Ignition Guide

Ignition® 8.3

Security Hardening Guide

inductive automation

(800) 266-7798

www.inductiveautomation.com

Table of Contents

Introduction

Welcome to Inductive Automation's Ignition Security Hardening Guide. Inductive Automation is committed to security and strives to make the product as secure as possible. This document is intended to provide recommendations on how to secure an Ignition installation.

Included in this document are guidelines specifically for the Ignition software, as well as general suggestions regarding the hardware and network where Ignition is installed. The steps provided are recommendations rather than requirements and should be considered in accordance with organizational policies and objectives.

Start with a Good Foundation

Ignition security should be considered within the context of its greater environment. "Defense in Depth" is a strategy that uses overlapping protective mechanisms supporting the ability of defenders to monitor and respond. Strong network segmentation should be applied. Consider adopting commonly accepted industry standards such as the ISA/IEC 62443 series of OT cybersecurity standards.

It is an industry best practice to adhere to a formal cybersecurity framework to align security controls with organizational objectives. The Cybersecurity and Infrastructure Security Agency (CISA) under the Department of Homeland Security (DHS) provides great free cybersecurity resources.

Secure Baseline

The following steps offer best practice recommendations for securing Ignition and its environment. These steps cover securing access to the Gateway and devices, user management, application security, auditing, the environment, and keeping Ignition up to date. Security controls should align with organizational goals and policy.

Step 1: Secure Gateway Communication

Enforcing secure communication (HTTPS using an SSL/TLS certificate) is the first and most important step towards securing an Ignition Gateway.

Secure Communication

An Ignition Gateway can be configured to provide modern, end-to-end secure communication using Transport Level Security (TLS) technologies by enabling HTTPS. Secure communication protects all Gateway traffic to Perspective sessions, Vision clients, Ignition Designers, and Ignition web configuration from attackers and eavesdroppers through strong encryption and standard technologies. TLS also helps protect against a class of security vulnerabilities known as "session hijacking" where a threat actor may exploit a valid computer session to gain access to an Ignition system. Examples of “session hijacking” are: Man-in-the-middle attack, cross-site scripting (XSS), or session sniffing.

Terminology note: “Secure Sockets Layer”, (SSL) is the predecessor to the Transport Layer Security (TLS) protocol. SSL is deprecated as a technology. However, the term “SSL” is still widely used to refer to secure communication and “TLS”. For example, modern digital certificates supporting TLS are commonly referred to as “SSL certificates”.

Steps to Enforce Secure Communication

Full instructions to Enable Secure Communication are available online. The broad steps are:

1. Obtain certificate. Secure communication (HTTPS) in Ignition requires an SSL/TLS Certificate installed in the Gateway. It is highly recommended to obtain genuine certificates issued from a reputable Certificate Authority (CA).
2. Install certificate in Ignition Gateway.
3. Force secure redirect. After SSL/TLS is enabled, all Clients, Designers, and web browsers are redirected to the HTTPS port if they try to use the standard HTTP port. By default, the HTTPS port is 8043. Consider the standard https port (443) when customer facing.
4. Consider certificate renewal. First consider organizational standards from IT or cybersecurity. Many SSL/TLS certificates have a cumbersome renewal lifecycle. The renewal process can be simplified and automated using the ACME (Automatic Certificate Management Environment) protocol. ACME is an automated framework for obtaining and renewing certificates. Let’s Encrypt is a “free, automated, and open certificate authority (CA)” that can renew certificates using an ACME server and an Internet connection. See Let’s Encrypt Guide for Ignition.

Disabling Older Cipher Suites

When HTTPS is enabled, cipher suites provide essential information on how to communicate secure data between the Ignition Gateway and the user’s browser. The user’s browser will notify the Ignition server which cipher suites the browser supports, and the most secure cipher suite supported by both will be used for communication. As a security assumption, consider the gateway to be protected at the level of the weakest allowed cipher suite. A threat agent can execute a “downgrade attack” by only offering the chosen suite. Cipher suites can be excluded in the Gateway settings under **Config** > **Web Server** > **Excluded Cipher Suites**.

The suites currently considered strong are ever-changing as security researchers discover flaws and create new algorithms. One source for information on the currently accepted list of strong suites is maintained by SSL Labs. SSL Labs provides a free online test if your Ignition server is accessible from the Internet. Their current list of strong cipher suites can be found in their “SSL/TLS Deployment Best Practices” guide under 2.3 User Secure Cipher Suites. In general, allowing older cipher suites helps with compatibility while fewer and stronger cipher suites improve security.

Configuring Strong Headers

Https headers are set to secure default values as described under Gateway Security HTTP Headers and Configuration. Ignition offers the opportunity to change the values of these headers in the Gateway Configuration (Ignition.conf) file.

Consider enabling HTTP Strict Transport Security (HSTS). HSTS ensures that web browsers connect over https. It is disabled by default in Ignition to allow communication over http. **Enabling HSTS is recommended if Ignition is being used over standard http port 80 and https port 443**. It is not recommended with default Ignition ports of 8088 and 8043.

1. Enable HSTS per this reference. In **Ignition.conf** file under "Java Additional Parameters" set:
    a. wrapper.java.additional.1=-Dignition.https.sts.maxAge=63072000
        i. Note: 63072000 seconds = 2 years.
    b. wrapper.java.additional.2=-Dignition.https.sts.includeSubDomains=false
    c. wrapper.java.additional.3=-Dignition.https.sts.preload=true
2. Setting "preload=true" means that the gateway URL can be registered with the HTST preload list. This will prevent HTTP from ever being used at all and will force HTTPS from the browser itself. To use it, it does require a user to register their site with https://hstspreload.org/ after setting preload=true.

Step 2: Restrict Ignition OS and File System Permissions

Complete the steps in ***Appendix A - Restrict Ignition Service Security*** to limit unneeded Ignition service permissions to the local operating system, file system, and network resources.

Ignition Privileged Users

By design, any users with direct or indirect access to modify projects (e.g. Designer, Gateway roles) are able to write Python applications (a.k.a. *programs*) that will be executed by the Ignition Gateway with all privileges granted to the Ignition process. This is the power of Ignition.

The implication is that an Ignition designer (person) is implicitly trusted with the (authorized) sum of all privileges of the Ignition Gateway process, regardless of individual credentials. This often includes access to devices, databases, and business systems.

Ignition Process - System Account and Persistent Connections

Ignition is designed to run 24/7, which includes a persistent operating system process and trusted communication connections. This includes access to devices (e.g. PLC, OPC UA) and databases.

The Ignition process needs to start automatically. It is executed by a user-specified service account on the Gateway. This means that Ignition is theoretically capable of all operating system actions of that account including privileges on the local host and larger network environment.

In order to protect the larger environment, it is important to apply the *Principle of Least Privilege* outside of Ignition. Consider the following tips:

- Minimize privileges of the Ignition computer and service account on the larger network.
- Minimize Ignition permissions on external databases and business systems.

