VAST
FEDERAL

White Paper

VAST Data Federal
Product Hardening Procedures

Version 2.1

All rights reserved. This document may be used free of charge. Selling without prior written consent is prohibited. Obtain permission before redistributing. In all cases, this copyright notice and disclaimer must remain intact.

THE CONTENTS OF THIS DOCUMENT ARE PROVIDED "AS IS," AND COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE DOCUMENT ARE SUITABLE FOR ANY PURPOSE; THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD-PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE DOCUMENT OR THE PERFORMANCE OR IMPLEMENTATION OF THE CONTENTS THEREOF.







Registered in the U.S. Trademark Office

CONFIDENTIAL

Executive Summary

The United States (U.S.) Federal Government is one of the largest purchasers of Information Technology (IT) products in the world with an estimated IT budget of over $200 billion for 2025. It is also one of the most rigorous enforcers of product security and compliance requirements since it is one of the most sought-after targets of highly valuable information. The requirements clearly listed within regulations such as United States (U.S.) Federal Acquisition Regulations (FAR) and the Defense Federal Acquisition Regulations (DFAR) must be met in order for hardware and software products and services to be eligible for purchase by the U.S. Federal Government. Additionally, by signing a contract with the U.S. Government, an IT vendor contractually promises compliance with all applicable federal or state requirements clearly, or even indirectly, specified in the contract. Certifications or representations found to be false, or delivery of non-compliant products, can expose VAST to civil or criminal penalties.

In order to be competitive in this customer vertical VAST products and services must be compliant to all applicable federal and state procurement and operational requirements, regulations and laws. This Product Hardening Guide (PHG) provides the steps necessary to configure the product to be in compliance with all applicable Defense Information Systems Agency (DISA) Security Technical Implementation Guides (STIGs) and Security Requirements Guides (SRGs). DISA STIGs and SRGs compromise the most stringent product security requirements across all technology sectors around the globe. The steps detailed in this guide cannot be modified without approval from the product author; however, customers can choose to not implement certain steps if it is determined that any particular step would impact mission accomplishment.

Customers should take great care when choosing to implement the procedures in chapters 6, 7 and 8. The STIGs have the ability to lock downan operating system to the point of being overly restrictive when measured against day-to-day operational needs. VAST is providing the steps that can be executed on the AI OS, but each risky chapter is precluded by an applicable warning. Questions to any procedure in this document should be sent to the point of contact identified in Chapter 1.3.

CONFIDENTIAL

Revision History

Name	Date	Changes	Version
Sabre Schnitzer	6/13/2024	Initial draft, content and structure	1.0
Sabre Schnitzer	10/14/2024	Content update.	1.1
Sabre Schnitzer	12/27/2024	Admin lockdown procedures added.	1.2
Sabre Schnitzer	1/8/2025	VMS account management added. USB port config.	1.3
Sabre Schnitzer	1/10/2025	VAST OS lockdown procedures.	1.4
Sabre Schnitzer	1/15/2025	Further procedures clarification	1.5
Sabre Schnitzer	2/6/2025	Additional procedures added	1.6
Sabre Schnitzer	3/2/2025	DoDIN APL Audit procedures added	1.7
Sabre Schnitzer	4/14/2025	Content corrections	1.8
Sabre Schnitzer	5/16/2025	Additional content corrections	1.9
Sabre Schnitzer	2/3/2026	Branding update. Content expansion.	2.0
Sabre Schnitzer	3/5/2026	Procedures update	2.1

CONFIDENTIAL

Table of Contents

Executive Summary 3
1.0 Document Objectives 7
1.1 Purpose 7
1.2 Scope 7
1.3 Point of Contact 7
2.0 VAST Operating System Overview 8
3.0 General Hardening Steps 8
3.1 Monitor VAST Security Advisories 8
3.2 Centralized Authentication 8
3.3 Secure Communications 8
4.0 STIG Hardening Steps 9
4.1 Application Server SRG 9
4.1.1 SRG-APP-000015-AS-000010 – TLS Communications 9
4.2 Red Hat Enterprise Linux 10
4.2.1 RHEL-08-010040 – DoD Warning Banner 10
4.2.2 RHEL-08-010040 – SSO and MFA 11
4.2.3 RHEL-08-010200 – Session Timeout 13
4.2.4 RHEL-08-030020 – Syslog and Notifications 14
4.2.5 RHEL-08-030310 – Auditing 16
4.3 Web Server 17
4.3.1 SRG-APP-000108-WSR-000066 – Networking and Alerting 17
5.0 VMS Account Lockdown Procedures 19
5.1 APSC-DV-001680 – APSC-DV-001730 – Password Policy 19
6.0 Administrative Account Lockdown 21
6.1 APSC-DV-001680 – APSC-DV-001730 – Password Policy 21
6.2 APSC-DV-001670 – Inactivity Lockout 23
6.3 APSC-DV-001760 – Password Minimum Lifetime 23
6.4 APSC-DV-001770 – Password Maximum Liftime 23
7.0 VAST OS Lockdown 24
7.1 RHEL-08-010421 - Page Poisoning 24
7.2 RHEL-08-010423 - SLUB/SLAB Poisoning 24
7.3 RHEL-08-010550 – Remote Root Access 25
7.4 RHEL-08-020015 – Unlock Time 25
7.5 RHEL-08-020353 – Default Account Permissions 25
7.6 RHEL-08-040004 – Page-Table Isolation 26
7.7 RHEL-08-040101 – Firewalld Enablement 26
7.8 RHEL-08-040125 – Tempt Folder Noexec 26

CONFIDENTIAL

7.9 RHEL-08-010383 – Defaults 27
7.10 RHEL-08-030570 – Chacl Auditing 27
7.11 RHEL-08-040021 – ATM Blacklisting 27
7.12 RHEL-08-040022 – CAN Blacklisting 28
7.13 RHEL-08-040023 – SCTP Blacklisting 28
7.14 RHEL-08-040024 – TIPC Blacklisting 28
7.15 RHEL-08-040025 – Cramfs Kernel Blacklisting 29
7.16 RHEL-08-040026 – Firewire Core Blacklisting 29
7.17 RHEL-08-040080 – USB Blacklisting 30
7.18 RHEL-08-040111 – Bluetooth Blacklisting 30
7.19 RHEL-08-040249 – IPv4 Forwarding 31
7.20 RHEL-08-040250 - IPv6 Forwarding 31
7.21 RHEL-08-040279 – IPv4 ICMP 32
7.22 RHEL-08-040284 – Namespace Disabling 33
8.0 Firewall Configuration 33
8.1 RHEL-08-040030 – Firewall Configuration 33
9.0 VASTOS STIG Script 36
10.0 Summary 72

CONFIDENTIAL

1.0 Document Objectives

This VAST AI OS Product Hardening Guide (PHG) provides U.S. Government and security-focused customers the configuration steps necessary to configure the VAST AI OS product to be in compliance with all applicable Defense Information Systems Agency (DISA) Security Technical Implementation Guides (STIG) and Security Requirement Guides (SRG). This PHG should be utilized in conjunction with the VAST AI OS Security Assessment Report (SAR). The SAR includes all applicable STIGs and SRGs and performs a complete control by control assessment. Each control has an official response regardless of the applicability of the control. All applicable controls have a validation procedure to verify that the applicability and configuration is valid. The PHG takes the executable steps from the STIG assessment and places them into a simple step-by-step implementation procedure.

Due to the fact that customers across the globe follow the U.S. Federal Government security recommendations due to their comprehensive product security nature, VAST is publishing this paper to demonstrate the product's ability to comply with these rigid security requirements in order to operate the product in its most secure fashion.

1.1 Purpose

The purpose of this white paper is to establish a standard hardening procedure for the VAST AI OS product so that configuration control can be achieved for security minded customers.

1.2 Scope

This PHG covers just the product discussed in this white paper. It does not address the other network and environment hardening steps that must be undertaken to ensure that the entire infrastructure is secure. To be utilized effectively, this PHG should be utilized on a VAST system that is incorporated into a hardened environment.

1.3 Point of Contact

If customers have any questions about any of the procedures identified in this white paper should reach out to the author at the following

Sabre Schnitzer, Federal Compliance Officer
VAST Data Federal

Sabre@vastfederal.com

703-785-0608

CONFIDENTIAL

2.0 VAST Operating System Overview

The VAST AI OS product is configured as a hardened system. However, rather than hiding behind an appliance claim, VAST provides customers with access to all areas of the product to demonstrate the product's security hardening posture. Too many technology companies hide their inner workings from vulnerability scans to hide insecure components. VAST determined very early on that we would provide customer access to all components in order to demonstrate unquestionable insight into our security posture.

3.0 General Hardening Steps

The hardening steps in this chapter are general in nature and should be applied to all AI OS installations regardless of customer segment that the product is deployed into.

3.1 Monitor VAST Security Advisories

The VAST security team creates and publishes advisories for security-related issues in VAST products. These advisories are posted on the VAST support portal located at https://support.vastdata.com/s/ In order to maintain a secure data management service, all customers need to be aware of the VAST security advisories so that patches can be applied before exploits become widely available.

For critical vulnerabilities, VAST provides patches as soon as feasible. The patches will be available for download from the support portal and can be applied through the VAST CLI. Each patch comes with release notes, which explains the vulnerability and the appropriate information about fixes.

3.2 Centralized Authentication

The VAST AI OS supports both local authentication and Lightweight Directory Access Protocol (LDAP) authentication. Local authentication must be used only as a fallback mechanism to access the AI OS product when LDAP services are unreachable.

Instructions for configuring Active Directory authentication are included in chapter 4.

3.3 Secure Communications

In order to secure and trust all communications to the product, customers should install a trusted server certification to enable trusted TLS communications to the product.

CONFIDENTIAL

4.0 STIG Hardening Steps

The steps below are specific in nature and should be applied to AI OS installations in all U.S. Government environments. Use by other customer verticals is acceptable so long as these procedures are not deviated from. Manipulation of the product outside of these steps without VAST approval is prohibited.

Each hardening step below is identified with the associated Vulnerability ID and/or STIG ID. This allows customers to cross reference the hardening step below with its associated STIG requirement as well as the VAST AI OS STIG assessment.

The following STIGs and SRGs are determined applicable to the VAST AI OS:

- Application Security and Development STIG
- Application Server SRG
- Container SRG
- Database SRG
- Docker STIG
- Network Device SRG
- Red Hat Enterprise Linux 8 SRG
- Web Server SRG

If a hardening step is repeated due to the step resolving more than one STIG control, then the step is only provided once in this document.

4.1 Application Server SRG

The following control steps are derived from the DISA Application Server SRG.

4.1.1 SRG-APP-000015-AS-000010 – TLS Communications

Follow these steps to configure trusted TLS communications.

Procedure:

1. Obtain an SSL certificate from a Certificate Authority (CA). The CA will provide you with two files: a certificate file and a key file. Choose anX.509 output file format containing ASCII (Base64) encoded data.

2. From a networked workstation, enter the IP address of the VAST AI OS into a browser. The VAST AI OS GUI will appear.

CONFIDENTIAL

3. Authenticate to the VAST AI OS with an authorized account. Upon successful authentication and authorization, the user will be granted into the system.

4. From the left navigation menu, select **Settings** and then **Certificates**.

5. From the Certificate for dropdown, select **VMS**.

6. Enter or upload the certificate file contents in the Certificate field and the key file content into the Key field.

7. When pasting the certificate file content, include the BEGIN CERTIFICATE and END

   CERTIFICATE lines, like this:

   -----BEGIN CERTIFICATE-----

   ...

   -----END CERTIFICATE-----

   When pasting the private key file content, include the BEGIN PRIVATE KEY and END PRIVATE KEY lines, like this:

   -----BEGIN PRIVATE KEY-----

   ...

   -----END PRIVATE KEY-----

8. Click **Update**.

Result: Your certificate is installed, and you can now browse to the VAST Web UI without your browser warning that the site is not secure.

4.2 Red Hat Enterprise Linux

The following control steps are derived from the DISA Red Hat Enterprise Linux STIG.

4.2.1 RHEL-08-010040 – DoD Warning Banner

Follow these steps to add a standard warning banner.

CONFIDENTIAL

Procedure:

1. From a networked workstation, enter the IP address of the VAST AI OS into a browser. The VAST AI OS GUI will appear.

2. Authenticate to the VAST AI OS with an authorized account. Upon successful authentication and authorization, the user will be granted into the system.

3. Navigate to **Settings** → **VMS**

4. In the **Login Banner field**, add the text that is desired. The login banner can contain any text you choose to specify. It appears on the login page of the VMS VAST Web UI and it appears after login in the command line of the VAST CLI. The banner is scrollable in the VAST Web UI and there is no limit on the length of text you can enter.

5. Click **Save**.

6. Click **Yes** to save your changes.

Result: The VAST AI OS will display a standard warning banner prior to logon to the GUI and immediately after logon via SSH.

4.2.2 RHEL-08-010040 – SSO and MFA

Follow these steps to configure SSO and MFA.

Prerequisite: Trusted TLS communications need to be configured prior to configuring SSO and MFA

Procedure:

1. From a networked workstation, enter the IP address of the VAST AI OS into a browser. The VAST AI OS GUI will appear.

2. Authenticate to the VAST AI OS with an authorized account. Upon successful authentication and authorization, the user will be granted into the system.

3. From the left navigation menu, navigate to the **Administrators** page, then select the **SAML** tab.

CONFIDENTIAL

4. Click **Add** new identity.

5. In the **General** section, add these details for the Identity Provider:

    a. **IdP name.** The name of the Identify Provider (e.g. Okta)

    b. **IdP Entity ID.** The Entity ID for the Identity Provider, typically obtained from the metadata.

    c. **Force Authenticate.** Forces authentication with the IdP for each sign-on.

7. In the **Metadata** section, enter these details:

    a. **Metadata URL.** The URL to the metadata on the IdP, usually in the form of https://<idp-url/sso/saml/metadata where idp-url is the URL of the IdP.

    b. **Local Metadata.** Use metadata stored locally on VMS. This is an alternative to including a Metadata URL.

    c. Paste metadata text in the box.

8. In the Assertions and Certificates section you can optionally enable and configure encryption for SAML assertions and responses. If enabled, you must also provide or upload certificates.

