

Splunk® Enterprise 6.4.0

Securing Splunk Enterprise

Generated: 5/11/2016 6:31 pm

Table of Contents

About securing Splunk Enterprise.....	1
About securing Splunk Enterprise.....	1
Ways you can secure Splunk.....	1
Install Splunk securely.....	3
Install Splunk Enterprise securely.....	3
Secure Splunk Enterprise on your network.....	3
Some best practices for your servers and operating system.....	4
Harden your configuration.....	5
Disable unnecessary Splunk Enterprise components.....	5
Users and role-based access control.....	7
Use access control to secure Splunk data.....	7
About user authentication.....	7
About configuring role-based user access.....	8
About defining roles with capabilities.....	10
Add and edit roles with Splunk Web.....	14
Add and edit roles with authorize.conf.....	16
Setting access to manager consoles and apps.....	19
Find existing users and roles.....	22
Delete all user accounts.....	22
Secure access for Splunk knowledge objects.....	23
Native Splunk Enterprise authentication.....	24
Set up user authentication with Splunk Enterprise native authentication.....	24
Configure users with Splunk Web.....	24
Configure users with the CLI.....	25
Add a user to a role with Splunk Web.....	26
Authentication with LDAP.....	27
Set up user authentication with LDAP.....	27
Manage Splunk user roles with LDAP.....	28
LDAP prerequisites and considerations.....	29
How Splunk works with multiple LDAP servers.....	30
Configure LDAP with Splunk Web.....	31
Map LDAP groups to Splunk roles in Splunk Web.....	38
Configure LDAP with the configuration file.....	39
Map LDAP groups and users to Splunk roles in the configuration files....	41

Table of Contents

Authentication with LDAP	
Test your LDAP configuration.....	42
Convert to LDAP from Splunk's built in authentication.....	43
Best practice for removing an LDAP user.....	44
Authentication using single sign-on with SAML.....	45
Configure single sign-on with SAML.....	45
Configure SSO with PingIdentity as your identity provider.....	46
Configure SSO with Okta as your identity provider.....	48
Configure SSO with AzureAD or ADFS as your Identity Provider.....	51
Map SAML groups to roles.....	53
Configure SAML SSO in the configuration files.....	54
Troubleshoot SAML SSO.....	57
Authentication using single sign-on with reverse proxy.....	59
About Single Sign-On using reverse proxy.....	59
Configure Single Sign-On with reverse proxy.....	62
Troubleshoot reverse-proxy SSO.....	65
Scripted authentication.....	67
Set up user authentication with external systems.....	67
Create the authentication script.....	68
Edit authentication.conf.....	71
Use PAM authentication.....	73
Use the getSearchFilter function to filter at search time.....	73
Securing Splunk Enterprise communications with SSL.....	75
About securing Splunk with SSL.....	75
About using SSL tools on Windows and Linux.....	78
Configure allowed and restricted SSL versions.....	79
Secure browser to Splunk Web communication.....	82
About securing Splunk Web.....	82
Turn on encryption (https) with Splunk Web.....	83
Turn on encryption (https) using web.conf.....	84
Secure Splunk Web with your own certificate.....	84
Troubleshoot your Splunk Web authentication.....	86

Table of Contents

Secure Splunk forwarder to indexer communication.....	88
About securing data from forwarders.....	88
Configure Splunk forwarding to use the default certificate.....	89
Configure Splunk forwarding to use your own certificates.....	91
Other configuration options.....	93
Validate your configuration.....	95
Troubleshoot your forwarder to indexer authentication.....	96
Secure distributed environments.....	98
About securing inter-Splunk communication.....	98
Securing distributed search heads and peers.....	98
Secure your deployment server and clients.....	98
Secure your clusters.....	100
Audit Splunk Enterprise activity.....	101
Use Splunk to audit your system activity.....	101
Search for audit events.....	101
Manage data integrity.....	102
Best practices for Splunk Enterprise security.....	104
Secure your service accounts.....	104
Deploy secure passwords across multiple servers.....	104
How pass4symkey hardens your configuration.....	105
Harden your KV store port.....	105
Use Splunk Enterprise Access Control Lists.....	106
Safeguards for risky commands.....	107
Appendix A: How to get SSL certificates.....	110
How to self-sign certificates.....	110
Next steps.....	114
How to get certificates signed by a third-party.....	114
How to prepare your signed certificates for Splunk.....	118
Self-sign certificates for Splunk Web.....	120
Get certificates signed by a third-party for Splunk Web.....	125
Determine your cipher suite.....	129

About securing Splunk Enterprise

About securing Splunk Enterprise

As soon as you set up and begin using your new Splunk Enterprise installation or upgrade, you should perform a few additional steps to ensure that Splunk Enterprise and your data are secure. Taking the proper steps to secure Splunk Enterprise reduces its attack surface and mitigates the risk and impact of most vulnerabilities.

To get started, see "[Ways you can secure Splunk](#)" in this manual.

A note about common criteria for Splunk

Splunk has submitted Splunk Enterprise 6.4.0 for Common Criteria evaluation. Until we are common criteria certified we do not recommend nor support configuring Splunk Enterprise in common criteria mode.

Ways you can secure Splunk

This manual documents some of the many ways to secure your Splunk configuration and describes how to do the following:

- Set up **user authentication** to create users and manage user access by assigning **roles**.
- Use SSL to secure communications for your Splunk configuration.
- Use **auditing** to secure your data.
- Harden your Splunk instances to reduce vulnerability and risk.

User authentication and role-based access control

Set up users and use roles to control access. Splunk allows you to configure users in three ways:

- Splunk's own built-in system, described in "[Set up user authentication with Splunk's built-in system.](#)"
- LDAP, described in "[Set up user authentication with LDAP.](#)"
- A scripted authentication API for use with an external authentication system, such as PAM or RADIUS, described in "[Set up user](#)

[authentication with external systems.](#)"

Once you've configured users you can assign roles that determine and control capabilities and access levels. For more information about roles and capabilities, read "[About role-based user access.](#)"

Encryption and authentication with SSL

Splunk comes with a set of default certificates and keys that, when enabled, provide encryption and data compression. You can also use your own certificates and keys to secure communications between your browser and Splunk Web as well as data sent from forwarders to a **receiver**, such as an indexer.

Under certain conditions, you can also secure communications in distributed search environments, configuration data sent to clients by a deployment server, and communications from Splunk Web to splunkd.

For more information about securing Splunk communications with SSL, see "[About securing Splunk with SSL](#)" in this manual.

Auditing

Keep an eye on activities within Splunk (such as searches or configuration changes).

- **Monitoring**
You can use monitoring to index all `.conf` files using a white list as a best practice, and for user attribution of changes, you can use the file system-based monitoring available out of the box on most Splunk-supported operating systems.
For more information about monitoring, see "Monitor Files and Directories" in the Getting Data In manual.
- **Audit events**
Watch your Splunk instance by monitoring audit events. Audit events are generated whenever anyone accesses any of your Splunk instances -- including any searches, configuration changes or administrative activities. Each audit event contains information that shows you what changed where and when and who implemented the change. Audit events are especially useful in distributed Splunk configurations for detecting configuration and access control changes across many Splunk Servers. To learn more, see "[Audit Splunk Enterprise activity](#)" in this manual.

Install Splunk securely

Install Splunk Enterprise securely

Take the following steps when downloading and installing Splunk

Verify Integrity

Verify your Splunk Enterprise download using hash functions such as Message Digest 5 (MD5) and SHA-512 to compare the hashes. Be sure to use a trusted version of OpenSSL. For example:

```
./openssl dgst -md5 <filename-splunk-downloaded.zip>
```

or

```
./openssl dgst -sha512 <filename-splunk-downloaded.zip>
```

Verify Signatures

You can verify the authenticity of the downloaded RPM package using our GnuPG Public key as follows

1. Download the GnuPG Public key file (yes, this link is over TLS).
2. Install the key using:

```
rpm --import <filename>
```

3. Verify the package signature using:

```
rpm -K <filename>
```

Secure Splunk Enterprise on your network

Under certain conditions, Splunk Enterprise ports can become susceptible to attacks. Prevent access by shielding your Splunk Enterprise configuration from the Internet.

If possible, use a host-based firewall to restrict access to Splunkweb, management, and data ports. Keep Splunk Enterprise within a host-based firewall. Have your remote users access Splunk Enterprise on a Virtual Private Network.

You also can protect Splunk Enterprise from attacks in the following ways:

- Restrict CLI security by restricting this port to local calls only, from behind a host firewall.
- Unless necessary, do not allow access to forwarders on any port.
- Install Splunk Enterprise on an isolated network segment that only trustworthy machines can access.
- Limit port accessibility to only necessary connections. The necessary connections are:
 - ◆ End users and administrators must access Splunkweb (TCP port 8000 by default).
 - ◆ Search heads must access search peers on the management port (TCP port 8089 by default).
 - ◆ Deployment clients must access the deployment server on the management port (TCP port 8089 by default).
 - ◆ Forwarders must access the index server data port (TCP port 9997 by default).
 - ◆ Remote CLI calls use the management port.

Some best practices for your servers and operating system

Operating System

Splunk strongly recommends hardening all Splunk server operating systems.

- If your organization does not have internal hardening standards, Splunk recommends the CIS hardening benchmarks.
- As a minimum, limit shell/command line access to your Splunk servers.

Splunk

- Configure redundant Splunk instances, both indexing a copy of the same data.
- Backup Splunk data and configurations, regularly.

- Execute a periodic recovery test by attempting to restore Splunk from backup.
- Verify your Splunk download using a hash function such as MD5 to compare the hashes. For example:

```
./openssl dgst md5 <filename-splunk-downloaded.zip>
```

Client browser

- Use a current version of a supported browser, such as Firefox or Internet Explorer.
- Use a client-side JavaScript blocker such as noscript on Firefox or Internet Explorer 8 Filters to help protect against XSS, XSRF, and similar exploits.
- Ensure that users have the latest Flash version installed.

Physical security

- Secure physical access to all Splunk servers.
- Ensure that Splunk end users practice sound physical and endpoint security.
 - ◆ Set a short time-out for Splunk Web user sessions. See Configure timeouts for more information.

Harden your configuration

Consider the following opportunities to secure your configuration:

- Use a configuration management tool, such as subversion, to provide version control for Splunk configurations.
- Integrate Splunk configuration changes into your existing change management framework.
- Configure Splunk to monitor its own configuration files and alert on changes.

Disable unnecessary Splunk Enterprise components

For single-server Splunk Enterprise deployments:

- Forwarders should not run Splunkweb and should not be configured to receive data on TCP or UDP ports or from other Splunk Enterprise instances.

For multiserver Splunk Enterprise deployments:

- Search heads should not receive data on TCP or UDP ports or from other Splunk Enterprise instances.
- If users are not logging in to Splunkweb on indexers in a distributed environment, Splunkweb should be disabled on the indexers.
- Forwarders should not run Splunkweb and should not be configured to receive data on TCP or UDP ports or from other Splunk instances.

Users and role-based access control

Use access control to secure Splunk data

This page is currently a work in progress; expect frequent near-term updates.

Role-based access control provides flexible and effective tools that you can use to protect Splunk data.

Splunk Enterprise masks data to the user is much like the way a relational database manages role-based access control. In some cases total segmentation of data may be necessary. In other cases, controlling the searches and results at the presentation layer (something you can do with many of our Splunk Apps) may meet your security needs.

Consider your use cases when deciding how to set up your configurations and whether role-based access might fit your needs. For example:

- For extremely sensitive data, where even allowing access to a system that might have sensitive data incurs legal risk, consider installing and configuring more than one instance of Splunk Enterprise, and then configuring each instance with the data for the appropriate audience.
- When intentionally or unintentionally exposing sensitive data to the wrong user might incur legal ramifications, then consider creating indexes specifically for privileged and non-privileged accounts and assigning them to roles created for each level of access.
- When there are security concerns but not so much legal risk, you can restrict access using Apps. For example, you can create an App with static dashboards and assign roles with lower clearance to those dashboards, limiting the type of information the user assigned to the role may access.
- Field encryption, search exclusions, and field aliasing to redacted data are also great ways to tighten up a limited search case.

About user authentication

Splunk Enterprise authentication allows you to add users, assign them to **roles**, and give those roles custom permissions as needed for your organization.

Splunk ships with support for three types of authentication systems:

- Splunk's own built-in system: In addition to Splunk's default roles of **Admin**, **Power** or **User**, you can define your own roles using a list of **capabilities**. Splunk with an Enterprise license comes with Splunk's built-in authentication enabled by default. See "[Set up user authentication with Splunk's built-in system](#)" for more information.
- LDAP: Splunk supports authentication with its internal authentication services or your existing LDAP server. See "[Set up user authentication with LDAP](#)" for more information.
- Scripted authentication API: Use scripted authentication to tie Splunk's authentication into an external authentication system, such as RADIUS or PAM. See "[Set up user authentication with external systems](#)" for more information.

Note: Authentication, including native authentication, LDAP, and scripted authentication, is not available in Splunk Free.

You can create and assign users to flexible roles either in Splunk Web or by editing `authorize.conf`. For more information about roles and capabilities, read "[About role-based user access.](#)"

Important: Splunk's built-in system always takes precedence over any external systems. This is the order in which Splunk authenticates a user:

1. Splunk built-in authentication
2. LDAP or Scripted authentication (if enabled).

About configuring role-based user access

If you're running Splunk Enterprise, you can create users with passwords and assign them to **roles**. Roles determine the access and **permissions** of any user assigned to that role.

For more information about users, see "[About user authentication.](#)"

By default, Splunk comes with the following roles predefined:

- **admin**: this role is intended for administrators who will manage all or most of the users, objects, and configuration and comes predefined with the most assigned **capabilities**.
- **power**: this role can edit all shared objects (saved searches, etc) and alerts, tag events, and other similar tasks.
- **user**: this role can create and edit its own saved searches, run searches, edit its own preferences, create and edit event types, and other similar tasks.
- **can_delete**: This role allows the user to delete by keyword. This capability is necessary when using the delete search operator.
- **sc_admin (Cloud only)**: This role allows users to create users and roles but does not grant any other admin capabilities.

You can also create custom roles and assign your users to those roles. When you create a custom role, you determine the following:

- **Allowed searches**: you can define the searches that an user assigned to the role is allowed to perform.
- **Role inheritance**: you can have your role inherit certain properties of one or more existing roles. Role inheritance is discussed later in this topic.
- **Assign capabilities**: you can specify the allowed actions (change their password, change forwarder settings, etc) of the user assigned to the role. See "[About defining roles with capabilities](#)" for more information.
- **Set allowed and default indexes**: you can limit access to specific indexes and set the index that is searched by default.

To create roles in Splunk Web, see "[Add and edit roles with Splunk Web](#)". To create roles by editing `authorize.conf`, see "[Add and edit roles with authorize.conf](#)"

Inheritance

As a rule, members of multiple roles inherit properties from the role with the broadest permissions.

How users inherit search filter restrictions

You can create roles that inherit the characteristics of other roles. Users assigned to multiple roles inherit properties from the assigned roles.

In the case of search filters, if a user is assigned to roles with different search filters, the filters are all combined and thus the restrictions of each role are applied.

For example, by default, the Power and User roles do not have search filters defined to restrict searches. If a user has a combination of these roles and another role with filters defined (for example, srchFilter=x), the user will inherit the restrictions of that role, despite the association with roles that have no filter.

How users inherit allowed indexes

In the case of allowed indexes, the user is given the highest level of access granted to any role to which they are assigned.

For example, if a user is assigned to the role "simple user" which limits access to one particular index, and also to a role "advanced user" which has more capabilities and allows access to all indexes, the user will have access to all indexes. If you wanted to grant the capabilities of the "advanced user" but continue to limit their index access to the single index defined for the "simple user", you should create a new role specifically for that user.

How users inherit capabilities

In the case of capabilities, the user is given the highest level of abilities granted to any role to which they are assigned.

For example, if a user is assigned to the role "admin" which has the most capabilities, and also to a role "advanced user" which a different set of capabilities, the user will have the capabilities of both roles.

About defining roles with capabilities

When you create a user in Splunk Web you assign that user to one role. See ["About role-based user access"](#) for more information.

Each role contains a set of **capabilities**. You can add or edit capabilities for new, existing, and default roles. For example, you might give a role the capability to add inputs or edit saved searches.

To add or change the capabilities to a role in Splunk Web, see ["Add and edit roles with Splunk Web."](#) To create roles by editing authorize.conf, see ["Add and edit roles with authorize.conf."](#)

List of available capabilities

This list shows the capabilities that you can add to any role. Check `authorize.conf` for the most up-to-date version of this list. The admin role has all the capabilities in this list except for the "delete_by_keyword" capability.

Capability name	What it lets you do
<code>accelerate_datamodel</code>	Enable or disable acceleration for data models.
<code>accelerate_search</code>	Enable or disable acceleration for reports. For a role to use this it must also have the <code>schedule_search</code> capability.
<code>admin_all_objects</code>	Access and modify any object in the system (user objects, search jobs, etc.). (Overrides any limits set in the objects.)
<code>change_authentication</code>	Change authentication settings and reload authentication.
<code>change_own_password</code>	User can change their own password.
<code>delete_by_keyword</code>	Use the "delete" operator in searches.
<code>edit_deployment_client</code>	Change deployment client settings.
<code>edit_deployment_server</code>	Change deployment server settings.
<code>edit_dist_peer</code>	Add and edit peers for distributed search.
<code>edit_forwarders</code>	Change forwarder settings.
<code>edit_httpauths</code>	Edit and end user sessions.
<code>edit_input_defaults</code>	Change default hostnames for input data.
<code>edit_monitor</code>	Add inputs and edit settings for monitoring files.
<code>edit_roles</code>	Edit roles and change user/role mappings.
<code>edit_roles_grantable</code>	<p>Edit roles and change user/role mappings for a limited set of roles.</p> <p>A role with the <code>edit_user</code> capability can assign any role to other users. To limit this ability, also assign the <code>edit_roles_grantable</code> capability and configure <code>grantableRoles</code> in <code>authorize.conf</code>. For example: <code>grantableRoles = role1;role2;role3</code></p>
<code>edit_scripted</code>	Create and edit scripted inputs.

<code>edit_search_head_clustering</code>	Edit search head clustering settings.
<code>edit_search_server</code>	Edit general distributed search settings like timeouts, heartbeats, and blacklists.
<code>edit_server</code>	Edit general server settings like server name, log levels, etc.
<code>edit_splunktcp</code>	Change settings for receiving TCP inputs from another Splunk instance.
<code>edit_splunktcp_ssl</code>	Can list or edit any SSL-specific settings for Splunk TCP input.
<code>edit_tcp</code>	Change settings for receiving general TCP inputs.
<code>edit_tcp_token</code>	Change TCP tokens. This is an admin capability and should only be assigned to system administrators.
<code>edit_udp</code>	Change settings for UDP inputs.
<code>edit_user</code>	<p>Create, edit, or remove users.</p> <p>A role with the <code>edit_user</code> capability can assign any role to other users. To limit this ability, also assign the <code>edit_roles_grantable</code> capability and configure <code>grantableRoles</code> in <code>authorize.conf</code>. For example: <code>grantableRoles = role1;role2;role3</code></p>
<code>edit_view_html</code>	Create, edit, or modify HTML-based views.
<code>edit_web_settings</code>	Change settings for <code>web.conf</code> .
<code>embed_reports</code>	Embed reports and disable embedding for embedded reports.
<code>export_results_is_visible</code>	Displays or hides the Export Results button in Splunk Web. The default value is to display the button.
<code>get_diag</code>	Use the <code>/streams/diag</code> endpoint to get a remote diag from a Splunk instance.
<code>get_metadata</code>	Use the "metadata" search processor.
<code>get_typeahead</code>	Use typeahead.
<code>indexes_edit</code>	Change index settings like file size and memory limits.
<code>input_file</code>	Add a file as an input.

license_tab	Access and change the license.
license_edit	Edit the license.
list_deployment_client	View deployment client settings.
list_deployment_server	View deployment server settings.
list_forwarders	View forwarder settings.
list_httpauths	View user sessions.
list_inputs	View list of various inputs, including input from files, TCP, UDP, scripts, etc.
output_file	Add a file as an output.
pattern_detect	Controls ability to see and use the Patterns tab in the Search view.
request_remote_tok	Get a remote authentication token.
rest_apps_management	Edit settings in the python remote apps handler.
rest_apps_view	List properties in the python remote apps handler.
rest_properties_get	Can get information from the services/properties endpoint.
rest_properties_set	Edit the services/properties endpoint.
restart_splunkd	Restart Splunk through the server control handler.
rtsearch	Run real-time searches.
run_debug_commands	Run debug commands.
schedule_search	Schedule saved searches, create and update alerts, and review triggered alert information.
schedule_rtsearch	Schedule real-time saved searches. In order for a user to use this capability their role must also have the <code>schedule_search</code> capability.
search	Run searches.
use_file_operator	Use the "file" search operator.

Windows-specific capabilities

If you are running Splunk on Windows, additional capabilities are provided to facilitate monitoring.

Capability name	What it lets you do
edit_win_eventlogs	Edit windows eventlogs.
edit_win_wmicnf	Edit wmi.conf.
list_win_localavailablelogs	List all local Windows event logs.

Add and edit roles with Splunk Web

When you create users, you assign them to roles that determine the level of access to Splunk Enterprise and the tasks that they can perform. Splunk Enterprise comes with a set of default roles that you can use. You can also create your own.

For information about roles and how capabilities and permissions are inherited, see ["About role-based user access."](#)

Note: Custom roles that inherit from Admin or Power users do not automatically inherit management access. For information about granting management access to custom roles, see ["Add access controls to custom roles."](#)

Add or edit a role

To create or edit roles in Splunk Web:

1. Click **Settings > Access Controls**.
2. Click **Access controls** page click **Roles**.
3. Click **New** or select and edit an existing role. Role names must use lowercase characters only. They cannot contain spaces, colons, or forward slashes.
4. Specify **Search restrictions** for this role. You create and restrict data access control and search capacity by specifying search restrictions.
 - **Restrict search terms:** You can create search strings that determine what data will (or will not) display for a user assigned to the role. See ["Search filter format"](#) in this topic.
 - **Restrict search time range:** specify over how large of a window of time this role can search.
 - **User-level concurrent search jobs limit:** specify the maximum number of search jobs that can be run at one time for users of this role.