    - API access is often safer than direct access.
- Segment network areas with “zones and conduits” or “accreditation boundaries”.

Finally, many customer implementations use multiple Ignition Gateways. Each gateway has its own Ignition process with associated credentials. Permissions may differ between gateways depending on the architecture.

Step 3: Lock Gateway Access

Review and set privileged Ignition Gateway access according to your needs.

The gateway web site is organized into six sections, **Home**, **Platform**, **Connections**, **Network**, **Services**, and **Diagnostics**. By default, the **Platform**, **Connections**, **Network**, **Services**, and **Diagnostics** sections are protected by the *“Authenticated/Roles/Administrator”* security level. Gateway sections and Designer access can be protected with role-based access control.

Details for setting up the Gateway Security Settings.

External Identity Providers can provide extra security when used for supplying gateway user access. Please see the Perspective Session Security section for more details on Identity Providers.

Step 4: Device, OPC, and MQTT Security

Operational Technology (OT) device connections have historically been made using native device communication protocols. Many PLC manufacturers created their own protocols for communication, and a variety are popular and in heavy use today. Many modern devices natively support standard communication protocols (OPC UA and MQTT). It is important to secure device communication. This can sometimes only be accomplished outside the protocol itself via “Defense in Depth” strategies (e.g. “encapsulation” via secure transport).

OPC UA Communication

The OPC UA standard offers built-in security that can be implemented with the server, client, or embedded directly on a device. One of the ways that this can be done is to encrypt all communication. Different devices/servers support different encryption levels. When possible, enable SignAndEncrypt security mode to encrypt all data sent over OPC UA.

The Ignition OPC Server requires trusting remote certificates for all secured inbound and outbound connections. Under **OPC UA** > **Security** on the gateway webpage, trusted certificates can be imported and quarantined certificates can be marked as trusted. Some third-party OPC Servers may require additional steps such as manually adding the client certificate.

OPC UA connections also support user authentication. Adhere to organizational standards with password complexity and change requirements. The Ignition OPC UA server includes a default "opcua-module" user source that contains one user with default credentials. This default user source allows the Ignition Gateway to initially connect as a UA client to its own UA server.

The best practice is to update credentials from default in the OPC UA Server Settings. This requires a matching update to the Ignition OPC UA Clients server loopback connection.

MQTT

MQTT includes built-in security features. It is recommended that data transferred between the Publisher and Broker (MQTT Transmission), as well as between the Broker and Subscriber (MQTT Engine), are configured with certificate-based TLS connections. In addition to this encryption, Username/Password Authentication is also supported and can be utilized to protect the data. MQTT also supports Access Control Lists (ACLs) which limit user access based on topic namespace. These security measures should be implemented whether the broker is local or hosted in the cloud. For more information on securing MQTT communication, see the Cirrus Link MQTT secure communication instructions. The same MQTT security strategy and standard should be applied with outside vendor hardware, software, or services.

Native Device Communication

Most native OT device protocols are weak or insecure by modern standards. Communication between Ignition and OT devices should be protected. Consider using available security options based on the protocol and hardware.

Many devices do not support certificates, encryption, or even authentication. Strong network segmentation is recommended according to industry accepted standards (e.g. IEC 62443) and best practices. It is often possible to encapsulate insecure traffic in a secure tunnel (e.g. VPN). In addition, many environments separate private OT networks. It is recommended to apply an overall “defense in depth” strategy when developing a secure network architecture. Ignition can participate well in these architectures, especially with multiple connected gateways. For example, Ignition Edge on an IPC can act as an edge-of-network node that can segment a vulnerable device from other network segments.

There are many architecture options and IT/OT technologies. It is important to choose appropriate security trust boundaries for the application. It is important to include security and architecture professionals to help identify best options.

Step 5: Identity and Access Management

When securing an Ignition-based application, consider both authentication and authorization. Authentication determines who is logging in, whereas authorization determines their privileges. Ignition has the following tools to help satisfy almost any kind of authentication and authorization strategy required:

- User Identification and Authentication: User Sources and Identity Providers
- Authorization: Role-Based Access Control, Location-Based Access Control through Security Zones
- Security Levels: a hierarchical, inheritance-based access control model which builds off of Roles, Security Zones, and other attribute sources

User Identification and Authentication

Ignition manages users through Identity Providers (IdP). Ignition has a built-in IdP, but can also connect to third-party IdPs such as Okta, Duo, and ADFS via SAML or OpenID Connect.

Ignition Identity Provider

The Ignition IDP supports three main user sources: [Internal Authentication](), [Database Authentication](), and [Active Directory Authentication]().

Internal Authentication

From the Gateway Web Page, users can be managed directly within Ignition. These users are local to the Ignition Gateway where they are defined. This default strategy is typically for smaller installations.

Database Authentication

The Database Authentication type uses an external database instead of storing data inside Ignition. Managing users is done via direct interaction with the database. This strategy is typically used for convenience, not the strongest possible security pattern (e.g. external Identity Providers).

Active Directory Authentication

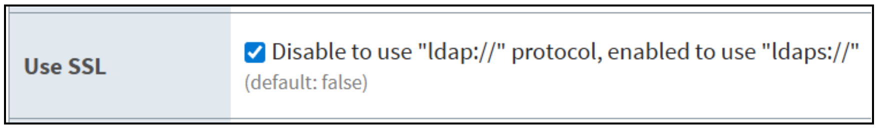
Enable Secure Lightweight Directory Access Protocol (LDAP) when using LDAP for Active Directory authentication. This is not the default setting.

The Active Directory Authentication profile uses Microsoft's Active Directory over LDAP to store all the users, roles, and more that make up an Authentication profile. Active Directory Groups are used for Ignition's roles and user-role mappings.

While using an Active Directory User Source, administration of users and roles is through Active Directory itself, and not manageable within Ignition. Thus, adding new users to an Active Directory User Source or modifying pre-existing users requires the modifications be made from Active Directory, usually through an Active Directory Administrator.

Using the Perspective Module, if a seamless experience is desired with no login prompt, consider using ADFS or another IdP instead of Ignition's internal IdP. See the Third-Party Identity Provider (Perspective) section below.

[LDAP Protocol Security]()
The Active Directory User Source communicates with a Microsoft Active Directory server through the LDAP protocol. To prevent snooping on authentication, encryption should be implemented. In the advanced options for a new Active Directory User Source, Ignition has a setting to force LDAPS.


