**16. Security Best Practices**

Automation controller out-of-the-box is deployed in a secure fashion for use to automate typical environments. However, managing certain operating system environments, automation, and automation platforms, may require some additional best practices to ensure security. This document describes best practices for automation in a secure manner.

**16.1. General best practices**

An application is only as secure as the underlying system. To secure Red Hat Enterprise Linux, start with the release-appropriate security guide:

* For Red Hat Enterprise Linux 8: <https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/security_hardening/index>
* For Red Hat Enterprise Linux 9: <https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html/security_hardening>

**16.2. Understand the architecture of Ansible and the controller**

Ansible and automation controller comprise a general purpose, declarative, automation platform. That means that once an Ansible playbook is launched (via the controller, or directly on the command line), the playbook, inventory, and credentials provided to Ansible are considered to be the source of truth. If policies are desired around external verification of specific playbook content, job definition, or inventory contents, these processes must be undertaken before the automation is launched (whether via the controller web UI, or the controller API).

These can take many forms. The use of source control, branching, and mandatory code review is best practice for Ansible automation. There are many tools that can help create process flow around using source control in this manner.

At a higher level, many tools exist that allow for creation of approvals and policy-based actions around arbitrary workflows, including automation; these tools can then use Ansible via the controller’s API to perform automation.

We recommend all customers of automation controller select a secure default administrator password at time of installation. See [Changing the Controller Admin Password](https://docs.ansible.com/automation-controller/latest/html/administration/tipsandtricks.html#tips-change-password) for more information.

Automation controller exposes services on certain well-known ports, such as port 80 for HTTP traffic and port 443 for HTTPS traffic. We recommend that you do not expose automation controller on the open internet, significantly reducing the threat surface of your installation.

**16.3. Granting access**

Granting access to certain parts of the system exposes security risks. Apply the following practices to help secure access:

* [Minimize administrative accounts](https://docs.ansible.com/automation-controller/latest/html/administration/security_best_practices.html#minimize-administrative-accounts)
* [Minimize local system access](https://docs.ansible.com/automation-controller/latest/html/administration/security_best_practices.html#minimize-local-system-access)
* [Remove access to credentials from users](https://docs.ansible.com/automation-controller/latest/html/administration/security_best_practices.html#remove-access-to-credentials-from-users)
* [Enforce separation of duties](https://docs.ansible.com/automation-controller/latest/html/administration/security_best_practices.html#enforce-separation-of-duties)

[**16.3.1. Minimize administrative accounts**](https://docs.ansible.com/automation-controller/latest/html/administration/security_best_practices.html#id2)

Minimizing the access to system administrative accounts is crucial for maintaining a secure system. A system administrator/root user can access, edit, and disrupt any system application. Keep the number of people/accounts with root access to as small of a group as possible. Do not give out *sudo* to *root* or *awx* (the controller user) to untrusted users. Know that when restricting administrative access via mechanisms like *sudo*, that restricting to a certain set of commands may still give a wide range of access. Any command that allows for execution of a shell or arbitrary shell commands, or any command that can change files on the system, is fundamentally equivalent to full root access.

In a controller context, any controller ‘system administrator’ or ‘superuser’ account can edit, change, and update any inventory or automation definition in the controller. Restrict this to the minimum set of users possible for low-level controller configuration and disaster recovery only.

[**16.3.2. Minimize local system access**](https://docs.ansible.com/automation-controller/latest/html/administration/security_best_practices.html#id3)

Automation controller, when used with best practices, should not require local user access except for administrative purposes. Non-administrator users should not have access to the controller system.

[**16.3.3. Remove access to credentials from users**](https://docs.ansible.com/automation-controller/latest/html/administration/security_best_practices.html#id4)

If an automation credential is only stored in the controller, it can be further secured. Services such as OpenSSH can be configured to only allow credentials on connections from specific addresses. Credentials used by automation can be different than credentials used by system administrators for disaster-recovery or other ad-hoc management, allowing for easier auditing.

