3 Network

This section provides guidance on for securing the network configuration of the system

3.1 Configure Network Devices

To reduce the attack surface of a system, unused devices should be disabled.

Note: This should not be considered a comprehensive list, you may wish to consider additions to those listed here for your environment.

3.1.1 Ensure IPv6 status is identified (Manual)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

Internet Protocol Version 6 (IPv6) is the most recent version of Internet Protocol (IP). It's designed to supply IP addressing and additional security to support the predicted growth of connected devices. IPv6 is based on 128-bit addressing and can support 340 undecillion, which is 340,282,366,920,938,463,463,374,607,431,768,211,456 unique addresses.

Features of IPv6

- Hierarchical addressing and routing infrastructure
- Statefull and Stateless configuration
- Support for quality of service (QoS)
- An ideal protocol for neighboring node interaction

Rationale:

IETF RFC 4038 recommends that applications are built with an assumption of dual stack. It is recommended that IPv6 be enabled and configured in accordance with Benchmark recommendations.

- IF - dual stack and IPv6 are not used in your environment, IPv6 may be disabled to reduce the attack surface of the system, and recommendations pertaining to IPv6 can be skipped.

Note: It is recommended that IPv6 be enabled and configured unless this is against local site policy

Impact:

IETF RFC 4038 recommends that applications are built with an assumption of dual stack.

When enabled, IPv6 will require additional configuration to reduce risk to the system.

Audit:

Run the following script to identify if IPv6 is enabled on the system:

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

{
   l_output=""
   ! grep -Pqs -- '^\h*0\b' /sys/module/ipv6/parameters/disable &&
l_output="- IPv6 is not enabled"
   if sysctl net.ipv6.conf.all.disable_ipv6 | grep -Pqs --
"^\h*net\.ipv6\.conf\.all\.disable_ipv6\h*=\h*1\b" && \
      sysctl net.ipv6.conf.default.disable_ipv6 | grep -Pqs --
"^\h*net\.ipv6\.conf\.default\.disable_ipv6\h*=\h*1\b"; then
      l_output="- IPv6 is not enabled"
   fi
   [ -z "$l_output" ] && l_output="- IPv6 is enabled"
   echo -e "\n$l_output\n"
}
```

Remediation:

Enable or disable IPv6 in accordance with system requirements and local site policy

Default Value:

IPv6 is enabled

References:

1. NIST SP 800-53 Rev. 5: CM-7

Additional Information:

Having more addresses has grown in importance with the expansion of smart devices and connectivity. IPv6 provides more than enough globally unique IP addresses for every networked device currently on the planet, helping ensure providers can keep pace with the expected proliferation of IP-based devices.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.8 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.		●	●
v7	9.2 Ensure Only Approved Ports, Protocols and Services Are Running Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1557, T1557.000, T1595, T1595.001, T1595.002	TA0008	M1042

3.1.2 Ensure wireless interfaces are disabled (Automated)

Profile Applicability:

- Level 1 - Server

Description:

Wireless networking is used when wired networks are unavailable.

Rationale:

-IF- wireless is not to be used, wireless devices can be disabled to reduce the potential attack surface.

Impact:

Many if not all laptop workstations and some desktop workstations will connect via wireless requiring these interfaces be enabled.

Audit:

Run the following script to verify no wireless interfaces are active on the system:

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

{
   l_output="" l_output2=""
   module_chk()
   {
      # Check how module will be loaded
      l_loadable="$(modprobe -n -v "$l_mname")"
      if grep -Pq -- '^\h*install \/bin\/(true|false)' <<< "$l_loadable";
then
         l_output="$l_output\n - module: \"$l_mname\" is not loadable:
\"$l_loadable\""
      else
         l_output2="$l_output2\n - module: \"$l_mname\" is loadable:
\"$l_loadable\""
      fi
      # Check is the module currently loaded
      if ! lsmod | grep "$l_mname" > /dev/null 2>&1; then
         l_output="$l_output\n - module: \"$l_mname\" is not loaded"
      else
         l_output2="$l_output2\n - module: \"$l_mname\" is loaded"
      fi
      # Check if the module is deny listed
      if modprobe --showconfig | grep -Pq -- "^\h*blacklist\h+$l_mname\b";
then
         l_output="$l_output\n - module: \"$l_mname\" is deny listed in:
\"$(grep -Pl -- "^\h*blacklist\h+$l_mname\b" /etc/modprobe.d/*)\""
      else
         l_output2="$l_output2\n - module: \"$l_mname\" is not deny listed"
      fi
   }
   if [ -n "$(find /sys/class/net/*/ -type d -name wireless)" ]; then
      l_dname=$(for driverdir in $(find /sys/class/net/*/ -type d -name
wireless | xargs -0 dirname); do basename "$(readlink -f
"$driverdir"/device/driver/module)";done | sort -u)
      for l_mname in $l_dname; do
         module_chk
      done
   fi
   # Report results. If no failures output in l_output2, we pass
   if [ -z "$l_output2" ]; then
      echo -e "\n- Audit Result:\n  ** PASS **"
      if [ -z "$l_output" ]; then
         echo -e "\n - System has no wireless NICs installed"
      else
         echo -e "\n$l_output\n"
      fi
   else
      echo -e "\n- Audit Result:\n  ** FAIL **\n - Reason(s) for audit
failure:\n$l_output2\n"
      [ -n "$l_output" ] && echo -e "\n- Correctly set:\n$l_output\n"
   fi
}
```