9. To enable encryption of SAML assertions, toggle **Enable Assertion Encryption**.
   If enabled, follow these steps to configure a certificate and key.

    a. Click **Add Certificate**.

    b. Paste an X.509 certificate in the box or click **Upload**, and upload an X.509 certificate file.

    c. Click **Save** to save the certificate.

    d. Click **Add Key**.

    e. Paste an X.509 key in box or click **Upload**, and upload an X.509 key file.

CONFIDENTIAL

f. Click Save.

10. To enable signatures on SAML assertion responses, toggle Enable assertion response signing. If enabled, follow these steps to configure a certificate and key. This is enabled independently of the Enable assertion encryption option. The certificate and key used for this option can be different from the ones used for Assertion Encryption.

    a. Click Add certificate.

    b. Paste an X.509 certificate in the box or click **Upload** and upload an X.509 certificate file.

    c. Click **Save** to save the certificate.

    d. Click **Add Key**.

    e. Paste an X.509 key in box or click **Upload** and upload an X.509 key file.

    f. Click **Save**.

11. Click **Save**

Result: The VAST AI OS will accept multifactor authentication for users.

4.2.3 RHEL-08-010200 – Session Timeout

Follow these steps to configure the Session Timeout.

Procedure:

1. From a networked workstation, enter the IP address of the VAST AI OS into a browser.
   The VAST AI OS GUI will appear.

2. Authenticate to the VAST AI OS with an authorized account.
   Upon successful authentication and authorization, the user will be granted into the system.

CONFIDENTIAL

3. From the left navigation menu, select **Settings** and then **Cluster**.

4. Under **General**, in the SSH Auto logout timeout field, enter the timeout in seconds.

5. Click Save and then Yes to confirm your changes.

Result: The VAST AI OS will disconnect users after a period of inactivity

4.2.4 RHEL-08-030020 – Syslog and Notifications

Follow these steps to configure notifications.

Procedure:

1. From a networked workstation and SSH client, authenticate to the VAST AI OS with an administrative account.

2. From the left navigation menu, select **Settings** and then **Notifications**.

3. To configure SMTP for sending email notifications, select **SMTP Setup** and complete the fields:

    a. **SMTP Host** - The host name of the SMTP server.
    For example: mail.company.com.

    b. **SMTP Port** - The port used by the SMTP server to send outgoing emails. The most commonly used port for SMTP is port 25, although some IPs deny its use in order to block spam. SMTP servers often support alternate ports, including port 587.

    c. **SMTP User** - User for SMTP host authentication.

    d. **SMTP Password** - The password for the SMTP user.

    e. **Use TLS** - Enable this setting to send emails over a TLS connection.

4. To set up email message properties and recipients, select **Email Setup** and complete the fields:

CONFIDENTIAL

a. **Email Sender** - The sender email address that is included in outgoing emails. This setting applies to all alarm notification emails.
Example: do_not_reply@company.com

b. **Email Subject** - The email subject to be included in outgoing emails. This optional setting applies to all alarm notification emails.

Example: VAST Alarm If you want VAST Cluster to include the alarm description as the email subject, leave this field blank.

c. **Email Recipients** - Default email recipients. These recipients receive notifications of all alarms except those triggered by events that have a different list of email recipients specified in the event definition or for which default notification actions are disabled. Enter as a comma-separated list of email addresses (no spaces).

Example: storage_admin@company.com, bsmith@company.com,

5. To specify a webhook for sending alarms to an external application (optional), select **Webhook Setup** and enter the details for the default webhook. This webhook is triggered by all events except those for this a custom webhook is defined or for which notification actions are disabled.

a. **Webhook URL** - The URL of the API endpoint of an external application, including parameters.

b. **Webhook Data** - The payload, if required, for the endpoint. You can use the $event variable to include the event message.

c. **Webhook Method** - Select the HTTP method you want to invoke with the trigger:
   i. POST
   ii. GET
   iii. PUT
   iv. PATCH
   v. DELETE

CONFIDENTIAL

6. To configure sending alarm information to a syslog server, select **Syslog Setup** and complete the fields:

    a. **Syslog Host** - Specify the syslog server's IP address.

    b. **Syslog Port** - Specify the port number that the server listens on for syslog requests. Default: 514

    c. **Syslog Protocol** - Specify either of the protocols for communicating with the remote syslog server:

        vi. TCP

        vii. UDP (default)

    d. The protocol you choose must be enabled on the syslog server.

    e. Enable VMS Audit - Toggle on (default) or off to enable or disable auditing of VMS operations.

    f. Enable Shell Audit - Toggle on or off (default) to enable or disable auditing of CNode and DNode shell commands.

    g. Enable IPMI Audit - Toggle on or off (default) to enable or disable auditing of CNode and DNode IPMI commands.

    h. Audit Logs Retention Enter the number of days to store audit logs on the syslog server.

7. Click Save.

Result: The VAST AI OS will notify specified individuals when specific events occur.

4.2.5 RHEL-08-030310 – Auditing

Follow these steps to configure Auditing.

Procedure:

1. From a networked workstation, enter the IP address of the VAST AI OS into a browser. The VAST AI OS GUI will appear.

CONFIDENTIAL

2. Authenticate to the VAST AI OS with an authorized account. Upon successful authentication and authorization, the user will be granted into the system.

3. Navigate to **Settings** → **Auditing**.

4. Add your user account to the RO view of the audit log – without this you will not be able to access audit logs.

5. Select the Enable Auditing slider.

6. You can optionally enable auditing globally for one or more protocols.

7. Click Save.

Result: The VAST AI OS will enable comprehensive auditing.

4.3 Web Server

The following control steps are derived from the DISA Web Server STIG.

4.3.1 SRG-APP-000108-WSR-000066 – Networking and Alerting

Procedure: VMS enables you to change some of the network configuration parameters of a working cluster from the VAST Web UI.

1. Authenticate to the product over a network connection.
2. From the left navigation menu, click **Settings** and then **Configure Network**.
3. Change the settings as needed:

    - **VMS IP** The IP address configured on the management interfaces on all CNodes. VAST Management Service (VMS) listens on this IP. The IP should be on the management subnet. This parameter is originally set by the **Management VIP** field in the Easy Install interface or the --mgmt_ip parameter in the configure_network.py script.

    - **NTP -** The IP(s) of up to two NTP server(s) that you want to use for time keeping.

CONFIDENTIAL

- **Default GW -** The IP address of the default gateway for the management network.

- **DNS -** The IP address(es) of up to two DNS servers that will forward DNS queries to the cluster.

- **External Netmask** The subnet mask of the management subnet.

- **IPMI Gateway** The IP of a default gateway for the IPMI interfaces on the CNodes and DNodes, if different from the management network default gateway.

- **IPMI Netmask** The subnet mask for the IPMI gateway.

- **External (MGMT IPs) +MTU definitions -** Click in the field to reconfigure the IPs of the external management interfaces on any or all of the nodes and associated configurations such as MTU. A list of all CNodes and DNodes in the cluster appears. You can click in the columns provided to change the following settings per node:

    - **External Eth MTU -** For dual-NIC CNodes where a NIC is directly connected to an external Ethernet network, use this field to set the MTU for that Ethernet network.

    - **External IB MTU -** For dual-NIC CNodes where a NIC is directly connected to an external InfiniBand data network, use this field to set the MTU for that InfiniBand network. Default: 2044

        - If NB IB type is Connected, the maximum IB NB MTU is 65520.
        - If NB IB type is Datagram, the maximum IB NB MTU is 4092.

    - **Mgmt IP** -Changes the management IP of the node.

    - **IPMI IP** - Changes the IPMI IP of the node.

- **Auto-ports-ext-iface -** Reallocates the external management IP to an interface:

    - **Outband**. Allocates the external management IP to the onboard left or right port, depending whether B2B is enabled or not.

    - **Inband**. Allocates the external management IP to the bond0 interface.

    - **Bond**. Creates a bond interface (bond1) on the two RJ45 ports, allowing for redundancy. Negates the ability to have a technician interface.

CONFIDENTIAL

- **Northband**. For clusters with dual NIC CNodes where one NIC is directly connected to an external client network, this option allocates the external management IP to the first port on the NIC that was allocated for external usage.

- **B2B - IPMI** Enables auto configuring the IPMI ports on the nodes with IP addresses according to the B2B template.

4. Click **Validate data in order to save**.

5. Review the summary of configuration changes. If you need to make any changes, click **Back to Edit the Values**, change them and click **Validate data in order to save again.**

6. When you are sure you want to go ahead with the changes, click **Approve & Save Pending Changes**.

Result: The VAST AI OS will enable comprehensive auditing.

5.0 VMS Account Lockdown Procedures

VAST Data recommends the local VMS accounts be set with the proper password complexity and lifetime policies to ensure that all local accounts created comply with STIG requirements.

5.1 APSC-DV-001680 – APSC-DV-001730 – Password Policy

Procedure:

1. Open a browser and enter the IP address of the AI OS.

2. Authenticate the product with the VASTDATA account.

3. From the left navigation menu, select **Settings** and then **VMS**.

4. The following password complexity policies can be set:

    - **Minimum number of characters.** Sets a minimum required number of characters for VMS passwords. *Set the value to 12.*

CONFIDENTIAL

- **Lowercase letters.** Enables a requirement that VMS manager passwords contain at least one lowercase character. *Set the value to 1.*

- **Uppercase letters.** Enables a requirement that VMS manager passwords contain at least one uppercase character. *Set the value to 1.*

- **Numbers**. Enables a requirement that VMS manager passwords contain at least one numeric character. *Set the value to 1.*

- **Special characters**. Enables a requirement that VMS manager passwords contain at least one non-alphanumeric character. *Set the value to 1.*

5. The following password rotation policies can be set:

    - **Prevent password changes within**. Enable this setting to prevent multiple password changes within a specified time frame. When you enable the setting with the slider, the default time frame is 24 hours. To modify, enter a number in the field provided and select hours, days or months from the dropdown. *Set the value to 24.*

    - **Password lockout**. Enable this setting to enforce password lockout after a specified number of failed attempts. When you enable the setting with the slider, the number is set by default to 10, which you can change in the field provided. *Set the value to 3.*

    - **Password expiration.** Enable this setting to expire VMS passwords after a specified time period. When you enable the setting with the slider, the expiration period is set to 180 days by default. To change the value, enter a number in the field provided and set the unit to Days or Months in the dropdown. The minimum password expiration time is one day. *Set the value to 35.*

    - **Prevent password reuse**. Enable this setting to prevent VMS manager users from reusing a specified number of previous passwords. When you enable this setting with the slider, the value is set by default to 8, which you can modify in the field provided. *Set the value to 12.*

Result. Performing these actions will bring all VMS local accounts to be in compliance with the applicable STIG controls

CONFIDENTIAL

6.0 Administrative Account Lockdown

VAST Data recommends that all day-to-day AI OS accounts be SSO/MFA accounts managed by the customers IDP infrastructure and not locally managed accounts. Following this recommendation ensures that all customer user account security policies are enforced by the AI OS without having to configure the product as an additional stovepipe. However, following this recommendation only secures off-box managed accounts and not the local accounts. The following steps will configure all VAST AI OS local accounts to be in compliance with all applicable STIG requirements.

*** * * WARNING * * ***

The Application STIG requires that all accounts have a maximum password life of 60 days. If customers choose to make this configuration, the local account passwords must be changed at least every 60 days at a minimum, or local accounts will be locked.

The VAST AI OS operates all background operations with the use of the root account. **If the root account is locked out for any reason, the product will continue to function but not be manageable.** Customers must understand the risks of implementing password life on all local accounts. Customers must have an SOP in place to change these accounts passwords on repeatable thresholds to ensure that neither the root nor vastdata account locks. If customers cannot implement this SOP, VAST Data Federal recommends that a complex password be set for all local accounts and these accounts only be used in an emergency.

The following steps are to be utilized ONLY after the customer understands the risks of implementing password complexity and lifetime requirements on local administration accounts.

6.1 APSC-DV-001680 – APSC-DV-001730 – Password Policy

Procedure: Modify the /etc/security/pwquality.conf file, which is managed by the pam_pwquality module, to configure minimum password length, required character classes (uppercase, lowercase, digits, special characters), and repetition limits.

- SSH to the IP address of the AI OS.
- Authenticate the product with the VASTDATA account.

- Enter the following command:
    - **sudo vi /etc/security/pwquality.conf**
- Make the following edits to the file:

```
# Minimum password length of 12 characters
        minlen=-12

# Set retry option to 3.
        retry=3

# Require at least one uppercase letter
        ucredit=-1

# Require at least one lowercase letter
        lcredit=-1

# Require at least one digit
        dcredit=-1

# Require at least one special character
        Ocredit=-1

# Limit repetition of the same character class to 4
        maxclassrepeat=4

# Limit repeating characters to 3
        maxrepeat = 3

# Require change of at least 4 character classes
        minclass = 4

# Require the change of at least 8 characters when passwords are changed
        difok = 8
```

Result: The appropriate password policy is applied to VAST OS local accounts.

CONFIDENTIAL

6.2 APSC-DV-001670 – Inactivity Lockout

Procedure: Add the following line to "**/etc/default/useradd**"

- INACTIVE=35

Result: Inactivity timeouts are set for VAST OS local accounts.

6.3 APSC-DV-001760 – Password Minimum Lifetime

Procedure: Configure the operating system to enforce 24 hours/1 day minimum password lifetime.

- Add, or modify the following line in the '**/etc/login.defs'** file:

  PASS_MIN_DAYS 1

Result: VAST OS local accounts have a minimum password life.

6.4 APSC-DV-001770 – Password Maximum Liftime

Configure the operating system to enforce a 60-day maximum password lifetime restriction.

Procedure: Add the following lines in /**etc/login.defs** (or modify the line to have the required value):

- PASS_MAX_DAYS 60
- Pass_warn_age – 7 notifcation

Result: VAST OS local accounts will have a password maximum lifetime limitation.

CONFIDENTIAL

*** * * WARNING * * ***

The following steps are to be utilized ONLY after the customer understands the risks of implementing performance impacting configuration changes to the product. These changes will bring the product into compliance with the identified control requirements but will impact the product's performance.

7.0 VAST OS Lockdown

The steps below detail the configuration steps needed to be taken by customers to bring the VAST AI OS into compliance with all applicable DISA Red Hat Enterprise Linux STIG.