- **User-level concurrent real-time search jobs limit:** Specify the number of real-time search jobs that can be run at the same time for a user of this role.
- **Role-level concurrent search jobs limit:** specify the maximum number of search jobs that can be run at one time for this role.
- **Role-level concurrent real-time search jobs limit:** Specify the number of real-time search jobs that can be run at the same time for this role.
- **Limit total jobs disk quota:** Specify the total disc space you want to dedicate to search jobs for each user assigned to the role.

5. In the **Inheritance** section, select roles that you want your new role from which you want to inherit capabilities and properties. A user assigned to multiple roles inherits properties from the role with the broadest permissions. See "[Role inheritance](#)" in the "[About role-based user access](#)" topic for more information.

6. In the **Capabilities** section, choose any individual capabilities you want to provide to this role. See "[About defining roles with capabilities](#)" for more information.

7. In **Indexes searched by default** specify the indexes that this role will automatically search if no index is specified in the search.

8. In **Indexes** select indexes the user is allowed to search. If you add at least one index, a user with this role will only be able to conduct searches on the index or indexes selected. If you do not specify any indexes at all, the user assigned to the role is able to search all indexes.

9. Click **Save**.

Search filter format

The Search filter field can include any of the following search terms:

- source=
- host=
- index=
- eventtype=
- sourcetype=
- search fields
- wildcards
- use **OR** to use multiple terms, or **AND** to make searches more restrictive

The search terms cannot include:

- saved searches
- time operators
- regular expressions
- any fields or modifiers Splunk Web can overwrite

Add and edit roles with `authorize.conf`

You can add or modify roles by editing `authorize.conf`. Users are assigned to roles which determine the level of access to Splunk and the tasks that they can perform with Splunk. For more information about roles and capabilities, read ["About role-based user access."](#)

Caution: Do not edit or delete any roles in `$SPLUNK_HOME/etc/system/default/authorize.conf`. This could break your admin capabilities. Edit this file in `$SPLUNK_HOME/etc/system/local/`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/`. For more information on configuration files in general, see "About configuration files" in the Admin Manual.

Note: Distributed search configurations have slightly different authorization needs. When you use search head pooling, you must make sure that the search heads and the search peers all use the same set of `authorize.conf` file(s). To make sure your authorization is properly set up for search pooling, see "How authorization works in distributed searches."

Add roles

Here's the syntax for adding roles through

`$SPLUNK_HOME/etc/system/local/authorize.conf`:

```
[role_<roleName>]
<attribute> = <value>
<attribute> = <value>
...
```

The `<roleName>` in the stanza header is the name you want to give your role. For example: `security, compliance, ninja`.

Role names must use lowercase characters only. They cannot contain spaces, colons, or forward slashes.

You can include these attributes in the role stanza:

- `<capability> = enabled`
 - ◆ You can add any number of capabilities to a role. See ["About defining roles with capabilities"](#) for more information.
 - ◆ Capabilities are disabled by default. To add a capability to a role, just set it to "enabled".
- `importRoles = <role>;<role>;...`
 - ◆ When set, the current role will inherit all the capabilities from `<role>`. Members assigned to multiple roles inherit properties from the role with the broadest permissions. See ["Role inheritance"](#) in the "About users and roles" topic for more information.
 - ◆ Separate multiple roles, if any, with semicolons.
- `srchFilter = <search_string>`
 - ◆ Use this field for fine-grained access controls. Searches for this role will be filtered by this expression. See ["Search filter format"](#) in this topic for more information.
- `srchTimeWin = <string>`
 - ◆ Maximum time span (in seconds) of a search executed by this role.
- `srchDiskQuota = <int>`
 - ◆ Maximum amount of disk space (MB) that can be taken by search jobs of a user that belongs to this role.
- `cumulativeSrchJobsQuota = <number>`
 - ◆ Maximum number of concurrently running historical searches that all members of this role can have.
 - ◆ **NOTE:** When a user belongs to multiple roles, then the user uses searches from the roles with the largest cumulative search quota first. When the quota that role is completely used up, roles with lower quotas are used.
- `cumulativeRTSrchJobsQuota = <number>`
 - ◆ Maximum number of concurrently running real-time searches that all members of this role can have.
 - ◆ **NOTE:** When a user belongs to multiple roles, then the user uses searches from the roles with the largest cumulative search quota first. When the quota that role is completely used up, roles with lower quotas are used.
- `srchJobsQuota = <int>`
 - ◆ Maximum number of concurrently running searches a member of this role can have.
- `rtSrchJobsQuota = <number>`
 - ◆ Maximum number of concurrently running real-time searches a member of this role can have.
- `srchIndexesDefault = <string>`
 - ◆ Semicolon delimited list of indexes to search when no index is specified.
 - ◆ These indexes can be wildcarded, with the exception that '*' does not match internal indexes.

- ◆ To match internal indexes, start with ''. All internal indexes are represented by '*'.
- `srchIndexesAllowed = <string>`
 - ◆ Semicolon delimited list of indexes this role is allowed to search.
 - ◆ Follows the same wildcarding semantics as `srchIndexesDefault`.

Note: You must reload authentication or restart Splunk after making changes to `authorize.conf`. Otherwise, your new roles will not appear in the **Role** list. To reload authentication, go to the **Manager > Authentication** section of Splunk Web. This refreshes the authentication caches, but does not boot current users.

Search filter format

The `srchFilter/` field can include any of the following search terms:

- `source=`
- `host=` and host tags
- `index=` and index names
- `eventtype=` and event type tags
- `sourcetype=`
- search fields
- wildcards
- use `OR` to use multiple terms, or `AND` to make searches more restrictive.

The search terms cannot include:

- saved searches
- time operators
- regular expressions
- any fields or modifiers Splunk Web can overwrite

Example of creating a role in `authorize.conf`

This example creates the role "ninja", which inherits capabilities from the default "user" role. `ninja` has almost the same capabilities as the default "power" role, except it cannot schedule searches. In addition:

- The search filter limits `ninja` to searching on `host=foo`.
- `ninja` is allowed to search all public indexes (those that do not start with underscore) and will search the indexes `mail` and `main` if no index is specified in the search.
- `ninja` is allowed to run 8 search jobs and 8 real-time search jobs concurrently. (These counts are independent.)

- ninja is allowed to occupy up to 500MB total space on disk for all its jobs.

```
[role_ninja]
rtsearch = enabled
importRoles = user
srchFilter = host=foo
srchIndexesAllowed = *
srchIndexesDefault = mail;main
srchJobsQuota = 8
rtSrchJobsQuota = 8
srchDiskQuota = 500
```

Setting access to manager consoles and apps

The local.meta file is handy for allowing you to grant and restrict access to certain parts of your Splunk instance. For example, you can:

- Restrict users in custom roles to a specific app
- Give users in custom roles the ability to access admin level features

Granting admin roles to users

Some management abilities that belong to the Admin role are unique to that specific label. These abilities are not automatically inherited from the Admin role when you configure a role in Splunk Web or `authorize.conf`.

For example, say you want to create a custom role that inherits all of the Admin abilities but has limited access to your search jobs. To do this, you would create a new role called "SpecialAdmin" and set it to inherit all of the capabilities of an Admin as described in ["About defining roles with capabilities"](#) then set your search limits ["About configuring role-based user access"](#).

"SpecialAdmin" will have every capability you assign to the Admin role. However it will not automatically have access to the same manager pages as the Admin, including:

- System related functionality: The ability to view or modify your system settings and cluster configuration, update or view your license, and restart Splunk.
- Data features: The ability to view or add data inputs, configure forwarding and receiving, manage indexes, and report acceleration.

- Distributed Environment: Set up distributed search, view deployment client and server status.
- Users and Authentication: Manage you LDAP configuration, users (other than their own) and roles.

Restricting access to specific apps

The `local.meta` file can also be used to restrict access.

For example, say you want to allow a user access to only one dashboard view. To accomplish this, you could create an app for that view and assign the user's role to that app. You should use `meta.local` to permit the role to view that app.

How to add and remove access via local.meta files

You can give or restrict access by editing the `local.meta` file to add the new role wherever you want it.

1. Locate the `local.meta` file. If you are editing access for the main search page (ie, the manager controls), look in `$(SPLUNK_HOME)/etc/system/metadata/`. If you want to edit access to a particular app, look in `$(SPLUNK_HOME)/etc/apps/<app_name>/metadata/`. If the directory for the desired location does not contain the file, you can copy the default version `default.meta` and rename it.

Note: Do **NOT** edit the `default.meta` file directly, you may need the default values in that file at a future time.

2. In the `local.meta` file, add the name of the new role to the stanza that corresponds with the desired access.

Default stanza	What it does
<code>[manager/accesscontrols]</code> <code>access = read : [*], write : [admin, power]</code>	Allow all users to read this app's contents, or access functions in the Splunk Manager page, depending on the directory you are in. Unless overridden by other metadata, allows only admin and power users to share objects into this app.
<code>[views] [manager/accesscontrols]</code> <code>access = read : [*], write : [admin]</code>	Determines the access controls for the Manager page access.

3. When you have made all of your changes, restart Splunk.

Examples

Example 1: A new role called "usermanager" only inherits capabilities from a user and has no searches or indexes inherited. The intent is to create a role that has no access to data and is solely used to create and manage user accounts.

To create this role you would edit the following stanza:

```
[manager/accesscontrols]
access = read : [ admin ], write : [ admin ]
```

To include the following:

```
[manager/accesscontrols]
access = read : [ admin, usermanager ], write : [ admin, usermanager ]
```

You have just given "usermanager" the ability to see and edit stuff in the "Access controls" pages in Manager.

Example 2: To enable the role "userview," to access but not edit the pages, only add the role to the read value:

```
[manager/accesscontrols]
access = read : [ admin, userview, usermanager ], write : [ admin,
usermanager ]
```

You can also grant access to read the manager pages to EVERY role using the wildcard:

```
[manager/accesscontrols]
access = read : [ * ], write : [ admin ]
```

Example 3: You want to have a subset of users who can only read sales data that you specify. To accomplish this you can create an app for the dashboard and then create a new role "salesusers."

In the `local.meta` file in your app directory (remember that you can create one from the `default.meta` file), you then edit the following stanza:

```
[viewstates]
access = read : [ * ], write : [ * ]
```

to read:

```
[viewstates]
access = read : [ salesusers ], write : [ admin ]
```

Find existing users and roles

First, locate the existing user or role in Splunk Web, click **System** in the main menu and select **Access Controls** to go to the Access Controls page. From there you can click on **Users** or **Roles** to search for your user and role. Splunk Search supports wildcards.

Splunk searches for the string you enter in all available fields by default. To search a particular field, specify that field.

For example, to search only email addresses, type "email=<email address or address fragment>."; or to search only the "Full name" field, type "realname=<name or name fragment>". To search for users in a given role, use "roles=".



Delete all user accounts

Remove all the user data (user accounts) from your Splunk installation by typing `./splunk clean` followed by the `userdata` argument. This deletes all the user accounts other than the default user accounts included with Splunk (admin, power, user).

Caution: Removing user data is irreversible; if you accidentally delete user data, you must re-add the accounts manually.

To remove all of the user accounts in the system:

```
./splunk clean userdata
```

To remove the user accounts in the system and force Splunk to skip the confirmation prompt:

```
./splunk clean userdata -f
```

Secure access for Splunk knowledge objects

As you use Splunk, you create a variety of Splunk **knowledge objects** such as **event types**, **tags**, **lookups**, **field extractions**, **workflow actions**, and **saved searches**. Splunk Web lets you restrict and expand access to knowledge objects within your Splunk implementation. You can use it to:

- Make an object available to users of all apps.
- Make an object available to users of a particular app.
- Restrict object access by user role.
- Disable or delete objects.
- Allow users to share or delete objects they do not own.

For more information about securing your knowledge objects, see "Manage knowledge object permissions" and "Disable or delete knowledge objects" in the Knowledge Manager Manual.

Native Splunk Enterprise authentication

Set up user authentication with Splunk Enterprise native authentication

Splunk Enterprise native **authentication** allows you to easily set up users within your system. Splunk Enterprise offers three ways to configure users, but native authentication always takes precedence over any external systems. This is the order in which users are authenticated:

1. Splunk Enterprise native authentication.
2. Either LDAP or Scripted authentication (if enabled). For more information, see ["Set up user authentication with LDAP"](#) and ["Set up user authentication with external systems."](#)

Note: LDAP and scripted authentication cannot be used together.

You can create new users and assign those users to **roles** with a role-based access control system in two ways:

- Use Splunk Web to create users and assign roles. For more information see ["Configure users with Splunk Web."](#)
- Use the CLI to create users and then assign them to roles with Splunk Web. For more information see ["Configure users with the CLI."](#)

Important naming guidelines when creating users and roles

Username stored in native authentication cannot contain spaces, colons, or forward slashes. Names are case-insensitive, for example: "Jacque", "jacque", "JacQue" are all the same to Splunk Enterprise.

Role names must use lowercase characters only. They cannot contain spaces, colons, or forward slashes.

Configure users with Splunk Web

To configure users and roles In Splunk Web:

1. Navigate to **Settings > Users and Authentication > Access controls**.
2. Click **Users**.
3. Click **New** or select an existing user to edit.
4. Specify or change the information for the user. You can specify the user's:
 - ◆ full name.
 - ◆ email address.
 - ◆ time zone. This allows users to view events and other information in their own time zone.
 - ◆ default app. This overrides the default app inherited from the user's role.
 - ◆ password.
5. Assign the user to an existing role or roles and click **Save**.

You can also create a role specifically for a user, defining exactly what access that user has to Splunk. You can then assign the user to that role. For information about roles, read "[About role-based user access](#)."

For information about managing user settings, see the Splunk Enterprise Administration Guide.

Configure users with the CLI

In the CLI, use the `add user` command. Here are some examples:

- To add a new administrator user with the password "changeme2":

```
./splunk add user admin2 -password changeme2 -role admin -auth admin:changeme
```

- To change an existing user's password to "fflanda":

```
./splunk edit user admin -password fflanda -role admin -auth admin:changeme
```

Important: Passwords with special characters that would be interpreted by the shell (for example '\$' or '!') must be either escaped or single-quoted. For example:

```
./splunk edit user admin -password 'fflanda$' -role admin -auth admin:changeme  
or
```

```
./splunk edit user admin -password fflanda\$ -role admin -auth admin:changeme
```

Add a user to a role with Splunk Web

You can add a user to a default role or to a custom role you create yourself. For more information, see ["About role-based user access."](#)

To add a user or users to a role with Splunk Web:

1. Click **Settings > Access Control > Access Controls** in the main menu.
2. Click **Users**.
3. Edit an existing user or create a new one.
4. Choose which role to map to from the **Role** list. Any custom roles you have created in `authorize.conf` will be listed here.

Authentication with LDAP

Set up user authentication with LDAP

Splunk supports three types of **authentication** systems:

- Splunk's own built-in system, described in ["Set up user authentication with Splunk's built-in system."](#)
- LDAP, described in the topic you're now reading.
- A scripted authentication API for use with an external authentication system, such as PAM or RADIUS, described in ["Set up user authentication with external systems."](#)

About configuring LDAP authentication for Splunk

Splunk allows user and role configuration for LDAP users and groups. You can configure one or many LDAP servers and map users and user groups from your servers to roles created in Splunk.

For more information about configuring multiple LDAP servers, see ["How Splunk works with multiple LDAP servers."](#)

Before you configure LDAP, take a look at ["LDAP prerequisites and considerations."](#)

How to configure Splunk to use LDAP

These are the main steps to configure Splunk to work with LDAP:

1. Configure one or more LDAP strategies (typically, you configure one strategy per LDAP server).
2. Map your LDAP groups to one or more Splunk roles.
3. If you have multiple LDAP servers, specify the connection order of their servers.

You can perform these steps in Splunk Web or by editing the configuration file. See ["Configure LDAP with Splunk Web"](#) or ["Configure LDAP with the configuration file"](#) for more information.

Authentication precedence

Splunk's built-in system always takes precedence over any external systems. This is the order in which Splunk authenticates a user:

1. Splunk built-in authentication
2. Either LDAP or scripted authentication, if one of these methods is enabled. For more information about scripted authentication see "[Set up user authentication with external systems.](#)"

Answers

Have questions? Visit Splunk Answers and see what questions and answers the Splunk community has around LDAP authentication with Splunk.

Manage Splunk user roles with LDAP

To configure Splunk to use LDAP authentication, first create a Splunk strategy for each LDAP server and then map Splunk roles to that server's groups. When a user attempts to log in, Splunk queries the server(s) to find the user. It grants the user permissions based on any roles associated with the LDAP groups the user is a member of.

When it comes to changing a user's permissions, you have several options:

- To change the permissions for a group of users, you can remap the LDAP group to a different Splunk role. You can also update the role itself to specify a different set of permissions for it. You do this on Splunk.
- To change the permissions for an individual user, you can move the user to an LDAP group mapped to a different Splunk role. You do this on the LDAP server.

Here are some other user management activities:

- To add a user to a Splunk role: First, on Splunk Web, make sure that you've mapped the Splunk role to an LDAP group. Then, on your LDAP server, add the user to that LDAP group.
- To remove a user from a Splunk role: On your LDAP server, remove the user from the corresponding LDAP group.

A user can have membership in several roles. In that case, Splunk gives the user access to all the capabilities available for any of those roles. For example, if the user is a member of both the docs and eng groups, and docs is mapped to "user" and eng is mapped to "admin", the user obtains all permissions assigned to both the "user" or "admin" roles.

Note: Splunk automatically checks LDAP membership information when a user attempts to log into Splunk. You do not need to reload the authentication configuration when adding or removing users.

LDAP prerequisites and considerations

Before configuring LDAP for Splunk, make the preparations described in this topic.

Determine your User and Group Base DN

Before you map your LDAP settings in Splunk, figure out your user and group base DN, or distinguished name. The DN is the location in the directory where authentication information is stored.

If group membership information for users is kept in a separate entry, enter a separate DN identifying the subtree in the directory where the group information is stored. Users and groups will be searched recursively on all the subnodes under this DN. If your LDAP tree does not have group entries, you can set the group base DN to the same as the user base DN to treat users as their own group. This requires further configuration, described later.

If you are unable to get this information, contact your LDAP Administrator for assistance.

Note: For best results when integrating Splunk with Active Directory, place your Group Base DN in a separate hierarchy than the User Base DN.

Additional considerations

When configuring Splunk to work with LDAP, note the following:

- Entries in Splunk Web and `authentication.conf` are case sensitive.
- Any user created locally through [Splunk native authentication](#) will have precedence over an LDAP user of the same name. For example, if the

LDAP server has a user with a username attribute (for instance, cn or uid) of 'admin' and the default Splunk user of the same name is present, the Splunk user will win. Only the local password will be accepted, and upon login the roles mapped to the local user will be in effect.

- The number of LDAP groups Splunk Web can display for mapping to roles is limited to the number your LDAP server can return in a query. You can use the **Search request size limit** and **Search request time limit** settings to configure this.
 - ◆ To prevent Splunk from listing unnecessary groups, use the `groupBaseFilter`. For example: `groupBaseFilter = (|(cn=SplunkAdmins)(cn=SplunkPowerUsers)(cn=Help Desk))`
 - ◆ If you must role map more than the maximum number of groups, you can edit `authentication.conf` directly. In this example, "roleMap_AD" specifies the name of the Splunk strategy. Each attribute/value pair maps a Splunk role to one or more LDAP groups:

```
[roleMap_AD]
admin = SplunkAdmins1;SplunkAdmins2
power = SplunkPowerUsers
user = SplunkUsers
```

- Splunk always uses LDAP protocol version 3, aka v3.

How Splunk works with multiple LDAP servers

Splunk can search against multiple LDAP servers when authenticating users. To configure multiple LDAP servers, you set up multiple LDAP "strategies," one for each LDAP server.

Once you create your strategies, you can then specify the order in which you want Splunk to query these strategies when searching for LDAP users. If you do not specify a search order, Splunk assigns a default "connection order" based on the order in which the strategies are created.

For more about the steps to configure LDAP strategies, see ["Configure LDAP with Splunk Web"](#) or ["Configure LDAP with the configuration file"](#) for more information.

How connection order works during a search

During authentication, Splunk searches based on the strategies you created for your servers in the specified connection order. Once Splunk locates the user on a server, it quits searching and takes those credentials. If the user also has credentials on a server later in the search order, Splunk ignores them.

For example, assume you configure and enable three strategies in this order: A, B, C. Splunk will search their servers in that same order: A, B, C. If it finds the user on A, it stops looking. It doesn't matter whether the user also exists on B and C; Splunk will only use A's credentials for that user. If it doesn't find the user on A, then it will continue to search the remaining servers: first B, then C.

If you later disable strategy A, Splunk will search the remaining strategies in the order: B, C.

You can change the connection order at any time by editing the strategies' properties in [Splunk Web](#) or by changing the order of the strategies in the `authSettings` attribute, as described in `Authentication.conf` spec file. More about editing this file for LDAP, see [Edit authentication.conf](#).

Important: Any user created locally through [Splunk's built-in authentication](#) will have precedence over an LDAP user of the same name. See "[About user authentication](#)", for details.

Configure LDAP with Splunk Web

This section describes how to configure LDAP through Splunk Web. If you want to configure LDAP by directly editing `authentication.conf`, see "[Configure LDAP with the configuration file](#)".

There are three main steps to configuring LDAP with Splunk Web:

1. Create an LDAP strategy.
2. Map LDAP groups to Splunk roles.
3. Specify the connection order (for multiple LDAP servers only)

Create an LDAP strategy

To create an LDAP strategy:

1. Click **Settings > Users and authentication > Access controls**.
3. Click **Authentication method**.
4. Check **LDAP**.
5. Click **Configure Splunk to use LDAP and map groups**. This takes you to the **LDAP strategies** page.
6. Click **New**. This takes you to the **Add new** page.
7. Enter an **LDAP strategy name** for your configuration.
8. Enter the **Host** name of your LDAP server. Be sure that your Splunk Server can resolve the host name. **Note: At this time, Splunk does not support IPv6 address formats for Windows.**
9. Enter the **Port** that Splunk should use to connect to your LDAP server.
 - By default LDAP servers listen on TCP port 389.
 - LDAPS (LDAP with SSL) defaults to port 636.
10. To turn on SSL, check **SSL enabled**.
 - This setting is recommended for security.
 - You must also have SSL enabled on your LDAP server.
11. Enter the **Bind DN**.
 - This is the distinguished name used to bind to the LDAP server.
 - This is typically, but not necessarily, the administrator. This user needs to have read access to all LDAP user and group entries you want to retrieve.
 - Leave blank if anonymous bind is sufficient.
12. Enter and confirm the **Bind DN password** for the binding user.
13. Specify the **User base DN**. You can specify multiple user base DN entries by separating them with semicolons.