Remediation:

Run the following script to disable any wireless interfaces:

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

{
   module_fix()
   {
      if ! modprobe -n -v "$l_mname" | grep -P -- '^\h*install
\/bin\/(true|false)'; then
         echo -e " - setting module: \"$l_mname\" to be un-loadable"
         echo -e "install $l_mname /bin/false" >>
/etc/modprobe.d/"$l_mname".conf
      fi
      if lsmod | grep "$l_mname" > /dev/null 2>&1; then
         echo -e " - unloading module \"$l_mname\""
         modprobe -r "$l_mname"
      fi
      if ! grep -Pq -- "^\h*blacklist\h+$l_mname\b" /etc/modprobe.d/*; then
         echo -e " - deny listing \"$l_mname\""
         echo -e "blacklist $l_mname" >> /etc/modprobe.d/"$l_mname".conf
      fi
   }
   if [ -n "$(find /sys/class/net/*/ -type d -name wireless)" ]; then
      l_dname=$(for driverdir in $(find /sys/class/net/*/ -type d -name
wireless | xargs -0 dirname); do basename "$(readlink -f
"$driverdir"/device/driver/module)";done | sort -u)
      for l_mname in $l_dname; do
         module_fix
      done
   fi
}
```

References:

1. NIST SP 800-53 Rev. 5: CM-7

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.8 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.		●	●
v7	15.4 Disable Wireless Access on Devices if Not Required Disable wireless access on devices that do not have a business purpose for wireless access.			●
v7	15.5 Limit Wireless Access on Client Devices Configure wireless access on client machines that do have an essential wireless business purpose, to allow access only to authorized wireless networks and to restrict access to other wireless networks.			●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1011, T1011.000, T1595, T1595.001, T1595.002	TA0010	M1028

3.1.3 Ensure bluetooth services are not in use (Automated)

Profile Applicability:

- Level 1 - Server
- Level 2 - Workstation

Description:

Bluetooth is a short-range wireless technology standard that is used for exchanging data between devices over short distances. It employs UHF radio waves in the ISM bands, from 2.402 GHz to 2.48 GHz. It is mainly used as an alternative to wire connections.

Rationale:

An attacker may be able to find a way to access or corrupt your data. One example of this type of activity is `bluesnarfing`, which refers to attackers using a Bluetooth connection to steal information off of your Bluetooth device. Also, viruses or other malicious code can take advantage of Bluetooth technology to infect other devices. If you are infected, your data may be corrupted, compromised, stolen, or lost.

Impact:

Many personal electronic devices (PEDs) use Bluetooth technology. For example, you may be able to operate your computer with a wireless keyboard. Disabling Bluetooth will prevent these devices from connecting to the system.

There may be packages that are dependent on the `bluez` package. If the `bluez` package is removed, these dependent packages will be removed as well. Before removing the `bluez` package, review any dependent packages to determine if they are required on the system.

-IF- a dependent package is required: stop and mask `bluetooth.service` leaving the `bluez` package installed.

Audit:

Run the following command to verify the `bluez` package is not installed:

```
# dpkg-query -s bluez &>/dev/null && echo "bluez is installed"
```

Nothing should be returned.
- OR -
- IF - the `bluez` package is required as a dependency:
Run the following command to verify `bluetooth.service` is not enabled:

```
# systemctl is-enabled bluetooth.service 2>/dev/null | grep 'enabled'
```

Nothing should be returned.
Run the following command to verify `bluetooth.service` is not active:

```
# systemctl is-active bluetooth.service 2>/dev/null | grep '^active'
```

Nothing should be returned.
Note: If the package is required for a dependency

- Ensure the dependent package is approved by local site policy
- Ensure stopping and masking the service and/or socket meets local site policy

Remediation:

Run the following commands to stop `bluetooth.service`, and remove the `bluez` package:

```
# systemctl stop bluetooth.service
# apt purge bluez
```

- OR -
- IF - the `bluez` package is required as a dependency:
Run the following commands to stop and mask `bluetooth.service`:

```
# systemctl stop bluetooth.service
# systemctl mask bluetooth.service
```

Note: A reboot may be required

References:

1. https://www.cisa.gov/tips/st05-015
2. NIST SP 800-53 Rev. 5: CM-7

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.8 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.		●	●
v7	9.2 Ensure Only Approved Ports, Protocols and Services Are Running Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1011, T1011.001	TA0010	M1042

3.2 Configure Network Kernel Modules

The Linux kernel modules support several network protocols that are not commonly used. If these protocols are not needed, it is recommended that they be disabled in the kernel.

Note: This should not be considered a comprehensive list of uncommon network protocols, you may wish to consider additions to those listed here for your environment.

3.2.1 Ensure dccp kernel module is not available (Automated)

Profile Applicability:

- Level 2 - Server
- Level 2 - Workstation

Description:

The Datagram Congestion Control Protocol (DCCP) is a transport layer protocol that supports streaming media and telephony. DCCP provides a way to gain access to congestion control, without having to do it at the application layer, but does not provide in-sequence delivery.

Rationale:

- IF - the protocol is not required, it is recommended that the drivers not be installed to reduce the potential attack surface.

Audit:

Run the following script to verify:
- IF - the `dccp` kernel module is available in ANY installed kernel, verify:

- An entry including `/bin/true` or `/bin/false` exists in a file within the `/etc/modprobe.d/` directory
- The module is deny listed in a file within the `/etc/modprobe.d/` directory
- The module is not loaded in the running kernel

- IF - the `dccp` kernel module is not available on the system, or pre-compiled into the kernel, no additional configuration is necessary

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

{
   a_output=() a_output2=() a_output3=() l_dl="" l_mod_name="dccp"
l_mod_type="net"
   l_mod_path="$(readlink -f /lib/modules/**/kernel/$l_mod_type | sort -u)"
   f_module_chk()
   {
      l_dl="y" a_showconfig=()
      while IFS= read -r l_showconfig; do
         a_showconfig+=("$l_showconfig")
      done < <(modprobe --showconfig | grep -P --
'\b(install|blacklist)\h+'"${l_mod_chk_name//-/_}"'\b')
      if ! lsmod | grep "$l_mod_chk_name" &> /dev/null; then
         a_output+=("  - kernel module: \"$l_mod_name\" is not loaded")
      else
         a_output2+=("  - kernel module: \"$l_mod_name\" is loaded")
      fi
      if grep -Pq -- '\binstall\h+'"${l_mod_chk_name//-
/_}"'\h+(\/usr)?\/bin\/(true|false)\b' <<< "${a_showconfig[*]}"; then
         a_output+=("  - kernel module: \"$l_mod_name\" is not loadable")
      else
         a_output2+=("  - kernel module: \"$l_mod_name\" is loadable")
      fi
      if grep -Pq -- '\bblacklist\h+'"${l_mod_chk_name//-/_}"'\b' <<<
"${a_showconfig[*]}"; then
         a_output+=("  - kernel module: \"$l_mod_name\" is deny listed")
      else
         a_output2+=("  - kernel module: \"$l_mod_name\" is not deny listed")
      fi
   }
   for l_mod_base_directory in $l_mod_path; do
      if [ -d "$l_mod_base_directory/${l_mod_name/-/\/}" ] && [ -n "$(ls -A
"$l_mod_base_directory/${l_mod_name/-/\/}")" ]; then
         a_output3+=("  - \"$l_mod_base_directory\"")
         l_mod_chk_name="$l_mod_name"
         [[ "$l_mod_name" =~ overlay ]] && l_mod_chk_name="${l_mod_name::-2}"
         [ "$l_dl" != "y" ] && f_module_chk
      else
         a_output+=(" - kernel module: \"$l_mod_name\" doesn't exist in
\"$l_mod_base_directory\"")
      fi
   done
   [ "${#a_output3[@]}" -gt 0 ] && printf '%s\n' "" " -- INFO --" " - module:
\"$l_mod_name\" exists in:" "${a_output3[@]}"
   if [ "${#a_output2[@]}" -le 0 ]; then
      printf '%s\n' "" "- Audit Result:" "  ** PASS **" "${a_output[@]}"
   else
      printf '%s\n' "" "- Audit Result:" "  ** FAIL **" " - Reason(s) for
audit failure:" "${a_output2[@]}"
      [ "${#a_output[@]}" -gt 0 ] && printf '%s\n' "- Correctly set:"
"${a_output[@]}"
   fi
}
```