Use SSL — ✓ Disable to use "ldap://" protocol, enabled to use "ldaps://"
(default: false)

Badge Authentication

Another secure method is RFID authentication support for the Ignition Identity Provider allowing you to associate badges with users. Entering your credentials on a screen allows others to see your username and therefore weakens your security. With RFID enabled, users do not have to enter their username and password, and instead must have a physical badge in order to log in to the Session. If your organization makes use of RFID badges, it is recommended that the Badge Authentication Method is enabled and set to default and Badge Secret is also enabled. Badge Secret will require the user to type in their password after scanning their badge. This added layer of security is useful in numerous situations. For example, in the event that a user's badge is stolen and used by another individual, the added layer of a required password would keep the thief from accessing the Session, since a password will still be required for access.

More information on the Badge Authentication Method can be found on our website.

Third-Party Identity Provider

Utilizing an external service allows you to utilize a company's existing IdP infrastructure, leveraging existing investments in web application security. External services also allow features that Ignition doesn't support natively such as multi-factor options like push notifications or biometrics. They generally are easy to tie in if IT has standardized other applications on a particular SSO solution.

Third party IdPs are supported for all major authentication and authorization areas of Ignition, including Perspective Clients, Vision Clients, the Designer, and the Gateway Webpage.

General Authentication Suggestions

User Accounts

To ensure User Account integrity, a strong password policy should be defined including password length and complexity requirements. Establishing a password expiration schedule and quickly removing former user accounts are strongly recommended. When using Ignition's Internal Authentication, these settings can be found under the Password Policy section. Generic accounts and password sharing should be avoided. For more security best practices, please refer to https://www.cisa.gov/shields-up.

Group Access and Disabling Auto Login

Generic logins pose a security risk in any system. If Auto Login is enabled, any user that launches a project is granted basic access. To mitigate this risk, Auto Login should be disabled and each user should have their own unique credentials.

Authorization

Once a user is authenticated, authorization determines what they have access to. Authorization can be determined by a variety of conditions including roles and location. When creating new users, it is a best practice to only create users for those who will need access to the environment. Every user credential should be unique to the particular user, avoiding shared or easy to guess usernames and passwords. Credentials should also be unique to the platform and not reused among other platforms. It is also a best practice to exercise the Principle of Least Privilege (PoLP), which requires that a user must be able to access only the information and resources that are necessary for the user's purpose. This will benefit the stability and security of the Ignition environment as users are limited to the scope that they have clearance for and possibly have familiarity with. Examples include preventing unapproved users from causing Gateway and/or project level modifications that can have adverse consequences to the Ignition environment or process.

Role-Based Security

Each user is granted privileges by assigning one or many roles. Roles are not default types but rather created custom during development. Roles can be defined inside Ignition or mapped to Active Directory groups or an IdP's attributes. The level of access granted by a given role is determined in Step 6: Implement Ignition-based Application Security.

Location-Based Security

A Security Zone is a list of Gateways, Computers, or IP addresses that are defined and grouped together. This group now becomes a zone which can have additional policies and restrictions placed on it. While Users and Roles restrict access to specific functions within the gateway, such as making certain controls read-only for certain users and read/write for others, Security Zones provide this functionality to the Gateway Network, location-based access control in Vision, Perspective, Alarm Notifications and Status, and History Provider and Tag Access. Security Zones allow the application to restrict access based on where the client connects from in addition to a user's role-based privileges. This allows for greater control over the type of information that is passing over the network, improving security and helping to keep different areas of the business separate, while still allowing them to interconnect.

Using Security Zones

Sometimes, in addition to knowing who the user is, it is important to know where they are sending a command from. An operator may have permissions to turn on a machine from an HMI, but if they were to log in to a project on a different Gateway in the network that had remote access to those tags, it might not be a good idea to let the operator write to those tags from a remote location where they can't see if the physical machine is clear to run.

This is where Security Zones come in. Security Zones themselves don't define the security, they instead define an area of the Gateway Network, breaking up Gateways and network locations into manageable zones that can then have a security policy set on them. Once there are zones defined, a security policy can be assigned to each zone, and a priority of zones can be set in the event that more than one zone applies in a given situation.

Information on how to set up Security Zones can be found on our website.

Security Levels

Security Levels are a user-defined hierarchy used to define roles to manage access permissions. This authorization system can be used to map roles managed in an Identity Provider (IdP) to Ignition roles. Creating security levels can be used to restrict certain people from being able to access specific functions within the gateway, such as access to certain Perspective sessions, visibility of components in a view, or read/write permissions. This will improve security and allow better management of the information and level of control specific users are granted on the network.

Using Security Levels

When working with the Gateway, it is important that all users are able to get the information or control they need in order for operations to run smoothly. An operator may need to be able to read and write to tags, and view the status of each plant. However, a manager may need to view the status of the plant and read the tag values, but should not be able to write to the tag values. Making sure that each user has the correct permissions is vital to the security of the operation.

With security levels, roles can be defined in order to allow certain users more or less control of the Gateway in flexible ways allowing granular access to specific security zones, projects, and other attributes. These roles can also be used in the Ignition Designer in order to limit a role's viewing access and/or read/write capabilities.

Security Levels are defined in the Gateway and arranged in an inherited tree structure. Each child (nested) level of the tree inherits the security of its parent levels. Consider a powerful permissions model where all Employees can access a screen in read-only mode, but only Plant-Floor Operator Employees can read / write. Since Plant Floor Operators are Employees, they can do everything all other Employees can do plus the specific things that only Plant Floor Operator Employees can do.

Tip: Complex Security Levels Rules can be created using Ignition's expression language, where a Security Level can be derived from the Authentication / Authorization context. This context can take into account who you are (logged in user identity / profile attributes), where you are (security zones assigned to you), what roles you have, the current time of day (in case users can only log into a system between the hours of 8am-5pm PT, for example), tag state representing the physical system from PLCs, or many other complex factors.

Information on how to set up Security Levels can be found online.

Step 6: Implement Ignition-Based Application Security

Ignition is a software platform for creating custom applications to suit your needs. These applications could be for HMI (Human Machine Interface), SCADA (Supervisory Control And Data Acquisition), Database Front End and more. Each of the applications require customized security settings based on your requirements. Ignition allows for security to be implemented at many levels (e.g. clients, projects, objects, actions, and even individual tags).

Ignition Vision Client Security

In Clients, security settings can be applied to individual windows or components. While users with different roles may view the same project from the client, the functionality and accessibility can change based on their assigned roles. Generally, higher-level access provides full functionality to all contents of a project, and lower level access is restricted to generalized read-only privileges. However, client security settings are flexible enough to accommodate security requirements.

Information on how to set-up Client Security can be found on our website.

Ignition Perspective Session Security

In Sessions, security is managed through Identity Providers (IdP). Ignition can either use an internal user source for an internal Identity Provider or leverage an external one. External Identity Providers offer a way for users to log into Ignition using credentials that are stored outside of Ignition and are great for solutions that require secure user access since many allow extra security functionality like Single Sign-On and Multi-Factor Authentication. Once the identity provider is set up in the Gateway, a security level can be assigned to a Perspective Session, which will grant only the users with the specified security level access to the Session. Generally, the higher-level access provides full functionality to all components of the project and lower level sets more restrictions, such as read/write privileges.