- Splunk uses this attribute to locate user information.
- You must set this attribute for authentication to work.

14. Enter the **User base filter** for the object class you want to filter your users on.

- This is recommended to return only applicable users. For example: (department=IT).
- Default value is empty, meaning no user entry filtering.

15. Enter the **User name attribute** that contains the user name.

- The username attribute cannot contain white spaces.
- In Active Directory, this is typically `sAMAccountName`, but you can also authenticate on other attributes, like `cn`.
- The value `uid` should work for most other configurations.

16. Enter the **Real name attribute** (common name) of the user.

- Typical values are `displayName` or `cn` (common name).

17. Enter an **Email attribute**

18. Enter the **Group mapping attribute**.

- This is the user attribute that group entries use to define their members.
- The default is `dn` for active directory; set this attribute only if groups are mapped using some other attribute besides user DN.
- For example, a typical attribute used to map users to groups is `dn`.

19. Enter the **Group base DN**. You can specify multiple group base DN entries by separating them with semicolons.

- This is the location of the user groups in LDAP.
- If your LDAP environment does not have group entries, you can treat each user as its own group:
 - ◆ Set `groupBaseDN` to the same value as `userBaseDN`. This means you will search for groups in the same place as users.
 - ◆ Next, set the `groupMemberAttribute` and `groupMappingAttribute` to the same attribute as `userNameAttribute`. This means the entry, when treated as a group, will use the username value as its only member.

- ◆ For clarity, you should probably also set `groupNameAttribute` to the same value as `userNameAttribute`.

Note: For best results when integrating Active Directory, place your Group Base DN in a separate hierarchy than the User Base DN.

20. Enter the **Static group search filter** for the object class you want to filter your static groups on.

- This is recommended to return only applicable groups. For example:
`(|(objectclass=groupofNames)(objectclass=groupofUniqueNames))`
- Default value is empty, meaning no static group entry filtering.

21. Enter the **Group name attribute**.

- This is the group entry attribute whose value stores the group name.
- This is usually `cn`.

22. Enter the **Static member attribute**.

- This is the group attribute whose values are the group's members.
- This is typically `member`, `uniqueMember`, or `memberUId`.

22. To expand nested groups, check **Nested groups**.

- This controls whether Splunk will expand nested groups using the 'memberof' attribute. Only check this if you have nested groups that leverage the 'memberof' attribute to resolve their members. On OpenLDAP, you need to explicitly enable the 'memberof' overlay.

23. Enter the **Dynamic group search filter** to retrieve dynamic groups, if any.

- This must match the object class of your dynamic groups definition to ensure that those groups get returned to Splunk. For example:
`(objectclass=groupOfURLs)`
- Default value is empty, meaning Splunk will not look for dynamic group entries during authentication and authorization.

24. Enter the **Dynamic member attribute**.

- This is the group attribute that uses the form of an LDAP search URL (such as `ldap:///o=Acme,c=US??sub?(objectclass=person)`) to define its members.

- This is typically `memberURL`.

25. If you check **Advanced settings**, there are several additional options you can set:

- **Enable referrals with anonymous bind only.**
 - ◆ This setting is on by default. Turn this off if you have no need for referrals.
 - ◆ Splunk can chase referrals with anonymous bind only. You must also have anonymous search enabled on your LDAP server.
 - ◆ If you are seeing long LDAP search timeouts (likely in Active Directory) and "Operations error" in `splunkd.log` for `ScopedLDAPConnection`, the issues might be related to referrals.
- **Search request size limit**
 - ◆ To avoid performance-related issues, you can set the search request size limit. Splunk will then request that the LDAP server return the specified maximum number of entries in response to a search request. In a large deployment with millions of users, setting this limit to a high value could result in a long response, depending on the search filter set in the LDAP strategy configuration. If this limit is reached, `splunkd.log` should contain a `size limit exceeded` message.
 - ◆ You should set the **search request time limit** and **search request size limit** values in conjunction with the `splunkweb` timeout property, described in "Configure user session timeouts". If you have a group that is not showing up in the Splunk console, it was likely excluded due to one of these limits. Tune these properties as needed.
 - ◆ To set the request size limit higher than 1000, you must also edit `max_users_to_precache` in `limits.conf` to accommodate the number of users you set for your request size limit.
- **Search request time limit**
 - ◆ To avoid performance-related issues, you can set the search request time limit. Splunk will then request that the LDAP server complete its search within the specified number of seconds. In a large deployment with millions of users, setting this limit to a high value could cause Splunk Web to timeout. If this limit is reached, `splunkd.log` should contain a `time limit exceeded` message.
 - ◆ You should set the **search request time limit** and **search request size limit** values in conjunction with the `splunkweb` timeout property, described in "Configure user session timeouts". If you have a group that is not showing up in the Splunk console, it was likely excluded due to one of these limits. Tune these properties as

needed.

- **Network socket timeout**

- ◆ This property is used to break the loop in the authentication chain when one of the LDAP servers in a multiple strategy configuration is unreachable due to network congestion or otherwise takes too long to respond. After waiting the specified number of seconds, the authentication process will continue with the next available strategy, if any.
- ◆ When an LDAP strategy is first created, Splunk validates the LDAP server/port and other parameters. If the LDAP server is down or one of the parameters cannot be validated at that time, the LDAP strategy does not get created.

26. Click **Save**.

Map your new LDAP groups to Splunk roles

Once you have configured Splunk to authenticate via your LDAP server, map your LDAP groups to [Splunk roles](#). If you do not use groups, you can map users individually.

Note: You can map either users or groups, but not both. If you are using groups, all users you want to access Splunk must be members of an appropriate group. Groups inherit capabilities from the highest level role they're a member of.

All users are visible in the **Users** page in Splunk Manager. To assign roles to groups in Splunk Web:

1. From the main menu, select **System > Users and Authentication > Access Controls**.
2. In the **Access Controls** page, click **Authentication method**.
3. Select the **LDAP** radio button then click **Configure Splunk to use LDAP and map groups**. This takes you to the **LDAP strategies** page.
4. Click **Map groups** in the Actions column for a specific strategy. This takes you to the **LDAP Groups** page. You can use the search field in the upper right corner of the page to qualify the list of groups; for example, to search for groups containing specific users.
5. Click on a group name. This takes you the mapping page, which includes a list of available roles and a list of LDAP users for that group.

6. To map a role to a group, click the arrow to the left of a role in the "Available Roles" list. This moves the group into the "Selected Roles" list. You can map multiple roles to the group.

7. Click **Save**. This takes you back to the **LDAP Groups** page.

8. Repeat the process for each group that you want to assign Splunk roles to.

Specify the server connection order

If you have enabled multiple LDAP strategies, you can specify the order in which Splunk searches their servers to find a user, as described in ["How Splunk works with multiple LDAP servers"](#).

By default, Splunk searches the servers in the order in which they were enabled. To change the connection (search) order, you need to edit the properties for each strategy individually:

1. From the main menu, select **System > Users and Authentication > Access Controls**.

2. Click **Authentication method**.

3. Select the **LDAP** radio button.

4. Click **Configure Splunk to use LDAP and map groups**. This takes you to the **LDAP strategies** page.

5. Click on the strategy whose connection order you want to specify. This takes you to the properties page for that strategy.

6. Edit the **Connection order** field near the top of the page. This field appears only if multiple strategies are enabled.

Note: The **Connection order** field does not appear when you initially create the strategy. It only appears when you later edit its properties. Also, the field will be grayed out if the strategy has been disabled.

7. Click **Save**.

8. Repeat the process for any other enabled strategy whose connection order you want to change.

Map LDAP groups to Splunk roles in Splunk Web

If you have configured Splunk to authenticate via your LDAP server, you can map your LDAP groups to [Splunk roles](#). If you do not use groups, you can also map LDAP users individually.

For information about setting up LDAP groups in Splunk Web, see "[Configure LDAP with Splunk Web](#)" in this manual.

Note: You can map either users or groups, but not both. If you are using groups, all users you want to access Splunk must be members of an appropriate group. Groups inherit capabilities from the highest level role they're a member of.

All users are visible in the **Users** page in Splunk Manager. To assign roles to groups in Splunk Web:

1. Click **Settings** in Splunk Web.
2. In the **Users and authentication** section, click **Access controls**.
3. Click **Authentication method**.
4. Select the **LDAP** radio button.
5. Click **Configure Splunk to use LDAP and map groups**. This takes you to the **LDAP strategies** page.
6. Click **Map groups** in the Actions column for a specific strategy. This takes you to the **LDAP Groups** page. You can use the search field in the upper right corner of the page to qualify the list of groups; for example, to search for groups containing specific users.
7. Click on a group name. This takes you the mapping page, which includes a list of available roles and a list of LDAP users for that group.
8. To map a role to a group, click the arrow to the left of a role in the "Available Roles" list. This moves the group into the "Selected Roles" list. You can map multiple roles to the group.
9. Click **Save**. This takes you back to the **LDAP Groups** page.
10. Repeat the process for each group that you want to assign Splunk roles to.

Configure LDAP with the configuration file

As an alternative to using Splunk Web to configure LDAP, you can directly edit the `authentication.conf` file.

This example steps you through the process of setting up `authentication.conf`. If you prefer to configure LDAP with Splunk Web, see ["Configure LDAP with Splunk Web"](#).

Note: If you configure LDAP authentication and decide later to return to using the default Splunk authentication, the simplest way is to move the existing `authentication.conf` file out of the way (for example, by renaming it to `authentication.conf.disabled`) and restart Splunk.

You can see some more examples at the end of the `authentication.conf` spec file.

Edit `authentication.conf` in `$SPLUNK_HOME/etc/system/local/`. For information on configuration files in general, see "About configuration files" In the Administration Manual.

Set authentication type and strategy name(s)

By default, Splunk uses its own authentication type. Change the type to LDAP in the `[authentication]` stanza:

```
[authentication]
authType = LDAP
authSettings = ldaphost1,ldaphost2
```

Note the following:

- Turn on LDAP by setting `authType = LDAP`.
- The `authSettings` attribute identifies one or more LDAP strategies. Each strategy has its own stanza, as described below.

Configure LDAP strategy stanzas

Each LDAP strategy needs its own stanza. Map the LDAP values to attribute/value pairs in the strategy's stanza.

Note: At this time, Splunk does not support IPv6 address formats for Windows.

Here's an example stanza for the "ldaphost1" strategy, specified earlier in the `authSettings` attribute:

```
[ldaphost1]
host = ldaphost1.domain.com
port = 389
SSLEnabled = 0
bindDN = cn=bind_user
bindDNpassword = bind_user_password
groupBaseDN = ou=Groups,dc=splunk,dc=com
groupBaseFilter = (objectclass=*)
groupMappingAttribute = dn
groupMemberAttribute = uniqueMember
groupNameAttribute = cn
realNameAttribute = displayName
userBaseDN = ou=People,dc=splunk,dc=com
userBaseFilter = (objectclass=*)
userNameAttribute = uid
```

Note: For best results when integrating Active Directory, place your Group Base DN in a separate hierarchy than the User Base DN.

Configure multiple LDAP strategies

Splunk can search across multiple LDAP servers, as described in ["How Splunk works with multiple LDAP servers"](#). To configure this, set the `authSettings` attribute to a comma-separated list of all strategies, in the order in which you want Splunk to query them. Then, specify separate stanzas for each strategy.

Map groups to roles

To map Splunk roles to a strategy's LDAP groups, you need to set up a `roleMap` stanza for that strategy. Each strategy requires its own `roleMap` stanza. This example maps roles for groups in the "ldaphost1" strategy. The syntax is `<Splunk RoleName> = <LDAP group string>`:

```
[roleMap_ldaphost1]
admin = SplunkAdmins
itusers = ITAdmins
```

Map users directly to roles

If you need to map users directly to Splunk roles, you can do so by setting the `groupBaseDN` to the value of `userBaseDN`. Also, set the attributes for `groupMappingAttribute`, `groupMemberAttribute`, and `groupNameAttribute` to the same attribute as `userNameAttribute`. For example:

```

[supportLDAP]
SSLEnabled = 0
bindDN = cn=Directory Manager
bindDNpassword = #####
groupBaseDN = ou=People,dc=splunksupport,dc=com
groupBaseFilter = (objectclass=*)
groupMappingAttribute = uid
groupMemberAttribute = uid
groupNameAttribute = uid
host = supportldap.splunksupport.com
port = 389
realNameAttribute = cn
userBaseDN = ou=People,dc=splunksupport,dc=com
userBaseFilter = (objectclass=*)
userNameAttribute = uid

[roleMap_supportLDAP]
admin = rlee;bsmith

```

Map LDAP groups and users to Splunk roles in the configuration files

Once you've set up LDAP authentication and users, you can map your LDAP groups and users to roles in Splunk Web. To set up LDAP for Splunk see ["Configure LDAP with the configuration file"](#) in this manual.

Map groups to roles

To map Splunk roles to a strategy's LDAP groups, you need to set up a `roleMap` stanza for that strategy. Each strategy requires its own `roleMap` stanza. This example maps roles for groups in the "ldaphost1" strategy:

```

[roleMap_ldaphost1]
admin = SplunkAdmins
itusers = ITAdmins

```

Map users directly to roles

If you need to map users directly to Splunk roles, you can do so by setting the `groupBaseDN` to the value of `userBaseDN`. Also, set the attributes for `groupMappingAttribute`, `groupMemberAttribute`, and `groupNameAttribute` to the same attribute as `userNameAttribute`. For example:

```

[supportLDAP]

```

```
SSLEnabled = 0
bindDN = cn=Directory Manager
bindDNpassword = #####
groupBaseDN = ou=People,dc=splunksupport,dc=com
groupBaseFilter = (objectclass=*)
groupMappingAttribute = uid
groupMemberAttribute = uid
groupNameAttribute = uid
host = supportldap.splunksupport.com
port = 389
realNameAttribute = cn
userBaseDN = ou=People,dc=splunksupport,dc=com
userBaseFilter = (objectclass=*)
userNameAttribute = uid

[roleMap_supportLDAP]
admin = rlee;bsmith
```

Test your LDAP configuration

If you find that Splunk is not able to connect to your LDAP server, try these troubleshooting steps:

1. Check `$SPLUNK_HOME/var/log/splunk/splunkd.log` for any authentication errors. Turn on DEBUG-level logging for AuthenticationManagerLDAP to get more information here. This can be done from the Splunk Web UI - Server Settings/Server Logging.
2. Remove any custom values you've added for **userBaseFilter** and **groupBaseFilter**.
3. Use `ldapsearch` to confirm that the variables you are specifying will return the expected entries:

```
ldapsearch -x ?h <ldap_host> ?p <ldap_port> ?D "bind_dn" -w "bind_passwd" -b "user_basedn" "userNameAttribute=*"
```

```
ldapsearch -x ?h <ldap_host> ?p <ldap_port> ?D "bind_dn" -w "bind_passwd" ?b "group_basedn" "groupNameAttribute=*"
```

If these commands return matching entries, then your backend LDAP system is properly configured. Continue to troubleshoot the Splunk LDAP strategy configuration.

Convert to LDAP from Splunk's built in authentication

If you move from built-in authentication to LDAP, it's important to note that the accounts created in Splunk are not automatically disabled and take precedence over LDAP accounts.

If you have converted from Splunk's built-in authentication system to LDAP, you might need to delete users from Splunk's built-in system to ensure that you're using LDAP credentials. This is only necessary if usernames are the same in both systems.

Secure local Splunk accounts

If you have configured Splunk to use LDAP authentication, it's important to be aware that all local accounts using Splunk built-in authentication are still present and active. This includes the "admin" account. You need to consider the security implications of this.

To remove all the current local accounts when enabling LDAP authentication:

- Move the `$(SPLUNK_HOME)/etc/passwd` file to `passwd.bak`.
- Create a blank `$(SPLUNK_HOME)/etc/passwd` file.
- Restart Splunk.

Keep in mind that local Splunk accounts can still be created when Splunk is in LDAP authentication mode. Also, any local Splunk accounts that must remain for backup or disaster-recovery purposes should use a very strong password.

When using LDAP, make sure that your LDAP implementation enforces:

- Strong password requirements for length and complexity.
- A low incorrect attempt threshold for password lockout.

Saved searches

If your LDAP usernames are the same as the names you previously used in the built-in system (but then deleted), saved searches should work without any conversion.

If you have existing saved searches created when your system was using Splunk's built-in authentication and you'd like to transfer them to an LDAP user of

a different name, edit the metadata:

1. Modify `$(SPLUNK_HOME)/etc/apps/<app_name>/metadata/local.meta` and swap the `owner = <username>` field under each `savedsearch` permission stanza to the corresponding LDAP username and save your changes.

2. Restart Splunk for your changes to take effect.

Best practice for removing an LDAP user

If you remove a user from your LDAP directory, Splunk does not automatically remove it from the Splunk native authentication directory. Usually this is not an issue, but if the user has global permissions of any sort, LDAP may generate errors.

To more information about working with LDAP users in Splunk, see ["Set up user authentication with LDAP"](#) in this manual.

Take the following steps to safely remove a user name from Splunk's directory:

1. First, back up the `$(HOME)/splunk/etc/users/$userid` folder.

2. Search the files under `$(HOME)/splunk/etc/apps/` for the user id string to see if the user owns any searches or objects with global permissions.

3. For any searches or objects that the user owns, change the owner. You change it an admin user or maintenance account, or whatever you prefer.

4. Check `splunkd.log` on the search head to make sure there are no further LDAP authentication errors.

5. Once you have redirected any object ownership, you can safely remove the `$(HOME)/splunk/etc/users/$userid` folder.

Authentication using single sign-on with SAML

Configure single sign-on with SAML

Splunk software lets you use SAML authentication for single sign-on (SSO). To enable SSO, use information provided by your supported IdP to configure Splunk to work with SAML.

Caution: Before you attempt to configure SSO in Splunk Cloud, contact Splunk Support and open a ticket requesting them to prepare your cloud deployment for SSO. When they have configured your deployment, they will notify you and provide the certificate required.

Prerequisites

- Either:
 - ◆ A running version of Splunk Enterprise
 - ◆ A managed deployment of Splunk Cloud. Self-service deployments of Splunk Cloud log in through the Splunk customer portal and cannot independently configure SAML SSO.
- An identity provider configured to provide the `role`, `realName`, and `mail` attributes. The supported identity providers are:
 - ◆ Ping Identity
 - ◆ Okta
 - ◆ Azure AD
 - ◆ AD FS
- An admin role with the `change_authentication` capability. This permission level lets you enable SAML and edit authentication settings on the Splunk search head.

1. Configure SAML SSO with:

- [Ping Identity](#).
- [Okta](#).
- [Azure AD or AD FS](#).

2. [Map SAML groups to Splunk Enterprise roles](#).

Configure SSO with PingIdentity as your identity provider

This task describes how to set up SSO for Splunk once you have configured PingIdentity as your Identity Provider (IdP). For information about configuring PingIdentity as an IdP, consult your Ping Federate documentation.

You can use the following task to configure Splunk to recognize and use your PingIdentity configuration. Then you can authorize groups on this IdP to log in by mapping them to Splunk user roles.

Caution: Before you attempt to configure SSO in Splunk Cloud, contact Splunk Support and open a ticket requesting them to prepare your cloud deployment for SSO. When they have configured your deployment, they will notify you and provide the certificate required.

Prerequisites

Verify that your system meets all of the requirements. See [Configure single sign-on with SAML](#).

1. In the **Settings** menu, select **Access Controls > Authentication method**.
2. Select **SAML** as your authentication type.
3. Click **Configure Splunk to use SAML**.
4. On the SAML Groups page, click **SAML Configuration**.
5. Browse and select your metadata file, or copy and paste your metadata directly into the text window. Refer to your IdP's documentation if you are not sure how to get your metadata file.
6. In **General Settings**, provide the following information.

Single Sign on URL.	This field is populated automatically by your selected metadata file. It is the protected endpoint on your IdP to which Splunk sends authentication requests. If you are using Splunk Cloud, open a support ticket to have the Splunk Cloud operations team open the port for communicating with the IdP. Your users use this URL for SSO login.
----------------------------	---

	To access the login page once SAML is enabled, append the full login URL (/account/login) with loginType=Splunk. Users can also log into their local Splunk account by navigating directly to ? splunkweb:port/en-US/account/login/loginType=Splunk
Single Log Out URL.	This field is populated automatically by the metadata file and is the IdP protocol endpoint. If you do not provide this URL, the user will not be logged out.
IdP's certificate path	This value can be a directory or a file, depending on your IdP requirements. If you provide a file, Splunk uses that file to validate authenticity of SAML response. If you provide a directory, Splunk looks for all the certificates that are present as children of the directory and tries to validate SAML response with each one of them, if Splunk fails to validate authenticity with all of them, response is not considered authentic.
Entity ID.	This field is the entity ID as configured in the SP connection entry in your IdP.
Sign AuthRequest.	Select this option.
Sign SAML Response.	Select this option.

7. In **Attribute Query**, provide the following information so you can create scheduled searches later.

Attribute Query URL.	This field is the endpoint on the IdP to which queries over SOAP are sent. The format is as follows: <urn:oasis:names:tc:SAML:2.0:attrname-format:uri>
Sign attribute query request	Verify that this field is selected.
Sign attribute query response	Verify that this field is selected.

8. In **Advanced settings**, provide the following information.

Attribute Alias Role	Use this field to specify a new attribute name on any IdP and then configure an alias on the Splunk side for any of the 3 attributes.
Attribute Alias Mail	Skip this field.
Attribute Alias Real Name	Skip this field.
FQDN - Host Name or IP of the load balancer	Set to: https://sh1.STACKID.splunkcloud.com. This setting works for a Splunk deployment with Single Search Head Setup or a Search Head Cluster Setup. You must provide an address if you use load balancing with a search head cluster.
(Optional) Redirect Port	Provide a redirect port for the load-balancer described in the previous field.

9. Click **Save**.

Next Step

[Map SAML groups to Splunk Enterprise roles](#)

Configure SSO with Okta as your identity provider

When you configure Splunk to use your SAML authentication system, you can authorize groups on your IdP to log in by mapping them to Splunk user roles.