7.1 RHEL-08-010421 - Page Poisoning

Procedure: Configure VAST OS to enable page poisoning with the following commands:

- $ sudo grubby --update-kernel=ALL --args="page_poison=1"

Add or modify the following line in "/etc/default/grub" to ensure the configuration survives kernel updates:

- GRUB_CMDLINE_LINUX="page_poison=1"

Result: Page Poisoning will be enabled on the product.

7.2 RHEL-08-010423 - SLUB/SLAB Poisoning

Procedure: Configure VAST OS to enable poisoning of SLUB/SLAB objects with the following commands:

- $ sudo grubby --update-kernel=ALL --args="slub_debug=P"

Add or modify the following line in "/etc/default/grub" to ensure the configuration survives kernel updates:

- GRUB_CMDLINE_LINUX="slub_debug=P"

CONFIDENTIAL

Result: SLUB/SLAB page poisoning is enabled.

7.3 RHEL-08-010550 – Remote Root Access

Procedure: Configure VAST OS to stop users from logging on remotely as the "root" user via SSH.

Edit the appropriate "/etc/ssh/sshd_config" file to uncomment or add the line for the "PermitRootLogin" keyword and set its value to "no":

- PermitRootLogin no

The SSH daemon must be restarted for the changes to take effect. To restart the SSH daemon, run the following command:

- $ sudo systemctl restart sshd.service

Result: Remote root access is disabled.

7.4 RHEL-08-020015 – Unlock Time

Procedure: Configure the operating system to lock an account until released by an administrator when three unsuccessful logon attempts occur in 15 minutes.

- Add/Modify the "/etc/security/faillock.conf" file to match the following line:
- unlock_time = 0

Result: Account lockout is set correctly.

7.5 RHEL-08-020353 – Default Account Permissions

Procedure: Configure the operating system to define default permissions for all authenticated users in such a way that the user can only read and modify their own files.

CONFIDENTIAL

Add or edit the lines for the "UMASK" parameter in the "/etc/bashrc", "/etc/csh.cshrc" and "/etc/profile"files to "077":

- UMASK 077

Result: Default account permissions are set to 077.

7.6 RHEL-08-040004 – Page-Table Isolation

Procedure: Configure RHEL 8 to enable kernel page-table isolation with the following command:

- $ sudo grubby --update-kernel=ALL --args="pti=on"

Add or modify the following line in "/etc/default/grub" to ensure the configuration survives kernel updates:

- GRUB_CMDLINE_LINUX="pti=on"

Result: Page-Table Isolation is enabled.

7.7 RHEL-08-040101 – Firewalld Enablement

Procedure: Configure "firewalld" to protect the operating system with the following command:

- $ sudo systemctl enable firewalld

Result: The firewall is enabled.

7.8 RHEL-08-040125 – Tempt Folder Noexec

Procedure: Configure the system so that /tmp is mounted with the "noexec" option by adding /modifying the /etc/fstab with the following line:

- /dev/mapper/rhel-tmp /tmp xfs defaults,nodev,nosuid,noexec 0 0

CONFIDENTIAL

Result: The temp folder is set with he noexec option.

7.9 RHEL-08-010383 – Defaults

Procedure: Define the following in the Defaults section of the /etc/sudoers file or a configuration file in the /etc/sudoers.d/ directory. Add these entries:

```
Defaults !targetpw
Defaults !rootpw
Defaults !runaspw
```

Remove any configurations that conflict with the above from the following locations:

```
/etc/sudoers
/etc/sudoers.d/
```

Result: The firewall policy is set to be compliant with the STIGs.

7.10 RHEL-08-030570 – Chacl Auditing

Procedure: Configure the audit system to generate an audit event for any successful/ unsuccessful use of the "chacl" command by adding or updating the following rule in the "/etc/audit/rules.d/audit.rules" file:

- -a always,exit -F path=/usr/bin/chacl -F perm=x -F auid>=1000 -F auid!=unset -k perm_mod

The audit daemon must be restarted for the changes to take effect.

Result: The audit policy is set to be compliant with the STIGs.

7.11 RHEL-08-040021 – ATM Blacklisting

Procedure: Add or update the following lines in the file "/etc/modprobe.d/blacklist.conf":

- install atm /bin/false
- blacklist atm

CONFIDENTIAL

Reboot the system for the settings to take effect.

Result: The ATM package compliant with the STIGs.

7.12 RHEL-08-040022 – CAN Blacklisting

Procedure: Add or update the following lines in the file "/etc/modprobe.d/blacklist.conf":

- install can /bin/false
- blacklist can

Reboot the system for the settings to take effect.

Result: The CAN package is compliant with the STIGs.

7.13 RHEL-08-040023 – SCTP Blacklisting

Procedure: Configure the operating system to disable the ability to use the SCTP kernel module.

Add or update the following lines in the file "/etc/modprobe.d/blacklist.conf":

- install sctp /bin/false
- blacklist sctp

Reboot the system for the settings to take effect.

Result: The SCTP package is compliant with the STIGs.

7.14 RHEL-08-040024 – TIPC Blacklisting

Procedure: Configure the operating system to disable the ability to use the TIPC protocol kernel module.

CONFIDENTIAL

Add or update the following lines in the file "/etc/modprobe.d/blacklist.conf":

- install tipc /bin/false
- blacklist tipc

Reboot the system for the settings to take effect.

Result: The TIPC package is compliant with the STIGs.

7.15 RHEL-08-040025 – Cramfs Kernel Blacklisting

Procedure: Configure the operating system to disable the ability to use the cramfs kernel module.

Add or update the following lines in the file "/etc/modprobe.d/blacklist.conf":

- install cramfs /bin/false
- blacklist cramfs

Reboot the system for the settings to take effect.

Result: The Cramfs package is compliant with the STIGs.

7.16 RHEL-08-040026 – Firewire Core Blacklisting

Procedure: Configure the operating system to disable the ability to use the firewire-core kernel module.

Add or update the following lines in the file "/etc/modprobe.d/blacklist.conf":

- install firewire-core /bin/false
- blacklist firewire-core

CONFIDENTIAL

Reboot the system for the settings to take effect.

Result: The firewall core package is compliant with the STIGs.

7.17 RHEL-08-040080 – USB Blacklisting

Procedure: Configure the operating system to disable the ability to use the USB Storage kernel module and the ability to use USB mass storage devices.

Add or update the following lines in the file "/etc/modprobe.d/blacklist.conf":

install usb-storage /bin/false

blacklist usb-storage

Reboot the system for the settings to take effect.

Result: USB devices are blocked on the system and compliant with the STIGs.

7.18 RHEL-08-040111 – Bluetooth Blacklisting

Procedure: Configure the operating system to disable the Bluetooth adapter when not in use.

Build or modify the "/etc/modprobe.d/bluetooth.conf" file with the following line:

- install bluetooth /bin/false

Disable the ability to use the Bluetooth kernel module.

- $ sudo vi /etc/modprobe.d/blacklist.conf

Add or update the line:

- blacklist bluetooth

CONFIDENTIAL

Reboot the system for the settings to take effect.

Result: Bluetooth devices are blocked on the system and compliant with the STIGs.

7.19 RHEL-08-040249 – IPv4 Forwarding

Procedure: Configure RHEL 8 to not forward IPv4 source-routed packets by default.

Add or edit the following line in a system configuration file, in the "/etc/sysctl.d/" directory:

- net.ipv4.conf.default.accept_source_route=0

Remove any configurations that conflict with the above from the following locations:

- /run/sysctl.d/*.conf
- /usr/local/lib/sysctl.d/*.conf
- /usr/lib/sysctl.d/*.conf
- /lib/sysctl.d/*.conf
- /etc/sysctl.conf
- /etc/sysctl.d/*.conf

Load settings from all system configuration files with the following command:

- $ sudo sysctl –system

Result: IPv4 forwarding is blocked on the system and compliant with the STIGs.

7.20 RHEL-08-040250 - IPv6 Forwarding

Procedure: Configure RHEL 8 to not forward IPv6 source-routed packets by default.

Add or edit the following line in a system configuration file, in the "/etc/sysctl.d/" directory:

- net.ipv6.conf.default.accept_source_route=0

CONFIDENTIAL

Remove any configurations that conflict with the above from the following locations:

- /run/sysctl.d/*.conf
- /usr/local/lib/sysctl.d/*.conf
- /usr/lib/sysctl.d/*.conf
- /lib/sysctl.d/*.conf
- /etc/sysctl.conf
- /etc/sysctl.d/*.conf

Load settings from all system configuration files with the following command:

- $ sudo sysctl –system

Result: IPv6 forwarding is blocked on the system and compliant with the STIGs.

7.21 RHEL-08-040279 – IPv4 ICMP

Procedure: Configure RHEL 8 to ignore IPv4 ICMP redirect messages.

Add or edit the following line in a system configuration file, in the "/etc/sysctl.d/" directory:

- net.ipv4.conf.all.accept_redirects = 0

Remove any configurations that conflict with the above from the following locations:

- /run/sysctl.d/*.conf
- /usr/local/lib/sysctl.d/*.conf
- /usr/lib/sysctl.d/*.conf
- /lib/sysctl.d/*.conf
- /etc/sysctl.conf
- /etc/sysctl.d/*.conf

Load settings from all system configuration files with the following command:

CONFIDENTIAL

- $ sudo sysctl –system

Result: IPv4 ICMP is disabled on the system and compliant with the STIGs.

7.22 RHEL-08-040284 – Namespace Disabling

Procedure: Configure RHEL 8 to disable the use of user namespaces by adding the following line to a file, in the "/etc/sysctl.d" directory:

Note: User namespaces are used primarily for Linux containers. If containers are in use, this requirement is not applicable.

- user.max_user_namespaces = 0

Remove any configurations that conflict with the above from the following locations:

- /run/sysctl.d/*.conf
- /usr/local/lib/sysctl.d/*.conf
- /usr/lib/sysctl.d/*.conf
- /lib/sysctl.d/*.conf
- /etc/sysctl.conf
- /etc/sysctl.d/*.conf

The system configuration files need to be reloaded for the changes to take effect. To reload the contents of the files, run the following command:

- $ sudo sysctl –system

Result: Namespaces are disabled on the system and compliant with the STIGs.

8.0 Firewall Configuration

8.1 RHEL-08-040030 – Firewall Configuration

CONFIDENTIAL

Procedure: The VASTOS firewall script is owned by Caleb Vorderbruggem**.**

The AI OS firewall script contents are listed below:

```
# Script: VMS OS Hardening
# Description: Apply additional security policies to vast os.
#   - Firwalld: Loops through all existed (non-excluded) interface on the host, creating a zone for each.
# Version: v0.2.0
# Date: 2025-02-04
# Usage ./vms_hardening.sh
#   - No Arguments Required


PRIMARY_INTERFACE=${PRIMARY_INTERFACE:-$(ip route | grep default | awk '{print $5}' | head -n1)}
declare -a EXCLUDED_INTERFACES=("lo" "docker0" "${PRIMARY_INTERFACE}")

function create_directories() {
  sudo mkdir -p /etc/firewalld/zones
  sudo mkdir -p /etc/firewalld/services
}

configure_firewalld() {
   sudo sed -i 's/^DefaultZone=.*/DefaultZone=drop/' /etc/firewalld/firewalld.conf
   sudo sed -i 's/^AllowZoneDrifting=.*/AllowZoneDrifting=no/' /etc/firewalld/firewalld.conf
}

configure_firewalld_zone_vms() {
   INTERFACE=$1
   firewall-cmd --permanent --new-zone=VMS
   firewall-cmd --permanent --zone=VMS --set-target=DROP
   firewall-cmd --permanent --zone=VMS --add-interface=${INTERFACE}
   firewall-cmd --permanent --zone=VMS --add-service={ssh,vast-vms}
}

configure_firewalld_zone_docker() {
   firewall-cmd --permanent --new-zone=docker
   firewall-cmd --permanent --zone=docker --set-target=DROP
   firewall-cmd --permanent --zone=docker --add-interface=docker0
```

CONFIDENTIAL

```
    firewall-cmd --permanent --zone=docker --add-port=1-65535/tcp
    firewall-cmd --permanent --zone=docker --add-port=1-65535/udp
    firewall-cmd --permanent --zone=docker --add-masquerade
}

configure_firewalld_zone() {
    INTERFACE=$1
    ZONE_NAME=${INTERFACE//./_}
    firewall-cmd --permanent --new-zone=${ZONE_NAME}
    firewall-cmd --permanent --zone=${ZONE_NAME} --set-target=DROP
    firewall-cmd --permanent --zone=${ZONE_NAME} --add-interface=${INTERFACE}
    firewall-cmd --permanent --zone=${ZONE_NAME} --add-port=1-65535/tcp
    firewall-cmd --permanent --zone=${ZONE_NAME} --add-port=1-65535/udp
}


configure_firewalld_services(){
cat << EOF > /etc/firewalld/services/vast-vms.xml
<?xml version="1.0" encoding="utf-8"?>
<service>
  <short>VAST Management System (HTTP)</short>
   <description>VAST Management System provides a web interface for managing all aspect of VAST OS.</description>
  <port protocol="tcp" port="443"/>
  <port protocol="tcp" port="80"/>
</service>
EOF
}

is_excluded() {
    local INTERFACE=$1
    for EXCLUDED in "${EXCLUDED_INTERFACES[@]}"; do
        if [[ "$INTERFACE" == "$EXCLUDED" ]]; then
            return 0
        fi
    done
    return 1
}

systemctl enable firewalld --now
```

CONFIDENTIAL

```
create_directories
configure_firewalld
configure_firewalld_services

firewall-cmd --reload

configure_firewalld_zone_docker
configure_firewalld_zone_vms ${PRIMARY_INTERFACE}

for INTERFACE in $(ip -o link show | awk -F': ' '{print $2}' | cut -d'@' -f1)
do
   is_excluded ${INTERFACE} && continue
   configure_firewalld_zone $INTERFACE
done

firewall-cmd --reload
```

9.0 VASTOS STIG Script

The contents of the VASTOS STIG script are available below. To get this script, please contact the author of this paper.

```
#!/usr/bin/env bash

# Script: VMS OS Hardening
# Description: Apply additional security policies to vast os.
# Version: v1.0.0
# Date: 2025-04-10
# Usage ./vms_hardening.sh --help

################################################################################
# Global Variables
################################################################################
DRY_RUN=false
HARDENING_LEVEL=1

################################################################################
# Utility Functions
################################################################################

function show_help {
```

