# Security guide

Use this guide to reduce risk and mitigate threats to IG security.

**Threats**

Understand and address security threats.

**Operating systems**

Secure your operating systems.

**Connections**

Secure network connections.

**Access**

Remove non-essential access and features, update patches, and manage cookies.

**Keys and Secrets**

Manage keys and secrets.

**Audit Trails**

Audit events in your deployment.

ForgeRock® Identity Platform serves as the basis for our simple and comprehensive Identity and Access Management solution. We help our customers deepen their relationships with their customers, and improve the productivity and connectivity of their employees and partners. For more information about ForgeRock and about the platform, see https://www.forgerock.com. The ForgeRock® Common REST API works across the platform to provide common ways to access web resources and collections of resources.

## Access

The following sections describe how to prevent unwanted access to your deployment, and reduce the amount of non-essential information that it provides.

### Use an IG service account

Install and run IG from a dedicated service account. This is optional when you are evaluating IG, but essential in production installations. For more information, see Create an IG service account.

### Remove non-essential access

Make sure that only authorized people can access your servers and applications through the appropriate network, using the appropriate ports, and by presenting strong-enough credentials.

Apply the principle of least privilege to IG logs and configuration directories. For more information, see Set up logs and configuration files.

Make sure that users connect to systems through the latest versions of TLS, and audit system access periodically.

Restrict access to your monitoring data by protecting the Prometheus Scrape Endpoint and Common REST Monitoring Endpoint. For information, see Protecting the monitoring endpoints.

Prevent IG from scanning for changes to routes. For information, see `scanInterval` in Router.

Disable administration endpoints and Studio by setting the IG run mode to `production`. For information, see Switching from development mode to production mode.

### Remove non-essential features

The more features you have turned on, the greater the attack surface. If something is not being used, uninstall it, disable it, or protect access to it.

### Update patches

Prevent the exploitation of security vulnerabilities by using up-to-date versions of IG and third-party software.

Review and follow the Ping Identity security advisories.

Follow similar lists from all of your vendors.

## Manage cookies

Increase the security of cookies genrated by IG or the protected application in the following ways:

- Change the default name of cookies to prevent them from being easily associated with an application.
- Create cookies with the `secure` flag to ensure that browsers cannot transmit the cookie over non-SSL.

  When cookies have the `secure` flag, the first hop of the connection between the user agent and protected application must be secure (over HTTPs); subsequent hops do not have to be secure. In this example, the first hop from the user agent to NGINX is secure, the subsequent hop to IG is not secure:

  ```
  User agent -> NGINX (https://acme.com) -> IG
  (http://gateway:8080)-> protected application
  (https://internal.app:8081)
  ```
- Create cookies with the `httpOnly` flag, to ensure that the cookie cannot be accessed through client-side scripts, and to mitigate any cross-site scripting attacks.

  Cookies are `httpOnly` by default in `admin.json`, JwtSession, CrossDomainSingleSignOnFilter, and FragmentFilter. In web container mode, check `WEB-INF/web.xml`. For more information, see Configuring stateful sessions.
- Set the `samesite` attribute of cookies to `STRICT` or `LAX`. For more information, see SameSite cookies.
- Set a timeout for cookies, to strike a good compromise between security and usability.

Harden an IG configuration by configuring the following objects:

- For stateful sessions in standalone mode, configure the `session.cookie` property in admin.json.
- For stateful sessions in web container mode, configure the container `WEB-INF/web.xml` file when you unpack the IG .war file. For more information, see Configuring stateful sessions.
- For stateless sessions, configure the `cookie` property of JwtSession.

- For authentication results, configure the `authCookie` property of CrossDomainSingleSignOnFilter.
- For the fragment part of a URI when a request triggers a login redirect, configure the `cookie` property of FragmentFilter.

## Threats

The following sections describe some of the possible threats to IG, which you can mitigate by following the instructions in this guide.

### Out-of-date software

Prevent the exploitation of security vulnerabilities by using up-to-date versions of IG and third-party software.