Caution: Before you attempt to configure SSO in Splunk Cloud, contact Splunk Support and open a ticket requesting them to prepare your cloud deployment for SSO. When they have configured your deployment, they will notify you and provide the certificate required.

Prerequisites

Verify that your system meets all of the requirements. See [Configure single sign-on with SAML](#).

1. In the **Settings** menu, select **Access Controls > Authentication method**.

2. Select **SAML** as your authentication type.
3. Click **Configure Splunk to use SAML**.
4. On the SAML Groups page, click **SAML Configuration**.
5. Browse and select your metadata file, or copy and paste your metadata directly into the text window. Refer to your IdP's documentation if you are not sure how to get your metadata file.
6. In **General Settings**, provide the following information.

Single Sign on URL.	<p>This field is populated automatically by your selected metadata file. It is the protected endpoint on your IdP to which Splunk sends authentication requests. If you are using Splunk Cloud, open a support ticket to have the Splunk Cloud operations team open the port for communicating with the IdP. Your users use this URL for SSO login.</p> <p>To access the login page once SAML is enabled, append the full login URL (<code>/account/login</code>) with <code>loginType=Splunk</code>. Users can also log into their local Splunk account by navigating directly to <code>?splunkweb:port/en-US/account/login/loginType=Splunk</code></p>
Single Log Out URL.	<p>This field is populated automatically by the metadata file and is the IdP protocol endpoint. If you do not provide this URL, the user will not be logged out.</p>
IdP's certificate path	<p>This value can be a directory or a file, depending on your IdP requirements. If you provide a file, Splunk uses that file to validate authenticity of SAML response. If you provide a directory, Splunk looks for all the certificates that are present as children of the directory and tries to validate SAML response with each one of them, if Splunk fails to validate authenticity with all of them, response is not considered authentic.</p>
Entity ID.	<p>This field is the entity ID as configured in the SP connection entry in your IdP.</p>
Sign AuthRequest.	<p>Select this option.</p>
	<p>Select this option.</p>

Sign SAML Response.	
----------------------------	--

7. These fields are not required for creating scheduled searches with Okta. You can skip them and move to step 8.

Attribute Query URL.	This field is the endpoint on the IdP to which queries over SOAP are sent. The format is as follows: <urn:oasis:names:tc:SAML:2.0:attrname-format:uri>
Sign attribute query request	Verify that this field is selected.
Sign attribute query response	Verify that this field is selected.

8. In **Advanced settings**, provide the following information.

Attribute Alias Role	Use this field to specify a new attribute name on any IdP and then configure an alias on the Splunk side for any of the 3 attributes.
Attribute Alias Mail	Skip this field for Okta.
Attribute Alias Real Name	Skip this field for Okta.
FQDN - Host Name or IP of the load balancer	Set to: https://sh1.STACKID.splunkcloud.com . This setting works for a Splunk deployment with Single Search Head Setup or a Search Head Cluster Setup. You must provide an address if you use load balancing with a search head cluster.
(Optional) Redirect Port	Provide a redirect port for the load balancer described in the previous field. For Okta set it to "0" (Zero).

9. Click **Save**.

Next Step

Configure SSO with AzureAD or ADFS as your Identity Provider

When you configure Splunk to use your SAML authentication system, you can authorize groups on your IdP to log in by mapping them to Splunk user roles.

Caution: Before you attempt to configure SSO in Splunk Cloud, contact Splunk Support and open a ticket requesting them to prepare your cloud deployment for SSO. When they have configured your deployment, they will notify you and provide the certificate required.

Prerequisites

Verify that your system meets all of the requirements. See [Configure single sign-on with SAML](#).

1. In the **Settings** menu, select **Access Controls > Authentication method**.
2. Select **SAML** as your authentication type.
3. Click **Configure Splunk to use SAML**.
4. On the SAML Groups page, click **SAML Configuration**.
5. Browse and select your metadata file, or copy and paste your metadata directly into the text window. Refer to your IdP's documentation if you are not sure how to get your metadata file.
6. In **General Settings**, provide the following information.

Single Sign on URL.	This field is populated automatically by your selected metadata file. It is the protected endpoint on your IdP to which Splunk sends authentication requests. If you are using Splunk Cloud, open a support ticket to have the Splunk Cloud operations team open the port for communicating with the IdP. Your users use this URL for SSO login.
----------------------------	---

	To access the login page once SAML is enabled, append the full login URL (/account/login) with loginType=Splunk. Users can also log into their local Splunk account by navigating directly to ? splunkweb:port/en-US/account/login/loginType=Splunk
Single Log Out URL.	This field is populated automatically by the metadata file and is the IdP protocol endpoint. If you do not provide this URL, the user will not be logged out.
IdP's certificate path	This value can be a directory or a file, depending on your IdP requirements. If you provide a file, Splunk uses that file to validate authenticity of SAML response. If you provide a directory, Splunk looks for all the certificates that are present as children of the directory and tries to validate SAML response with each one of them, if Splunk fails to validate authenticity with all of them, response is not considered authentic.
Entity ID.	This field is the entity ID as configured in the SP connection entry in your IdP.
Sign AuthRequest.	Select this option.
Sign SAML Response.	Select this option.

7. Skip the Attribute Query Information section for Azure AD, or AD FS and go to step 8.

8. In **Advanced settings**, provide the following information.

Attribute Alias Role	<p>Use this field to specify a new attribute name on any IdP and then configure an alias on the Splunk side for any of the 3 attributes.</p> <p>If you have configured the AD FS built-in "Role" attribute to be returned and this has the AD group information, specify http://schemas.microsoft.com/ws/2008/06/identity/claims/role. This value tells Splunk the attribute that contains the role information in the SAML response returned.</p> <p>If you have configured Azure AD, specify http://schemas.microsoft.com/ws/2008/06/identity/claims/groups.</p>

Attribute Alias Mail	Populate this field if you use Azure AD as your IdP. This value maps the alias to the user email addresses in the SAML response returned. Enter <code>http://schemas.microsoft.com/identity/claims/displayname</code> .
Attribute Alias Real Name	Populate this field if you use Azure AD as your IdP. This value tells Splunk Enterprise where to map the real name in the SAML response returned. Enter <code>http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress</code> .
FQDN - Host Name or IP of the load balancer	Set to: <code>https://sh1.STACKID.splunkcloud.com</code> . This setting works for a Splunk deployment with Single Search Head Setup or a Search Head Cluster Setup. You must provide an address if you use load balancing with a search head cluster.
(Optional) Redirect Port	Provide a redirect port for the load balancer described in the previous field. For Okta set it to "0" (Zero).

9. Click **Save**.

Next Step

[Map SAML groups to Splunk Enterprise roles](#)

Map SAML groups to roles

When you configure Splunk to use your SAML authentication system, you can authorize groups on your SAML server to log in by mapping them to Splunk user roles. You can map multiple groups to a single user role.

Prerequisites

[Configure SAML SSO](#)

1. In the **Settings** menu, select **Access Controls > Authentication method**.
2. Select **SAML** as your authentication type.
3. Click **Configure Splunk to use SAML**.

4. On the SAML Groups page, click **New Group** or click **Edit** for a group you want to modify.

5. Provide a name for the group.

6. Determine the roles that you want to assign to this group by moving the desired roles from the left column to the right column.

7. Click **Save**.

After you configure SAML SSO and map groups to role, you can distribute the login URL to your users.

Configure SAML SSO in the configuration files

This topic explains how to set up SSO for SAML v2 using configuration files:

- Configure `authentication.conf` and `web.conf` in Splunk Enterprise
- Configure your identity provider

Configure `authentication.conf`

Configure the following stanza in `authentication.conf`

```
[authentication]
authSettings = saml_settings
authType = SAML
[rolemap_SAML]
admin = Super Admin;
power = Power Admin;
user = <list roles> Admin;Employee;
[saml_settings]
entityId = <entityid>
idpAttributeQueryUrl = <path to the Attribute query> https://your
path/idp/attrsvc.saml2
idpCertPath = <path to the idp cert in Splunk>
/home/user/splunk/saml-install/etc/auth/ping_idp.crt
idpSSOUrl = <path to the sso url> https://your path/idp/SSO.saml2
idpSLOUrl = <Logout url. If not specified, this will be treated as a
typical sso and the logout button will be disabled> https://your
path/idp/SLO.saml2 #
redirectPort=443
attributeQueryTTL = 3600
signAuthnRequest = true
```

```
signedAssertion = true
attributeQueryRequestSigned = true
attributeQueryResponseSigned = true
attributeQuerySoapPassword = <your password>
attributeQuerySoapUsername = <your username>
```

To configure single sign-on with Azure AD or ADFS, add the following additional attributes:

`nameIDFormat` = (optional) Specify the format of the subject that is returned in the SAML response. AzureAD returns a string to identify the subject and this attribute lets you optionally specify a different format (we recommend email address). This can be useful for auditing and saved searches. To specify email address as the format, use:

```
urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress
```

`role` = Populate this field if you use Azure AD for SSO or ADFS. This value tells Splunk Enterprise the attribute that supplies role information in the SAML response returned. For Azure AD, use:

```
http://schemas.microsoft.com/ws/2008/06/identity/claims/groups
```

`email` = This value maps the alias to the user email addresses in the SAML response returned. For Azure AD, use:

```
http://schemas.microsoft.com/identity/claims/displayname
```

`realName` = This tells Splunk Enterprise where to map the real name in the SAML response returned. For Azure AD use:

```
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress.
```

</pre>

Working with users not assigned to a role

Currently, Splunk software requires attribute query support in the IdP.

If SAML users do not have assigned roles, you can still include them by configuring the `skipAttributeQueryForUsers` and `defaultRoleIfMissing` settings to include those users. For example:

```
defaultRoleIfMissing = user1,user2,user3
skipAttributeQueryRequestForUsers=user1,user2,user3
```

Note that `defaultRoleMissing` cannot be used instead of the `skipAttributeQueryForUsers`. These attributes work in tandem and you must configure both attributes.

Attribute Query Request for Ping Identity

The following is a workaround for the Attribute Query Request field and should be included in your stanza. This is because the existing SSL configuration is too strong to work with Ping Identity, causing the attribute query to fail. You can add the following workaround to weaken the SSL cipher to accommodate Ping Identity.

```
#cipherSuite = TLSv1+MEDIUM:@STRENGTH
cipherSuite = ALL:!aNULL:!eNULL:!LOW:!EXP:RC4+RSA:+HIGH:+MEDIUM
defaultRoleIfMissing = user
skipAttributeQueryRequestForUsers=admin,username,anotherusername
sslKeysfile = <path_to_saml_signing_cert_and_keys>
sslKeysfilePassword = <password_for_saml_keys>
```

Configure `web.conf` and optionally add a failure redirect address

Add the following values to your settings stanza in `web.conf`

```
[settings]
appServerPorts = 7065 <make sure this attribute is enabled>
ssoAuthFailureRedirect =
http://10.140.31.19:7000/ui/en-us/account/sso_error <this is your
custom user redirect for failed logins>
```

Configure your identity provider

Now you must configure your IdP to import Splunk software metadata. To import Splunk software metadata on your IdP, make sure that the `AuthnRequest` signing and `AttributeQuery` request signing setting is compatible on Splunk software and the IdP:

1. Export the IdP certificate onto a file in your Splunk software instance.
2. Make sure that `authentication.conf` points to this certificate in the SAML configuration stanzas.
3. Import the Splunk software server certificate (`server.pem`) into the IdP for signature verification.

Note that you can export Splunk software metadata using the `/saml/spmetadata` endpoint on Splunk Web. You can also access the `SAML-sp-metadata` endpoint on `splunkd`.

Troubleshoot SAML SSO

Here are some common issues and how to resolve them.

Issue

You experience the following message:

```
ERROR AuthenticationManagerSAML - Requesting user info from ID returned
an error Error in Attribute query request, AttributeQueryTransaction
err=Cannot resolve hostname, AttributeQueryTransaction descr=Error
resolving: Name or service not known, AttributeQueryTransaction
statusCode=502
```

To mitigate this issue

- Make sure that the `cipherSuite` is specified correctly in the SAML stanza.

For example:

```
◆ cipherSuite = TLSv1+MEDIUM:@STRENGTH
◆ cipherSuite =
  ALL:!aNULL:!eNULL:!LOW:!EXP:RC4+RSA:+HIGH:+MEDIUM
```

- Make sure all SOAP password requirements are met.
- Make sure your SSL settings for SAML are configured correctly in `authentication.conf`.

Issue

You experience the following message:

```
ERROR UserManagerPro - user="samluser1" had no roles
```

To mitigate this issue

- Make sure that `rolemap_SAML` contains the correct role mapping with ";" at the end of each role name.

Issue

You experience the following message:

```
ERROR AuthenticationManagerSAML - Attribute query request failed.
Status code=urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal, Status
```

msg=No attributes found for requested subject

To mitigate this issue

- Make sure that the `role`, `mail`, and `realName` attributes are mapped to be returned back as part of `AuthnRequest` and the Attribute Query Request.

Issue

User cannot log in after successful assertion validation. No valid Splunk role is found in the local mapping or in the assertion.

To mitigate this issue

- Make sure that `rolemap_SAML` stanza contains proper mapping between roles returned from IdP and the appropriate Splunk role.
- Make sure there are no spaces between, before, or after each role defined in `authentication.conf`. For example:

```
user = User;Employee
```

Issue

Authentication is configured as SAML and the settings appear to be correct, but the login screen shows the page for Splunk authentication instead.

To mitigate this issue

- Make sure in `web.conf` that `appServerPorts` is set to a valid port and not '0'.

Authentication using single sign-on with reverse proxy

About Single Sign-On using reverse proxy

Splunk Single Sign-on (SSO) lets you use a reverse proxy to handle Splunk **authentication**, meaning that once the user has logged into their proxy, they can seamlessly access Splunk Web (and presumably any other applications configured to your proxy).

The reverse proxy implementation of Splunk Enterprise SSO supports logging into Splunk Enterprise via Splunk Web only. Since the implementation relies on cookies to save authentication information, SSO cannot be used for CLI authentication to Splunk Enterprise. Invoking `https://localhost:8089` (or the assigned management port) still requires independent authentication.

To utilize SSO using reverse proxy, you need the following:

- A Proxy Server (Splunk Enterprise supports IIS or Apache).
- An LDAP Server or other external authentication system.
- A working Splunk Enterprise configuration.

For more information about how to configure these items and set up SSO, see [Configure Single Sign-On](#)

How it works

Splunk Enterprise administrators and users invoke Splunk Web via a proxy URL that is deployed with Splunk Web. The proxy authenticates the incoming request against your authentication system. Upon successful authentication the proxy sets a request header with the authenticated identity's attribute and sends this information to Splunk Enterprise.

Splunk Enterprise accepts the incoming HTTP request from the proxy, and if Splunk Enterprise recognizes the user contained in the header, the user bypasses the login page and is automatically authorized.

For successful single sign-on, all requests from the proxy to Splunk Web must include this authenticated header. If the header is not included in a request, then the user is returned to the login page or an error page, depending on your

configuration. Splunk continues using this authenticated header until the identity closes the browser session.

How Splunk processes the proxy request

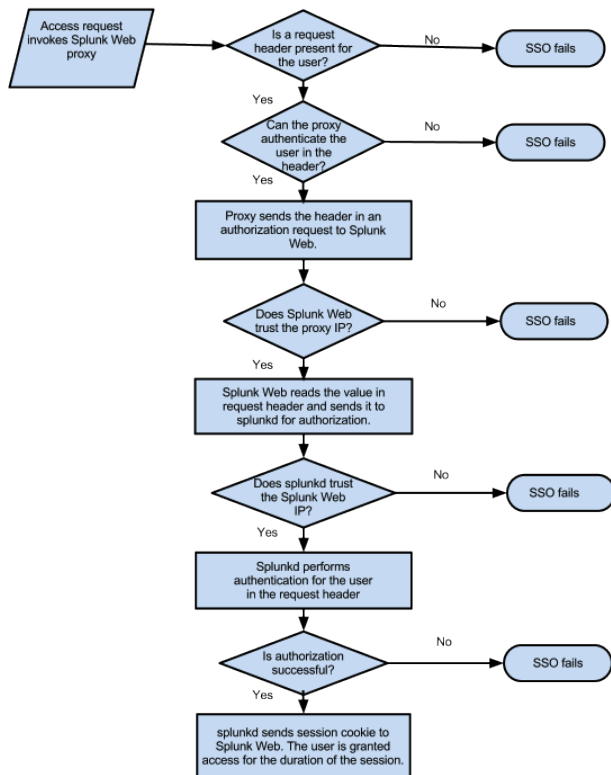
When the proxy server makes a request to Splunk Web, Splunk Web looks to the `trustedIP` value in `web.conf` to verify that the proxy's IP is on the trusted IP list.

If the IP is not trusted, the request is rejected and the sign-on attempt fails. If the IP address is trusted, then Splunk Web queries for the identity in the request header and sends `splunkd` an authorization request containing that header information.

Upon receiving the authorization request from Splunk Web, `splunkd` verifies whether the incoming IP address of the client (e.g. usually Splunk Web) matches the value of the `trustedIP` property of the `server.conf` file.

If the IP addresses are not in the `trustedIP` list the request is rejected and the sign-on attempt fails. The user is either returned to a login page or shown an error page, depending upon your `SSOmode` configuration in `web.conf`. For more on this attribute and other configuration information, see [Configure Splunk Single Sign-On](#).

If the IP is trusted, then `splunkd` uses the information contained in the request header and conducts the authorization process.



How Splunk authorizes the user

Splunk first checks to see if the given identity and role matches any of the users in your Splunk native user configuration. If Splunk fails to find a match there, it looks to see if there are any LDAP matches. (For information about how Splunk authenticates users, see [Set up user authentication with LDAP](#) in this manual.)

If no match is found and the user contained in the header cannot be authorized, then the browser redirects to an error page.

If a match is found, Splunk authorizes the user and checks to see if an existing session is present. If a session already exists, Splunk uses that session identifier and creates the necessary cookies to allow the user access to Splunk Web. If a session does not exist, then Splunk creates a new session as well as the necessary cookies for Splunk Web authorization.

Once the cookies are created, Splunk Web resumes its normal flow. Any subsequent access to Splunk via the proxy URL does not require re-authorization as long as the request header contains the trusted identity and until the user closes the browser session.

Configure Single Sign-On with reverse proxy

Before you configure reverse proxy-based SSO with Splunk Enterprise, make sure you have the following:

- A Proxy Server (Splunk Enterprise supports IIS or Apache) configured as a reverse proxy to authenticate to external systems.
- An LDAP Server or other external authentication system provisioned with appropriate groups and users for your proxy to authenticate against.
- A working Splunk Enterprise configuration that is either configured to use the same external authentication system as your proxy (usually LDAP) or that has native Splunk Enterprise users that match the user and group IDs contained in your external authentication system.

Configuring SSO with reverse proxy requires the following steps:

1. Edit the properties on your proxy server to authenticate against your external authentication system.
2. Edit the Splunk Enterprise `server.conf` file.
3. Edit the Splunk Enterprise `web.conf` file.

Note: For optimal security, any HTTP header-based solutions should be implemented over a TLS/SSL enabled deployment.

Configure `server.conf`

Edit the `trustedIP` in the `general settings` stanza to add the IP address that will make secure authentication requests to `splunkd`. This is typically Splunk Web and therefore the `localhost`. You can only enter one IP address per `splunkd` instance.

```
trustedIP=127.0.0.1
```

If no IP addresses are provided in the `trustedIP` list, Splunk SSO is disabled by default.

Configure `web.conf`

To enable SSO, configure the following in the `[settings]` stanza in `web.conf` (`$SPLUNK_HOME/etc/system/local`):

```

SSOMode = strict
trustedIP = 127.0.0.1,10.3.1.61,10.1.8.81
remoteUser = X-Remote-User
tools.proxy.on = True

```

Attribute	Default	Value
SSOMode	no	<p>The <code>SSOMode</code> attribute determines whether the Splunk Web SSO operates in <code>strict</code> or <code>permissive</code> mode.</p> <p>Strict mode restricts authentication to identities that match the IP addresses listed in <code>trustedIP</code> property. If the IP attempting to connect does not match any IP address, an error page appears to the user. Strict mode is recommended for SSO.</p> <p>Permissive mode also restricts authentication to requests from IPs found in the <code>trustedIP</code> list. In permissive mode, if the IP attempting to connect does not match any IP address, a login page is displayed to allow the user to re-authenticate.</p>
trustedIP	n/a	Set this to the IP address of the authenticating proxy or proxies. Specify a single address or a comma-separated list of addresses; IP ranges and netmask notation are not supported.
remoteUser	REMOTE_USER	<p>The <code>remoteUser</code> attribute determines the authenticated identity's attribute that is passed by the proxy server via the HTTP request header. Splunk defaults this value to <code>REMOTE_USER</code> but any LDAP attribute can be passed via this request header as long as the proxy sets this attribute properly after authentication. When you configure your <code>remoteUser</code> attribute, you must also configure the <code>RequestHeader</code> property in your proxy configuration to pass the identity's attribute to Splunk. This process is described in "About Splunk Single Sign-On".</p>

		The default Splunk header used is <code>REMOTE_USER</code> , but if your proxy uses a different header, you can change the name of the header here.
<code>tools.proxy.on</code>	<code>false</code>	To use Splunk SSO, set <code>tools.proxy.on</code> to <code>true</code> When set to "false," Splunk Enterprise uses the IP address of the computer logging on, however, in Splunk Enterprise SSO, it is the proxy that is requesting login on behalf of the user. Since requests are rejected if the IP address is not listed in the <code>trustedIP</code> property, setting this value to <code>True</code> means that Splunk Web looks at the proxy's IP address.

If you host Splunk Web behind a proxy that does not place Splunk Web at the proxy's root, you may also need to configure the `root_endpoint` setting in `$SPLUNK_HOME/etc/system/local/web.conf`.

For example if your proxy hosts Splunk Web at "yourhost.com:9000/splunk", `root_endpoint` should be set to `/splunk`.

For example:

```
root_endpoint=/lzone
ProxyPass /lzone http://splunkweb.splunk.com:8000/lzone
ProxyPassReverse /lzone http://splunkweb.splunk.com:8000/lzone
```

In the above example, Splunk Web is accessed via `http://splunk.example.com:8000/lzone` instead of `http://splunk.example.com:8000/`.