CONFIDENTIAL

```
    cat << EOF
VMS OS Hardening Script

Usage: ./vms_hardening.sh [OPTIONS]

Options:
  --help              Show this help message and exit
  --dry-run           Check compliance without making any changes
  --level <1|2>       Specify hardening level (default: 1)
                      Level 1: Basic security settings
                      Level 2: Advanced security settings that may impact performance (firewalld is always reconfigured)

Examples:
  ./vms_hardening.sh                # Apply level 1 hardening
  ./vms_hardening.sh --level 2      # Apply level 1 and 2 hardening
  ./vms_hardening.sh --dry-run      # Check compliance without making changes

EOF
    exit 0
}

function parse_args {
    while [[ $# -gt 0 ]]; do
        case "$1" in
            --help)
                show_help
                ;;
            --dry-run)
                DRY_RUN=true
                shift
                ;;
            --level)
                if [[ "$2" =~ ^[1-2]$ ]]; then
                    HARDENING_LEVEL=$2
                    shift 2
                else
                    error "Invalid hardening level. Use 1 or 2."
                fi
                ;;
            *)
                error "Unknown option: $1. Use --help for usage information."
                ;;
        esac
    done
}

print_green() {
    echo -e -n "\033[0;32m$1\033[0m"
```

CONFIDENTIAL

```
}

print_red() {
   echo -e -n "\033[0;31m$1\033[0m"
}

print_yellow() {
   echo -e -n "\033[0;33m$1\033[0m"
}

error() {
   local message="$1"
   local exit_code="${2:-1}"

   # Print error message in red to stderr
   echo -e "\033[0;31mERROR: $message\033[0m" >&2

   exit "$exit_code"
}

################################################################################
# Password Policy (minlen) [APSC-DV-001680, APSC-DV-001730]
# Configure [OK]
# Check     [OK]
################################################################################
PASSWORD_POLICY_MIN_LENGTH=12

function password_policy_min_length_check {
   # Find `minlen` at start of line | Only count last occurance | Check value
   grep -E "^minlen\s*=\s*(-?[0-9]+)" /etc/security/pwquality.conf | tail -1 | awk -v min=${PASSWORD_POLICY_MIN_LENGTH} -F=
'{gsub(/[[:space:]]/, "", $2); if ($2 >= min) exit 0; else exit 1}'
}

function password_policy_min_length_configure {
   if ! grep -q "^minlen\s*=" /etc/security/pwquality.conf; then
      # minlen doesn't exist, append it
      echo "minlen = ${PASSWORD_POLICY_MIN_LENGTH}" >> /etc/security/pwquality.conf
   else
      # minlen exists but may be incorrect, update it
      sed -i "s/^minlen\s*=.*$/minlen = ${PASSWORD_POLICY_MIN_LENGTH}/" /etc/security/pwquality.conf
   fi
}

################################################################################
# Password Policy (ucredit) [APSC-DV-001680, APSC-DV-001730]
# Configure [OK]
# Check     [OK]
################################################################################
```

CONFIDENTIAL

```
PASSWORD_POLICY_UCREDDIT=-1

function password_policy_ucredit_check {
   # Find `ucredit` at start of line | Only count last occurance | Check value
   grep -E "^ucredit\s*=\s*(-?[0-9]+)" /etc/security/pwquality.conf | tail -1 | awk -v ucredit=${PASSWORD_POLICY_UCREDDIT} -F=
'{gsub(/[[:space:]]/, "", $2); if ($2 == ucredit) exit 0; else exit 1}'
}

function password_policy_ucredit_configure {
   if ! grep -q "^ucredit\s*=" /etc/security/pwquality.conf; then
      # ucredit doesn't exist, append it
      echo "ucredit = ${PASSWORD_POLICY_UCREDDIT}" >> /etc/security/pwquality.conf
   else
      # ucredit exists but may be incorrect, update it
      sed -i "s/^ucredit\s*=.*$/ucredit = ${PASSWORD_POLICY_UCREDDIT}/" /etc/security/pwquality.conf
   fi
}

################################################################################
# Password Policy (lcredit) [APSC-DV-001680, APSC-DV-001730]
# Configure [OK]
# Check     [OK]
################################################################################
PASSWORD_POLICY_LCREDDIT=-1

function password_policy_lcredit_check {
   # Find `lcredit` at start of line | Only count last occurance | Check value
   grep -E "^lcredit\s*=\s*(-?[0-9]+)" /etc/security/pwquality.conf | tail -1 | awk -v lcredit=${PASSWORD_POLICY_LCREDDIT} -F=
'{gsub(/[[:space:]]/, "", $2); if ($2 == lcredit) exit 0; else exit 1}'
}

function password_policy_lcredit_configure {
   if ! grep -q "^lcredit\s*=" /etc/security/pwquality.conf; then
      # lcredit doesn't exist, append it
      echo "lcredit = ${PASSWORD_POLICY_LCREDDIT}" >> /etc/security/pwquality.conf
   else
      # lcredit exists but may be incorrect, update it
      sed -i "s/^lcredit\s*=.*$/lcredit = ${PASSWORD_POLICY_LCREDDIT}/" /etc/security/pwquality.conf
   fi
}

################################################################################
# Password Policy (dcredit) [APSC-DV-001680, APSC-DV-001730]
# Configure [OK]
# Check     [OK]
################################################################################
PASSWORD_POLICY_DCREDDIT=-1
```

CONFIDENTIAL

```
function password_policy_dcredit_check {
   # Find `dcredit` at start of line | Only count last occurance | Check value
   grep -E "^dcredit\s*=\s*(-?[0-9]+)" /etc/security/pwquality.conf | tail -1 | awk -v dcredit=${PASSWORD_POLICY_DCREDDIT} -F=
'{gsub(/[[:space:]]/, "", $2); if ($2 == dcredit) exit 0; else exit 1}'
}

function password_policy_dcredit_configure {
   if ! grep -q "^dcredit\s*=" /etc/security/pwquality.conf; then
      # lcredit doesn't exist, append it
      echo "dcredit = ${PASSWORD_POLICY_DCREDDIT}" >> /etc/security/pwquality.conf
   else
      # lcredit exists but may be incorrect, update it
      sed -i "s/^dcredit\s*=.*$/dcredit = ${PASSWORD_POLICY_DCREDDIT}/" /etc/security/pwquality.conf
   fi
}

###############################################################################
# Password Policy (ocredit) [APSC-DV-001680, APSC-DV-001730]
# Configure [OK]
# Check     [OK]
###############################################################################
PASSWORD_POLICY_OCREDIT=-1

function password_policy_ocredit_check {
   # Find `maxrepeat` at start of line | Only count last occurance | Check value
   grep -E "^ocredit\s*=\s*(-?[0-9]+)" /etc/security/pwquality.conf | tail -1 | awk -v ocredit="${PASSWORD_POLICY_OCREDIT}" -F=
'{gsub(/[[:space:]]/, "", $2); if ($2 == ocredit) exit 0; else exit 1}'
}

function password_policy_ocredit_configure {
   if ! grep -q "^ocredit\s*=" /etc/security/pwquality.conf; then
      # lcredit doesn't exist, append it
      echo "ocredit = ${PASSWORD_POLICY_OCREDIT}" >> /etc/security/pwquality.conf
   else
      # lcredit exists but may be incorrect, update it
      sed -i "s/^ocredit\s*=.*$/ocredit = ${PASSWORD_POLICY_OCREDIT}/" /etc/security/pwquality.conf
   fi
}

###############################################################################
# Password Policy (maxrepeat) [APSC-DV-001680, APSC-DV-001730]
# Configure [OK]
# Check     [OK]
###############################################################################
PASSWORD_POLICY_MAXREPEAT=2

function password_policy_maxrepeat_check {
   # Find `maxrepeat` at start of line | Only count last occurance | Check length
```

CONFIDENTIAL

```
    grep -E "^maxrepeat\s*=\s*([0-9]+)" /etc/security/pwquality.conf | tail -1 | awk -v maxrepeat=${PASSWORD_POLICY_MAXREPEAT} -F= '{gsub(/[[:space:]]/, "", $2); if ($2 == maxrepeat) exit 0; else exit 1}'
}

function password_policy_maxrepeat_configure {
  if ! grep -q "^maxrepeat\s*=" /etc/security/pwquality.conf; then
    # lcredit doesn't exist, append it
    echo "maxrepeat = ${PASSWORD_POLICY_MAXREPEAT}" >> /etc/security/pwquality.conf
  else
    # lcredit exists but may be incorrect, update it
    sed -i "s/^maxrepeat\s*=.*$/maxrepeat = ${PASSWORD_POLICY_MAXREPEAT}/" /etc/security/pwquality.conf
  fi
}

################################################################################
# Inactivity Lockout [APSC-DV-001670]
# Configure [OK]
# Check     [OK]
################################################################################
INACTIVE_DAYS=35

function inactivity_lockout_check {
  grep -E "^INACTIVE\s*=\s*${INACTIVE_DAYS}" /etc/default/useradd >/dev/null 2>&1
}

function inactivity_lockout_configure {
  if ! grep -q "^INACTIVE\s*=" /etc/default/useradd; then
    # INACTIVE doesn't exist, append it
    echo "INACTIVE=${INACTIVE_DAYS}" >> /etc/default/useradd
  else
    # INACTIVE exists but may be incorrect, update it
    sed -i "s/^INACTIVE\s*=.*$/INACTIVE=${INACTIVE_DAYS}/" /etc/default/useradd
  fi
}

################################################################################
# Password Minimum Lifetime [APSC-DV-001760]
# Configure [OK]
# Check     [OK]
################################################################################
PASSWORD_MIN_DAYS=1

function password_min_days_check {
  # Check if PASS_MIN_DAYS=1 is set in /etc/login.defs
  grep -E "^PASS_MIN_DAYS\s+${PASSWORD_MIN_DAYS}" /etc/login.defs >/dev/null 2>&1
}

function password_min_days_configure {
```

CONFIDENTIAL

```
    if ! grep -q "^PASS_MIN_DAYS" /etc/login.defs; then
        # PASS_MIN_DAYS doesn't exist, append it
        echo "PASS_MIN_DAYS ${PASSWORD_MIN_DAYS}" >> /etc/login.defs
    else
        # PASS_MIN_DAYS exists but may be incorrect, update it
        sed -i "s/^PASS_MIN_DAYS\s\+.*$/PASS_MIN_DAYS ${PASSWORD_MIN_DAYS}/" /etc/login.defs
    fi

    # Apply to existing users as well
    for USER in $(awk -F: '($3 >= 1000) && ($7 != "/sbin/nologin") {print $1}' /etc/passwd); do
        chage --mindays ${PASSWORD_MIN_DAYS} ${USER}
    done
}

################################################################################
# Password Maximum Lifetime [APSC-DV-001770]
# Configure [OK]
# Check     [OK]
################################################################################
PASSWORD_MAX_DAYS=60

function password_max_days_check {
    # Check if PASS_MAX_DAYS=60 is set in /etc/login.defs
    grep -E "^PASS_MAX_DAYS\s+${PASSWORD_MAX_DAYS}" /etc/login.defs >/dev/null 2>&1
}

function password_max_days_configure {
    if ! grep -q "^PASS_MAX_DAYS" /etc/login.defs; then
        # PASS_MAX_DAYS doesn't exist, append it
        echo "PASS_MAX_DAYS ${PASSWORD_MAX_DAYS}" >> /etc/login.defs
    else
        # PASS_MAX_DAYS exists but may be incorrect, update it
        sed -i "s/^PASS_MAX_DAYS\s\+.*$/PASS_MAX_DAYS ${PASSWORD_MAX_DAYS}/" /etc/login.defs
    fi

    # Apply to existing users as well
    for USER in $(awk -F: '($3 >= 1000) && ($7 != "/sbin/nologin") {print $1}' /etc/passwd); do
        chage --maxdays ${PASSWORD_MAX_DAYS} ${USER}
    done
}

################################################################################
# Password Expiration Warning [APSC-DV-001770]
# Configure [OK]
# Check     [OK]
################################################################################
PASSWORD_WARN_AGE=7
```

CONFIDENTIAL

```
function password_warn_age_check {
   # Check if PASS_WARN_AGE=7 is set in /etc/login.defs
   grep -E "^PASS_WARN_AGE\s+${PASSWORD_WARN_AGE}" /etc/login.defs >/dev/null 2>&1
}

function password_warn_age_configure {
   if ! grep -q "^PASS_WARN_AGE" /etc/login.defs; then
      # PASS_WARN_AGE doesn't exist, append it
      echo "PASS_WARN_AGE ${PASSWORD_WARN_AGE}" >> /etc/login.defs
   else
      # PASS_WARN_AGE exists but may be incorrect, update it
      sed -i "s/^PASS_WARN_AGE\s\+.*$/PASS_WARN_AGE ${PASSWORD_WARN_AGE}/" /etc/login.defs
   fi

   # Apply to existing users as well
   for USER in $(awk -F: '($3 >= 1000) && ($7 != "/sbin/nologin") {print $1}' /etc/passwd); do
      chage --warndays ${PASSWORD_WARN_AGE} ${USER}
   done
}

################################################################################
# Page Poisoning (memory protection) [RHEL-08-010421]
# Configure [OK]
# Check     [OK]
################################################################################
function page_poisoning_check {
   # Check if page_poison=1 is in GRUB_CMDLINE_LINUX
   grep -E "^GRUB_CMDLINE_LINUX=\".*page_poison=1.*\"" /etc/default/grub >/dev/null 2>&1
}

function page_poisoning_configure {
   # Check if GRUB_CMDLINE_LINUX exists
   if grep -q "^GRUB_CMDLINE_LINUX=" /etc/default/grub; then
      # Extract current value
      current_value=$(grep "^GRUB_CMDLINE_LINUX=" /etc/default/grub | cut -d"" -f2)

      # Check if page_poison already exists
      if echo "$current_value" | grep -q "page_poison="; then
         # Replace existing page_poison value
         new_value=$(echo "$current_value" | sed 's/page_poison=[^ "]*/page_poison=1/g')
      else
         # Add page_poison=1 to existing parameters
         new_value="$current_value page_poison=1"
      fi

      # Update GRUB_CMDLINE_LINUX
      sed -i "s|^GRUB_CMDLINE_LINUX=\".*\"|GRUB_CMDLINE_LINUX=\"$new_value\"|" /etc/default/grub
   else
```