Information on how to set up Perspective Session Security can be found on our website.

Perspective Views Security

Added security to a Perspective View allows more granularity of the security in a Perspective Session. An operator, for example, may need to view a Perspective Session but shouldn't be allowed to access a user management View. The operator can be granted access to the Session, but the user management View can be restricted to a higher-level role. Adding security levels to both the Perspective Session and Views allows only the roles with proper privileges to be allowed to view or edit content and thus improving the security of the Perspective Session and control of information on a project.

Information on how to set up Perspective Views Security can be found on our website.

Component Scripting Security

Both Vision and Perspective contain role-based component scripting security to ensure that non-privileged users are not allowed to run scripts that can be potentially harmful to operations or manipulate information that a user does not have clearance for.

More information on Vision and Perspective component scripting on security can be found on our website.

Designer Security

When several users are all working on the same project, managing changes to the project can become cumbersome. By default, all users with Designer access can modify, delete, save, and publish all resources available in the Designer. In some situations, it is desirable to limit what each user can do in the Designer. Ignition has several built-in Designer restriction methods to help in these scenarios.

See project security in the Designer documentation for information about protecting Ignition resources.

Tag Security

Tag security is often the best way to configure security for data access. By defining security on a tag, you are applying those rules for that tag across all windows/views and components that use the specified tag in the project. This is opposed to configuring security on each component that controls the tag.

If a user opens a window or view that has components that are bound to a tag that the user doesn't have clearance to read or write to, the component will get a forbidden overlay.
You can add read/write security to individual tags through the Designer. Custom Access Rights must be set to set permissions based on Security Levels.

Named Queries

Use named queries when possible. It is a poor security practice to dynamically generate SQL query strings, especially with end user input. This could lead to issues such as: SQL injection attacks, performance problems, maintainability challenges, and possible unintended privilege escalation or remote code execution.

One of Ignition’s key features is the ability to easily log, edit and retrieve data from SQL databases. By default, all database interaction is limited to defined queries on the Ignition Gateway, which may be called from clients based on the credentials of the user. These queries can be parameterized to allow for dynamic database interaction while ensuring only relevant data is accessible. It is recommended to only use parameters for individual variables. Another good practice can be limiting the user input for these parameters when building the Vision or Perspective UI as much as possible, such as only allowing the user to select from a pre-populated list in a drop down versus allowing the input to be typed into a text field.

This feature can be turned off to allow any SQL query to be run directly from an open client. This practice should be discouraged on production systems. While this can be powerful for adding flexibility to the platform, it also leaves the data potentially exposed. If client-authored queries are enabled, be especially sure to use SSL and not use auto-login or any shared accounts.

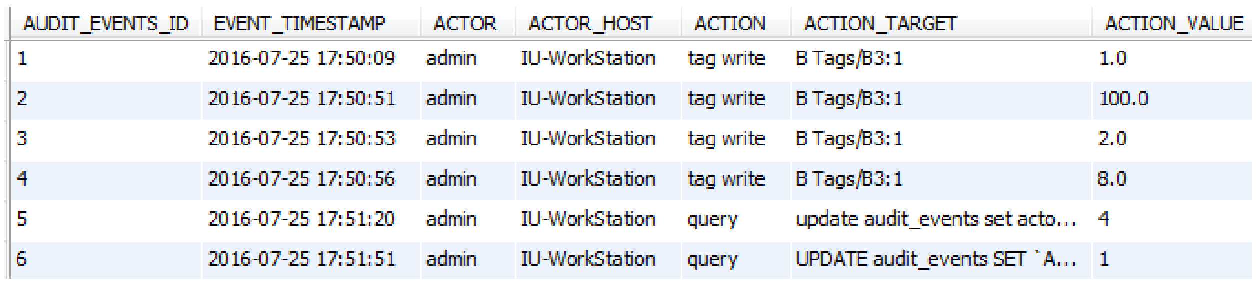
Access to named queries can be limited using the normal Ignition permission model including roles and security zones.

Step 7: Set Up Audit Logging

Enable Audit Logging and follow best practices and organizational standards regarding audit data. This feature records all user actions to a database. Use a dedicated audit database user with “APPEND ONLY” (e.g. SELECT, INSERT, not UPDATE, DELETE) privileges.

Audit Profiles allow Ignition to record details about specific events that occurred. Audit Profiles are simple to set up and immediately start to record events. By default, only tag writes, SQL UPDATE, SQL INSERT, and SQL DELETE statements are recorded. This allows you to keep track of which user wrote to which tag, or modified which table. Furthermore, a time-stamp is recorded, so you can easily track the changes, outline, and order of events.

Once some changes have been made to a tag or a database table, Ignition will begin recording.



AUDIT_EVENTS_ID	EVENT_TIMESTAMP	ACTOR	ACTOR_HOST	ACTION	ACTION_TARGET	ACTION_VALUE
1	2016-07-25 17:50:09	admin	IU-WorkStation	tag write	B Tags/B3:1	1.0
2	2016-07-25 17:50:51	admin	IU-WorkStation	tag write	B Tags/B3:1	100.0
3	2016-07-25 17:50:53	admin	IU-WorkStation	tag write	B Tags/B3:1	2.0
4	2016-07-25 17:50:56	admin	IU-WorkStation	tag write	B Tags/B3:1	8.0
5	2016-07-25 17:51:20	admin	IU-WorkStation	query	update audit_events set acto...	4
6	2016-07-25 17:51:51	admin	IU-WorkStation	query	UPDATE audit_events SET `A...	1

More Information regarding Audit Profiles can be found on our website.

Step 8: Protect Database(s)

Different databases offer different authentication options. We recommend not using a database owner account such as **root** or **sa**. A separate user account with limited privileges should be created for the database connection with the Ignition Gateway.

Limit Ignition access. Create separate Ignition database users and apply the principle of least privilege. This uses separate database connections in Ignition. For example, the Ignition Audit database user should have “APPEND ONLY” (e.g. SELECT, INSERT, not UPDATE, DELETE) privileges. A separate Ignition database account may need the full CRUD (Create, Read, Update, Delete) capabilities for records, but may not need database admin privileges, or even the capability to drop tables.

Secure network communication. Most modern databases also support SSL encryption of the connection between Ignition and the database. If the database is running on a different server, SSL can be enabled by following information available for your database’s JDBC driver and internal security settings. Refer to the documentation for your database for more information on enabling SSL JDBC connections from Ignition.

Step 9: Lock Down the Operating System (OS)

Ignition is a *distributed application development environment* that provides a powerful tool set to interface with physical systems and business applications. In order to properly secure Ignition it is important to understand how Ignition fits within the operating environment.