You would next make it visible to the proxy by mapping it in `httpd.conf`:

```
ProxyPass /lzone http://splunkweb.splunk.com:8000/lzone
ProxyPassReverse /lzone http://splunkweb.splunk.com:8000/lzone
```

Session management

Since there is no simple log out for a session and Splunk Enterprise will preserve a session as long as the correct header information is contained in the proxy header, you should set your proxy's session timeout value with this in mind.

If you need to end a session before the timeout has occurred, you can use the REST end point along with the session identifier to destroy the session:

```
curl -s -uadmin:changeme -k -X DELETE
https://localhost:8089/services/authentication/httpauth-tokens/990cb3e61414376554a39e39
```

Troubleshoot reverse-proxy SSO

Splunk Web provides an interface that allows you to analyze the environment and the run-time data to help you debug your deployment. This page can be accessed via the proxy or the direct URL. The request headers will not be available if you do not access this page through the proxy server.

This URL is located at:

```
http://YourSplunkServer:8000/debug/sso
```

Important: This debug page is not available by default. In order to make the page available, two steps must be completed. First, the role that is accessing this end point must have the `web_debug` capability, which the admin role has by default. Second, in `web.conf`, the setting `enableWebDebug=true` must be configured.

Consider the following when using the troubleshooting page to analyze your deployment:

- Compare the IP provided as the **Splunk trusted IP** with that of the **Host IP**. The values must be the same (they should be the IP of your proxy). If they are not the same in the troubleshooting page, you must edit the `trustedIP` value in `server.conf`.
- Check the value for **Incoming request IP received by splunkweb** to make sure that it displays your client's IP address. If the IP does not match that of your client, you must:
 - ◆ Edit `web.conf` to correct this.
 - ◆ Make sure that `tools.proxy.on` is set to `true`.
- Make sure that your proxy is providing a header. Check the **Authorization** field under **Other HTTP Headers**. If there is no value present, check the `http.conf` file in your proxy to make sure that the remote header attribute value is properly set. Splunk is configured to accept the remote header value of `X_REMOTE_USER`, which is the default for most proxies. If your proxy's remote header is different, and you wish to keep that value, you can edit the remote header value in `web.conf` to change the header that Splunk will accept. See [Configure SSO](#) for more information.

- Make sure that Splunk Web is creating a cookie to send to splunkd. Check the **Cookie** field under **Other HTTP headers** to make sure that a cookie is set. If a cookie is not set, then check your `web.conf` file to make sure your file is properly configured. [Configure SSO](#) for more information.

Scripted authentication

Set up user authentication with external systems

Splunk ships with support for four types of authentication systems:

- [Splunk's built-in system](#)
- [LDAP](#)
- [Single sign-on](#)
- A scripted authentication API for use with an external authentication system, such as PAM or RADIUS, described in this section.

Important: Splunk's built-in system always takes precedence over any external systems.

This is the order in which Splunk authenticates a user for LDAP:

1. Splunk built-in authentication or SSO.
2. LDAP or scripted authentication (if enabled). For more information about LDAP, see "[Set up user authentication with LDAP](#)".

How scripted authentication works

In scripted authentication, a user-generated Python script serves as the middleman between the Splunk server and an external authentication system such as PAM or RADIUS.

The API consists of a few functions that handle communications between Splunk and the authentication system. You need to create a script with handlers that implement those functions.

To use your authentication system with Splunk, make sure the authentication system is running and then do the following:

1. Create a Python authentication script. See "[Create the authentication script](#)" for the procedure.
2. Enable your script by editing `authentication.conf` to specify scripted authentication and associated settings. See "[Edit authentication.conf](#)" for the procedure.

Examples

Splunk provides several example authentication scripts and associated configuration files, including one set for RADIUS and another for PAM. There is also a simple script called `dumbScripted.py`, which focuses on the interaction between the script and Splunk.

You can use an example script and configuration file as the starting point for creating your own script. You must modify them for your environment.

You can find these examples in

`$SPLUNK_HOME/share/splunk/authScriptSamples/`. That directory also contains a README file with information on the examples, as well as additional information on setting up the connection between Splunk and external systems.

Important: Splunk does not provide support for these scripts, nor does it guarantee that they will fully meet your authentication and security needs. They are meant to serve as examples that you can modify or extend as needed.

Create the authentication script

To use your authentication system with Splunk, make sure the authentication system is running and then do the following:

1. Create a Python authentication script. See "[Create a Python script](#)" in this topic for the procedure.
2. Test the new script. See "[Test the script](#)" in this topic for the procedure.
3. Enable your script by editing `authentication.conf` to specify scripted authentication and associated settings. See "[Edit authentication.conf](#)" for the procedure.

Create a Python script

You must create a Python script that implements these authentication functions:

- `userLogin`
- `getUserInfo`

- getUsers

The Splunk server will call these functions as necessary, either to authenticate user login or to obtain information on a user's roles.

The script can optionally also include a handler for this function:

- getSearchFilter

This table summarizes the authentication functions, their arguments, and their return values:

Function	Description	Argument string	Return value string
userLogin	Login with user credentials.	--username=<username> --password=<password> (values passed one per line over <code>stdin</code>)	fail (safely passed over <code>stdout</code>)
getUserInfo	Return a user's information, including name and role(s).	--username=<username>	--status=success fail --userInfo=<userId>;<username>;<realname>;<roles> Note the following: <ul style="list-style-type: none"> • <code>userInfo</code> must specify a semicolon. • <code><userId></code> is deprecated; you should specify <code><username></code> associated semicolon. • <code><username></code> is required. • <code><realname></code> is optional, but its semicolon is required. • <code><roles></code> is required. To return multiple roles, use semicolons to separate the roles. For example: <code>admin:power</code> • This example returns just the roles named "docsplunk": <pre>--status=success --userInfo=;docsplunk;;admin:power</pre>
getUsers	Return information for all Splunk	none	--status=success fail --userInfo=<userId>;<username>;<realname>;<roles> --userInfo=<userId>;<username>;<realname>;<roles> --userInfo=<userId>;<username>;<realname>;<roles> ...

	users.		<p>Note the following:</p> <ul style="list-style-type: none"> • See <code>getUserInfo</code> for information on how to use to return each user's information. • Separate each user's information by a vertical tab character. • <code><roles></code> is required. To return multiple roles, use colons to separate the roles. For example: <code>admin:power</code>
<code>getSearchFilter</code>	Optional. Returns the filters applied specifically to this user, along with those applied to the user's roles. The filters are OR'd together.	<code>--username=<username></code>	<pre>--status=success fail --search_filters=<filters> --search_filter=<filter> ...</pre> <p>Note: User-based search filters are optional and not recommended. A better approach is to assign filters to roles and then assign users to the roles.</p> <p>For more information, see "Use the <code>getSearchFilter</code> function to filter at search time"</p>

See the example scripts for detailed information on how to implement these functions.

Test the script

Since the communication between Splunk and the script occurs via `stdin` and `stdout`, you can test the script interactively in your command shell, without needing to call it from Splunk. Be sure to send one argument per line and end each function call with an EOF (Ctrl-D).

Test each function individually, using this pattern:

```
> python [script] [function name]
[pass arguments here, one per line]
[send eof, with Ctrl-D]
[output appears here, check that it's correct]
>
```

The following example shows a debugging session that does some simple testing of a fictional script called "example.py", with two users "alice" and "bob". "alice" is a member of the "admin" and "super" roles, and "bob" is a member of

the "user" role.

```
> python example.py userLogin
--username=alice
--password=correctpassword
<send an EOF>
--status=success
> python example.py userLogin
--username=bob
--password=wrongpassword
<send an EOF>
--status=fail
> python example.py getUsers
<no arguments for this function, send an EOF>
--status=success --userInfo=bob;bob;bob;user
--userInfo=alice;alice;alice;admin:super
> python example.py getUserInfo
--username=bob
<send an EOF>
--status=success --userInfo=bob;bob;bob;user
> python example.py getUserInfo
--username=userdoesnotexist
<send an EOF>
--status=fail
>
```

Important: This is just an example of how to go about testing a script. It does not attempt to perform exhaustive debugging of any real script.

Edit authentication.conf

To use your authentication system with Splunk, make sure the authentication system is running and then do the following:

1. Create and test a Python authentication script. See ["Create the authentication script"](#) for the procedure.
2. Edit authentication.conf to enable your authentication script. See ["Enable your script"](#) in this topic.
3. Edit authentication.conf to set your cache duration. See ["Set cache durations"](#) in this topic.

Enable your script

Once you create a Python script to implement authentication, you update the `authentication.conf` in `$SPLUNK_HOME/etc/system/local/` to enable your script. You can also copy and edit a sample `authentication.conf` from `$SPLUNK_HOME/share/splunk/authScriptSamples/`.

Specify `Scripted` as your authentication type under the `[authentication]` stanza heading:

```
[authentication]
authType = Scripted
authSettings = script
```

Set script variables under the `[script]` stanza heading. For example:

```
[script]
scriptPath = $SPLUNK_HOME/bin/python $SPLUNK_HOME/bin/<scriptname.py>
```

Set cache durations

To significantly speed authentication performance when using scripted authentication, make use of Splunk's authentication caching capability. You do so by adding the optional `[cacheTiming]` stanza. Each script function (except `getSearchFilter`) has a settable `cacheTiming` attribute, which turns on caching for that function and specifies its cache duration. For example, to specify the cache timing for the `getUserInfo` function, use the `getUserInfoTTL` attribute. Caching for a function occurs only if its associated attribute is specified.

The `cacheTiming` settings specify the frequency at which Splunk calls your script to communicate with the external authentication system. You can specify time in seconds (s), minutes (m), hours (h), days (d), etc. Typically, you'll limit the cache frequency to seconds or minutes. If a unit is not specified, the value defaults to seconds. So, a value of "5" is equivalent to "5s".

This example shows typical values for the caches:

```
[cacheTiming]
userLoginTTL      = 10s
getUserInfoTTL   = 1m
getUsersTTL      = 2m
```

You'll want to set `userLoginTTL` to a low value, since this determines how long user login/password validity is cached.

To refresh all caches immediately, use the CLI command `reload auth`:

```
./splunk reload auth
```

Note: This command does not boot current users off the system.

You can also refresh caches in Splunk Web:

1. In the **System** menu, under **Users and authentication** select **Access controls**.
2. Click **Authentication method**.
3. Click **Reload authentication configuration** to refresh the caches.

Each specified function, except `getUsers`, has a separate cache for each user. So, if you have 10 users logged on and you've specified the `getUserInfoTTL` attribute, the `getUserInfo` function will have 10 user-based caches. The `getUsers` function encompasses all users, so it has a single, global cache.

Use PAM authentication

You can configure Splunk for PAM authentication by following the steps in the example directory's README, which is located at `$(SPLUNK_HOME)/share/splunk/authScriptSamples/`.

If you are still unable to authenticate, then edit `/etc/pam.d/pamauth` and add this line:

```
auth sufficient pam_unix.so
```

Use the `getSearchFilter` function to filter at search time

This function is optional and can be used to implement a user-based filter at search time. When `getSearchFilter` is enabled, Splunk will call it every time a search is run. A user-based search filter supplements any filters specified for that user's role(s). The returned filter(s) will be applied to each search, along with any configured at the role level. Caching of the filter does not occur with this function.

Note: User-based search filters are optional and not recommended. A better approach is to assign search filters to roles and then assign users to the appropriate roles.

To enable `getSearchFilter` function, set the `scriptSearchFilters` parameter in `authentication.conf`:

```
[script]
scriptPath = $SPLUNK_HOME/bin/python $SPLUNK_HOME/bin/<scriptname.py>
scriptSearchFilters = 1
```

Note: In previous releases, `getSearchFilter` could also be used to implement search filters for users who had been authenticated by Splunk's built-in system. This is no longer the case. Starting with 4.2, Splunk calls `getSearchFilter` only for users who have been authenticated by scripted auth.

In addition, if a call to `getSearchFilter` fails, Splunk will cancel the user's search and return an error message. This ensures that users cannot view results from unauthorized searches.

Securing Splunk Enterprise communications with SSL

About securing Splunk with SSL

This section describes the types of Splunk configurations you may want to secure with SSL.

About the default certificates

Splunk ships with, and is configured to use, a set of default certificates. These certificates will discourage casual snoopers but could still leave you vulnerable because the root certificate that ships with Splunk is the same root certificate in every download, and anyone with the same root certificate can authenticate.

The default certificates are generated and configured at startup and can be found in `$SPLUNK_HOME/etc/auth/`. They are set to expire three years after they are generated and new certificates must be created and configured at that time.

- For information about the default certificate for Splunk Web, see ["Turn on encryption \(https\) with Splunk Web."](#) or ["Turn on encryption \(https\) using web.conf."](#)
- For information about SSL for forwarding with the default certificate, see ["Configure Splunk forwarding to use the default certificate."](#)

Ways you can secure Splunk Enterprise

You can apply encryption and/or authentication using your own certificates for:

- Communications between the browser and **Splunk Web**
- Communication from Splunk **forwarders** to **indexers**
- Other types of communication, such as communications between Splunk instances over the management port

The table below describes the most common scenarios and the default SSL settings:

Type of exchange	Client function	Server function	Encryption	Certificate Authentication	Common Name	Type of data
------------------	-----------------	-----------------	------------	----------------------------	-------------	--------------

					checking	exchanged
Browser to Splunk Web	Browser	Splunk Web	NOT enabled by default	dictated by client (browser)	dictated by client (browser)	search term results
Inter-Splunk communication	Splunk Web	splunkd	enabled by default	NOT enabled by default	NOT enabled by default	search term results
Forwarding	splunkd as a forwarder	splunkd as an indexer	NOT enabled by default	NOT enabled by default	NOT enabled by default	data to be indexed

Communications between the browser and Splunk Web

Browser to Splunk Web data most commonly consists of search requests and returned data.

Data encryption (HTTPS) can be easily turned on using Splunk Web, or by editing the **configuration files**. Keep in mind that encryption with the default certificate protects against casual listening but is not fully secure.

For better security, replace the default certificates with certificates signed by a trusted CA. We strongly recommend using CA certs rather than signing your own in this case. Unless you have the ability to add your CA to the certificate stores in every browser that will access Splunk Web, a self-signed certificate is considered untrusted by users' browsers. For more information, see ["About securing Splunk Web."](#)

Splunk forwarders to indexers

Data sent from forwarders to indexers is the data that your indexers use for searches and reports. Depending upon your organization and the nature and format of the data being transmitted and Splunk configuration, this data may or may not be readable or sensitive.

Securing sensitive raw data helps to avoid snooping and man-in-the-middle attacks.

You can turn on SSL encryption using the default certificate to provide encryption and compression. However, communication using the default certificate does not provide secure authentication, as the certificate password is supplied with every installation of Splunk. The default certificates are set to expire three years after initial startup, and forwarder to indexer communications will fail at this point.

For better security, require certificate authentication using a self- or CA-signed certificate. A certificate signed by a known and mutually trusted Certificate Authority is considered more secure by outside parties than a certificate you sign yourself. For more information about using certificates with Splunk forwarders and indexers, see "[About securing data from forwarders.](#)"

Other Splunk SSL communications

Other Splunk communications happen between different instances of Splunk over the management port, usually but not always in a distributed environment. An example of this is configuration data sent by a deployment server to clients.

SSL encryption in this scenario is enabled by default. For most configurations this is adequate and is the recommended security method. However, if you do need to secure your communications with SSL authentication, we've provided some guidelines to help you in "[About securing Splunk to Splunk communication](#)" in this manual.

Getting your certificates

If you are experienced with SSL certificates, you can create them as you normally would and go straight to configuring your Splunk instances to use them.

If you need help getting your certificates together, we've provided very simple examples using OpenSSL commands. (OpenSSL ships with Splunk)

- [How to self-sign certificates](#)
- [How to get third-party certificates](#)
- [How to self-sign certificates for Splunk Web](#)
- [How to get third-party certificates for Splunk Web](#)

What to do when you have your certificates

The following topics provide more information about configuring Splunk to use your certificates once you have them:

- [Secure Splunk Web with your own certificate](#)
- [Configure Splunk forwarding to use your own certificates](#)
- [About securing inter-splunk communication](#)

About using SSL tools on Windows and Linux

This manual describes how to configure Splunk to use default, self-signed, or Certificate Authority signed certificates. For those who may not have certificates, we also provide simple examples for generating the certificates and keys using the command line and a version of OpenSSL that is packaged with Splunk.

Using the OpenSSL command-line examples

This manual provides a few basic examples for creating certificates using Splunk's version of OpenSSL in the command line. In order to perform these tasks you must have root administrator permissions. If you are working on a remote or virtual machine, you may have to take an extra step to ensure that you are able to perform all tasks:

- When working on a Windows platform, you may need to open the command line as the administrator: In the Start Menu, right click the .exe application and select **run as administrator**.
- When working on a *nix platform, you might need to use sudo to log in as the root administrator.

For more information about the differences between Windows and *nix, see the Administration Guide.

About SSL tools

Splunk ships with a recent version of OpenSSL at `$(SPLUNK_HOME)/splunk/lib`. For 6.0, Splunk supports OpenSSL with FIPS 140-2 enabled.

A variety of other SSL tools are available for purchase and download that you can use to create and set up certificates. If you do choose to use OpenSSL for certificate configuration, we strongly recommend that you use the version that ships with Splunk to avoid compatibility issues. To make sure that you are using the version provided with Splunk, set your environment to the version in `$(SPLUNK_HOME)/splunk/lib` or `$(SPLUNK_HOME)\splunk\bin` for Windows:

The following is an example of the library path for *nix:

```
export LD_LIBRARY_PATH=$(SPLUNK_HOME)/splunk/lib
```

The following is an example of the path for Windows:

```
set PATH = %PATH%;%SPLUNK_HOME%\bin
```

About FIPS

FIPS uses government-certified versions of some algorithms to meet regulatory guidelines. It should not be considered a security enhancement by itself and may potentially make your system slower. Splunk recommends that you only enable FIPS if it is a regulatory requirement for your environment.

If you are considering whether to enable FIPS, keep the following in mind:

- Splunk only supports OpenSSL with FIPS 140-2 (64-bit x86) on Linux, Windows, and Solaris.
- Though FIPS is disabled by default, it will be automatically enabled if you are running Splunk on a Linux machine with a kernel in FIPS mode.
- The FIPS module disables the use of some cryptographic algorithms in the instance of Python that Splunk uses to run apps (such as md5 and rc4). Make sure that any Splunk apps you intend to run are certified to run in FIPS mode and don't have dependencies on these algorithms.

To enable FIPS:

Before you start Splunk for the first time, edit

`$SPLUNK_HOME/etc/splunk-launch.conf` to add the following line:

```
SPLUNK_FIPS=1
```

Note: Upgrading a non-FIPS install to FIPS is not supported by Splunk ? the decision to use FIPS must be made at the original installation.

Configure allowed and restricted SSL versions

Splunk Enterprise 6.2 provides the `sslVersions` keyword to restrict older versions of protocols. SSLv3 is shipped out of box to support easy upgrades but should be disabled as soon as upgrades are complete. By default, Splunk Enterprise allows communications on SSLv3 and all subsequent versions.

When Splunk Enterprise is [configured in FIPS mode](#), SSLv2 and SSLv3 are always disabled regardless of any additional configuration.

CAUTION: In response to the v3 "POODLE" vulnerability, Splunk Enterprise strongly recommends updating your configuration to remove SSLv3 as upgrades are applied to your environment.

1. In `web.conf`, update the `sslVersions` attribute to list or limit the versions (separated by commas) you want Splunk Enterprise to support. By default this attribute is set to `*`, `-sslv2`, which is any version newer than SSLv2 (not recommended). For 6.2 the allowed SSL versions are:

- SSLv2 (not recommended)
- SSLv3 (not recommended)
- TLS1.0 (not recommended)
- TLS1.1
- TLS1.2

For example:

```
sslVersions = tls1.1, tls1.2
```

Syntax options

To select all supported versions use `"*"`:

```
sslVersions = *
```

To include all versions TLS1.0 or newer use `"tls"`:

```
sslVersions = tls
```

To restrict a particular version prefix it with `"-"` :

```
sslVersions = *, -ssl3
```

Note: When Splunk Enterprise is configured in FIPS mode, SSLv2 and SSLv3 are always disabled regardless of this configuration.

2. In `inputs.conf`, update the `sslVersions` attribute to list or limit the versions (separated by commas) you want Splunk Enterprise to support.

```
sslVersions = ssl2, tls1.1, tls1.2
```

You can use `"*"` to select all supported versions:

```
sslVersions = *
```

Simply use `"tls"` to include all versions TLS1.1 or newer:

```
sslVersions = tls
```

The prefix a version with `"-"` to restrict a particular version:

```
sslVersions = *, -ssl3
```

3. Configure forwarders to be compatible with your indexer. Changing or limiting the SSL versions (and restricting SSLv3) can create compatibility issues with forwarders, particularly those that run earlier versions of Splunk Enterprise. For

forwarders running 6.2 you can mitigate compatibility issues by also updating each forwarder's `inputs.conf` and `web.conf` settings in addition to your indexer.

Update any forwarders to 6.2 to be consistent with your indexer and the SSL settings (For purposes of backward compatibility, 6.0 can support up to `tls1.1`.)

4. Configure your `server.conf` file to accept connections with clients (for example, `web.conf` by editing the `sslVersions` attribute so that it is the same as your client(s).

Secure browser to Splunk Web communication

About securing Splunk Web

Information transmitted to **Splunk Web** mostly consists of search requests and results.

Note that browser to Splunk Web transmission does not always need to be secured. For example, if your users only access Splunk Web from a local browser behind the same firewall as Splunk Web, security may not be a concern. In this case simple encryption using Splunk's default certificates might be adequate.