Remediation:

Run the following script to unload and disable the `dccp` module:
- IF - the `dccp` kernel module is available in ANY installed kernel:

- Create a file ending in `.conf` with `install dccp /bin/false` in the `/etc/modprobe.d/` directory
- Create a file ending in `.conf` with `blacklist dccp` in the `/etc/modprobe.d/` directory
- Run `modprobe -r dccp 2>/dev/null; rmmod dccp 2>/dev/null` to remove `dccp` from the kernel

- IF - the `dccp` kernel module is not available on the system, or pre-compiled into the kernel, no remediation is necessary

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

{
   a_output2=() a_output3=() l_dl="" l_mod_name="dccp" l_mod_type="net"
   l_mod_path="$(readlink -f /lib/modules/**/kernel/$l_mod_type | sort -u)"
   f_module_fix()
   {
      l_dl="y" a_showconfig=()
      while IFS= read -r l_showconfig; do
         a_showconfig+=("$l_showconfig")
      done < <(modprobe --showconfig | grep -P --
'\b(install|blacklist)\h+'"${l_mod_chk_name//-/_}"'\b')
      if  lsmod | grep "$l_mod_chk_name" &> /dev/null; then
         a_output2+=(" - unloading kernel module: \"$l_mod_name\"")
         modprobe -r "$l_mod_chk_name" 2>/dev/null; rmmod "$l_mod_name"
2>/dev/null
      fi
      if ! grep -Pq -- '\binstall\h+'"${l_mod_chk_name//-
/_}"'\h+(\/usr)?\/bin\/(true|false)\b' <<< "${a_showconfig[*]}"; then
         a_output2+=(" - setting kernel  module: \"$l_mod_name\" to
\"$(readlink -f /bin/false)\"")
         printf '%s\n' "install $l_mod_chk_name $(readlink -f /bin/false)" >>
/etc/modprobe.d/"$l_mod_name".conf
      fi
      if ! grep -Pq -- '\bblacklist\h+'"${l_mod_chk_name//-/_}"'\b' <<<
"${a_showconfig[*]}"; then
         a_output2+=(" - denylisting kernel module: \"$l_mod_name\"")
         printf '%s\n' "blacklist $l_mod_chk_name" >>
/etc/modprobe.d/"$l_mod_name".conf
      fi
   }
   for l_mod_base_directory in $l_mod_path; do # Check if the module exists
on the system
      if [ -d "$l_mod_base_directory/${l_mod_name/-/\/}" ] && [ -n "$(ls -A
"$l_mod_base_directory/${l_mod_name/-/\/}")" ]; then
         a_output3+=("  - \"$l_mod_base_directory\"")
         l_mod_chk_name="$l_mod_name"
         [[ "$l_mod_name" =~ overlay ]] && l_mod_chk_name="${l_mod_name::-2}"
         [ "$l_dl" != "y" ] && f_module_fix
      else
         printf '%s\n' " - kernel module: \"$l_mod_name\" doesn't exist in
\"$l_mod_base_directory\""
      fi
   done
   [ "${#a_output3[@]}" -gt 0 ] && printf '%s\n' "" " -- INFO --" " - module:
\"$l_mod_name\" exists in:" "${a_output3[@]}"
   [ "${#a_output2[@]}" -gt 0 ] && printf '%s\n' "" "${a_output2[@]}" ||
printf '%s\n' "" " - No changes needed"
   printf '%s\n' "" " - remediation of kernel module: \"$l_mod_name\"
complete" ""
}
```

References:

1. NIST SP 800-53 Rev. 5: SI-4, CM-7

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.8 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.		●	●