The best practice is for an Ignition Gateway to be the main/only application on a host. Apply strong segmentation. Isolation makes sense from a security perspective because Ignition has significant capabilities. The compromise of an Ignition host, even through other applications can lead to the compromise of Ignition capabilities.

We already restricted the Ignition service account in Step 2: Restrict Ignition OS and File System Permissions

Remove Unneeded Programs

Each program and dependency library introduce potential entry points for an attacker. Removing unnecessary software limits vulnerabilities. Programs should be run using the minimum credentials required, avoiding privileged users when possible. A strong approach to enforcing allowed software is called “application whitelisting”, where only approved applications are allowed to run and all others are disabled. Removing unnecessary programs is always a best practice, even with other security controls in place.

Runtime (Java) Security

Uninstall Java and .NET if possible. Ignition is bundled with a purpose-built Java runtime that is kept up-to-date by staying on a current version of Ignition. Unless required by other applications on an Ignition Gateway, it is recommended that externally installed runtimes such as Java and the .NET Frameworks be uninstalled to protect from known and future vulnerabilities. If absolutely needed, keep these runtimes up to date.

Patches and Service Packs

Keep up-to-date on OS patches and Service Packs. Align with IT and OT policies and procedures.

Remote Services

Remote services should be disabled unless managed as part of organizational IT or OT policy.

- On Windows, Remote Registry and Windows Remote Management should be disabled.
- On Linux and Mac OS, disable root for everything but the ‘physical’ console.

OS Firewalls, Network communication, and Endpoint Protection

This is mentioned in other steps. However, consider Operating System level security features in addition to external implementations as part of a “Defense in Depth” strategy. The Ignition service account needs limited access (described in other areas) to the file system and network ports. It is reasonable to use OS or third-party security controls.

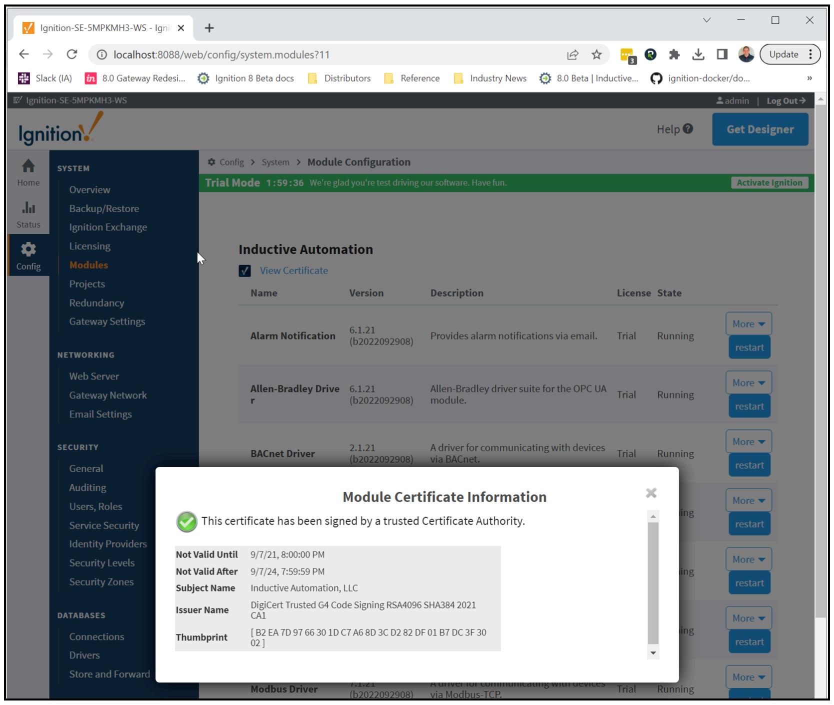
Step 10: Verify That Ignition Software is Genuine

Verify that Ignition software is genuine as part of your deployment and upgrade process. The Online User Manual provides Inductive Automation certificate information and instructions for verifying that Ignition software is genuine.

Verifying Trusted Code (Modules)

Ignition Modules will not run under normal configuration with self-signed, expired or untrusted certificates. Verify the authenticity of Ignition modules in the Gateway web page. Be careful and thoughtful with the use of third-party modules, including management of your own custom modules (as applicable).

Gateway Configuration Page -> Modules shows modules grouped by certificate (*View Certificate*).


Config > System > Module Configuration
Trial Mode 1:59:36 We're glad you're test driving our software. Have fun.
Activate Ignition
Get Designer

Module Certificate Information

This certificate has been signed by a trusted Certificate Authority.

Not Valid Until	9/7/21, 8:00:00 PM
Not Valid After	9/7/24, 7:59:59 PM
Subject Name	Inductive Automation, LLC
Issuer Name	DigiCert Trusted G4 Code Signing RSA4096 SHA384 2021 CA1
Thumbprint	[B2 EA 7D 97 66 30 1D C7 A6 8D 3C D2 82 DF 01 B7 DC 3F 30 02]

Step 11: Endpoint Detection and Response

Ignition can work successfully with Endpoint Detection and Response (EDR) technologies when they are implemented correctly. Endpoint protection is recommended, especially when aligned with company policy. EDR technologies can vary drastically in system impact, by product and configuration. Inductive Automation recommends testing in advance and monitoring performance. Disabling the agent can help determine the source of performance issues. It should be possible to achieve EDR protection and a performant Ignition system.

Networking

The Ignition Gateway needs to communicate on all ports necessary based on configuration. See the Ignition ports reference page of the manual for information on what ports are used for external communication: Ignition Port Reference | Ignition User Manual

From an Ignition perspective, other OS services and processes can be restricted in access to network communication.

File system

The Ignition service needs read/write access to the file system where Ignition is installed (reference Appendix A). The default install directories for Ignition on different operating systems can be found in the user manual here: Ignition Default Installation Directories

Ignition can be an I/O heavy application based on configuration and usage. Anti-virus scans on the install directories impact Ignition performance. Similarly, scans related to the Ignition process can impact performance. This should be considered when configuring these scans and their timing. In most cases, the Ignition installation directory should be excluded from persistent monitoring based on performance. Consider other approaches (e.g. scheduled scans) to protect these I/O intense areas. Consider the same for active memory resident protection agents.

Clients

On systems running Ignition client launcher applications (Vision Client Launcher, Perspective Workstation, and Ignition Designer) the ~/.ignition directory (or redirected directory) should allow executables such as java, 7zip, and chromium as well as jars etc. File-association custom icons on windows require the installation of a DLL when installing the launchers (this feature can be opted out of during launcher install if needed and the default icons are used) If planning to use deeplinks the protocols needed for the respective launcher should be allowed (perspective:// vision:// and designer://).