- For information about the default certificate for Splunk Web, see ["Turn on encryption \(https\) with Splunk Web."](#) or ["Turn on encryption \(https\) using web.conf."](#)
- For information about SSL for forwarding with the default certificate, see ["Configure Splunk forwarding to use the default certificate."](#)

To turn on basic encryption, see ["Turn on encryption \(https\) with Splunk Web."](#)

On the other hand, if your Splunk configuration lives in a distributed environment where Splunk Web is accessed from browsers outside of firewalls from varied locations, stronger security should be implemented using signed certificates. For information about configuring Splunk Web to use signed certificates, see ["Secure Splunk Web using your own certificate."](#)

There are several ways you can use signed certificates to improve security for your browser to Splunk Web communications:

- **For secured encryption with authentication, you can replace the default certificate with a signed certificate.**
You replace the default certificate provided by Splunk with one that you request from a trusted Certificate Authority. This is the most secure option and recommended if security is a concern.
For more information about obtaining CA certificates for Splunk, see ["Get certificates signed by a third-party for Splunk Web."](#)
Note that you may also use self-signed certificates to secure authentication, however, because they are signed by you rather than a

known and trusted Certificate Authority, browsers will not have you as a CA in their certificate store and as a result will not trust you or your certificates. For self-signed certificates to be effective you would need the ability to add your certificate to a the certificate store of every single browser that will access Splunk Web.

For more information about creating self-signed certificates for Splunk, see ["Self-sign certificates for Splunk Web."](#)

- **When you use a signed certificate, you can further strengthen your SSL configuration by turning on common name checking.**

Common name checking adds an extra layer of security by requiring that the common name provided in the certificates on each communicating instance are a match. You can enable common name checking when setting up your certificate and instruct Splunk to check for that common name when authenticating.

For more information about configuring Splunk to use certificates and learn more about common name checking, see ["Secure Splunk Web using your own certificate."](#)

Turn on encryption (https) with Splunk Web

This topic explains how to use Splunk Web to enable HTTPS for browser to Splunk Web communication. Splunk can listen on HTTPS or HTTP, but not both.

The simple encryption that can be turned on in Splunk Web uses the default certificate that is provided in the "out of box" installation. Since every installation provides the same default certificate, this method is not highly secure. If security is a priority, Splunk strongly recommends you change the default certificate and configure authentication for better security. See ["Secure Splunk Web with your own certificate"](#) for information about replacing the default certificates.

To enable HTTPS with Splunk Web:

1. In Splunk Web, select **System > System settings**.
2. For **Enable SSL (HTTPS) in Splunk Web**, select the **Yes** radio button.

Splunk is already set to point to the default certificates when encryption is turned on, so no further action is needed.

3. Restart Splunk Web.

You must now prepend "https://" to the URL you use to access Splunk Web.

Turn on encryption (https) using web.conf

You can enable HTTPS through the `web.conf` configuration file. If it is not already present in your local directory, copy the default version of the file from `$SPLUNK_HOME/etc/system/default` to your local directory `$SPLUNK_HOME/etc/system/local/` or your own custom application directory in `$SPLUNK_HOME/etc/apps/`. For information on configuration files in general, see "About configuration files."

The encryption that can be turned on in the task described here is not secure. If security is a priority, Splunk strongly recommends you change the default certificate and configure authentication for better security. See "[Secure Splunk Web with your own certificate](#)" for information about replacing the default certificates.

To enable HTTPS through `web.conf`:

1. Set the `enableSplunkWebSSL` attribute to `true`:

```
[settings]
httpport = <https port number>
enableSplunkWebSSL = true
```

Note: By default, Splunk is set to point to the default certificates whenever encryption is turned on.

2. Restart Splunk Web, as described in "Start Splunk".

You must now prepend "https://" to the URL you use to access Splunk Web.

Secure Splunk Web with your own certificate

This example assumes that you have already generated self-signed certificates or purchased third-party certificates. If you have not done this and are unsure how to proceed, we've provided some simple examples:

- [Self-sign certificates for Splunk Web.](#)
- [Get a certificates signed by a third-party for Splunk Web.](#)

Note: Splunk Web does not currently support password-protected private keys. You should remove the password from your key before configuring Splunk Web for the certificate.

Before you begin: Copy your certificates to a new folder

Copy the server certificate to `$SPLUNK_HOME/etc/auth/splunkweb` or to your own certificate repository in `$SPLUNK_HOME/etc/auth`.

In the following example our web certificate is called `mySplunkWebCertificate.pem` and our private key is called `mySplunkWebPrivateKey.key`:

*nix:

```
# cp $SPLUNK_HOME/etc/auth/mycerts/mySplunkWebCertificate.pem
   $SPLUNK_HOME/etc/auth/mycerts/mySplunkWebPrivateKey.key
   $SPLUNK_HOME/etc/auth/splunkweb
```

Windows:

```
copy $SPLUNK_HOME\etc\auth\mycerts\mySplunkWebCertificate.pem
   $SPLUNK_HOME\etc\auth\splunkweb\
```

```
copy $SPLUNK_HOME\etc\auth\mycerts\mySplunkWebPrivateKey.key
   $SPLUNK_HOME\etc\auth\splunkweb\
```

Note: Do not overwrite or delete the existing certificates located in `$SPLUNK_HOME/etc/auth/splunkweb/`. The certificates at this location are automatically generated upon startup, meaning that any changes you make will be overwritten at startup. Instead, in the next steps, we will rewrite the relevant configuration file to point to your new certificate location.

Configure Splunk Web to use the key and certificate files

Note: Splunk Web does not support passwords for private keys, so you must remove the password from the key before using the key to secure Splunk Web.

1. In `$SPLUNK_HOME/etc/system/local/web.conf` (or any other applicable location, if you are using a deployment server), make the following changes to the `[settings]` stanza:

The following is an example of an edited settings stanza:

```
[settings]
enableSplunkWebSSL = true
```

```
privKeyPath = etc/auth/splunkweb/mySplunkWebPrivateKey.key
caCertPath = etc/auth/splunkweb/mySplunkWebCertificate.pem
```

2. Restart Splunk Web:

```
# $SPLUNK_HOME/bin/splunk restart splunkweb
```

Important note about the `caCertPath` attribute

When you use the `caCertPath` attribute to specify a new default certificate, make sure that the certificate file you specify:

- Contains at least a server SSL certificate and a certificate authority (CA) certificate.
- Contains multiple certificates in the proper sequence:
 - ◆ The server's SSL certificate.
 - ◆ Any intermediate certificates, if required.
 - ◆ The root certificate, if required.
- Uses absolute paths to certificate files, for best results. While you can use relative paths, those paths are relative to `$SPLUNK_HOME`, and you cannot change this setting.

The `caCertPath` attribute cannot point to a file that contains only a CA certificate, because with only a CA certificate, SSL cannot properly establish a secure connection. If you specify a certificate file that only contains a CA certificate, Splunk Web over HTTPS will not work.

For more information on how to set the `caCertPath` attribute, read about `web.conf` in the Admin Manual.

Verify your configuration

If your configuration is successful, you should be able to view your certificate using your browser tools. You will also see successful connection messages in `web_service.log` (located in `$SPLUNK_HOME/var/log/splunk`).

If you are not able to verify successful connection, see "[Troubleshoot your Splunk Web authentication](#)" for troubleshooting tips.

Troubleshoot your Splunk Web authentication

If you are unable to verify your certificate configuration, you can use the `web_service.log` in `$SPLUNK_HOME/var/log/splunk` to view and troubleshoot any errors that occur upon restart.

Look for SSL configuration warnings. For example, if you provide an incorrect path to the server certificate declared in `caCertPath`, Splunk Web fails to start and the following error appears:

```
2010-12-21 16:25:02,804 ERROR [4d11455df3182e6710] root:442 - [Errno 2] No such file or directory:
```

```
'/opt/splunk/share/splunk/mycerts/mySplunkWebCertificate.pem'
```

Note: If the private key is provided in `privKeyPath` is password protected, no error is provided but your browser won't load Splunk Web.

See ["Self-sign certificates for Splunk Web"](#) or ["Get certificates signed by a third party for Splunk Web"](#) for information about removing your password.

Secure Splunk forwarder to indexer communication

About securing data from forwarders

Forwarders send raw data to your **indexers**. This data can be vulnerable to snooping and corruption. If data is forwarded outside of a closed or co-located network, or if your data is very sensitive you should use SSL certificates to secure your data.

Using the default certificates will discourage casual snoopers but could still leave you vulnerable because the root certificate that ships with Splunk is the same root certificate in every download, and anyone with the same root certificate can authenticate. The default certificates are generated and configured at startup and can be found in `$SPLUNK_HOME/etc/auth/`.

Important: If you use the default certificates, keep in mind that they are set to expire three years after they are generated and new certificates must be created and configured at that time using one of the methods described in this manual.

For information about setting up SSL with the default certificate, see ["Configure Splunk forwarding to use the default certificate."](#)

To ensure that no one can easily snoop on your traffic or send data to your indexers, we recommend that you use new signed certificates that are either self-signed or purchased from a third-party certificate authority. To configure your forwarders and indexers to use certificates, see ["Configure Splunk forwarding to use your own certificates."](#)

There are several ways you can use self or CA-signed certificates to improve security for your forwarder to indexer:

- **You can replace the default certificates with certificates signed by your own root CA.**
You replace the default certificate provided by Splunk with one that you generate and sign yourself. For information about generating and self-signing certificates, see ["How to self-sign certificates."](#)
- **You can replace the default certificates with certificates signed by a trusted certificate authority.**

See ["How to get certificates signed by a third-party."](#)

- **You can further strengthen security by configuring common name checking.**

Common name checking adds an extra layer of security by requiring that the common name provided in the certificates on each indexer match the common name specified in the configuration file on the forwarder. You can also configure multiple certificates with different common names and distribute them to your indexers. You enable common name checking when setting up your certificate. See ["Configure Splunk forwarding to use your own certificates"](#) for more information.

Configure Splunk forwarding to use the default certificate

The default root certificate that ships with Splunk is the same root certificate in every download. That means that anyone who has downloaded Splunk has server certificates that have been signed by the same root certificate and would be able to authenticate to your certificates. To ensure that no one can easily snoop on your traffic or wrongfully send data to your indexers, we recommend that you replace them with signed certificates.

Important: The default certificates are set to expire three years after they are generated and new certificates must be created and configured at that time using one of the methods described in this manual.

To configure your forwarders to use certificates signed by your own root CA or a third-party CA see ["Configure Splunk forwarding to use your own certificates."](#)

In this topic we describe how to:

- Configure the indexer to use the default certificates that ship with Splunk
- Configure the forwarder to use the default certificates that ship with Splunk

Note: When configuring multiple forwarders, configure each forwarder to use the default certificate separately.

Set up the indexer to use the default server certificate

1. In `$(SPLUNK_HOME)/etc/system/local/inputs.conf` (or the appropriate directory of any app you are using to distribute your forwarding configuration), set up the

following stanzas:

For this example, we are using port 9997 to receive data from forwarders.

```
[SSL]
rootCA = $SPLUNK_HOME/etc/auth/cacert.pem
serverCert = $SPLUNK_HOME/etc/auth/server.pem
password = password
```

```
[splunktcp-ssl:9997]
disabled=0
```

Where `rootCA` is the path to the CA public key and `serverCert` is the path to the default server certificate.

The default certificate can be found at `$SPLUNK_HOME/etc/auth/server.pem`.

Note: When using the default certificate it is not necessary to set `requireClientCert = true` since we do not need to check the validity of the default server certificate.

2. Restart splunkd:

```
$SPLUNK_HOME/bin/splunk restart splunkd
```

Configure the forwarder

Set up your forwarder to use the same default certificates as your indexer and configure the forwarder to send data to the configured listening port.

In the following example, the indexer IP address is 10.1.12.112.

1. Define the following stanzas in `$SPLUNK_HOME/etc/system/local/outputs.conf` (or in the appropriate directory of any app you are using to distribute your forwarding configuration):

```
[tcpout]
defaultGroup = splunkssl
[tcpout:splunkssl]
server = 10.1.12.112:9997
sslVerifyServerCert = false
sslRootCAPath = $SPLUNK_HOME/etc/auth/cacert.pem
sslCertPath = $SPLUNK_HOME/etc/auth/server.pem
sslPassword = password
```

Where `rootCA` is the path to the CA public key and `serverCert` is the path to the default server certificate.

Note again that you do not need to set `sslVerifyServerCert = true`. It is not necessary to ask the forwarders to check the validity of the default server certificate.

2. Restart splunkd:

```
# $SPLUNK_HOME/bin/splunk restart splunkd
```

Next steps

Next, you should check your connection to make sure your configuration works. See "[Validate your configuration](#)" for more information.

Configure Splunk forwarding to use your own certificates

This topic describes how to configure Splunk to send data from your forwarders to your indexers using your own SSL certificates. Using certificates to secure the data from forwarders helps ensure that your data is securely encrypted when in transit on your network. This topic describes the following steps:

- Configure the indexer(s) to use a new signed certificate, as described in this topic.
- Configure the forwarder(s) to use a new signed certificate, as described in this topic.

Before you begin, you must procure and prepare your certificates. Make sure your certificates are PEM files in x509 format and that your key is in RSA format. If you need help, we've provided a few simple examples to help you create and prepare your own certificates. See "[About securing data from forwarders](#)" and "[About securing inter-Splunk communication](#)" for more information.

You can also create multiple certificates (signed by the same CA) with different common names and distribute those to your indexers for added security.

Configure your indexer to use your certificates

1. Copy your server certificate and CA public certificate into an accessible folder on the indexer(s) you intend to configure.

For this example, we are calling these certificates `myNewServerCertificate.pem` and `myCACertificate.pem` and are placing the certificates in the following path:

```
$SPLUNK_HOME/etc/auth/mycerts/
```

Warning: If you configure `inputs.conf` or `outputs.conf` in an app directory, the password is NOT encrypted and the clear-text value remains in the file. For this reason, you may prefer to create different certificates (signed by the same root CA) to use when configuring SSL in app directories.

2. Configure `inputs.conf` on the indexer(s) to use the new server certificate. In `$SPLUNK_HOME/etc/system/local/inputs.conf` (or in the appropriate directory of any app you are using to distribute your forwarding configuration), set up the [SSL] stanzas:

Note that this example uses port 9997 (which is the default) to receive data from the forwarders.

```
[SSL]
rootCA = $SPLUNK_HOME/etc/auth/mycerts/myCACertificate.pem
serverCert = $SPLUNK_HOME/etc/auth/mycerts/myNewServerCertificate.pem
password = <server certificate private key password>
cipherSuite = <your chosen cipher suite (optional)>
```

```
[splunktcp-ssl:9997]
compressed = true
```

When you edit the file in `$SPLUNK_HOME/etc/system/local/inputs.conf`, Splunk encrypts the password and overwrites the clear-text server certificate password you provided when you restart Splunk.

Note: We do not recommend that you disable `tls` compression, as it can cause bandwidth issues.

3. Restart `splunkd`.

```
# $SPLUNK_HOME/bin/splunk restart splunkd
```

Configure your forwarders to use your certificates

1. Copy the server certificate (this can be your existing server certificate or a new certificate you created specifically for this forwarder. For convenience we are calling it `myNewServerCertificate.pem` in this example) and the CA public certificate `myCACertificate.pem` into an accessible folder on the forwarders you plan to configure. For this example, we are placing them in

```
$SPLUNK_HOME/etc/auth/mycerts/.
```

Warning: If you configure `inputs.conf` or `outputs.conf` in an app directory, the password is NOT encrypted and the clear-text value remains in the file. For this reason, you may prefer to create different certificates (signed by the same root

CA) to use when configuring SSL in app directories.

2. Define the [SSL] stanza in `$SPLUNK_HOME/etc/system/local/outputs.conf` (or in the appropriate directory of any app you are using to distribute your forwarding configuration):

```
[tcpout]
defaultGroup = splunkssl

[tcpout:splunkssl]
server = 10.1.12.112:9997
sslRootCAPath = $SPLUNK_HOME/etc/auth/mycerts/myCACertificate.pem
sslCertPath = $SPLUNK_HOME/etc/auth/mycerts/myNewServerCertificate.pem
sslPassword = default
sslVerifyServerCert = true
```

Note that when you save the file in

`$SPLUNK_HOME/etc/system/local/outputs.conf`, Splunk encrypts and overwrites the clear-text server certificate password when splunkd restarts.

3. Restart splunkd.

```
# $SPLUNK_HOME/bin/splunk restart splunkd
```

Other configuration options

To forward data to more than one indexer

To configure a forwarder to authenticate to multiple indexers, simply add their HOST:PORT address as a comma-separated list in the "server" configuration parameter of the target group definition stanza.

The following example uses the same certificate for the indexer and the forwarders:

```
[tcpout]
defaultGroup = splunkssl

[tcpout:splunkssl]
server = 10.1.12.112:9997,10.1.12.111:9999
sslRootCAPath = $SPLUNK_HOME/etc/auth/mycerts/myCACertificate.pem
sslCertPath =
$SPLUNK_HOME/etc/certs/auth/mycerts/myNewServerCertificate.pem
sslPassword = <your server private key password>
sslVerifyServerCert = true
```

The following example uses the certificate `myForwarderCertificate.pem`

```
[tcpout]
defaultGroup = splunkssl
[tcpout:splunkssl]
server = 10.1.12.112:9997,10.1.12.111:9999
sslRootCAPath = $SPLUNK_HOME/etc/auth/mycerts/myCACertificate.pem
sslCertPath = $SPLUNK_HOME/etc/auth/mycerts/myForwarderCertificate.pem
sslPassword = <your server certificate private key password>
sslVerifyServerCert = true
```

To forward data to multiple indexers using certificates with different common names

You can create and configure one server certificate for each indexer by configuring the forwarder's `outputs.conf` with one server-specific `[SSLConfig]` stanza per indexer.

If you have created one server certificate per indexer and you have set a unique common name (CN) in each indexer certificate to be checked by the forwarders, you will need to configure one `[tcpout-server://HOST:PORT]` configuration stanza per indexer in `outputs.conf`. This is so that you can specify which common name to check for which indexer.

The example below shows a forwarder that will check that the indexer `@10.1.12.112` displays a server certificate with `CN="phobos"` and that the indexer `@10.1.12.113` displays a server certificate with `CN="deimos"`. If one of those conditions is not matched, the forwarder will refuse to send data to that indexer :

```
[tcpout]
defaultGroup = splunkssl

[tcpout:splunkssl]
server = 10.1.12.112:9997

[tcpout-server://10.1.12.112:9997]
sslRootCAPath = $SPLUNK_HOME/etc/certs/myCACertificate.pem
sslCertPath = $SPLUNK_HOME/etc/certs/myNewServerCertificate.pem
sslPassword = <your server private key password>
sslVerifyServerCert = true
sslCommonNameToCheck = phobos

[tcpout-server://10.1.12.113:9997]
sslRootCAPath = $SPLUNK_HOME/etc/certs/myCACertificate.pem
sslCertPath = $SPLUNK_HOME/etc/certs/myNewServerCertificate.pem
sslPassword = <your server private key password>
sslVerifyServerCert = true
sslCommonNameToCheck = deimos
```

This ensures that the forwarder will only send data to an indexer presenting a certificate signed by the same CA and that this certificate was issued to someone with the expected name (CN). Splunk matches the name in the Splunk configuration file ("sslCommonNameToCheck") and the common name signed into the indexer's certificate.

Next steps

Next, you should check your connection to make sure your configuration works. See "[Validate your configuration](#)" for more information.

Validate your configuration

Before you deploy your configuration, you can use `splunkd.log` to validate and troubleshoot your configuration. `Splunkd.log` is located on your indexer and forwarder at `$SPLUNK_HOME/var/log/splunkd.log`.

On the indexer, look for the following or similar messages at the start-up sequence to verify a successful connection:

```
02-06-2011 19:19:01.552 INFO TcpInputProc - using queueSize 1000
02-06-2011 19:19:01.552 INFO TcpInputProc - SSL
cipherSuite=ALL:!aNULL:!eNULL:!LOW:!EXP:RC4+RSA:+HIGH:+MEDIUM
02-06-2011 19:19:01.552 INFO TcpInputProc - supporting SSL v2/v3
02-06-2011 19:19:01.555 INFO TcpInputProc - port 9997 is reserved for
splunk 2 splunk (SSL)
02-06-2011 19:19:01.555 INFO TcpInputProc - Port 9997 is compressed
02-06-2011 19:19:01.556 INFO TcpInputProc - Registering metrics callback
for: tcpin_connections
```

On the forwarder, look for the following or similar messages at the start-up sequence to verify a successful connection:

```
02-06-2011 19:06:10.844 INFO TcpOutputProc - Retrieving configuration
from properties
02-06-2011 19:06:10.848 INFO TcpOutputProc - found Whitelist
forwardedindex.0.whitelist , RE : forwardedindex.0.whitelist
02-06-2011 19:06:10.848 INFO TcpOutputProc - found Whitelist
forwardedindex.1.blacklist , RE : forwardedindex.1.blacklist
02-06-2011 19:06:10.848 INFO TcpOutputProc - found Whitelist
forwardedindex.2.whitelist , RE : forwardedindex.2.whitelist
02-06-2011 19:06:10.850 INFO TcpOutputProc - Will retry at max backoff
sleep forever
02-06-2011 19:06:10.850 INFO TcpOutputProc - Using SSL for server
10.1.12.112:9997, sslCertPath=/opt/splunk/etc/aut/server.pem
02-06-2011 19:06:10.854 INFO TcpOutputProc - ALL Connections will use
```

```
SSL with sslCipher=  
02-06-2011 19:06:10.859 INFO TcpOutputProc - initializing single  
connection with retry strategy for 10.1.12.112:9997
```

For help troubleshooting your configuration issues, see "[Troubleshoot your forwarder to indexer configuration](#)" in this manual.

Troubleshoot your forwarder to indexer authentication

1. Check `$SPLUNK_HOME/var/log/splunk/splunkd.log` (indexer and forwarder) for errors. On the indexer, check for the messages from the TCP input processor `TcpInputProc`. On the forwarder, check the messages from the TCP output processor `TcpOutputProc`.
2. Increase the logging level of the appropriate processors on the indexer and the forwarder in `$SPLUNK_HOME/etc/log.cfg`.

On the forwarder, set `category.TcpOutputProc=DEBUG`, on the indexer set `category.TcpInputProc=DEBUG`.

3. Restart Splunk for these to take effect and observe the start-up sequence for the pertinent component. Most configuration issues are explicitly revealed by this method.
4. Check the SSL configuration as it is seen by Splunk using `btool`.

