoints and send information to administrative clients connected via the registry.

Securing this endpoint with IceSSL is necessary to prevent a malicious program from potentially hijacking a server by registering its endpoints first. The property definition shown below demonstrates how to limit access to this endpoint to trusted Ice servers:

```
IceSSL.TrustOnly.Server.IceGrid.Registry.Server=CN="Server"
```

Internal Endpoint

IceGrid nodes and registry replicas use the internal endpoint to communicate with the registry. For example, nodes connect to the internal endpoint of each active registry, and registry slaves establish a session with their master via this endpoint.

The internal endpoint must be secured with IceSSL to prevent malicious Ice applications from gaining access to sensitive functionality that is intended to be used only by nodes and registry replicas. You can restrict access to this endpoint with the following property:

```
IceSSL.TrustOnly.Server.IceGrid.Registry.Internal=CN="IceGrid Node";CN="IceGrid Registry"
```

Session Manager Endpoint

The session manager endpoint is used by Glacier2 to create IceGrid client sessions. The functionality exposed by this endpoint is unrestricted so you must either secure it or disable it (this endpoint is disabled by default). The property shown below demonstrates how to configure IceSSL so that only Glacier2 routers are accepted by this endpoint:

```
IceSSL.TrustOnly.Server.IceGrid.Registry.SessionManager=CN="Glacier2 Router Client"
```

In this example, `Glacier2 Router Client` is the common name of the Glacier2 router used by clients to create IceGrid client sessions.

Administrative Session Manager Endpoint

Glacier2 routers use the registry's administrative session manager endpoint to create IceGrid administrative sessions. The functionality exposed by this endpoint is unrestricted, so you must either secure it or disable it (this endpoint is disabled by default). The property shown below demonstrates how to configure IceSSL so that only Glacier2 routers are accepted by this endpoint:

```
IceSSL.TrustOnly.Server.IceGrid.Registry.AdminSessionManager=CN="Glacier2 Router Admin"
```

In this example, `Glacier2 Router Admin` is the common name of the Glacier2 router used by clients to create IceGrid administrative sessions. Note that if you use a single Glacier2 router instance for both client and administrative sessions, you will need to use the same common name to restrict access to both session manager endpoints:

```
IceSSL.TrustOnly.Server.IceGrid.Registry.SessionManager=CN="Glacier2 Router Client"
IceSSL.TrustOnly.Server.IceGrid.Registry.AdminSessionManager=CN="Glacier2 Router Client"
```

Outgoing Connections

The registry establishes outgoing connections to other registries and nodes. You should configure the `IceSSL.TrustOnly.Client` property to restrict connections to these trusted peers:

```
IceSSL.TrustOnly.Client=CN="IceGrid Registry";CN="IceGrid Node"
```

The registry can also connect to Glacier2 routers and permission verifier objects. To allow connections to these services, you must include in this property the common names of Glacier2 routers that create client or administrative sessions, as well as the common names of servers that host the permission verifier objects.

Understanding the Node Endpoints

An IceGrid node has only one endpoint, which is used for internal communications with the registry. As a result, it should be configured to accept connections only from IceGrid registries:

```
IceSSL.TrustOnly.Server=CN="IceGrid Registry"
```

A node also establishes outgoing connections to the registry's internal endpoint, as well as the `Ice.Admin` endpoint of deployed servers. You should configure the `IceSSL.TrustOnly.Client` property as shown below to verify the identity of these peers:

```
IceSSL.TrustOnly.Client=CN="Server";CN="IceGrid Registry"
```

Understanding the Administrative Endpoints with IceGrid

By default, IceGrid sets the endpoints of a deployed server's `Ice.Admin` adapter to `tcp -h 127.0.0.1`. This setting is already quite secure because it only accepts connections from processes running on the same host. However, since you already need to configure IceSSL so that a server can authenticate with the IceGrid registry (servers connect to the registry to register their endpoints), you might as well use a secure endpoint for the `Ice.Admin` adapter and configure it to accept connections only from IceGrid nodes:

```
IceSSL.TrustOnly.Server.Ice.Admin=CN="IceGrid Node"
```

This is only necessary if the `Ice.Admin` endpoint is enabled (which it is by default).

You can also set the `IceSSL.TrustOnly.Client` property so that the server is only permitted to connect to the IceGrid registry:

```
IceSSL.TrustOnly.Client=CN="IceGrid Registry"
```

If your server invokes on other servers, you will need to modify this setting to allow secure connections to them.

See Also

- IceSSL
- Configuring IceSSL
- Setting up a Certificate Authority
- Glacier2 Integration with IceGrid
- Resource Allocation using IceGrid Sessions
- IceGrid Administrative Sessions
- Well-Known Objects