Tips

- When selecting "Cloud Anti-malware" rules, or as applicable, always keep "Detection" settings higher than "Prevention". Start low, monitor, then increase.
- If needed, some EDR agents allow "Sensor Visibility Exclusions" (creates a blind spot), which could be used with the Ignition installation directory if performance is unacceptable due to endpoint protection.
    - This is more useful for testing than production. It really should not be needed.
- In Dev/Test/QA setups when stress testing Ignition for disk performance disable "Detect/Quarantine on Write". Consider applicability for your production environment.

Step 12: Hardening the Environment

Firewalls and ports

Firewalls should be in place to restrict unneeded network traffic. The best practice is to close all ports by default and open those that are necessary. Ignition default Gateway ports are listed in the online user manual.

Redundant Servers

Firewalls must be set up on any server doing redundancy in order to protect the redundancy system from external attacks. For redundancy support, the firewall on the main server should be configured to accept connections on the Gateway Network port from the back-up server IP address. It is recommended to configure that firewall to reject Gateway Network port incoming connections from any other source unless Gateway Network traffic is expected from other Gateways.

Gateway Network

Ignition's Gateway Network communication is configured directionally, where one gateway is configured with an outgoing connection that is accepted as an incoming connection on the other gateway. When configuring this connection across different networks and/or a firewall it is recommended to configure the gateway in the more secure network/zone as the outgoing connection. This allows you to only need to configure outgoing firewall port rules to allow the Gateway Network communication across the firewall.

The Gateway Network was upgraded to Protobuf from Java Serialization in Ignition 8.3. This upgrade provides improved security for the Gateway Network. Though 8.3+ gateways can communicate with 8.1 gateways with the “Allow Java Serialization” option enabled, this is only recommended when needed for 8.3+ to 8.1 Gateway Network communication and should be left disabled otherwise for best security practice.

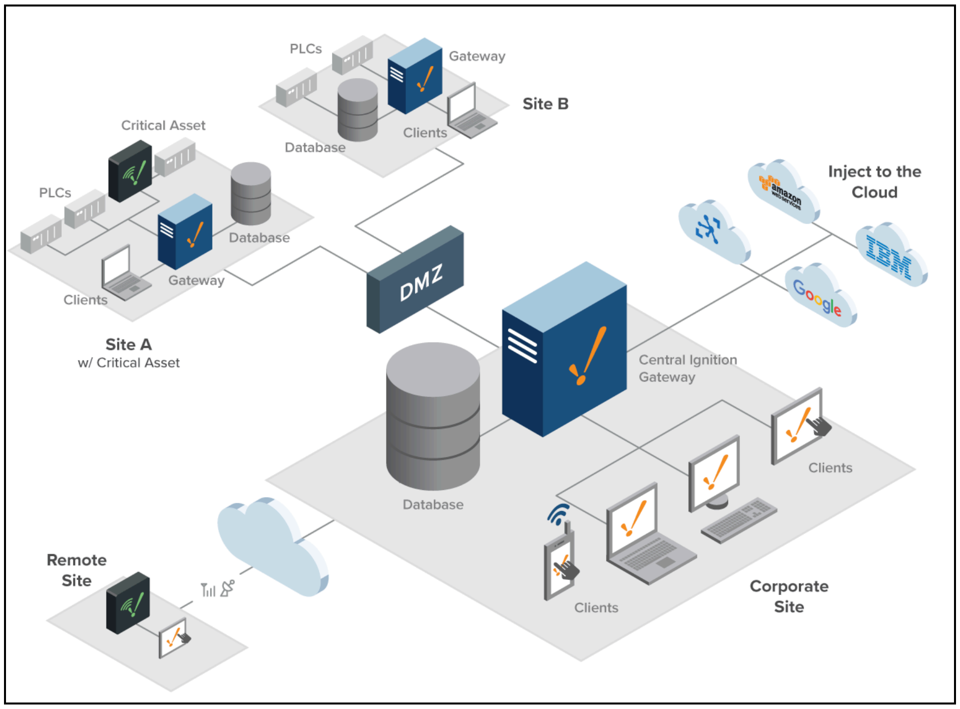
In 8.3+ Ignition Gateways, the Gateway Network defaults to “Require Two Way Auth” enabled in the Gateway Network settings. This requires manually moving the remote gateway’s cert to the local gateway in addition to approval of certificates. This was done as a best practice for security and should be left at default unless required to be changed.

DMZ Architecture

A DMZ Architecture (referred to as a “demilitarized zone”) contains a subnetwork that accommodates the organization’s exposed, outward connecting services. In general, it acts as a point of contact between the organization’s internal network and untrusted networks, such as the internet.

The goal of this architecture is to add an extra layer of security to the local area’s network, allowing the local network to access what is exposed in the DMZ and keeping the rest of the network protected behind a firewall.

We recommend consulting with a network security professional to help identify the best network options for your organization, and whether a DMZ is the right choice for your network topology.



Step 13: Stay Informed and Keep Ignition Up-To-Date

Inductive Automation recognizes that software security requires constant effort and maintenance. It is important to keep Ignition up to date to protect from known vulnerabilities.

1. Subscribe to Trust Center Updates on the Inductive Automation Security Portal
2. Keep Ignition up to date according to your company plan (e.g. after testing, might always "lag" a version)
3. Read release notes for new versions of Ignition (download portal).
4. Have an upgrade plan and follow a checklist
5. Keep up with informational resources
    a. Inductive Automation Forum and tech advisories
    b. Cybersecurity & Infrastructure Security Agency (CISA) *United States Computer Emergency Readiness Team* (US-CERT) site for advisories and recommendations.

Step 14: Secure the API

With the introduction of Ignition 8.3, a RESTful API for gateway information and configuration has been implemented on the Ignition platform. This also applies to the Web Dev Module, which allows user-defined API endpoints in Ignition 8.1 or 8.3.

Tokens

Any API endpoints that edit gateway settings through POST/PUT requests require a gateway generated token. The same applies to GET requests that touch critical gateway configuration information. Tokens are created in the Ignition Gateway and should be saved and maintained in a secure location, like a password manager or vault. Sensitive tokens should only be shared with those who need gateway API access. Always require secure HTTPS connections via token configuration in Production environments. The same is recommended for all environments.

Concurrency and Rate Limiting

Ignition does not broadly implement rate limiting. Inductive Automation recommends using external tools such as application load balancers or reverse proxies where feasible, especially for production and systems that are broadly network-accessible.

Step 15: Secrets Management

Ignition 8.3 introduces Secrets Management, a customizable system for securely storing and managing sensitive information, such as passwords and API tokens. Inductive Automation recommends the use of Secrets Management.

Secrets Management System Setup

Most users gain some benefit from this system without any effort. However, strong implementation requires customer key management, which includes significant disadvantages in the event of lost keys. Customers should not proceed with custom keys until getting to a point of high security and maturity.