CONFIDENTIAL

```
    # GRUB_CMDLINE_LINUX doesn't exist, create it
    echo 'GRUB_CMDLINE_LINUX="page_poison=1"' >> /etc/default/grub
  fi

  # Update grub configuration
  if command -v grub2-mkconfig >/dev/null 2>&1; then
    grub2-mkconfig -o /boot/grub2/grub.cfg
  elif command -v update-grub >/dev/null 2>&1; then
    update-grub
  else
    echo "WARNING: Could not update grub configuration. Please run grub-mkconfig manually."
  fi
}

################################################################################
# SLUB/SLAB Poisoning [RHEL-08-010423]
# Configure [OK]
# Check     [OK]
################################################################################
function slub_poisoning_check {
  # Check if slub_debug=P is in GRUB_CMDLINE_LINUX
  grep -E "^GRUB_CMDLINE_LINUX=\".*slub_debug=P.*\"" /etc/default/grub >/dev/null 2>&1
}

function slub_poisoning_configure {
  # Check if GRUB_CMDLINE_LINUX exists
  if grep -q "^GRUB_CMDLINE_LINUX=" /etc/default/grub; then
    # Extract current value
    current_value=$(grep "^GRUB_CMDLINE_LINUX=" /etc/default/grub | cut -d"" -f2)

    # Check if slub_debug already exists
    if echo "$current_value" | grep -q "slub_debug="; then
      # Replace existing slub_debug value
      new_value=$(echo "$current_value" | sed 's/slub_debug=[^ "]*/slub_debug=P/g')
    else
      # Add slub_debug=P to existing parameters
      new_value="$current_value slub_debug=P"
    fi

    # Update GRUB_CMDLINE_LINUX
    sed -i "s|^GRUB_CMDLINE_LINUX=\".*\"|GRUB_CMDLINE_LINUX=\"$new_value\"|" /etc/default/grub
  else
    # GRUB_CMDLINE_LINUX doesn't exist, create it
    echo 'GRUB_CMDLINE_LINUX="slub_debug=P"' >> /etc/default/grub
  fi

  # Update grub configuration
  if command -v grub2-mkconfig >/dev/null 2>&1; then
```

CONFIDENTIAL

```
    grub2-mkconfig -o /boot/grub2/grub.cfg
  elif command -v update-grub >/dev/null 2>&1; then
    update-grub
  else
    echo "WARNING: Could not update grub configuration. Please run grub-mkconfig manually."
  fi
}

################################################################################
# Page-Table Isolation [RHEL-08-040004]
# Configure [OK]
# Check     [OK]
################################################################################
function pti_check {
  # Check if pti=on is in GRUB_CMDLINE_LINUX
  grep -E "^GRUB_CMDLINE_LINUX=\".*pti=on.*\"" /etc/default/grub >/dev/null 2>&1
}

function pti_configure {
  # Check if GRUB_CMDLINE_LINUX exists
  if grep -q "^GRUB_CMDLINE_LINUX=" /etc/default/grub; then
    # Extract current value
    current_value=$(grep "^GRUB_CMDLINE_LINUX=" /etc/default/grub | cut -d"" -f2)

    # Check if pti already exists
    if echo "$current_value" | grep -q "pti="; then
      # Replace existing pti value
      new_value=$(echo "$current_value" | sed 's/pti=[^ "]*/pti=on/g')
    else
      # Add pti=on to existing parameters
      new_value="$current_value pti=on"
    fi

    # Update GRUB_CMDLINE_LINUX
    sed -i "s|^GRUB_CMDLINE_LINUX=\".*\"|GRUB_CMDLINE_LINUX=\"$new_value\"|" /etc/default/grub
  else
    # GRUB_CMDLINE_LINUX doesn't exist, create it
    echo 'GRUB_CMDLINE_LINUX="pti=on"' >> /etc/default/grub
  fi

  # Update grub configuration
  if command -v grub2-mkconfig >/dev/null 2>&1; then
    grub2-mkconfig -o /boot/grub2/grub.cfg
  elif command -v update-grub >/dev/null 2>&1; then
    update-grub
  else
    echo "WARNING: Could not update grub configuration. Please run grub-mkconfig manually."
  fi
```

CONFIDENTIAL

```
}

################################################################################
# Disable Remote Root Access [RHEL-08-010550]
# Configure [OK]
# Check     [OK]
################################################################################
function disable_root_login_check {
   # Check if PermitRootLogin is set to no in sshd_config
   grep -E "^PermitRootLogin\s+no" /etc/ssh/sshd_config >/dev/null 2>&1
}

function disable_root_login_configure {
   if grep -q "^PermitRootLogin" /etc/ssh/sshd_config; then
      # PermitRootLogin exists, update it
      sed -i 's/^PermitRootLogin.*/PermitRootLogin no/' /etc/ssh/sshd_config
   else
      # PermitRootLogin doesn't exist, add it
      echo "PermitRootLogin no" >> /etc/ssh/sshd_config
   fi

   # Restart SSH service to apply changes
   systemctl restart sshd.service
}

################################################################################
# Interactive User Home Directories [RHEL-08-010570]
# Configure [OK]
# Check     [OK]
################################################################################
function get_home_partition() {
   # Find the filesystem containing /home
   # Try to get the specific partition for /home first
   local home_mount=$(findmnt -n -o SOURCE --target /home 2>/dev/null)

   # If /home doesn't have its own partition, find the partition containing it
   if [ -z "$home_mount" ]; then
      home_mount=$(df /home | awk 'NR==2 {print $1}')
   fi

   echo "$home_mount"
}

function nosuid_home_check {
   local home_partition=$(get_home_partition)

   # Check if nosuid option is present for the home partition in fstab
   grep -E "^${home_partition}.*nosuid" /etc/fstab >/dev/null 2>&1 ||
```

CONFIDENTIAL

```
    grep -E "\s/home\s.*nosuid" /etc/fstab >/dev/null 2>&1
}

function nosuid_home_configure {
    local home_partition=$(get_home_partition)

    # Check if the partition is already in fstab
    if grep -q "^${home_partition}" /etc/fstab; then
        # Check if nosuid is already set
        if ! grep -E "^${home_partition}.*nosuid" /etc/fstab >/dev/null 2>&1; then
            # Add nosuid to existing options
              sed   -i   "s|^\(${home_partition}[[:space:]]\+[^[:space:]]\+[[:space:]]\+[^[:space:]]\+[[:space:]]\+\)\([^[:space:]]\+\)|\\1\\2,nosuid|"
/etc/fstab
        fi
    elif grep -q "\s/home\s" /etc/fstab; then
        # /home entry exists but without nosuid
        if ! grep -E "\s/home\s.*nosuid" /etc/fstab >/dev/null 2>&1; then
            # Add nosuid to existing options
            sed -i "s|\(.*[[:space:]]\+/home[[:space:]]\+[^[:space:]]\+[[:space:]]\+\)\([^[:space:]]\+\)|\\1\\2,nosuid|" /etc/fstab
        fi
    else
        echo "WARNING: No entry for /home found in /etc/fstab. Please add it manually with the nosuid option."
        return 1
    fi

    # Remount the filesystem to apply changes
    if ! $DRY_RUN; then
        mount -o remount /home 2>/dev/null || true
    fi
}

################################################################################
# Permanent Account Lockout (Unlock Time) [RHEL-08-020015]
# Configure [OK]
# Check     [OK]
################################################################################
function faillock_unlock_time_check {
    grep -E "^\s*unlock_time\s*=\s*0\s*$" /etc/security/faillock.conf >/dev/null 2>&1
}

function faillock_unlock_time_configure {
    if [ ! -f /etc/security/faillock.conf ]; then
        touch /etc/security/faillock.conf
    fi

    # Check if unlock_time is already configured
    if grep -E "^\s*unlock_time\s*=" /etc/security/faillock.conf >/dev/null 2>&1; then
        # Replace existing setting
```

CONFIDENTIAL

```
        sed -i 's/^\s*unlock_time\s*=.*$/unlock_time = 0/' /etc/security/faillock.conf
    else
        # Add the setting
        echo "unlock_time = 0" >> /etc/security/faillock.conf
    fi
}

################################################################################
# Default Account Permissions - bashrc [RHEL-08-020353]
# Configure [OK]
# Check     [OK]
################################################################################
function umask_bashrc_check {
    # Check if UMASK=077 is set in /etc/bashrc
    grep -E "^\s*umask\s+077" /etc/bashrc >/dev/null 2>&1
}

function umask_bashrc_configure {
    if grep -q "^\s*umask\s\+" /etc/bashrc; then
        # Replace existing UMASK setting
        sed -i 's/^\s*umask\s\+[0-9][0-9][0-9]/umask 077/' /etc/bashrc
    else
        # Add UMASK setting
        echo "umask 077" >> /etc/bashrc
    fi
}

################################################################################
# Default Account Permissions - csh.cshrc [RHEL-08-020353]
# Configure [OK]
# Check     [OK]
################################################################################
function umask_cshrc_check {
    # Check if UMASK=077 is set in /etc/csh.cshrc
    grep -E "^\s*umask\s+077" /etc/csh.cshrc >/dev/null 2>&1
}

function umask_cshrc_configure {
    if grep -q "^\s*umask\s\+" /etc/csh.cshrc; then
        # Replace existing UMASK setting
        sed -i 's/^\s*umask\s\+[0-9][0-9][0-9]/umask 077/' /etc/csh.cshrc
    else
        # Add UMASK setting
        echo "umask 077" >> /etc/csh.cshrc
    fi
}

################################################################################
```

CONFIDENTIAL

```
# Default Account Permissions - profile [RHEL-08-020353]
# Configure [OK]
# Check     [OK]
################################################################################
function umask_profile_check {
   # Check if UMASK=077 is set in /etc/profile
   grep -E "^\s*umask\s+077" /etc/profile >/dev/null 2>&1
}

function umask_profile_configure {
   if grep -q "^\s*umask\s\+" /etc/profile; then
      # Replace existing UMASK setting
      sed -i 's/^\s*umask\s\+[0-9][0-9][0-9]/umask 077/' /etc/profile
   else
      # Add UMASK setting
      echo "umask 077" >> /etc/profile
   fi
}

################################################################################
# Temporary Folder No Execute [RHEL-08-040125]
# Configure [OK]
# Check     [OK]
################################################################################
function get_tmp_mount() {
   # Find the filesystem containing /tmp
   local tmp_mount=$(findmnt -n -o SOURCE --target /tmp 2>/dev/null)

   # If /tmp doesn't have its own mount, check if it's a tmpfs
   if [ -z "$tmp_mount" ]; then
      # Check if /tmp is listed in fstab
      tmp_mount=$(grep -E '\s/tmp\s' /etc/fstab | awk '{print $1}')
   fi

   echo "$tmp_mount"
}

function tmp_noexec_check {
   # Check if /tmp has its own mount
   local tmp_mount=$(get_tmp_mount)

   # If we found a mount for /tmp, check if noexec is set
   if [ -n "$tmp_mount" ]; then
      # Check if noexec option is present for the /tmp mount in fstab
      grep -E "^${tmp_mount}.*noexec" /etc/fstab >/dev/null 2>&1 ||
      grep -E "\s/tmp\s.*noexec" /etc/fstab >/dev/null 2>&1
   else
      # Check if /tmp is mounted with noexec currently (could be tmpfs)
```

CONFIDENTIAL

```
        findmnt -n /tmp | grep -q "noexec"
    fi
}

function tmp_noexec_configure {
    local tmp_mount=$(get_tmp_mount)

    # If /tmp is already in fstab
    if grep -q "\s/tmp\s" /etc/fstab; then
        # Check if noexec is already set
        if ! grep -E "\s/tmp\s.*noexec" /etc/fstab >/dev/null 2>&1; then
            # Add noexec to existing options
            sed -i "s|\(.*[[:space:]]\+/tmp[[:space:]]\+[^[:space:]]\+[[:space:]]\+\)\([^[:space:]]\+\)|\1\2,noexec|" /etc/fstab
        fi
    else
        # /tmp is not in fstab, we need to add it
        # Check if it's a tmpfs
        if findmnt -n /tmp | grep -q "tmpfs"; then
            echo "tmpfs /tmp tmpfs defaults,noexec,nosuid,nodev 0 0" >> /etc/fstab
        else
            echo "WARNING: /tmp does not have its own mount point. Please configure /tmp in /etc/fstab manually with the noexec option."
            return 1
        fi
    fi

    # Remount /tmp to apply the changes
    if ! $DRY_RUN; then
        mount -o remount /tmp 2>/dev/null || true
    fi
}

################################################################################
# Sudo Configuration - targetpw [RHEL-08-010383]
# Configure [OK]
# Check     [OK]
################################################################################
function sudo_targetpw_check {
    # Check if "Defaults !targetpw" exists in sudoers files
    grep -r "^[[:space:]]*Defaults[[:space:]]\+!targetpw" /etc/sudoers /etc/sudoers.d/ 2>/dev/null | grep -v "#" > /dev/null
}

function sudo_targetpw_configure {
    # Create /etc/sudoers.d directory if it doesn't exist
    if [ ! -d /etc/sudoers.d ]; then
        mkdir -p /etc/sudoers.d
        chmod 750 /etc/sudoers.d
    fi
```