Review and follow the Ping Identity security advisories.

Follow similar lists from all of your vendors.

### Reconnaissance

The initial phase of an attack sequence is often reconnaissance. Limit the amount of information available to attackers during reconnaissance, as follows:

- Avoid using words that help to identify IG in error messages, such as those produced by the entity in a StaticResponseHandler. For information, see StaticResponseHandler.
- Use the lowest level of logging necessary. For example, consider logging at the `ERROR` or `WARNING` level, instead of `TRACE` or `MESSAGE`. For information, see Changing the global log level.

### Cross-site scripting

When you are using a StaticResponseHandler, secure responses from cross-site scripting attacks, as follows:

- Sanitize any external input, such as the request, before incorporating it in the response.
- Specify `Content-Type` in the `headers` property of StaticResponseHandler when an entity is used. (Required by default, from IG 7.)
- Set the response header `X-Content-Type-Options: nosniff` to prevent the user agent from interpreting the response entity as a different content type. (Set by default, from IG 7.)

- Set a restrictive value in the `Cache-Control` response header. For example, setting `Cache-Control: private` indicates that all or part of the response message is intended for a single user and MUST NOT be cached by a shared cache.

## Compromised passwords

Despite efforts to improve how people manage passwords, users have more passwords than ever before, and many use weak passwords. You are strongly encouraged to use a password manager to generate secure passwords. You can use identity and access management services to avoid password proliferation, and you can ensure the safety of passwords that you cannot eliminate.

Manage passwords for server administration securely. Passwords supplied to IG can be provided in files, through environment variables, or as system property values. Choose the approach that is most appropriate and secure for your deployment.

## Misconfiguration

Misconfiguration can arise from bad or mistaken configuration decisions, and from poor change management. Depending on the configuration error, features can stop working in obvious or subtle ways, and potentially introduce security vulnerabilities.

The following behaviour can be caused by misconfiguration:

- Routes fail to load, or succeed in loading but cause unexpected behaviour.

  For example, if a configuration change prevents the server from making HTTPS connections, many applications can no longer connect, and the problem is detected immediately. However, if a configuration change allows insecure TLS protocol versions or cipher suites for HTTPS connections, some applications negotiate insecure TLS, but appear to continue to work properly.
- Access policy is not correctly enforced.

  Incorrect parameters for secure connections and incorrect Access Control Instructions (ACI) can lead to overly permissive access to data, and potentially to a security breach.
- The server fails to restart.

  Although failure to start a server is not directly a threat to security, it can affect service availability.

To guard against bad configuration decisions, implement good change management:

- For all enabled features, document why they are enabled and what your configuration choices mean. This implies a review of configuration settings, including default settings that you accept.

- Validate configuration decisions with thorough testing.
- Maintain a record of your configurations and the changes applied.

  For example, use a filtered audit log. Use version control software for any configuration scripts and to record changes to configuration files.
- Maintain a record of external changes to the system, such as changes to operating system configuration, and updates to software, such as the JVM that introduces security changes.

### Unauthorized access

Data theft can occur when access policies are too permissive, and when the credentials to gain access are too easily cracked. It can also occur when the data is not protected, when administrative roles are too permissive, and when administrative credentials are poorly managed.

### Poor risk management

Threats can arise when plans fail to account for outside risks. To mitigate risk, develop appropriate answers to at least the following questions:

- What happens when a server or an entire data center becomes unavailable?
- How do you remedy a serious security issue in the service, either in the IG software or the connected systems?
- How do you validate mitigation plans and remedial actions?
- How do client applications work when the IG offline?

  If client applications require always-on services, how do your operations ensure high availability, even when a server goes offline?

For critical services, test expected operation and disaster recovery operation.

## Operating systems

When you deploy IG, familiarize yourself with the recommendations for the host operating systems that you use. For comprehensive information about securing operating systems, see the CIS Benchmark documentation.

### System updates

Over the lifetime of a deployment, the operating system might be subject to vulnerabilities. Some vulnerabilities require system upgrades, whereas others require only configuration changes. All updates require proactive planning and careful testing.