While Ignition provides a default encryption key that is used to encrypt all secrets on the Gateway, it is recommended to establish a custom Secrets Management System for enhanced security since the default key matches every Ignition installation. This system involves creating a unique Root Key, an Encryption Key Set, and a dedicated environment password, all unique to your installation. This process creates new encrypted files that secure your secrets and are essential for rotating keys and decrypting data.

Modifying Root Keys includes trade offs. It requires customer management of keys. The advantage is that backups and Ignition data and resources are cryptographically protected at rest. Please thoroughly read the responsibilities and implications of the Secrets Management System in the User Manual. Inductive Automation, including Support, is unable to help recover secrets data without encryption keys. Note: this does not impact data externally written to databases, etc.

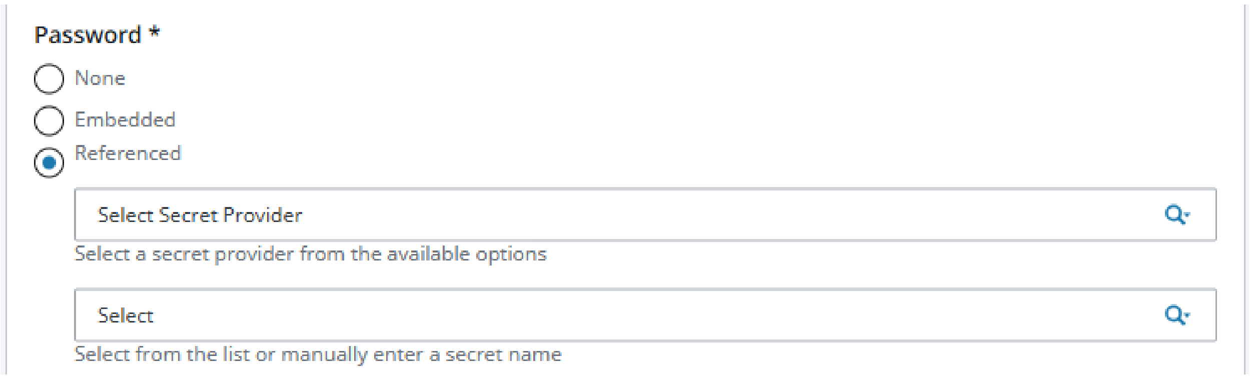
Key CLI Tool

The Secrets Management Key CLI tool is shipped with Ignition for users to directly manage the system's encryption keys from the command line. This tool exposes essential commands for tasks such as generating, resetting, removing, and rotating keys.

See the Secrets Management Key CLI Tool page for more information on the available commands.

Secret Input Methods

The Secrets Management system replaces traditional password fields across the Gateway, offering new options for securing system configurations, devices, and profiles that require a secret.


Password *

None [✓ Referenced] Embedded

Select Secret Provider

Select a secret provider from the available options

Select

Select from the list or manually enter a secret name

- **None**: This option confirms no password is required for the corresponding Gateway system (if applicable).
- **Embedded**: Secrets are entered directly into a field. Once saved, the secret is encrypted and stored within the specific configuration where it was specified. The encrypted secret is managed internally and can only be exposed using an encryption key.

- **Referenced**: The secret is retrieved by selecting a Secret Provider and referencing the stored secret’s name.

More information about Secret Input Methods can be found in our User Manual.

Secret Providers

Secret Providers allow users to utilize Ignition as a Secrets Management service through the use of referenced secrets. Currently, the Internal Secret Provider is the only type available. Secrets within the Internal provider are managed by Ignition using an embedded strategy, stored as files on the Gateway, and referenced by an assigned name to facilitate re-use. Upcoming 8.3.x+ Secret Provider support anticipates: HashiCorp Vault, followed by various other standards including Cloud providers.

Information on how to set up Secret Providers can be found in our User Manual.

Appendix A - Restrict the Ignition Service Security

It is important to apply the Principle of Least Privilege to the Ignition service account and OS file system. This reduces the potential impact of an accidental scripting error or malicious script executed or scheduled by a privileged user. A sophisticated cyberattack could attempt this vector even if you do not use scripting.

Tip: Keep in mind the need to grant OS permissions if Ignition requires such in the future. It is important to thoroughly test in your environment before deploying to production. Incorrectly applying these settings will often result in the Ignition service failing to start. The main "testing" step is simply starting the Ignition service. It can be useful to temporarily use the SYSTEM/root account while troubleshooting restrictions.

Recommended Windows Service Restrictions

A default Windows installation will run Ignition under the SYSTEM user in most environments. This grants Ignition excessive access to the OS and possibly network resources compared to what is normally needed.

The best practice is to create a dedicated local windows account (e.g. svc-ign) for the Ignition service.

- Without good reason (e.g. mature enterprise management), this should not be a domain account
  - If using a domain account, ensure proper restriction; work with IT / OT
- Remove all group memberships from the service account (including *Users* and *Administrators*).
- Set Ignition service to "Log on as" the dedicated account in the service properties (services.msc)
- Update security policy (Local Security Policy (secpol.msc) or Group Policy)
  - Add "Log in as a service" to security policy
  - Add "Deny log on locally" security policy

Restrict file system access.

- Grant the Ignition service account *Full control* access to Ignition installation directory (typically "Inductive Automation").
  - Restrict access to the Ignition directory. Note, the installer will need access to upgrade.
- Set deny access settings for the service account on other directories not needed by the Ignition service. Common examples below:
  - C:\Windows
  - Windows Temp folder
  - C:\Program Files(x86)

    - C:\ProgramData
    - C:\Users
- Change the Ignition temp directory in the **ignition.conf** file (\Ignition\data)
    - Incorrect settings will cause the Ignition service to fail to start. This is easy to recognize and undo.
    - Set temp directory #Java Additional Parameters. Example:
        - wrapper.java.additional.8=-Djava.io.tmpdir="C:/Program Files/Inductive Automation/Ignition/local/temp"
        - Note: the Java additional parameter number 8 and file path above are examples.
            - The file path is recommended as a new folder in the Ignition install directory.
        - Note: if the specified directory does not exist the service will fail to start. You likely need to create the folder.
    - Test start the Ignition service. A valid temp directory is required for Ignition to function.
        - If Ignition "breaks" (won't start), simply delete or comment out (start line with #) the above configuration line and correct the problem (likely tmpdir).
- Grant file system access as needed. For any other directories used by Ignition processes like automated gateway backups, core historian archival, and scripts that need to touch the file system, make sure those directories do not have access denied and that the specific type of access required (read, write, read/write, etc.) is given to the dedicated Ignition service account.

