

UNCLASSIFIED



RED HAT ENTERPRISE LINUX (RHEL) 9 STIG ANSIBLE DOCUMENTATION

Version 2, Release 5

02 July 2025

Developed by DISA for the DOD

UNCLASSIFIED

TABLE OF CONTENTS

	Page
1. BACKGROUND.....	1
2. INSTALLATION	2
2.1 Installing Ansible	2
2.2 Extracting.....	2
3. CONFIGURATION.....	3
3.1 Simple	3
3.2 Custom	3
4. COMPLIANCE EXTRACTION	4
5. OTHER CONSIDERATIONS.....	5
5.1 Reboot for SELINUX Configuration Changes	5

1. BACKGROUND

Ansible is an open source, cross-platform configuration management solution used to define and enforce system and application configurations. This package provides Ansible configurations that implement most of the Red Hat Enterprise Linux 9 STIG. While the content has been tested during development, all possible system and environmental factors could not be tested. Before using this content in a production environment, perform testing with the intended settings in a test environment. There is no mandate to use this content; it is published as a resource to assist in the application of security guidance to systems. Use it in the manner and to the extent that it assists with this goal.

2. INSTALLATION

The following instructions are for standalone installation using [ansible-playbook](#) for testing purposes. A production environment may additionally use Ansible Tower. See [here](#) for details.

2.1 Installing Ansible

Newer versions of Ansible are in the Red Hat Enterprise Linux 9 [EPEL](#) repository. To install it, run the following:

```
sudo yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
sudo yum install ansible
```

For other installation methods, see [here](#).

2.2 Extracting

Unzip the `rhel9STIG-ansible.zip`.

3. CONFIGURATION

3.1 Simple

To apply the default STIG Ansible configuration to the local machine only, run the **enforce.sh** script to enforce the STIG. Additionally note that Ansible will refuse to reboot the local machine automatically (refer to section 5.1). To tailor the configuration, follow the steps in the next section.

3.2 Custom

To customize, create a YAML (.yaml) file containing just the variables to customize from the variables named in the **roles/rhel9STIG/defaults/main.yaml** file. This file contains configuration data to define which configuration settings to manage and the values for these settings. Edit the newly created configuration file in a text editor to best suit each system's requirements as needed. For example, to turn off STIG rule ID 258117, set the "Manage" attribute equal to **False**. To set STIG rule ID 258107's minimum password length to 20, set the "**_etc_security_pwquality_conf_Line**" attribute to **'minlen = 20'**.

```
rhel9STIG_stigrule_258117_Manage: False
rhel9STIG_stigrule_258117__etc_login_defs_Line: 'ENCRYPT_METHOD SHA512'

rhel9STIG_stigrule_258107_Manage: True
rhel9STIG_stigrule_258107__etc_security_pwquality_conf_Line: 'minlen = 20'
```

To use the newly created, custom variables file, edit **site.yaml** to include it. See the highlighted lines to add below:

```
- hosts: localhost
  gather_facts: no
  vars_files:
    - /path/to/custom/vars.yaml
  roles:
    - rhel9STIG
```

For more information on variables, see [here](#). For more information on YAML, see [here](#).

4. COMPLIANCE EXTRACTION

This compliance extraction methodology returns results based on a system's compliance with the enforcement content. This may be different from STIG compliance. For example, multiple values may be allowed by the STIG but will be marked as “fail” if the value does not match the single exact value in the enforcement content. Additionally, if a value is customized in such a way to violate a STIG rule it will be marked as “pass” since it matches the enforcement content’s expected value.

At the completion of a successful Ansible playbook play content extraction of the configuration results into XCCDF results can be performed via an Ansible callback plugin. Use of this plugin can be controlled via modification of the follow variable in the ansible.cfg file to include the name of the plugin to use:

```
[defaults]
callback_whitelist = stig_xml
```

Configuration of the plugin is controlled via creation/modification of the following environment variables:

- **export**
STIG_PATH=/path/to/stig/U_Red_Hat_Enterprise_Linux_9_V2R5_Manual_STIG-xccdf.xml
- **export XML_PATH=/path/where/to/write/results.xml**

The above environmental variables control the plugin writing the XCCDF results to the file **XML_PATH** using the STIG at path **STIG_PATH**. The XCCDF results file is output by default to **/tmp/tmpxxxxxx/xccdf-results.xml** where **tmpxxxxxx** is a randomly generated folder.

Note: The STIG provided above should match the STIG release and version number for which the Ansible content is built.

Ansible provides means of checking compliance without enforcement called **--check** (aka “dry run”). To use this mode, run the following:

```
ansible-playbook -v -b -i /dev/null --check site.yml
```

5. OTHER CONSIDERATIONS

5.1 Reboot for SELINUX Configuration Changes

If this content makes changes to the SELINUX configuration, it will restart the system to allow the new settings to take effect. If an automatic restart is not desired, disable management of these rules.

When run against the local system and a reboot is required for a setting change, Ansible will refuse to reboot the system with the following message:

```
fatal: [localhost]: FAILED! => {"changed": false, "elapsed": 0, "msg":  
"Running reboot with local connection would reboot the control node.",  
"rebooted": false}
```

The system may be rebooted manually to apply the changes requiring it. For remote systems, Ansible will perform the remote system reboot.