CONFIDENTIAL

```
    # Check if the setting already exists (but possibly commented)
    if grep -r "Defaults[[:space:]]\+!targetpw" /etc/sudoers /etc/sudoers.d/ 2>/dev/null | grep -v "#" > /dev/null; then
        # Already set correctly, nothing to do
        return 0
    elif grep -r "Defaults[[:space:]]\+targetpw" /etc/sudoers /etc/sudoers.d/ 2>/dev/null | grep -v "#" > /dev/null; then
        # Exists but incorrectly set
        if grep -q "Defaults[[:space:]]\+targetpw" /etc/sudoers; then
            # In main sudoers file
            sed -i 's/^[[:space:]]*Defaults[[:space:]]\+targetpw/Defaults !targetpw/' /etc/sudoers
        else
            # In a sudoers.d file
            file=$(grep -r "Defaults[[:space:]]\+targetpw" /etc/sudoers.d/ 2>/dev/null | grep -v "#" | cut -d: -f1 | head -1)
            if [ -n "$file" ]; then
                sed -i 's/^[[:space:]]*Defaults[[:space:]]\+targetpw/Defaults !targetpw/' "$file"
            fi
        fi
    else
        # Doesn't exist, create it
        echo "Defaults !targetpw" > /etc/sudoers.d/hardening_targetpw
        chmod 440 /etc/sudoers.d/hardening_targetpw
    fi
}

################################################################################
# Sudo Configuration - rootpw [RHEL-08-010383]
# Configure [OK]
# Check     [OK]
################################################################################
function sudo_rootpw_check {
    # Check if "Defaults !rootpw" exists in sudoers files
    grep -r "^[[:space:]]*Defaults[[:space:]]\+!rootpw" /etc/sudoers /etc/sudoers.d/ 2>/dev/null | grep -v "#" > /dev/null
}

function sudo_rootpw_configure {
    # Create /etc/sudoers.d directory if it doesn't exist
    if [ ! -d /etc/sudoers.d ]; then
        mkdir -p /etc/sudoers.d
        chmod 750 /etc/sudoers.d
    fi

    # Check if the setting already exists (but possibly commented)
    if grep -r "Defaults[[:space:]]\+!rootpw" /etc/sudoers /etc/sudoers.d/ 2>/dev/null | grep -v "#" > /dev/null; then
        # Already set correctly, nothing to do
        return 0
    elif grep -r "Defaults[[:space:]]\+rootpw" /etc/sudoers /etc/sudoers.d/ 2>/dev/null | grep -v "#" > /dev/null; then
        # Exists but incorrectly set
        if grep -q "Defaults[[:space:]]\+rootpw" /etc/sudoers; then
```

CONFIDENTIAL

```
        # In main sudoers file
        sed -i 's/^[[:space:]]*Defaults[[:space:]]\+rootpw/Defaults !rootpw/' /etc/sudoers
    else
        # In a sudoers.d file
        file=$(grep -r "Defaults[[:space:]]\+rootpw" /etc/sudoers.d/ 2>/dev/null | grep -v "#" | cut -d: -f1 | head -1)
        if [ -n "$file" ]; then
            sed -i 's/^[[:space:]]*Defaults[[:space:]]\+rootpw/Defaults !rootpw/' "$file"
        fi
    fi
  else
    # Doesn't exist, create it
    echo "Defaults !rootpw" > /etc/sudoers.d/hardening_rootpw
    chmod 440 /etc/sudoers.d/hardening_rootpw
  fi
}

#############################################################################
# Sudo Configuration - runaspw [RHEL-08-010383]
# Configure [OK]
# Check     [OK]
#############################################################################
function sudo_runaspw_check {
  # Check if "Defaults !runaspw" exists in sudoers files
  grep -r "^[[:space:]]*Defaults[[:space:]]\+!runaspw" /etc/sudoers /etc/sudoers.d/ 2>/dev/null | grep -v "#" > /dev/null
}

function sudo_runaspw_configure {
  # Create /etc/sudoers.d directory if it doesn't exist
  if [ ! -d /etc/sudoers.d ]; then
    mkdir -p /etc/sudoers.d
    chmod 750 /etc/sudoers.d
  fi

  # Check if the setting already exists (but possibly commented)
  if grep -r "Defaults[[:space:]]\+!runaspw" /etc/sudoers /etc/sudoers.d/ 2>/dev/null | grep -v "#" > /dev/null; then
    # Already set correctly, nothing to do
    return 0
  elif grep -r "Defaults[[:space:]]\+runaspw" /etc/sudoers /etc/sudoers.d/ 2>/dev/null | grep -v "#" > /dev/null; then
    # Exists but incorrectly set
    if grep -q "Defaults[[:space:]]\+runaspw" /etc/sudoers; then
      # In main sudoers file
      sed -i 's/^[[:space:]]*Defaults[[:space:]]\+runaspw/Defaults !runaspw/' /etc/sudoers
    else
      # In a sudoers.d file
      file=$(grep -r "Defaults[[:space:]]\+runaspw" /etc/sudoers.d/ 2>/dev/null | grep -v "#" | cut -d: -f1 | head -1)
      if [ -n "$file" ]; then
        sed -i 's/^[[:space:]]*Defaults[[:space:]]\+runaspw/Defaults !runaspw/' "$file"
      fi
```

CONFIDENTIAL

```
        fi
    else
        # Doesn't exist, create it
        echo "Defaults !runaspw" > /etc/sudoers.d/hardening_runaspw
        chmod 440 /etc/sudoers.d/hardening_runaspw
    fi
}

################################################################################
# World-Writable Directory Ownership [RHEL-08-010710]
# Configure [OK]
# Check     [OK]
################################################################################
function get_system_accounts() {
    # Get list of system accounts (group IDs less than 1000)
    # Include vastdata regardless of its GID
    local system_groups=$(awk -F: '($3 < 1000) {print $1}' /etc/group)
    echo "$system_groups vastdata"
}

function world_writable_dirs_check {
    # Find world-writable directories on local filesystems
    local incorrect_dirs=0
    local system_accounts=$(get_system_accounts)

    # Get a list of world-writable directories
    while IFS= read -r dir; do
        # Skip if directory doesn't exist or is a symlink
        [ ! -d "$dir" ] || [ -L "$dir" ] && continue

        # Get directory group owner
        group_owner=$(stat -c '%G' "$dir" 2>/dev/null)

        # Check if group owner is in the list of system accounts
        if ! echo "$system_accounts" | grep -qw "$group_owner"; then
            incorrect_dirs=$((incorrect_dirs + 1))
            break
        fi
    done < <(find / -path /proc -prune -o -path /sys -prune -o -path /dev -prune -o -type d -perm -0002 -print 2>/dev/null)

    return $incorrect_dirs
}

function world_writable_dirs_configure {
    local system_accounts=$(get_system_accounts)

    # Find world-writable directories on local filesystems
    while IFS= read -r dir; do
```

CONFIDENTIAL

```
        # Skip if directory doesn't exist or is a symlink
        [ ! -d "$dir" ] || [ -L "$dir" ] && continue

        # Get directory group owner
        group_owner=$(stat -c '%G' "$dir" 2>/dev/null)

        # Check if group owner is in the list of system accounts
        if ! echo "$system_accounts" | grep -qw "$group_owner"; then
            # Change group owner to vastdata
            chgrp vastdata "$dir" 2>/dev/null
            echo "Changed group ownership of $dir to vastdata"
        fi
    done < <(find / -path /proc -prune -o -path /sys -prune -o -path /dev -prune -o -type d -perm -0002 -print 2>/dev/null)
}

################################################################################
# Chacl Auditing [RHEL-08-030570]
# Configure [OK]
# Check     [OK]
################################################################################
function chacl_audit_check {
    # The correct rule pattern we're looking for
      local    rule_pattern="^\s*-a\s+always,exit\s+-F\s+path=/usr/bin/chacl\s+-F\s+perm=x\s+-F\s+auid>=1000\s+-F\s+auid!=unset\s+-
k\s+perm_mod"

    # Check each rules file in /etc/audit/rules.d/
    for rules_file in /etc/audit/rules.d/*.rules; do
        if [ -f "$rules_file" ]; then
            # Check for the correct implementation
            if grep -E "$rule_pattern" "$rules_file" > /dev/null 2>&1; then
                # Found a correct implementation
                return 0
            fi
        fi
    done

    # No correct implementation found
    return 1
}

function chacl_audit_configure {
    # The correct rule we want to implement
    local correct_rule="-a always,exit -F path=/usr/bin/chacl -F perm=x -F auid>=1000 -F auid!=unset -k perm_mod"

    # Remove any existing chacl rules from all files first
    for file in /etc/audit/rules.d/*.rules; do
        if [ -f "$file" ]; then
            sed -i '/path=\/usr\/bin\/chacl/d' "$file"
```

CONFIDENTIAL

```
        fi
    done

    # Add the correct rule to audit.rules
    echo "$correct_rule" >> /etc/audit/rules.d/audit.rules

    # Restart the audit daemon to apply changes
    if command -v systemctl > /dev/null 2>&1; then
        systemctl restart auditd
    else
        service auditd restart
    fi
}

################################################################################
# ATM Module Blacklisting [RHEL-08-040021]
# Configure [OK]
# Check     [OK]
################################################################################
function atm_blacklist_check {
    # Check if both required lines exist in any modprobe.d config file
    local install_line_exists=0
    local blacklist_line_exists=0

    # Check each file in /etc/modprobe.d/
    for config_file in /etc/modprobe.d/*.conf; do
        if [ -f "$config_file" ]; then
            if grep -q "^install[[:space:]]\+atm[[:space:]]\+/bin/false" "$config_file"; then
                install_line_exists=1
            fi
            if grep -q "^blacklist[[:space:]]\+atm" "$config_file"; then
                blacklist_line_exists=1
            fi
        fi
    done

    # Both lines must exist for the check to pass
    [ $install_line_exists -eq 1 ] && [ $blacklist_line_exists -eq 1 ]
}

function atm_blacklist_configure {
    # The file to modify
    local blacklist_file="/etc/modprobe.d/blacklist.conf"

    # Create the file if it doesn't exist
    if [ ! -f "$blacklist_file" ]; then
        touch "$blacklist_file"
    fi
```

CONFIDENTIAL

```
    # Remove any existing ATM blacklist configurations from all files
    for config_file in /etc/modprobe.d/*.conf; do
        if [ -f "$config_file" ]; then
            sed -i '/^install[[:space:]]\+atm[[:space:]]\+\/bin\/false/d' "$config_file"
            sed -i '/^blacklist[[:space:]]\+atm/d' "$config_file"
        fi
    done

    # Add the correct configurations to blacklist.conf
    echo "install atm /bin/false" >> "$blacklist_file"
    echo "blacklist atm" >> "$blacklist_file"
}

################################################################################
# CAN Module Blacklisting [RHEL-08-040022]
# Configure [OK]
# Check     [OK]
################################################################################
function can_blacklist_check {
    # Check if both required lines exist in any modprobe.d config file
    local install_line_exists=0
    local blacklist_line_exists=0

    # Check each file in /etc/modprobe.d/
    for config_file in /etc/modprobe.d/*.conf; do
        if [ -f "$config_file" ]; then
            if grep -q "^install[[:space:]]\+can[[:space:]]\+/bin/false" "$config_file"; then
                install_line_exists=1
            fi
            if grep -q "^blacklist[[:space:]]\+can" "$config_file"; then
                blacklist_line_exists=1
            fi
        fi
    done

    # Both lines must exist for the check to pass
    [ $install_line_exists -eq 1 ] && [ $blacklist_line_exists -eq 1 ]
}

function can_blacklist_configure {
    # The file to modify
    local blacklist_file="/etc/modprobe.d/blacklist.conf"

    # Create the file if it doesn't exist
    if [ ! -f "$blacklist_file" ]; then
        touch "$blacklist_file"
    fi
```

CONFIDENTIAL

```
    # Remove any existing CAN blacklist configurations from all files
    for config_file in /etc/modprobe.d/*.conf; do
        if [ -f "$config_file" ]; then
            sed -i '/^install[[:space:]]\+can[[:space:]]\+\/bin\/false/d' "$config_file"
            sed -i '/^blacklist[[:space:]]\+can/d' "$config_file"
        fi
    done

    # Add the correct configurations to blacklist.conf
    echo "install can /bin/false" >> "$blacklist_file"
    echo "blacklist can" >> "$blacklist_file"
}

################################################################################
# SCTP Module Blacklisting [RHEL-08-040023]
# Configure [OK]
# Check     [OK]
################################################################################
function sctp_blacklist_check {
    # Check if both required lines exist in any modprobe.d config file
    local install_line_exists=0
    local blacklist_line_exists=0

    # Check each file in /etc/modprobe.d/
    for config_file in /etc/modprobe.d/*.conf; do
        if [ -f "$config_file" ]; then
            if grep -q "^install[[:space:]]\+sctp[[:space:]]\+/bin/false" "$config_file"; then
                install_line_exists=1
            fi
            if grep -q "^blacklist[[:space:]]\+sctp" "$config_file"; then
                blacklist_line_exists=1
            fi
        fi
    done

    # Both lines must exist for the check to pass
    [ $install_line_exists -eq 1 ] && [ $blacklist_line_exists -eq 1 ]
}

function sctp_blacklist_configure {
    # The file to modify
    local blacklist_file="/etc/modprobe.d/blacklist.conf"

    # Create the file if it doesn't exist
    if [ ! -f "$blacklist_file" ]; then
        touch "$blacklist_file"
    fi
```

CONFIDENTIAL

```
    # Remove any existing SCTP blacklist configurations from all files
    for config_file in /etc/modprobe.d/*.conf; do
        if [ -f "$config_file" ]; then
            sed -i '/^install[[:space:]]\+sctp[[:space:]]\+\/bin\/false/d' "$config_file"
            sed -i '/^blacklist[[:space:]]\+sctp/d' "$config_file"
        fi
    done

    # Add the correct configurations to blacklist.conf
    echo "install sctp /bin/false" >> "$blacklist_file"
    echo "blacklist sctp" >> "$blacklist_file"
}

################################################################################
# TIPC Module Blacklisting [RHEL-08-040024]
# Configure [OK]
# Check     [OK]
################################################################################
function tipc_blacklist_check {
    # Check if both required lines exist in any modprobe.d config file
    local install_line_exists=0
    local blacklist_line_exists=0

    # Check each file in /etc/modprobe.d/
    for config_file in /etc/modprobe.d/*.conf; do
        if [ -f "$config_file" ]; then
            if grep -q "^install[[:space:]]\+tipc[[:space:]]\+/bin/false" "$config_file"; then
                install_line_exists=1
            fi
            if grep -q "^blacklist[[:space:]]\+tipc" "$config_file"; then
                blacklist_line_exists=1
            fi
        fi
    done

    # Both lines must exist for the check to pass
    [ $install_line_exists -eq 1 ] && [ $blacklist_line_exists -eq 1 ]
}

function tipc_blacklist_configure {
    # The file to modify
    local blacklist_file="/etc/modprobe.d/blacklist.conf"

    # Create the file if it doesn't exist
    if [ ! -f "$blacklist_file" ]; then
        touch "$blacklist_file"
    fi
```