Recommended Linux Service Restrictions

The best security practice is to create a local Linux user for the Ignition service. This user should have no home directory or shell access. The command below creates an example user called svc-ign with these restrictions:

```
None
sudo useradd -M -s /sbin/nologin svc-ign
```

Ignition security in Linux environments can be improved within the systemd file for the Ignition service. To persist this systemd file configuration across upgrades, an override .conf file for the systemd file should be used. To create that override file:

- In /etc/systemd/system/ Create an Ignition-Gateway.service.d directory
- In the created Ignition-Gateway.service.d directory create a .conf file for the systemd Ignition service with the naming convention of your liking(ex. - 00-SecurityHardening.conf) file that contains the following (Where User is whatever you named the service account)

```None
#

[Service]
# Make user service account
User=svc-ign

# Truncate allowed capabilities
CapabilityBoundingSet=
AmbientCapabilities=

# Disallow gaining of new privileges at runtime
NoNewPrivileges=yes

# Setup a dedicated /tmp/systemd-... temp directory for the systemd unit
PrivateTmp=yes

# Setup an isolated /dev mount with only psuedo-devices; do not use this
if you need access to serial ports, for example
PrivateDevices=yes

# Mount root filesystem as read-only
ProtectSystem=strict
# Mount home directory with tmpfs overlay
ProtectHome=tmpfs
# Prevent changes to clock
ProtectClock=yes
# Protect kernel variables, read-only
ProtectKernelTunables=yes
# Prevent kernel module loading
ProtectKernelModules=yes
# Prevent use of cgroups
ProtectControlGroups=yes

# Allow read/write to Ignition install path
ReadWritePaths=/usr/local/bin/ignition

# Only allow INET and UNIX sockets
RestrictAddressFamilies=AF_INET AF_INET6 AF_UNIX

# Deny use of Linux namespaces
RestrictNamespaces=yes

# Lock down personality system calls
LockPersonality=yes
```

```
# Prevent setuid/setgid setting on files
RestrictSUIDSGID=yes

# Use UMask to set permissions for install directory
UMask=022
```

Ignition can also be installed as the service account using the user argument documented in the arguments table of the Ignition manual: [Arguments Table | Ignition User Manual]

In some environments Ignition may need access to directories outside of the install directory. For example, when using an automated gateway backup schedule, Core Historian archival, or purposefully reading/writing to designated file system areas via scripting. In these cases:

- Choose or create a directory dedicated to that purpose.
- Make the service account the owner of that directory (or set permissions as needed).
- Add that directory to the ReadWritePaths systemd unit in the .conf file. See example below:

```
None
#      Allow      read/write      to      Ignition      install      path
# and other directories Ignition needs
ReadWritePaths=/usr/local/bin/ignition               /var/gatewayBackups
/var/coreHistArch
```

Appendix B - Containerized Deployments

Inductive Automation publishes an OCI image for Ignition, allowing it to run on various container runtimes such as Docker, containerd, Podman, etc. See Ignition User Manual on Docker Image. This includes Github links, settings, and best practices. The Inductive Automation Forum is a good resource to discuss Containerized Deployment strategies with the Ignition user community.

Running containers in production requires extra scrutiny on the host system and container runtimes. Consider industry standard benchmarks as a baseline effort for hardening your container hosts and runtimes, such as the CIS Benchmark for Docker.

The following sections offer additional hardening options that can be applied to containerized deployments of Ignition.

Running as non-privileged user

The Ignition Docker image already defaults to run as a non-root user, which is important for ensuring that Ignition doesn't have more access than it needs. The `ignition` user within the container is defined as UID/GID `2003`. Ideally, this UID/GID doesn't overlap with users/groups on the host system.

You can also configure your container runtime to leverage user namespaces, such as in this linked example for Docker. This approach allows the container to believe it is a given UID (even `0` or `root`) when in reality it is executing on the host as an unprivileged user in a designated high-number UID/GID range, greatly reducing the impact of any container escape scenario.

Minimizing container capabilities

By default, containers already run with a limited set of privileges and capabilities. You can restrict the capabilities of the Ignition container even more in many cases, further reducing possible vulnerability vectors in your deployment.

Consider dropping all of the default capabilities and conditionally adding any capabilities that your use case specifically requires. For example, in Docker, you could supply `--cap-drop=ALL` to drop all capabilities and add any that are needed back in with `--cap-add=<key>`. Typically, you'll be able to drop more of the default capabilities without impact to Ignition.

Finally, you can also apply a setting to disable container processes from acquiring new privileges. In Docker, this is via the `--security-opt=no-new-privileges:true` flag in the container configuration.

Set resource limits

We typically recommend scheduling Ignition workloads to dedicated hardware. However, even in those situations, it may still be beneficial to set hard limits on memory usage for your Ignition containers. This can help prevent a denial-of-service situation where memory use grows unbounded and causes general system stability issues.

For example, Docker uses the `-m` flag to limit a container's access to memory. You can apply memory limits to the container itself and then configure the Java Virtual Machine (JVM) for Ignition to use a specified percentage of that limit for its heap memory allocations. With the memory limits applying to overall usage (JVM heap, direct buffers, and other container processes such as healthchecks), it is good to start with conservative figures to prevent premature Out-of-Memory (OOM) kills. The **correct percentages to use will vary** based on the specific projects running within Ignition.

To configure Ignition to use a specific percentage (e.g. `60` percent) of total container memory, supply the following JVM and Wrapper Args:

```
wrapper.java.initmemory=0
wrapper.java.maxmemory=0
-XX:InitialRAMPercentage=60
-XX:MaxRAMPercentage=60
```

⚠️ **WARNING:** Only use memory percentages if you *are* enforcing container memory resource limits. With no limits applied, the JVM will apply the percentage based on total host system memory, causing possible interference with other workloads.

Leverage image pinning

When using the Ignition container image, you should typically pin to specific version tags in your configuration. For example, use the image reference `inductiveautomation/ignition:8.3.1` where the image tag `8.3.1` is more specific than tags such as `8.3` or `latest`. Avoid using `latest` tags in production.

As a general practice, consider including the image digest in your image references (e.g. `inductiveautomation/ignition:8.3.1@sha256:8080ce4eb64b7e16e4ec90867d9ef52019dfca16f181879c00346058da2ab1cd`). When you use image references in this format, the specific digest is enforced and the tag is actually ignored. This helps you ensure that you're always using a specific image and gives you better audit capabilities on what images you're using.

Appendix C - Ignition Cloud Edition

When running Ignition Cloud Edition, managing security is a shared responsibility between the Cloud Service Provider (CSP), Inductive Automation, and the asset owner.

The CSP is responsible for security of the cloud. This means that the CSP protects and secures the infrastructure that runs the services offered by the CSP.

Customers are responsible for security in the cloud. When using any service provided by the CSP, you're responsible for properly configuring the service and your applications, including Ignition, in addition to ensuring that your data is secure.

Inductive Automation is responsible for providing security updates via Ignition software updates. Inductive Automation has a process in place to make sure that customers get the latest security updates through various means, such as nightlies, critical updates, and the release train.

More information regarding the AWS Shared Responsibility Model can be found on the AWS website.

More information regarding the Azure Shared Responsibility Model can be found on the Azure website.