For the operating systems used in production, put a plan in place for avoiding and resolving security issues. The plan should answer the following questions:

- How does your organization become aware of system security issues early?

  This could involve following bug reports, mailing lists, forums, and other sources of information.
- How do you test security fixes, including configuration changes, patches, service packs, and system updates?

  Validate the changes first in development, then in one or more test environments, then in production in the same way you would validate other changes to the deployment.
- How do you roll out solutions for security issues?

  In some cases, fixes might involve both changes to the service, and specific actions by those who use the service.
- What must you communicate about security issues?
- How must you respond to security issues?

Software providers often do not communicate what they know about a vulnerability until they have a way to mitigate or fix the problem. Once they do communicate about security issues, the information is likely to become public knowledge quickly. Make sure that you can expedite resolution of security issues.

To resolve security issues quickly, make sure that you are ready to validate any changes that must be made. When you validate a change, check that the fix resolves the security issue. Validate that the system and IG software continue to function as expected in all the ways they are used.

## System audits

System audit logs make it possible to uncover system-level security policy violations that are not recorded in IG, such as unauthorized access to IG files. Such violations are not recorded in IG logs or monitoring information.

Also consider how to prevent or at least detect tampering. A malicious user violating security policy is likely to try to remove evidence of how security was compromised.

## Unused features

By default, operating systems include many features, accounts, and services that IG software does not require. Each optional feature, account, and service on the system brings a risk of additional vulnerabilities. To reduce the surface of attack, enable only required features, system accounts, and services. Disable or remove those that are not needed for the deployment.

The features needed to run and manage IG software securely include the following:

- A Java runtime environment, required to run IG software.
- Software to secure access to service management tools; in particular, when administrators access the system remotely.
- Software to secure access for remote transfer of software updates, backup files, and log files.
- Software to manage system-level authentication, authorization, and accounts.
- Firewall software, intrusion-detection/intrusion-prevention software.
- Software to allow auditing access to the system.
- System update software to allow updates that you have validated previously.
- If required for the deployment, system access management software such as SELinux.
- Any other software that is clearly indispensable to the deployment.

Consider the minimal installation options for your operating system, and the options to turn off features.

Consider configuration options for system hardening to further limit access even to required services.

For each account used to run a necessary service, limit the access granted to the account to what is required. This reduces the risk that a vulnerability in access to one account affects multiple services across the system.

Make sure that you validate the operating system behavior every time you deploy new or changed software. When preparing the deployment and when testing changes, maintain a full operating system with IG software that is not used for any publicly available services, but only for troubleshooting problems that might stem from the system being *too* minimally configured.

## Network connections

Protect network traffic by using HTTPS where possible, and secure communications during stateless sessions by signing and/or encrypting JWTs. For information about configuring IG for HTTPS client-side and HTTPS server-side, see the Installation guide.

*Recommendations for incoming connections (from clients to IG).*

| Protocol | Recommendations |
|---|---|

| Protocol | Recommendations |
|---|---|
| HTTP | HTTP connections that are not protected by SSL/TLS use cleartext messages. When you permit insecure connections, you cannot prevent client applications from sending sensitive data. For example, a client could send unprotected credentials in an HTTP Authorization header. Even if the server were to reject the request, the credentials would already be leaked to any eavesdroppers.<br><br>Always use HTTPS for connections up to a load-balancer or proxy in front of the web application or server. |
| HTTPS | Follow industry-standard TLS recommendations for Security/Server Side TLS.<br><br>Use a secure version of TLS/SSL to connect to TLS-protected endpoints with HTTP connection handlers, such as ClientHandler and ReverseProxyHandler. TLS protocols below 1.2 are not considered secure.<br><br>Some client applications require a higher level of trust, such as clients with additional privileges or access. Client application deployers might find it easier to manage public keys as credentials than to manage user name/password credentials. Client applications can use SSL client authentication.<br><br>When using IG REST to LDAP gateway, use HTTPS to protect client connections. |
| JMX | Secure JMX access with the SSL/TLS-related properties, such as `use-ssl` and others. |
| SSH | IG administration tools can connect securely.<br><br>Administrators should use SSH when changing the IG configuration or binaries.<br><br>The user account for running IG should not be the same user account for connecting remotely.<br><br>Secure Copy (SCP) uses SSH to transfer files securely. SCP is an appropriate protocol for copying backup data, for example. |