On the indexer :

```
$SPLUNK_HOME/bin/splunk cmd btool inputs list --debug
```

On the forwarder :

```
$SPLUNK_HOME/bin/splunk cmd btool outputs list --debug
```

Common problems

- The path to the server certificate file set as the value of `serverCert` in `inputs.conf` is wrong, or the file cannot be read. This will generate the following error :

```
12-16-2010 16:07:30.965 ERROR SSLCommon - Can't read certificate file  
/opt/splunk/etc/auth/server.pem errno=33558530 error:02001002:system  
library:fopen:No such file or directory
```

- The password to the RSA private key contained in the server certificate file is wrong.

```
12-07-2010 07:56:45.663 ERROR SSLCommon - Can't read key file  
/opt/splunk/etc/auth/server.pem
```

On *nix, you can manually test the password of the RSA key contained in the file with the comand:

```
# openssl rsa -in /opt/splunk/etc/auth/server.pem -text
```

On Windows, you can manually test the password of the RSA key using the following command:

```
>openssl.exe rsa -in "c:\Program Files\Splunk\etc\auth\server.pem" -text
```

Secure distributed environments

About securing inter-Splunk communication

This topic discusses the following "Splunk to Splunk" types of communications and how you can secure them:

- [Securing distributed search heads and peers](#)
- [Securing deployment server and clients](#)
- [About securing clusters](#)

Securing distributed search heads and peers

Distributed search configurations share search information, knowledge objects and app and configuration information over the management port.

Communication between search heads and peers relies on public-key encryption. Upon startup, Splunk generates a private key and public key on your Splunk installation. When you configure distributed search on the search head, the public keys are distributed by search heads to peers and those keys are used to secure communication. This default configuration provides built-in encryption as well as data compression that improves performance.

It is possible to swap these generated keys out with your own keys. However, it is not recommended and considered generally unnecessary. To configure public-key encryption for distributed search setups, you create your keys and distribute them to your search heads and peers. To learn more about distributing key files to distributed search peers, look in the section on configuring distributed search in the Distributed Search manual: "Distribute the key files".

Secure your deployment server and clients

Communication between the deployment server and client involve configuration information over the management port. These components do not transmit search data.

SSL encryption using the default certificates, which is enabled out-of-the-box, is strongly recommended as the minimum for any data communicated over the

management port. Splunk's default SSL configuration provides built-in encryption as well as data compression that improves performance.

It's important to know that configuring certificate authentication for a deployment server and clients impacts the rest of your configuration:

- Splunk Web will fail to authenticate unless you configure it to also use your certificate.
- The CLI will be unable to communicate with the deployment server.

However, you may find it necessary in certain distributed configurations, perhaps where extremely sensitive server configuration data is sent to a variety of locations outside your firewall.

With that in mind, we've provided some guidelines for configuring SSL for Splunk to Splunk communications.

To configure SSL certificates between search heads or peers, you should:

1. Create one or more certificates using the same root CA.
2. Distribute the certificates to your deployment server and clients.
3. Edit `server.conf` to provide the location of your certificates:

```
[sslConfig]
enableSplunkdSSL = true
sslKeysfile = server.pem
sslKeysfilePassword = password
caCertFile = cacert.pem
caPath = $SPLUNK_HOME/etc/auth
```

4. Edit `server.conf` to authenticate against your certificates by adding the following attribute to the `[sslConfig]` stanza in previous step:

```
requireClientCert = true
```

Important: This `requireClientCert` is set to "false" by default. If you change it to true to force Splunk to check your client's certificates, Splunk Web and the CLI will also be checked for certificates. Your CLI connection will no longer work because your CLI is unable to present a certificate as a client.

5. Edit `web.conf` to present a certificate signed by the same root CA so that Splunk Web can connect to the server.

The following is an example of an edited settings stanza:

```
[settings]
enableSplunkWebSSL = true
privKeyPath = etc/auth/splunkweb/mySplunkWebPrivateKey.key
caCertPath = etc/auth/splunkweb/mySplunkWebCertificate.pem
cipherSuite = <your chosen cipher suite (optional)>
```

Note: Splunk Web does not support passwords, so you must remove the password from the private key. For more information, see ["Get certificates signed by a third party for Splunk Web."](#)

Secure your clusters

Splunk provides a "shared secret" capability to help your clustering nodes authenticate to each other. When you set up a cluster, you determine the shared secret. You give each node the same secret.

For optimal performance, Splunk recommends that you use the default SSL encryption for your inter-Splunk data communication (already turned on by default) and then leverage Splunk's built-in "shared secret" functionality.

For more information about setting a shared secret using Splunk Web, see "Enable the master node".

For more information about using the shared secret in the CLI, see "Configure the cluster in server.conf" in *Managing Indexers and Clusters*.

Audit Splunk Enterprise activity

Use Splunk to audit your system activity

Knowing what is happening in your system is vital to keeping it secure.

To make the most of your system and keep it secure, we recommend the following best practices:

- Perform a periodic review of Splunk access and audit logs.
- Perform a periodic review of Splunk server audit and security logs.
- Perform a periodic review of all Splunk users and roles.

Search for audit events

Search audit events in Splunk Web or in Splunk's CLI. To do this, pipe your searches to the audit command. The audit search command is most useful if audit event signing *has* been configured. However, if you want to search for all audit events where audit event signing *has not* been configured (or to skip integrity validation) you may search the whole audit index.

- To search for all audit events, specify the `_audit` index:

```
index=_audit
```

This search returns all audit events.

- Pipe your search to the audit command:

```
index=_audit | audit
```

This search returns the entire audit index, and processes the audit events it finds through the `audit` command.

Narrow your search before piping to the audit command. However, you can only narrow the time range, or constrain by a single host. This is because each host has its own ID number sequence. Since sequential IDs exist to enable detection of gaps in audit events, narrowing a search across multiple hosts causes false gap detection.

The field that contains the status of the event is called "validity". Values can be:

- VALIDATED - no gap before this event and event signature matches
- TAMPERED - event signature does not match
- NO SIGNATURE - the signature was not found

The field that contains the gap status is called "gap". Values can be:

- TRUE - a gap was found
- FALSE - no gap was found
- N/A - no id was found

Manage data integrity

The Splunk Enterprise data integrity control feature provides a way to verify the integrity of data that is indexed.

When you enable data integrity control for an index, Splunk Enterprise computes hashes (using SHA 256) on every slice of data and stores those hashes so that you can go back later and verify the integrity of your data.

How it works

When you enable data integrity control, Splunk Enterprise computes hashes on every slice of newly indexed raw data and writes it to a `11Hashes` file. When the bucket rolls from hot to warm, Splunk Enterprise computes a hash on the contents of the `11Hashes` and stores the computed hash in `12Hash`. Both hash files are stored in the `rawdata` directory for that bucket.

Note that data integrity control hashes newly indexed data, data coming from a forwarder should be secured and encrypted with SSL. For more information, see [About securing Splunk with SSL](#).

Check your hashes to validate your data

To check Splunk Enterprise data, run the following CLI command to verify the integrity of an index or bucket:

```
./splunk check-integrity -bucketPath [ bucket path ] [ verbose ]
```

```
./splunk check-integrity -index [ index name ] [ verbose ]
```

Configure data integrity control

To configure Data Integrity Control, edit `indexes.conf` to enable the `enableDataIntegrityControl` attribute for each index. The default value for all indexes is `false` (off).

```
enableDataIntegrityControl=true
```

Data Integrity in clustered environments

In a clustered environment, the cluster master and all the peers must run Splunk Enterprise 6.3 to enable accurate index replication. Push the bundle with the data integrity configuration from the cluster master to the peers.

For more information about managing configurations in a clustered environment, see the Distributed Deployment Manual

Optionally modify the size of your data slice

By default, data slices are set to 128kb, which means that a data slice is created and hashed every 128KB. You can optionally edit `indexes.conf` to specify the size of each slice.

```
rawChunkSizeBytes = 131072
```

Store and secure your data hashes

For optimal security, you can optionally store your hashes outside the system where the data is hosted, such as a different server. To avoid naming conflicts, store your secured hashes in separate directories.

Regenerate hashes

If you lose your hashes for a bucket, Use the following CLI command to re-generate hash files on a bucket or index. This command extracts the hashes embedded in the journal:

```
./splunk generate-hash-files -bucketPath [ bucket path ] [ verbose ]
```

```
./splunk generate-hash-files -index [ index name ] [ verbose ]
```

Best practices for Splunk Enterprise security

Secure your service accounts

Practice the principle of least privilege by running Splunk as an unprivileged user rather than using a privileged account such as root or Administrator.

- On Unix or Linux, use the "splunk" user created with the PKG or RPM packages, or create your own user that only has privilege and ownership over `$SPLUNK_HOME`.
- On Windows, the local system context is often the best choice. However, if you require communication using a windows communication channel, such as WMI, use a restricted access account.

Deploy secure passwords across multiple servers

At initial startup, Splunk creates a file `$SPLUNK_HOME/etc/auth/splunk.secret`. This file contains a key used to encrypt some of your authentication information in configuration files:

- `web.conf`: Your SSL passwords on every instance.
- `authentication.conf`: Your LDAP passwords, if you have any.
- `inputs.conf`: Your SSL passwords, if you use `splunktcp-ssl`.
- `outputs.conf`: Your SSL passwords, if you use `splunktcp-ssl`.
- `server.conf`: `pass4symmkey`, if you have one.

When Splunk starts, if it detects a clear-text password, in one of these settings, it will create or overwrite the configuration in the equivalent local folder with the encrypted password.

Note: If the `pass4symmkey` or `SSLPassword` is specified in a default apps file, the password is obfuscated in the local version of the file upon restart. The default version of the file remains in clear text. However, if the file is listed using `curl` or a `splunkd` endpoint, the passwords appear encrypted.

When deploying Splunk on multiple servers, you can do the following to encrypt these passwords and ensure that they are consistent across your deployment. You should perform these steps at initial deployment and also any time you need

to deploy a new password for your instances:

1. Configure one Splunk instance and modify any passwords as necessary. (If this is a new configuration, do not start any other instances yet.)
2. Restart the configured instance to encrypt the passwords in the file. Password information is stored in clear text until it is encrypted at restart.
3. Copy the encrypted `splunk.secret` file from your configured instance to all of your other instances.
4. Start all new instances to which you copied the file or restart existing instances if you are distributing a modified file after deployment.

Note: This procedure is not relevant to search head clusters. In a search head cluster, the captain replicates its `splunk.secret` file to all other cluster members during initial deployment of the cluster, so you do not need to copy it manually. As part of its normal operation, the cluster also automatically replicates any credentials that are stored by apps for their own use.

How pass4symkey hardens your configuration

If a clear-text `pass4symkey` is specified in `etc/apps/myapp/default/server.conf`, an obfuscated version gets written in `system/local/server.conf` upon restart.

This is the intended behavior. Usually configuration files in default are read only, and the information is also written to the local file, which is editable.

When the configuration is listed using `curl` or a `splunkd` endpoint, the `pass4SymmKey` appears encrypted. If the configuration location is read-only like, Splunk likewise writes to local. (This is also how `sslPassword` works.)

For more information on using `pass4symkey`, see "Configuration methods."

Harden your KV store port

In your search head cluster, port 8191 is opened to the network so that other members of the cluster can replicate KV store data. This creates an additional service exposure in clustered environments.

We recommend that you secure your environment by restricting the 8191 port access to members of your search head cluster.

For more information about working with KVstore, see [About the app key value store](#)

Use Splunk Enterprise Access Control Lists

To help secure your Splunk configuration, use the Splunk Enterprise Access Control Lists (ACLs) to limit the IP addresses that can access various parts of your networks.

To configure ACLs, you edit `server.conf` and `inputs.conf` to specify the IP addresses that will be accepted or rejected for various communications.

How to set up ACLs

The addresses are separated by commas or spaces. You can provide the addresses in the following formats:

- A single IPv4 or IPv6 address. For example: `10.1.1.2.3`, `fe80::4a3`.
- A CIDR block of addresses. For example: `10/8`, `fe80:1234/32`.
- A DNS name, possibly with an `*` used as a wildcard, for example: `myhost.example.com`, `*.splunk.com`.
- A single `*` which matches anything (this is the default value).

To add addresses that you wish to include, you add the addresses in one of the formats described below. To exclude an address you prefix the address with '!'.

Rules are applied in order, and the first one to match is used. For example, `!10.1/16, *` will allow connections from everywhere except the `10.1.*.*` network.

Where to set up ACLs

You can secure IP addresses for the following connections by editing the `[Accept from]` value:

- To instruct a node to only accept replicated data from other nodes with specific IPs, edit the `httpServer` stanza in `server.conf`. If you set this attribute, you must make sure that you include the IP addresses of all other peers in the cluster. For more information about

clusters, see "About clusters and index replication" For more information about editing `server.conf`, see `server.conf`.

- To restrict TCP communications to specific IP addresses, edit the `tcp` stanza in `inputs.conf`. Be careful, as this will overwrite the output values in `server.conf` if the information conflicts.
- To restrict TCP communications that use SSL to specific IP addresses, edit the `tcp-ssl` stanza in `inputs.conf`.
- To restrict your indexer to accept data only from forwarders with specific IP addresses, edit the `splunktcp` stanza in `inputs.conf`. This prevents someone from spoofing your forwarders and possibly corrupting your data.
- If your forwarder to indexer communications are secured with SSL, edit the `splunktcp-ssl` stanza in `inputs.conf` to restrict your indexer to only accept data from forwarders with specific IP addresses.
- To restrict UDP communications to specific IP addresses, edit the `UDP` stanza in `inputs.conf`.

For more information about editing `inputs.conf`, see `inputs.conf`

Safeguards for risky commands

Splunk Enterprise contains built-in search processing language (SPL) safeguards to warn you when you are about to unknowingly run a search that contains commands that might be a security risk. This warning appears when you click a link or type a URL that loads a search that contains risky commands.

The warning does not appear when you create ad hoc searches.

This warning alerts you to the possibility of unauthorized actions by a malicious user. Unauthorized actions include:

- Copying or transferring data (data exfiltration)
- Deleting data
- Overwriting data

A possible scenario when this might occur is when a malicious person creates a search that includes commands that exfiltrate or damage data. The malicious

person then sends an unsuspecting user a link to the search. The URL contains a query string (q) and a search identifier (sid), but the sid is expired. The malicious person hopes the user will use the link and the search will run.

Commands that trigger the warning

The commands that trigger this warning are listed here:

- collect
- crawl
- dump
- delete
- input
- outputcsv
- outputlookup
- runshellscript
- script
- sendalert
- sendemail
- tscollect

Actions in the warning dialog box

Instead of running the search immediately, Splunk Enterprise analyzes the search for risky commands. If one or more risky commands are identified, a warning dialog box appears. You have the option to cancel, run, or investigate the search.

Cancel

Closes the warning dialog box. The search does not run and the search is removed from the Search bar. Closing the dialog box, by clicking the Close button (X), is the same as clicking **Cancel**.

Run

Runs the search.

Investigate

Displays the search in the Search bar so that you can review the SPL. Use this option to copy the syntax of the search. Send a copy of the search, along with any information about the source of the link, to your system administrator.

Turning off the warning

Only users with Write permission can edit the `web.conf` file to turn off the warning dialog box.

You can turn off the warning for a specific command, or for all of the risky commands.

Turn off the warning for a specific command

1. Copy the `commands.conf.spec` file, which is located in the `$SPLUNK_HOME/etc/system/default` directory.
2. Paste the copy of the file in the `$SPLUNK_HOME/etc/system/local` directory.
3. Locate the command and change the setting from `is_risky = true` to `is_risky = false`.
4. Restart Splunk Enterprise.

Turn off the warning for all of the commands

1. Open the `web.conf` file. This file is located in the `$SPLUNK_HOME/etc/system/default/` directory.
2. Change the `enable_risky_command_check` parameter to `false`.
3. Restart Splunk Enterprise.

See also

In the *Admin Manual*:

About configuration files
commands.conf file
web.conf file

Appendix A: How to get SSL certificates

How to self-sign certificates

This topic describes one way you can use OpenSSL to self-sign certificates for securing forwarder-to-indexer and Inter-Splunk communication.

If you already possess or know how to generate the needed certificates, you can skip this topic and go directly to the configuration steps, described later in this manual:

- [How to prepare your signed certificates for Splunk](#)
- [Configure Splunk forwarding to use your own certificates](#)
- [About securing inter-Splunk communication](#)

Self-signed certificates are best used for data communication that happens within an organization or between known entities. If you are for any reason communicating with unknown entities we recommend using CA-signed certificates to secure your data.

Before you begin

In this discussion, `$SPLUNK_HOME` refers to the Splunk installation directory. On Windows, Splunk is installed at `C:\Program Files\splunk` by default. For most Unix platforms, the default installation directory is at `/opt/splunk`; for Mac OS, it is `/Applications/splunk`. See the Administration Guide to learn more about working with Windows and *nix.

Make sure to use the version of OpenSSL provided with Splunk by setting your environment to the version in `$SPLUNK_HOME/splunk/lib` in *nix or `$SPLUNK_HOME\splunk\bin` in Windows. To do this, you can:

- Run `$SPLUNK_HOME/bin/setSplunkEnv` before you create your certificates.

or

- Navigate to `$SPLUNK_HOME/bin/` and use `./openssl` to run the certificate-generating commands.

Create a new directory for your certificates

Create a new directory to work from when creating your certificates. In our example, we are using `$(SPLUNK_HOME)/etc/auth/mycerts`:

```
# mkdir $(SPLUNK_HOME)/etc/auth/mycerts
# cd $(SPLUNK_HOME)/etc/auth/mycerts
```

Splunk strongly recommends that you create a new folder and do not overwrite the existing certificates stored in `$(SPLUNK_HOME)/etc/auth`. Working in a new directory protects the certificates that ship with Splunk in `$(SPLUNK_HOME)/etc/auth` and lets you use them for other Splunk components as necessary.

Create the root certificate

First you create a root certificate that serves as your root certificate authority. You use this root CA to sign the server certificates that you generate and distribute to your Splunk instances.

Generate a private key for your root certificate

1. Create a key to sign your certificates.

In *nix:

```
# openssl genrsa -des3 -out myCAPrivateKey.key 2048
```

In Windows you may need to append the location of the `openssl.cnf` file:

```
>openssl genrsa -des3 -out myCAPrivateKey.key 2048
```

2. When prompted, create a password for the key.

When the step is completed, the private key `myCAPrivateKey.key` appears in your directory.

Generate and sign the certificate

1. Generate a new Certificate Signing Request (CSR):

In *nix:

```
# openssl req -new -key myCAPrivateKey.key -out myCACertificate.csr
```

In Windows:

```
>openssl req -new -key myCAPrivateKey.key -out myCACertificate.csr
-config $(SPLUNK_HOME)\openssl.cnf
```

2. When prompted, enter the password you created for the private key in `$(SPLUNK_HOME)/etc/auth/mycerts/myCAPrivateKey.key`.

3. Provide the requested certificate information, including the common name if you plan to use common name checking in your Splunk configuration.

A new CSR `myCACertificate.csr` appears in your directory.

4. Use the CSR `myCACertificate.csr` to generate the public certificate:

In *nix:

```
# openssl x509 -req -in myCACertificate.csr -sha256 -signkey myCAPrivateKey.key -CAcreateserial -out myCACertificate.pem -days 1095
```

In Windows:

```
>openssl x509 -req -in myCACertificate.csr -sha256 -signkey myCAPrivateKey.key -CAcreateserial -out myCACertificate.pem -days 1095
```

5. When prompted, enter the password for the private key `myCAPrivateKey.key`.

A new file `myCACertificate.pem` appears in your directory. This is the public CA certificate that you will distribute to your Splunk instances.

Create the server certificate

Now that you have created a root certificate to serve as your CA, you must create and sign your server certificate.

Important: This example shows you how to create a new private key and server certificate. You can distribute this server certificate to all forwarders, indexers as well your Splunk instances that communicate on the management port. If you want to use a different common names for each instance, you simply repeat the process described here to create different certificates (each with a different common name) for your Splunk instances.

For example, if configuring multiple forwarders, you can use the following example to create the certificate `myServerCertificate.pem` for your indexer, then create another certificate `myForwarderCertificate.pem` using the same root CA and install that certificate on your forwarder. Note that an indexer will only accept a properly generated and configured certificate from a forwarder that is signed by the same root CA.

See "[Configure Splunk forwarding to use your own certificates](#)" for more

information about configuring your forwarders and indexers.

Generate a key for your server certificate

1. Generate a new RSA private key for your server certificate. In this example we are again using DES3 encryption and a 2048 bit key length:

In *nix:

```
# openssl genrsa -des3 -out myServerPrivateKey.key 2048
```

In Windows:

```
# openssl genrsa -des3 -out myServerPrivateKey.key 2048
```

2. When prompted, create a new password for your key.

A new key `myServerPrivateKey.key` is created. This key will be used to encrypt the outgoing data on any Splunk instance where you install it as part of the server certificate.

Generate and sign a new server certificate

1. Use your new server private key `myServerPrivateKey.key` to generate a CSR for your server certificate.

In *nix:

```
# openssl req -new -key myServerPrivateKey.key -out  
myServerCertificate.csr
```

In Windows:

```
openssl req -new -key myServerPrivateKey.key -out  
myServerCertificate.csr -config $SPLUNK_HOME\openssl.cnf
```

2. When prompted, provide the password to the private key

`myServerPrivateKey.key`.

3. Provide the requested information for your certificate, including a Common Name if you plan to configure Splunk to authenticate via common-name checking.

A new CSR `myServerCertificate.csr` appears in your directory.

4. Use the CSR `myServerCertificate.csr` and your CA certificate and private key to generate a server certificate.

In *nix:

```
# openssl x509 -req -in myServerCertificate.csr -sha256 -CA  
myCACertificate.pem -CAkey myCAPrivateKey.key -CAcreateserial -out  
myServerCertificate.pem -days 1095
```

In Windows:

```
# openssl x509 -req -in myServerCertificate.csr -sha256 -CA  
myCACertificate.pem -CAkey myCAPrivateKey.key -CAcreateserial -out  
myServerCertificate.pem -days 1095
```

5. When prompted, provide the password for the certificate authority private key `myCAPrivateKey.key`. Make sure to sign this with your private key and not the server key you just created.

A new public server certificate `myServerCertificate.pem` appears in your directory.

Next steps

You should now have the following files in the directory you created, which is everything you need to configure indexers, forwarders, and Splunk instances that communicate over the management port:

- `myServerCertificate.pem`
- `myServerPrivateKey.key`
- `myCACertificate.pem`

Now that you have the certificates you need, you must prepare your server certificate (including appending any intermediate certificates), and then configure Splunk to find and use them:

- See ["How to prepare your signed certificates for Splunk"](#) to learn how to set up your certificates to work with Splunk.
- See ["Configure Splunk forwarding to use your own certificates"](#) to learn more about configuring certificate authentication for forwarding.
- See [About securing inter-Splunk communication](#) to learn more about configuring certificate authentication for Splunk to Splunk communications.