CONFIDENTIAL

```
    # Remove any existing TIPC blacklist configurations from all files
    for config_file in /etc/modprobe.d/*.conf; do
        if [ -f "$config_file" ]; then
            sed -i '/^install[[:space:]]\+tipc[[:space:]]\+\/bin\/false/d' "$config_file"
            sed -i '/^blacklist[[:space:]]\+tipc/d' "$config_file"
        fi
    done

    # Add the correct configurations to blacklist.conf
    echo "install tipc /bin/false" >> "$blacklist_file"
    echo "blacklist tipc" >> "$blacklist_file"
}

################################################################################
# Cramfs Kernel Module Blacklisting [RHEL-08-040025]
# Configure [OK]
# Check     [OK]
################################################################################
function cramfs_blacklist_check {
    # Check if both required lines exist in any modprobe.d config file
    local install_line_exists=0
    local blacklist_line_exists=0

    # Check each file in /etc/modprobe.d/
    for config_file in /etc/modprobe.d/*.conf; do
        if [ -f "$config_file" ]; then
            if grep -q "^install[[:space:]]\+cramfs[[:space:]]\+/bin/false" "$config_file"; then
                install_line_exists=1
            fi
            if grep -q "^blacklist[[:space:]]\+cramfs" "$config_file"; then
                blacklist_line_exists=1
            fi
        fi
    done

    # Both lines must exist for the check to pass
    [ $install_line_exists -eq 1 ] && [ $blacklist_line_exists -eq 1 ]
}

function cramfs_blacklist_configure {
    # The file to modify
    local blacklist_file="/etc/modprobe.d/blacklist.conf"

    # Create the file if it doesn't exist
    if [ ! -f "$blacklist_file" ]; then
        touch "$blacklist_file"
    fi
```

CONFIDENTIAL

```
    # Remove any existing Cramfs blacklist configurations from all files
    for config_file in /etc/modprobe.d/*.conf; do
        if [ -f "$config_file" ]; then
            sed -i '/^install[[:space:]]\+cramfs[[:space:]]\+\/bin\/false/d' "$config_file"
            sed -i '/^blacklist[[:space:]]\+cramfs/d' "$config_file"
        fi
    done

    # Add the correct configurations to blacklist.conf
    echo "install cramfs /bin/false" >> "$blacklist_file"
    echo "blacklist cramfs" >> "$blacklist_file"
}

################################################################################
# Firewire Core Module Blacklisting [RHEL-08-040026]
# Configure [OK]
# Check     [OK]
################################################################################
function firewire_blacklist_check {
    # Check if both required lines exist in any modprobe.d config file
    local install_line_exists=0
    local blacklist_line_exists=0

    # Check each file in /etc/modprobe.d/
    for config_file in /etc/modprobe.d/*.conf; do
        if [ -f "$config_file" ]; then
            if grep -q "^install[[:space:]]\+firewire-core[[:space:]]\+/bin/false" "$config_file"; then
                install_line_exists=1
            fi
            if grep -q "^blacklist[[:space:]]\+firewire-core" "$config_file"; then
                blacklist_line_exists=1
            fi
        fi
    done

    # Both lines must exist for the check to pass
    [ $install_line_exists -eq 1 ] && [ $blacklist_line_exists -eq 1 ]
}

function firewire_blacklist_configure {
    # The file to modify
    local blacklist_file="/etc/modprobe.d/blacklist.conf"

    # Create the file if it doesn't exist
    if [ ! -f "$blacklist_file" ]; then
        touch "$blacklist_file"
    fi
```

CONFIDENTIAL

```
    # Remove any existing Firewire blacklist configurations from all files
    for config_file in /etc/modprobe.d/*.conf; do
        if [ -f "$config_file" ]; then
            sed -i '/^install[[:space:]]\+firewire-core[[:space:]]\+\/bin\/false/d' "$config_file"
            sed -i '/^blacklist[[:space:]]\+firewire-core/d' "$config_file"
        fi
    done

    # Add the correct configurations to blacklist.conf
    echo "install firewire-core /bin/false" >> "$blacklist_file"
    echo "blacklist firewire-core" >> "$blacklist_file"
}

################################################################################
# USB Storage Module Blacklisting [RHEL-08-040080]
# Configure [OK]
# Check     [OK]
################################################################################
function usb_storage_blacklist_check {
    # Check if both required lines exist in any modprobe.d config file
    local install_line_exists=0
    local blacklist_line_exists=0

    # Check each file in /etc/modprobe.d/
    for config_file in /etc/modprobe.d/*.conf; do
        if [ -f "$config_file" ]; then
            if grep -q "^install[[:space:]]\+usb-storage[[:space:]]\+/bin/false" "$config_file"; then
                install_line_exists=1
            fi
            if grep -q "^blacklist[[:space:]]\+usb-storage" "$config_file"; then
                blacklist_line_exists=1
            fi
        fi
    done

    # Both lines must exist for the check to pass
    [ $install_line_exists -eq 1 ] && [ $blacklist_line_exists -eq 1 ]
}

function usb_storage_blacklist_configure {
    # The file to modify
    local blacklist_file="/etc/modprobe.d/blacklist.conf"

    # Create the file if it doesn't exist
    if [ ! -f "$blacklist_file" ]; then
        touch "$blacklist_file"
    fi
```

CONFIDENTIAL

```
    # Remove any existing USB Storage blacklist configurations from all files
    for config_file in /etc/modprobe.d/*.conf; do
        if [ -f "$config_file" ]; then
            sed -i '/^install[[:space:]]\+usb-storage[[:space:]]\+\/bin\/false/d' "$config_file"
            sed -i '/^blacklist[[:space:]]\+usb-storage/d' "$config_file"
        fi
    done

    # Add the correct configurations to blacklist.conf
    echo "install usb-storage /bin/false" >> "$blacklist_file"
    echo "blacklist usb-storage" >> "$blacklist_file"
}

################################################################################
# Bluetooth Module Blacklisting [RHEL-08-040111]
# Configure [OK]
# Check     [OK]
################################################################################
function bluetooth_blacklist_check {
    # Check if both required lines exist in any modprobe.d config file
    local install_line_exists=0
    local blacklist_line_exists=0

    # Check each file in /etc/modprobe.d/
    for config_file in /etc/modprobe.d/*.conf; do
        if [ -f "$config_file" ]; then
            if grep -q "^install[[:space:]]\+bluetooth[[:space:]]\+/bin/false" "$config_file"; then
                install_line_exists=1
            fi
            if grep -q "^blacklist[[:space:]]\+bluetooth" "$config_file"; then
                blacklist_line_exists=1
            fi
        fi
    done

    # Both lines must exist for the check to pass
    [ $install_line_exists -eq 1 ] && [ $blacklist_line_exists -eq 1 ]
}

function bluetooth_blacklist_configure {
    # The file to modify
    local blacklist_file="/etc/modprobe.d/blacklist.conf"

    # Create the file if it doesn't exist
    if [ ! -f "$blacklist_file" ]; then
        touch "$blacklist_file"
    fi
```

CONFIDENTIAL

```
    # Remove any existing Bluetooth blacklist configurations from all files
    for config_file in /etc/modprobe.d/*.conf; do
        if [ -f "$config_file" ]; then
            sed -i '/^install[[:space:]]\+bluetooth[[:space:]]\+\/bin\/false/d' "$config_file"
            sed -i '/^blacklist[[:space:]]\+bluetooth/d' "$config_file"
        fi
    done

    # Add the correct configurations to blacklist.conf
    echo "install bluetooth /bin/false" >> "$blacklist_file"
    echo "blacklist bluetooth" >> "$blacklist_file"
}

################################################################################
# IPv4 Source Routing [RHEL-08-040249]
# Configure [OK]
# Check     [OK]
################################################################################
function ipv4_source_route_check {
    # Check if the parameter is set correctly in any file
    local correct_setting="net.ipv4.conf.default.accept_source_route=0"
    local setting_exists=0

    # Check all configuration files
    for conf_file in /etc/sysctl.conf /etc/sysctl.d/*.conf /usr/lib/sysctl.d/*.conf /usr/local/lib/sysctl.d/*.conf /run/sysctl.d/*.conf; do
        if [ -f "$conf_file" ]; then
            # Check if the correct setting exists
            if grep -q "^$correct_setting$" "$conf_file"; then
                setting_exists=1
                break
            fi
        fi
    done

    # Check if conflicting settings exist
    local conflicting_exists=0
    for conf_file in /etc/sysctl.conf /etc/sysctl.d/*.conf /usr/lib/sysctl.d/*.conf /usr/local/lib/sysctl.d/*.conf /run/sysctl.d/*.conf; do
        if [ -f "$conf_file" ]; then
            # Check if any conflicting setting exists
            if grep -q "^net.ipv4.conf.default.accept_source_route=[^0]" "$conf_file"; then
                conflicting_exists=1
                break
            fi
        fi
    done

    # Check current runtime value
```

CONFIDENTIAL

```
    local current_value=$(sysctl -n net.ipv4.conf.default.accept_source_route 2>/dev/null || echo "error")

    # Return success if setting exists with correct value, no conflicts exist, and runtime value is correct
    [ $setting_exists -eq 1 ] && [ $conflicting_exists -eq 0 ] && [ "$current_value" = "0" ]
}

function ipv4_source_route_configure {
    local correct_setting="net.ipv4.conf.default.accept_source_route=0"
    local config_file="/etc/sysctl.d/99-sysctl.conf"

    mkdir -p /etc/sysctl.d/

    # Remove any conflicting settings from all config files
    for conf_file in /etc/sysctl.conf /etc/sysctl.d/*.conf /usr/lib/sysctl.d/*.conf /usr/local/lib/sysctl.d/*.conf /run/sysctl.d/*.conf; do
        if [ -f "$conf_file" ]; then
            # Remove any setting (correct or incorrect)
            sed -i '/^net.ipv4.conf.default.accept_source_route=/d' "$conf_file"
        fi
    done

    # Add the correct setting to our config file
    echo "$correct_setting" >> "$config_file"

    # Apply the changes
    sysctl --system
}

################################################################################
# IPv6 Source Routing [RHEL-08-040250]
# Configure [OK]
# Check     [OK]
################################################################################
function ipv6_source_route_check {
    # Check if the parameter is set correctly in any file
    local correct_setting="net.ipv6.conf.default.accept_source_route=0"
    local setting_exists=0

    # Check all configuration files
    for conf_file in /etc/sysctl.conf /etc/sysctl.d/*.conf /usr/lib/sysctl.d/*.conf /usr/local/lib/sysctl.d/*.conf /run/sysctl.d/*.conf; do
        if [ -f "$conf_file" ]; then
            # Check if the correct setting exists
            if grep -q "^$correct_setting$" "$conf_file"; then
                setting_exists=1
                break
            fi
        fi
    done
```

CONFIDENTIAL

```
    # Check if conflicting settings exist
    local conflicting_exists=0
    for conf_file in /etc/sysctl.conf /etc/sysctl.d/*.conf /usr/lib/sysctl.d/*.conf /usr/local/lib/sysctl.d/*.conf /run/sysctl.d/*.conf; do
        if [ -f "$conf_file" ]; then
            # Check if any conflicting setting exists
            if grep -q "^net.ipv6.conf.default.accept_source_route=[^0]" "$conf_file"; then
                conflicting_exists=1
                break
            fi
        fi
    done

    # Check if IPv6 is enabled
    if [ -f /proc/sys/net/ipv6/conf/default/accept_source_route ]; then
        # Check current runtime value
        local current_value=$(sysctl -n net.ipv6.conf.default.accept_source_route 2>/dev/null || echo "error")

        # Return success if setting exists with correct value, no conflicts exist, and runtime value is correct
        [ $setting_exists -eq 1 ] && [ $conflicting_exists -eq 0 ] && [ "$current_value" = "0" ]
    else
        # IPv6 is disabled, so we consider the check as passed
        [ $setting_exists -eq 1 ] && [ $conflicting_exists -eq 0 ]
    fi
}

function ipv6_source_route_configure {
    local correct_setting="net.ipv6.conf.default.accept_source_route=0"
    local config_file="/etc/sysctl.d/99-sysctl.conf"

    mkdir -p /etc/sysctl.d/

    # Remove any conflicting settings from all config files
    for conf_file in /etc/sysctl.conf /etc/sysctl.d/*.conf /usr/lib/sysctl.d/*.conf /usr/local/lib/sysctl.d/*.conf /run/sysctl.d/*.conf; do
        if [ -f "$conf_file" ]; then
            # Remove any setting (correct or incorrect)
            sed -i '/^net.ipv6.conf.default.accept_source_route=/d' "$conf_file"
        fi
    done

    # Add the correct setting to our config file
    echo "$correct_setting" >> "$config_file"

    # Apply the changes if IPv6 is enabled
    if [ -f /proc/sys/net/ipv6/conf/default/accept_source_route ]; then
        sysctl --system
    fi
}
```

CONFIDENTIAL

```
################################################################################
# IPv4 ICMP Redirects [RHEL-08-040279]
# Configure [OK]
# Check     [OK]
################################################################################
function ipv4_icmp_redirects_check {
    # Check if the parameter is set correctly in any file
    local correct_setting="net.ipv4.conf.all.accept_redirects=0"
    local setting_exists=0

    # Check all configuration files
    for conf_file in /etc/sysctl.conf /etc/sysctl.d/*.conf /usr/lib/sysctl.d/*.conf /usr/local/lib/sysctl.d/*.conf /run/sysctl.d/*.conf; do
        if [ -f "$conf_file" ]; then
            # Check if the correct setting exists
            if grep -q "^$correct_setting$" "$conf_file"; then
                setting_exists=1
                break
            fi
        fi
    done

    # Check if conflicting settings exist
    local conflicting_exists=0
    for conf_file in /etc/sysctl.conf /etc/sysctl.d/*.conf /usr/lib/sysctl.d/*.conf /usr/local/lib/sysctl.d/*.conf /run/sysctl.d/*.conf; do
        if [ -f "$conf_file" ]; then
            # Check if any conflicting setting exists
            if grep -q "^net.ipv4.conf.all.accept_redirects=[^0]" "$conf_file"; then
                conflicting_exists=1
                break
            fi
        fi
    done

    # Check current runtime value
    local current_value=$(sysctl -n net.ipv4.conf.all.accept_redirects 2>/dev/null || echo "error")

    # Return success if setting exists with correct value, no conflicts exist, and runtime value is correct
    [ $setting_exists -eq 1 ] && [ $conflicting_exists -eq 0 ] && [ "$current_value" = "0" ]
}

function ipv4_icmp_redirects_configure {
    local correct_setting="net.ipv4.conf.all.accept_redirects=0"
    local config_file="/etc/sysctl.d/99-sysctl.conf"

    mkdir -p /etc/sysctl.d/

    # Remove any conflicting settings from all config files
    for conf_file in /etc/sysctl.conf /etc/sysctl.d/*.conf /usr/lib/sysctl.d/*.conf /usr/local/lib/sysctl.d/*.conf /run/sysctl.d/*.conf; do
```