*Recommendations for outgoing connections (from IG to another service.)*

| Client | Recommendations |
|---|---|
| Common Audit event handlers | Configure ForgeRock Common Audit event handlers to use HTTPS when connecting to external log services. |
| OAuth 2.0-based HTTP authorization mechanisms | HTTP authorization can be based on OAuth 2.0, where IG servers act as resource servers, and make requests to resolve OAuth 2.0 tokens.<br><br>Use HTTPS to protect the connections to OAuth2ResourceServerFilter and AuthorizationCodeOAuth2ClientFilter. For information, see OAuth2ResourceServerFilter and AuthorizationCodeOAuth2ClientFilter. |

### Message-level security

Server protocols such as HTTP, LDAP, and JMX rely on TLS to protect connections. To enforce secure communication, configure TLS as follows:

- Configure IG for HTTPS server-side in `admin.json` for standalone mode, in the container for web container mode. See the Installation guide.
- Configure IG for HTTPS client-side by configuring trust managers and key managers. See the Configure IG For HTTPS (client-side).

When negotiating connection security, the server and client must use a common security protocol and cipher suite. In ClientTlsOptions and ServerTlsOptions, define lists of security protocols and cipher suites. For security, use the most recent protocols and ciphers that the client supports. Clients with older TLS implementations might not support the most recent protocols and ciphers.

## Keys and secrets

IG uses cryptographic keys for encryption, signing, and securing network connections, and passwords. The following sections describe how to secure keys and secrets in your deployment.

### Update cryptography

Different algorithms and methods are discovered and tested over time, and communities of experts decide which are the most secure for different uses. Use up-to-date cryptographic methods and algorithms to generate keys.

### Use strong keys

Small keys are easily compromised. Use at least the HTTPS://wiki.mozilla.org/Security/Server_Side_TLS#Intermediate_compatibility_.28recommended.29[ recommended key size].

In JVM, the default ephemeral Diffie-Hellman (DH) key size is 1024 bits. To support stronger ephemeral DH keys, and protect against weak keys, consider setting the following system property to increase the DH key size: `jdk.tls.ephemeralDHKeySize=2048`. From Tomcat 8.5.37, the default DH key size is 2048-bit.

For more information, see Customizing size of ephemeral Diffie-Hellman keys

### Rotate keys

Rotate keys regularly to:

- Limit the amount of data protected by a single key.
- Reduce dependence on specific keys, making it easier to migrate to stronger algorithms.
- Prepare for when a key is compromised. The first time you try key rotation shouldn't be during a real-time recovery.
- Conform to internal business compliance requirements.

For more information, see Rotating keys.

## Audits and logs

### Audit trails

Audits in IG record access to a route. Audit logs in operating systems detect system login attempts and changes to the software.

The IG audit logging service adheres to the log structure common across the ForgeRock Identity Platform. For information, see Auditing your deployment.

Prevent logging of sensitive data for audit events by excluding fields from the audit logs. For information, see Including or excluding audit event fields in logs.

### Log files

Logs in IG contain informational, error, and warning events, to troubleshoot and debug transactions and events that take place within the IG instance.

Protect logs from unauthorised access, and make sure they contain a minimum of sensitive or personally identifiable information that could be used in attacks.

When using a CaptureDecorator, mask captured header and attribute values to avoid disclosing information, such as token values or passwords. For information, see CaptureDecorator.

Limit the number of repeat log messages to prevent log flow attacks, by adding a custom `logback.xml` with a `DuplicateMessageFilter`. For information, see Limit repetitive log messages.