How to get certificates signed by a third-party

This topic describes one way you can use the version of OpenSSL that ships with Splunk Enterprise to obtain third-party certificates that you can use to secure your forwarder-to-indexer and inter-Splunk communication.

To get certificates that you can use to secure for browser-to-Splunk Web communication, see ["Get certificates signed by a third-party for Splunk Web."](#)

If you already possess or know how to generate the certificates you can, skip this topic and go directly to the configuration steps, which are described later in this manual:

- [Configure Splunk forwarding to use your own certificates](#)
- [About securing inter-Splunk communication](#)

Note: If you plan to use multiple common names in your configurations, you can repeat the steps described here to create a different server certificate using the same root CA for each instance with it's own common name and then configure your Splunk instances to use them. See ["Configure Splunk forwarding to use your own certificates"](#) for more information about configuring your forwarders and indexers.

Before you begin

In this discussion, `$SPLUNK_HOME` (`%SPLUNK_HOME%` on Windows) refers to the Splunk Enterprise installation directory. On Windows, you might need to set this variable at the command line or in the Environment tab in the System Properties dialog.

On Windows, the Splunk Enterprise directory is at `C:\Program Files\Splunk` by default. For most Unix platforms, the default installation directory is at `/opt/splunk`. For Mac OS, it is `/Applications/splunk`. See the Administration Guide to learn more about working with Windows and *nix.

Make sure that you are using the version of OpenSSL provided with Splunk Enterprise by setting your environment to the version in `$SPLUNK_HOME/splunk/lib` in *nix or `%SPLUNK_HOME%/splunk/bin` in Windows.

Create a new directory for your certificates

Create a new directory to work from when creating your certificates. In our example, we are using `$SPLUNK_HOME/etc/auth/mycerts`:

```
# mkdir $SPLUNK_HOME/etc/auth/mycerts
# cd $SPLUNK_HOME/etc/auth/mycerts
```

Splunk strongly recommends that you make a new folder so that you do not overwrite the existing certificates in `$SPLUNK_HOME/etc/auth` for your new certificates and keys. Working in a new directory protects the certificates that

ship with Splunk and lets you use them for other Splunk components as necessary.

Request your server certificate

Create and sign a Certificate Signing Request (CSR) to send to your Certificate Authority.

Important: This example shows you how to create a new private key and request a server certificate. You can distribute this server certificate to all forwarders, indexers as well your Splunk instances that communicate on the management port. If you want to use a different common names for each instance, you simply repeat the process described here to create different certificates (each with a different common name) for your Splunk instances.

For example, if configuring multiple forwarders, you can use the following example to create the certificate `myServerCertificate.pem` for your indexer, then create another certificate `myForwarderCertificate.pem` using the same root CA and install that certificate on your forwarder. Note that an indexer will only accept a properly generated and configured certificate from a forwarder that is signed by the same root CA.

See "[Configure Splunk forwarding to use your own certificates](#)" for more information about configuring your forwarders and indexers.

Generate a private key for your server certificate

1. Create a new private key. The following example uses DES3 encryption and a 2048 bit key length, we recommend a key length of 2048 or higher.

In *nix:

```
openssl genrsa -des3 -out myServerPrivateKey.key 2048
```

In Windows:

```
openssl genrsa -des3 -out myServerPrivateKey.key 2048 -config  
$SPLUNK_HOME\openssl.cnf
```

2. When prompted, create a password for your key.

When you are done, a new private key `myServerPrivateKey.key` is created in your directory. You will use this key to sign your Certificate Signing Request (CSR).

Generate a new Certificate Signing Request (CSR)

1. Use your private key `myServerPrivateKey.key` to generate a CSR for your server certificate:

In *nix:

```
openssl req -new -key myServerPrivateKey.key -out  
myServerCertificate.csr
```

In Windows:

```
openssl req -new -key myServerPrivateKey.key -out  
myServerCertificate.csr -config $SPLUNK_HOME\openssl.cnf
```

2. When prompted, provide the password you created for your private key `myServerPrivateKey.key`.

3. Provide the requested information for your certificate. To use common-name checking, make sure to provide a Common Name when entering your certificate details.

When you are done, a new CSR `myServerCertificate.csr` appears in your directory.

Download and verify the server certificate and public key

1. Send your CSR to your Certificate Authority (CA) to request a new server certificate. The request process varies based on the Certificate Authority you use.

2. When it's ready, download the new server certificate from your Certificate Authority. For the examples in this manual, let's call this `myServerCertificate.pem`.

3. Also download your Certificate Authority's public CA certificate. For the examples in this manual, let's call this `myCACertificate.pem`.

If your Certificate Authority does not provide you with certificates in PEM format, you must convert them using the OpenSSL command appropriate to your existing file type, consult your OpenSSL documentation for more information about converting different file types.

4. View the contents to make sure it has everything you need:

- The `?Issuer?` entry should refer to your CA's information.
- The `?Subject?` entry should show the information (country name, organization name, Common Name, etc) that you entered when creating the CSR earlier.

Next steps

You should now have the following files in the directory you created, which is everything you need to configure indexers, forwarders, and Splunk instances that communicate over the management port:

- `myServerCertificate.pem`
- `myServerPrivateKey.key`
- `myCACertificate.pem`

Now that you have the certificates you need, you must prepare your server certificate (including appending any intermediate certificates), and then configure Splunk to find and use your certificates:

- See ["How to prepare your signed certificates for Splunk"](#) to learn how to set up your certificates to work with Splunk.
- See ["Configure Splunk forwarding to use your own certificates"](#) to learn more about configuring certificate authentication for forwarding.
- See ["About securing inter-Splunk communication"](#) to learn more about configuring certificate authentication for inter-Splunk communications.

How to prepare your signed certificates for Splunk

Once you have your certificates, you must combine the server certificate and your keys into a single file that Splunk can use.

If you do not have your certificates and need help getting them, we provide some basic examples using OpenSSL in the following topics:

- [How to self-sign certificates.](#)
- [How to get certificates signed by a third party.](#)

Note: To configure SSL for Splunk, make sure your certificates and public key are in x509 format and that your private key is in RSA format.

Create a single PEM file

Consolidate the signed server certificate, the server private key and the CA public key in a single PEM file.

For the examples here, we are using the file names described in ["How to self-sign certificates"](#) and ["How to get certificates signed by a third party."](#)

The following is an example for *nix:

```
# cat myServerCertificate.pem myServerPrivateKey.key  
myCACertificate.pem > myNewServerCertificate.pem
```

The following is an example for Windows:

```
>type myServerCertificate.pem myServerPrivateKey.key  
myCACertificate.pem > myNewServerCertificate.pem
```

Once created, the contents of the file `myNewServerCertificate` should contain, in the following order:

- The server certificate (`myServerCertificate.pem`)
- The private key (`myServerPrivateKey.key`)
- The certificate authority public key (`myCACertificate.pem`)

Here's an example of a properly concatenated certificate:

```
-----BEGIN CERTIFICATE-----  
MIICUTCCAboCCQCscBkn/xey1TANBqkqhkiG9w0BAQUFADBtMQswCQYDVQQGEwJV  
...  
<Server Certificate>  
...  
8/PZr3EuXYk1c+N5hgIQys5a/HIn  
-----END CERTIFICATE-----  
-----BEGIN RSA PRIVATE KEY-----  
Proc-Type: 4, ENCRYPTED  
DEK-Info: DES-EDE3-CBC, CFCECC7976725DE5  
  
S+DPcQ012Z1bk71N3cBqr/nwEXPNDQ4uqttecCd3iGMV3B/WSOWAQxcWzhe9JnIsl  
...  
<Server Private Key ? Passphrase protected>  
...  
-----END RSA PRIVATE KEY-----  
-----BEGIN CERTIFICATE-----  
MIICUTCCAboCCQCscBkn/xey1TANBqkqhkiG9w0BAQUFADBtMQswCQYDVQQGEwJV  
...  
<Certificate Authority Public Key>  
...  
8/PZr3EuXYk1c+N5hgIQys5a/HIn
```

-----END CERTIFICATE-----

How to configure certificate chains

To use multiple certificates, append the intermediate certificate to the end of the server's certificate file. You can add as many certificates you need to in decreasing order of hierarchy, up to the root.

The certificates should be concatenated in the following order:

```
[ server certificate]
[ intermediate certificate]
[ root certificate (if required) ]
```

So for example, a certificate chain might look like this:

```
-----BEGIN CERTIFICATE-----
... (certificate for your server)...
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
... (the intermediate certificate)...
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
... (the root certificate for the CA)...
-----END CERTIFICATE-----
```

Next steps

Now that you have the certificates you need, you must configure Splunk to find and use them:

- See "[Configure Splunk forwarding to use your own certificates](#)" to learn more about configuring certificate authentication for forwarding.
- See "[About securing inter-Splunk communication](#)" to learn more about configuring certificate authentication for inter-Splunk communications.

Self-sign certificates for Splunk Web

This topic provides basic examples for creating the self-signed certificates in the command line using the version of OpenSSL included with Splunk.

There are multiple ways you can create signed certificates, depending upon your organizations policies, your platform, and the tools that you are using. If you have already generated these certificates and key, or if you are experienced in

generating certificates, you can skip this task and go directly to the configuration topic ["Secure Splunk Web with your own certificate"](#) in this manual.

Since self-signed certificates are signed by your organization, they are not contained in browser certificate stores. As a result, web browsers consider self-signed certificates "untrusted". This produces a warning page to users and may even prevent access for the user.

Self-signed certificates are best used for browser to Splunk Web communication that happens within an organization or between known entities where you can add your own CA to all browser stores that will contact Splunk Web. For any other scenario, CA-signed certificates are recommended. See ["Get certificates signed by a third party for Splunk Web"](#) for more information.

Before you begin

In this discussion, `$SPLUNK_HOME` refers to the Splunk installation directory. On Windows, Splunk is installed at `C:\Program Files\splunk` by default. For most Unix platforms, the default installation directory is at `/opt/splunk`; for Mac OS, it is `/Applications/splunk`. See the Administration Guide to learn more about working with Windows and *nix.

Make sure that you are using the version of OpenSSL provided with Splunk by setting your environment to the version in `$SPLUNK_HOME/splunk/lib` in *nix or `$SPLUNK_HOME/splunk/bin` in Windows.

Generate a new root certificate to be your Certificate Authority

1. Create a new directory to host your certificates and keys. For this example we will use `$SPLUNK_HOME/etc/auth/mycerts`.

We recommend that you place your new certificates in a different directory than `$SPLUNK_HOME/etc/auth/splunkweb` so that you don't overwrite the existing certificates. This ensures that you are able to use the certificates that ship with Splunk in `$SPLUNK_HOME/etc/auth/splunkweb` for other Splunk components as necessary.

Note: If you created a self-signed certificate as described in ["How to self-sign certificates"](#), you can copy that root certificate into your directory and skip to the next step: "Create a new private key for Splunk Web".

2. Generate a new RSA private key. This example uses 2048 bit length:

```
# openssl genrsa -des3 -out myCAPrivateKey.key 2048
```

Note that in Windows you may need to append the location of the `openssl.cnf` file:

```
>openssl genrsa -des3 -out myCAPrivateKey.key 2048 -config  
$SPLUNK_HOME\openssl.cnf
```

Our example uses a key length of 2048 (minimum recommended), but you can specify key lengths above 2048 if they are supported by your browser.

3. When prompted, create a password.

The private key `myCAPrivateKey.key` appears in your directory. This is your root certificate private key.

4. Generate a certificate signing request using the root certificate private key `myCAPrivateKey.key`:

In *nix:

```
# openssl req -new -key myCAPrivateKey.key -out myCACertificate.csr
```

In Windows:

```
>openssl req -new -key myCAPrivateKey.key -out myCACertificate.csr  
-config $SPLUNK_HOME\openssl.cnf
```

5. Provide the password to the private key `myCAPrivateKey.key`.

A new CSR `myCACertificate.csr` appears in your directory.

6. Use the CSR to generate a new root certificate and sign it with your private key:

In *nix:

```
# openssl x509 -req -in myCACertificate.csr -signkey myCAPrivateKey.key  
-out myCACertificate.pem -days 3650
```

In Windows:

```
>openssl x509 -req -in myCACertificate.csr -signkey myCAPrivateKey.key  
-out myCACertificate.pem -days 3650 -config $SPLUNK_HOME\openssl.cnf
```

7. When prompted, provide for the password to the private key

`myCAPrivateKey.key`.

A new certificate `myCACertificate.pem` appears in your directory. This is your public certificate.

Create a new private key for Splunk Web

1. Generate a new private key:

In *nix:

```
# openssl genrsa -des3 -out mySplunkWebPrivateKey.key 2048
```

In Windows:

```
>openssl genrsa -des3 -out mySplunkWebPrivateKey.key 2048 -config  
$SPLUNK_HOME\openssl.cnf
```

2. When prompted, create a password.

A new key, `mySplunkWebPrivateKey.key` appears in your directory.

3. Remove the password from your key. (Splunk Web does not currently support password-protected private keys.)

In *nix:

```
# openssl rsa -in mySplunkWebPrivateKey.key -out  
mySplunkWebPrivateKey.key
```

In Windows:

```
>openssl rsa -in mySplunkWebPrivateKey.key -out  
mySplunkWebPrivateKey.key -config $SPLUNK_HOME\openssl.cnf
```

You can make sure your password is gone by issuing the following command:

In *nix:

```
# openssl rsa -in mySplunkWebPrivateKey.key -text
```

In Windows:

```
>openssl rsa -in mySplunkWebPrivateKey.key -text -config  
$SPLUNK_HOME\openssl.cnf
```

You should be able to read the contents of your certificate without providing a password.

Create and sign a server certificate

1. Create a new certificate signature request using your private key `mySplunkWebPrivateKey.key`:

In *nix:

```
# openssl req -new -key mySplunkWebPrivateKey.key -out  
mySplunkWebCert.csr
```

In Windows:

```
>openssl req -new -key mySplunkWebPrivateKey.key -out  
mySplunkWebCert.csr -config $SPLUNK_HOME\openssl.cnf
```

The CSR `mySplunkWebCert.csr` appears in your directory.

2. Self-sign the CSR with the root certificate private key `myCAPrivateKey.key`:

In *nix:

```
# openssl x509 -req -in mySplunkWebCert.csr -CA myCACertificate.pem  
-CAkey myCAPrivateKey.key -CAcreateserial -out mySplunkWebCert.pem  
-days 1095
```

In Windows:

```
>openssl x509 -req -in mySplunkWebCert.csr -CA myCACertificate.pem  
-CAkey myCAPrivateKey.key -CAcreateserial -out mySplunkWebCert.pem  
-days 1095 -config $SPLUNK_HOME\openssl.cnf
```

3. When prompted, provide the password to the root certificate private key `myCAPrivateKey.key`.

The certificate `mySplunkWebCert.pem` is added to your directory. This is your server certificate.

Create a single PEM file

Combine your server certificate and public certificates, in that order, into a single PEM file.

Here's an example of how to do this in Linux:

```
# cat mySplunkWebCert.pem myCACertificate.pem >  
mySplunkWebCertificate.pem
```

Here's an example in Windows:

```
# type mySplunkWebCert.pem myCACertificate.pem >  
mySplunkWebCertificate.pem
```


Set up certificate chains

To use multiple certificates, append the intermediate certificate to the end of the server's certificate file in the following order:

```
[ server certificate]
[ intermediate certificate]
[ root certificate (if required) ]
```

So for example, a certificate chain might look like this:

```
-----BEGIN CERTIFICATE-----
... (certificate for your server)...
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
... (the intermediate certificate)...
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
... (the root certificate for the CA)...
-----END CERTIFICATE-----
```

Next steps

Now that you have your certificates, you need to distribute them and configure Splunkd and Splunk Web to use them. See ["Secure Splunk Web with your own certificate"](#) in this manual for more information.

Get certificates signed by a third-party for Splunk Web

This topic provides basic examples for creating the third-party signed certificates necessary to configure Splunk Web for SSL authentication and encryption.

The steps provided here are not tasks specific to Splunk. There are multiple ways you can create these certificates, depending upon your organization's policies, your network structure and the tools that you are using. If you have already generated these certificates and key, or if you are experienced with third-party certificates, you may prefer to skip this step and go directly to the configuration topic in this manual at ["Secure Splunk Web with your own certificate."](#)

Before you begin

In this discussion, `$SPLUNK_HOME` refers to the Splunk installation directory. On Windows, Splunk is installed at `C:\Program Files\splunk` by default. For most Unix platforms, the default installation directory is at `/opt/splunk`; for Mac OS, it is `/Applications/splunk`. See the Administration Guide to learn more about working with Windows and *nix.

Make sure that you are using the version of OpenSSL provided with Splunk by setting your environment to the version in `$SPLUNK_HOME/lib` in *nix or `$SPLUNK_HOME/bin` in Windows.

Create a new private key for Splunk Web

1. Create a new directory to host your own certificates and keys. In this example we will use `$SPLUNK_HOME/etc/auth/mycerts`.

We recommend that you place your new certificates in a different directory than `$SPLUNK_HOME/etc/auth/splunkweb` so that you don't overwrite the existing certificates. This ensures that you can use the certificates that ship with Splunk for other Splunk components as necessary.

2. Generate a new private key. Our example uses a key length of 1024, but you can specify key lengths above 1024 if they are supported by your browser. For sensitive data we recommend a length of 2048 or higher when possible.

```
# openssl genrsa -des3 -out mySplunkWebPrivateKey.key 1024
```

Note that in Windows you may need to append the location of the `openssl.cnf` file:

```
>openssl genrsa -des3 -out mySplunkWebPrivateKey.key 1024 -config  
$SPLUNK_HOME\openssl.cnf
```

3. Create a password when prompted.

A new private key `mySplunkWebPrivateKey.key` is added to your directory. You can use this key to sign your CSR.

4. Remove the password from the private key (Splunk Web does not support private key passwords):

```
# openssl rsa -in mySplunkWebPrivateKey.key -out  
mySplunkWebPrivateKey.key
```

You can use the following command to make sure that your password was successfully removed:

```
# openssl rsa -in mySplunkWebPrivateKey.key -text  
You will be prompted to enter the passphrase for the original key. (to authorize  
the removal of the passphrase from the key)
```

If the password was successfully removed, you can view the certificate contents without providing a password.

Create a Certificate Authority (CA) request and obtain your server certificate

1. Create a new certificate signature request using your private key

`mySplunkWebPrivateKey.key`:

In *nix:

```
# openssl req -new -key mySplunkWebPrivateKey.key -out  
mySplunkWebCert.csr
```

In Windows:

```
>openssl req -new -key mySplunkWebPrivateKey.key -out  
mySplunkWebCert.csr -config $SPLUNK_HOME\openssl.cnf
```

Note for Windows platforms: If you see an error similar to this:

Unable to load config info from

`c:\build-amd64-5.0.2-20130120-1800\splunk/ssl/openssl.cnf`

Try typing the following in your command prompt then run the `openssl` command again:

```
set OPENSSL_CONF=c:/Program Files/Splunk/openssl.cnf
```

2. Use this CSR `mySplunkWebCert.csr` to request a new signed certificate from your Certificate Authority (CA). The process for requesting a signed certificate varies depending on how your Certificate Authority handles a certificate signature request. Contact your CA for more information.

3. Download the server certificate returned by your Certificate Authority. For this example, let's call it "`mySplunkWebCert.pem`."

4. Download your Certificate Authority's public CA certificate. For this example, let's call it "`myCAcert.pem`."

5. Make sure that both the server certificate and the public CA certificate are both in PEM format. If the certificates are not in PEM format, convert them using the `openssl` command appropriate to your existing file type. Here's an example of a command that you can use for DER formats:

```
x509 -in input.crt -inform DER -out output.crt -outform PEM/x509 -in input.crt
-inform DER -out output.crt -outform PEM
```

6. Check both certificates to make sure they have the necessary information and are not password protected.

```
# openssl x509 -in myCACert.pem -text
# openssl x509 -in mySplunkWebCert.pem -text
>openssl x509 -in myCACert.pem -text -config $SPLUNK_HOME\openssl.cnf
>openssl x509 -in mySplunkWebCert.pem -text -config
$SPLUNK_HOME\openssl.cnf
```

The issuer information for `mySplunkWebCert.pem` should be the subject information for `myCACert.pem` (unless you are using intermediary certificates).

Combine your certificate and keys into a single file

Combine your server certificate and public certificate, in that order, into a single PEM file.

Set up certificate chains

To use multiple certificates, append the intermediate certificate to the end of the server's certificate file in the following order:

```
[ server certificate]
[ intermediate certificate]
[ root certificate (if required) ]
```

So for example, a certificate chain might look like this:

```
-----BEGIN CERTIFICATE-----
... (certificate for your server)...
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
... (the intermediate certificate)...
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
... (the root certificate for the CA)...
-----END CERTIFICATE-----
```

Note that the root CA that signed the intermediate certificate as well as all intermediary certificates must be in the browser certificate stores.

Next steps

Configure Splunk's Web.conf file to find and use your certificates for authentication. See ["Secure Splunk Web with your own certificate"](#) for more information.

Determine your cipher suite

You can select and specify a cipher suite for inter-Splunk, Splunk Web, and Splunk forwarder to indexer communications. You add your cipher suite by appending a line at the end of your server SSL configuration stanza.

The following is an example of how you would updated `inputs.conf` when configuring forwarder to indexer certificate authentication:

```
[splunktcp-ssl:9998]
[SSL]
password = password
requireClientCert = false
rootCA = $SPLUNK_HOME/etc/auth/cacert.pem
serverCert = $SPLUNK_HOME/etc/auth/server.pem
cipherSuite = AES256-SHA256:DHE-RSA-AES256-SHA256
```

To see which ciphers are available to you:

```
$SPLUNK_HOME/bin/splunk cmd openssl ciphers -v
  $SPLUNK_HOME/bin/splunk cmd openssl ciphers -v "TLSv1.2"
  $SPLUNK_HOME/bin/splunk cmd openssl ciphers -v "HIGH"
```

Cipher suites are available to you based on your version of OpenSSL. To see which version of OpenSSL you are running:

```
$SPLUNK_HOME/bin/splunk cmd openssl version
```