[**16.3.4. Enforce separation of duties**](https://docs.ansible.com/automation-controller/latest/html/administration/security_best_practices.html#id5)

Different pieces of automation may need to access a system at different levels. For example, you may have low-level system automation that applies patches and performs security baseline checking, while a higher-level piece of automation deploys applications. By using different keys or credentials for each piece of automation, the effect of any one key vulnerability is minimized, while also allowing for easy baseline auditing.

**16.4. Available resources**

Several resources exist in the controller and elsewhere to ensure a secure platform. Consider utilizing the following functionality:

* [Audit and logging functionality](https://docs.ansible.com/automation-controller/latest/html/administration/security_best_practices.html#audit-and-logging-functionality)
* [Existing security functionality](https://docs.ansible.com/automation-controller/latest/html/administration/security_best_practices.html#existing-security-functionality)
* [External account stores](https://docs.ansible.com/automation-controller/latest/html/administration/security_best_practices.html#external-account-stores)
* [Django password policies](https://docs.ansible.com/automation-controller/latest/html/administration/security_best_practices.html#django-password-policies)

[**16.4.1. Audit and logging functionality**](https://docs.ansible.com/automation-controller/latest/html/administration/security_best_practices.html#id6)

For any administrative access, it is key to audit and watch for actions. For the system overall, this can be done via the [built in audit support](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html/security_hardening/auditing-the-system_security-hardening) and via the built-in logging support.

For automation controller, this is done via the built-in Activity Stream support that logs all changes within the controller, as well as via the automation logs.

Best practices dictate collecting logging and auditing centrally, rather than reviewing it on the local system. It is recommended that automation controller be configured to use whatever IDS and/or logging/auditing (Splunk) is standard in your environment. automation controller includes built-in logging integrations for Elastic Stack, Splunk, Sumologic, Loggly, and more. See [Logging and Aggregation](https://docs.ansible.com/automation-controller/latest/html/administration/logging.html#ag-logging) for more information.

[**16.4.2. Existing security functionality**](https://docs.ansible.com/automation-controller/latest/html/administration/security_best_practices.html#id7)

Do not disable SELinux, and do not disable the controller’s existing multi-tenant containment. Use the controller’s role-based access control (RBAC) to delegate the minimum level of privileges required to run automation. Use Teams in the controller to assign permissions to groups of users rather than to users individually. See [Role-Based Access Controls](http://docs.ansible.com/automation-controller/4.5/html/userguide/security.html#rbac-ug) in the *Automation Controller User Guide*.

[**16.4.3. External account stores**](https://docs.ansible.com/automation-controller/latest/html/administration/security_best_practices.html#id8)

Maintaining a full set of users just in the controller can be a time-consuming task in a large organization, prone to error. Automation controller supports connecting to external account sources via [LDAP](https://docs.ansible.com/automation-controller/latest/html/administration/ldap_auth.html#ag-auth-ldap), [SAML 2.0](https://docs.ansible.com/automation-controller/latest/html/administration/ent_auth.html#ag-auth-saml), and certain [OAuth providers](https://docs.ansible.com/automation-controller/latest/html/administration/social_auth.html#ag-social-auth). Using this eliminates a source of error when working with permissions.

[**16.4.4. Django password policies**](https://docs.ansible.com/automation-controller/latest/html/administration/security_best_practices.html#id9)

Controller admins can leverage Django to set password policies at creation time via AUTH\_PASSWORD\_VALIDATORS to validate controller user passwords. In the custom.py file located at /etc/tower/conf.d of your controller instance, add the following code block example:

AUTH\_PASSWORD\_VALIDATORS **=** [

{

'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',

'OPTIONS': {

'min\_length': 9,

}

},

{

'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',

},

]

For more information, see [Password management in Django](https://docs.djangoproject.com/en/3.2/topics/auth/passwords/#module-django.contrib.auth.password_validation) in addition to the example posted above.

Be sure to restart your controller instance for the change to take effect. See [Starting, Stopping, and Restarting the Controller](https://docs.ansible.com/automation-controller/latest/html/administration/init_script.html#ag-restart-tower) for detail.