CONFIDENTIAL

```
    if [ -f "$conf_file" ]; then
        # Remove any setting (correct or incorrect)
        sed -i '/^net.ipv4.conf.all.accept_redirects=/d' "$conf_file"
    fi
done

# Add the correct setting to our config file
echo "$correct_setting" >> "$config_file"

# Apply the changes
sysctl --system
}

################################################################################
# User Namespace Disabling [RHEL-08-040284]
# Configure [OK]
# Check     [OK]
################################################################################
function user_namespace_check {
    # Check if the parameter is set correctly in any file
    local correct_setting="user.max_user_namespaces=0"
    local setting_exists=0

    # Check all configuration files
    for conf_file in /etc/sysctl.conf /etc/sysctl.d/*.conf /usr/lib/sysctl.d/*.conf /usr/local/lib/sysctl.d/*.conf /run/sysctl.d/*.conf; do
        if [ -f "$conf_file" ]; then
            # Check if the correct setting exists
            if grep -q "^$correct_setting$" "$conf_file"; then
                setting_exists=1
                break
            fi
        fi
    done

    # Check if conflicting settings exist
    local conflicting_exists=0
    for conf_file in /etc/sysctl.conf /etc/sysctl.d/*.conf /usr/lib/sysctl.d/*.conf /usr/local/lib/sysctl.d/*.conf /run/sysctl.d/*.conf; do
        if [ -f "$conf_file" ]; then
            # Check if any conflicting setting exists
            if grep -q "^user.max_user_namespaces=[^0]" "$conf_file"; then
                conflicting_exists=1
                break
            fi
        fi
    done

    # Check current runtime value
    local current_value=$(sysctl -n user.max_user_namespaces 2>/dev/null || echo "error")
```

CONFIDENTIAL

```
    # Return success if setting exists with correct value, no conflicts exist, and runtime value is correct
    [ $setting_exists -eq 1 ] && [ $conflicting_exists -eq 0 ] && [ "$current_value" = "0" ]
}

function user_namespace_configure {
    local correct_setting="user.max_user_namespaces=0"
    local config_file="/etc/sysctl.d/99-sysctl.conf"

    mkdir -p /etc/sysctl.d/

    # Remove any conflicting settings from all config files
    for conf_file in /etc/sysctl.conf /etc/sysctl.d/*.conf /usr/lib/sysctl.d/*.conf /usr/local/lib/sysctl.d/*.conf /run/sysctl.d/*.conf; do
        if [ -f "$conf_file" ]; then
            # Remove any setting (correct or incorrect)
            sed -i '/^user.max_user_namespaces=/d' "$conf_file"
        fi
    done

    # Add the correct setting to our config file
    echo "$correct_setting" >> "$config_file"

    # Apply the changes
    sysctl --system
}

################################################################################
# Firewalld [RHEL-08-040030, RHEL-08-040101]
# Configure [OK]
# Check     [NO]
################################################################################
function firewalld_get_primary_interface() {
    echo ${FIREWALLD_PRIMARY_INTERFACE:-$(ip route | grep default | awk '{print $5}' | head -n1)}
}

function firewalld_excluded_interfaces {
    declare -a EXCLUDED_INTERFACES=("lo" "docker0" "${PRIMARY_INTERFACE}")
    echo ${EXCLUDED_INTERFACES}
}

function firewalld_create_directories() {
    sudo mkdir -p /etc/firewalld/zones
    sudo mkdir -p /etc/firewalld/services
}

function firewalld_set_conf() {
    sudo sed -i 's/^DefaultZone=.*/DefaultZone=drop/' /etc/firewalld/firewalld.conf
    sudo sed -i 's/^AllowZoneDrifting=.*/AllowZoneDrifting=no/' /etc/firewalld/firewalld.conf
```

CONFIDENTIAL

```
}

function firewalld_configure_zone_vms() {
    INTERFACE=$1
    firewall-cmd --permanent --new-zone=VMS >/dev/null 2>&1
    firewall-cmd --permanent --zone=VMS --set-target=DROP >/dev/null 2>&1
    firewall-cmd --permanent --zone=VMS --add-interface=${INTERFACE} >/dev/null 2>&1
    firewall-cmd --permanent --zone=VMS --add-service={ssh,vast-vms} >/dev/null 2>&1
}

function firewalld_configure_zone_docker() {
    firewall-cmd --permanent --new-zone=docker >/dev/null 2>&1
    firewall-cmd --permanent --zone=docker --set-target=DROP >/dev/null 2>&1
    firewall-cmd --permanent --zone=docker --add-interface=docker0 >/dev/null 2>&1
    firewall-cmd --permanent --zone=docker --add-port=1-65535/tcp >/dev/null 2>&1
    firewall-cmd --permanent --zone=docker --add-port=1-65535/udp >/dev/null 2>&1
    firewall-cmd --permanent --zone=docker --add-masquerade >/dev/null 2>&1
}

function firewalld_configure_zone() {
    INTERFACE=$1
    ZONE_NAME=${INTERFACE//./_}
    firewall-cmd --permanent --new-zone=${ZONE_NAME} >/dev/null 2>&1
    firewall-cmd --permanent --zone=${ZONE_NAME} --set-target=DROP >/dev/null 2>&1
    firewall-cmd --permanent --zone=${ZONE_NAME} --add-interface=${INTERFACE} >/dev/null 2>&1
    firewall-cmd --permanent --zone=${ZONE_NAME} --add-port=1-65535/tcp >/dev/null 2>&1
    firewall-cmd --permanent --zone=${ZONE_NAME} --add-port=1-65535/udp >/dev/null 2>&1
}

function firewalld_configure_services(){
cat << EOF > /etc/firewalld/services/vast-vms.xml
<?xml version="1.0" encoding="utf-8"?>
<service>
<short>VAST Management System (HTTP)</short>
<description>VAST Management System provides a web interface for managing all aspect of VAST OS.</description>
<port protocol="tcp" port="443"/>
<port protocol="tcp" port="80"/>
</service>
EOF
}

function firewalld_is_excluded() {
    local INTERFACE=$1
    local EXCLUDED_INTERFACES=$(firewalld_excluded_interfaces)
    for EXCLUDED in "${EXCLUDED_INTERFACES[@]}"; do
        if [[ "$INTERFACE" == "$EXCLUDED" ]]; then
            return 0
        fi
```

CONFIDENTIAL

```
    done
    return 1
}

function firewalld_configure {
    systemctl enable firewalld --now

    firewalld_create_directories
    firewalld_set_conf
    firewalld_configure_services

    firewall-cmd --reload >/dev/null 2>&1

    firewalld_configure_zone_docker
    firewalld_configure_zone_vms $(firewalld_get_primary_interface)

    for INTERFACE in $(ip -o link show | awk -F': ' '{print $2}' | cut -d'@' -f1)
    do
        firewalld_is_excluded ${INTERFACE} && continue
        firewalld_configure_zone $INTERFACE
    done

    firewall-cmd --reload >/dev/null 2>&1
}

################################################################################
# Main Loop
################################################################################

function run_check() {
    local check_name="$1"
    local check_function="$2"
    local configure_function="$3"
    local control_id="$4"

    echo -e -n "Checking ${check_name} [${control_id}]..."
    if ${check_function}; then
        echo -e -n "[ "$(print_green  OK); echo " ]"
    else
        if $DRY_RUN; then
            echo -e -n "[ "$(print_yellow  "NOT COMPLIANT"); echo " ]"
        else
            echo -e -n "[ "$(print_red  CONFIGURING); echo " ]"
            ${configure_function}
        fi
    fi
}
```

CONFIDENTIAL

```
function hardening_level_one {
    echo -e "\n=== Level 1 Hardening - Basic Security Settings ==="

    run_check "password policy min length" password_policy_min_length_check password_policy_min_length_configure "APSC-DV-001680, APSC-DV-001730"
    run_check "password policy uppercase requirement" password_policy_ucredit_check password_policy_ucredit_configure "APSC-DV-001680, APSC-DV-001730"
    run_check "password policy lowercase requirement" password_policy_lcredit_check password_policy_lcredit_configure "APSC-DV-001680, APSC-DV-001730"
    run_check "password policy digit requirement" password_policy_dcredit_check password_policy_dcredit_configure "APSC-DV-001680, APSC-DV-001730"
    run_check "password policy special char requirement" password_policy_ocredit_check password_policy_ocredit_configure "APSC-DV-001680, APSC-DV-001730"
    run_check "password policy max repeat" password_policy_maxrepeat_check password_policy_maxrepeat_configure "APSC-DV-001680, APSC-DV-001730"
    run_check "inactivity lockout" inactivity_lockout_check inactivity_lockout_configure "APSC-DV-001670"
    run_check "password minimum lifetime" password_min_days_check password_min_days_configure "APSC-DV-001760"
    run_check "password maximum lifetime" password_max_days_check password_max_days_configure "APSC-DV-001770"
    run_check "password expiration warning" password_warn_age_check password_warn_age_configure "APSC-DV-001770"
}

# The following steps are to be utilized ONLY after the customer understands the risks of implementing performance impacting configuration changes to the product.
# These changes will bring the product into compliance with the identified control requirements but will impact the product's performance.
function hardening_level_two {
    echo -e "\n=== Level 2 Hardening - Advanced Security Settings ==="

    echo -e -n "Configuring firewalld [RHEL-08-040030, RHEL-08-040101]..."
    if $DRY_RUN; then
        echo -e -n "[ "$(print_yellow  "WOULD CONFIGURE"); echo " ]"
    else
        firewalld_configure
        echo -e -n "[ "$(print_green  OK); echo " ]"
    fi

    run_check "page poisoning (memory protection)" page_poisoning_check page_poisoning_configure "RHEL-08-010421"
    run_check "SLUB/SLAB poisoning (memory protection)" slub_poisoning_check slub_poisoning_configure "RHEL-08-010423"
    run_check "page-table isolation (Meltdown mitigation)" pti_check pti_configure "RHEL-08-040004"
    run_check "disable remote root login" disable_root_login_check disable_root_login_configure "RHEL-08-010550"
    run_check "nosuid for user home directories" nosuid_home_check nosuid_home_configure "RHEL-08-010570"
    run_check "permanent account lockout (unlock time)" faillock_unlock_time_check faillock_unlock_time_configure "RHEL-08-020015"
    run_check "default account permissions in bashrc" umask_bashrc_check umask_bashrc_configure "RHEL-08-020353"
    run_check "default account permissions in csh.cshrc" umask_cshrc_check umask_cshrc_configure "RHEL-08-020353"
    run_check "default account permissions in profile" umask_profile_check umask_profile_configure "RHEL-08-020353"
    run_check "temporary folder noexec" tmp_noexec_check tmp_noexec_configure "RHEL-08-040125"
    run_check "sudo configuration - targetpw" sudo_targetpw_check sudo_targetpw_configure "RHEL-08-010383"
    run_check "sudo configuration - rootpw" sudo_rootpw_check sudo_rootpw_configure "RHEL-08-010383"
    run_check "sudo configuration - runaspw" sudo_runaspw_check sudo_runaspw_configure "RHEL-08-010383"
```

CONFIDENTIAL

```
    run_check "world-writable directory ownership" world_writable_dirs_check world_writable_dirs_configure "RHEL-08-010710"
    run_check "chacl command auditing" chacl_audit_check chacl_audit_configure "RHEL-08-030570"
    run_check "ATM module blacklisting" atm_blacklist_check atm_blacklist_configure "RHEL-08-040021"
    run_check "CAN module blacklisting" can_blacklist_check can_blacklist_configure "RHEL-08-040022"
    run_check "SCTP module blacklisting" sctp_blacklist_check sctp_blacklist_configure "RHEL-08-040023"
    run_check "TIPC module blacklisting" tipc_blacklist_check tipc_blacklist_configure "RHEL-08-040024"
    run_check "Cramfs module blacklisting" cramfs_blacklist_check cramfs_blacklist_configure "RHEL-08-040025"
    run_check "Firewire-core module blacklisting" firewire_blacklist_check firewire_blacklist_configure "RHEL-08-040026"
    run_check "USB Storage module blacklisting" usb_storage_blacklist_check usb_storage_blacklist_configure "RHEL-08-040080"
    run_check "Bluetooth module blacklisting" bluetooth_blacklist_check bluetooth_blacklist_configure "RHEL-08-040111"
    run_check "IPv4 source routing" ipv4_source_route_check ipv4_source_route_configure "RHEL-08-040249"
    run_check "IPv6 source routing" ipv6_source_route_check ipv6_source_route_configure "RHEL-08-040250"
    run_check "IPv4 ICMP redirects" ipv4_icmp_redirects_check ipv4_icmp_redirects_configure "RHEL-08-040279"
    run_check "user namespaces" user_namespace_check user_namespace_configure "RHEL-08-040284"
}

function main {
    parse_args "$@"

    if $DRY_RUN; then
        echo "Running in dry-run mode. No changes will be made."
    fi

    hardening_level_one

    if [ "$HARDENING_LEVEL" -ge 2 ]; then
        hardening_level_two
    fi

    echo -e "\nHardening process completed."
}

main "$@"
```

10.0 Summary

The steps above detail the configuration steps needed to be taken by customers to bring the VAST AI OS into compliance with all applicable DISA STIGs and SRGs.

CONFIDENTIAL