v7	9.2 Ensure Only Approved Ports, Protocols and Services Are Running Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1068, T1068.000, T1210, T1210.000	TA0008	M1042

3.2.2 Ensure tipc kernel module is not available (Automated)

Profile Applicability:

- Level 2 - Server
- Level 2 - Workstation

Description:

The Transparent Inter-Process Communication (TIPC) protocol is designed to provide communication between cluster nodes.

Rationale:

- IF - the protocol is not being used, it is recommended that kernel module not be loaded, disabling the service to reduce the potential attack surface.

Audit:

Run the following script to verify:
- IF - the `tipc` kernel module is available in ANY installed kernel, verify:

- An entry including `/bin/true` or `/bin/false` exists in a file within the `/etc/modprobe.d/` directory
- The module is deny listed in a file within the `/etc/modprobe.d/` directory
- The module is not loaded in the running kernel

- IF - the `tipc` kernel module is not available on the system, or pre-compiled into the kernel, no additional configuration is necessary

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

{
   a_output=() a_output2=() a_output3=() l_dl="" l_mod_name="tipc"
l_mod_type="net"
   l_mod_path="$(readlink -f /lib/modules/**/kernel/$l_mod_type | sort -u)"
   f_module_chk()
   {
      l_dl="y" a_showconfig=()
      while IFS= read -r l_showconfig; do
         a_showconfig+=("$l_showconfig")
      done < <(modprobe --showconfig | grep -P --
'\b(install|blacklist)\h+'"${l_mod_chk_name//-/_}"'\b')
      if ! lsmod | grep "$l_mod_chk_name" &> /dev/null; then
         a_output+=("  - kernel module: \"$l_mod_name\" is not loaded")
      else
         a_output2+=("  - kernel module: \"$l_mod_name\" is loaded")
      fi
      if grep -Pq -- '\binstall\h+'"${l_mod_chk_name//-
/_}"'\h+(\/usr)?\/bin\/(true|false)\b' <<< "${a_showconfig[*]}"; then
         a_output+=("  - kernel module: \"$l_mod_name\" is not loadable")
      else
         a_output2+=("  - kernel module: \"$l_mod_name\" is loadable")
      fi
      if grep -Pq -- '\bblacklist\h+'"${l_mod_chk_name//-/_}"'\b' <<<
"${a_showconfig[*]}"; then
         a_output+=("  - kernel module: \"$l_mod_name\" is deny listed")
      else
         a_output2+=("  - kernel module: \"$l_mod_name\" is not deny listed")
      fi
   }
   for l_mod_base_directory in $l_mod_path; do
      if [ -d "$l_mod_base_directory/${l_mod_name/-/\/}" ] && [ -n "$(ls -A
"$l_mod_base_directory/${l_mod_name/-/\/}")" ]; then
         a_output3+=("  - \"$l_mod_base_directory\"")
         l_mod_chk_name="$l_mod_name"
         [[ "$l_mod_name" =~ overlay ]] && l_mod_chk_name="${l_mod_name::-2}"
         [ "$l_dl" != "y" ] && f_module_chk
      else
         a_output+=(" - kernel module: \"$l_mod_name\" doesn't exist in
\"$l_mod_base_directory\"")
      fi
   done
   [ "${#a_output3[@]}" -gt 0 ] && printf '%s\n' "" " -- INFO --" " - module:
\"$l_mod_name\" exists in:" "${a_output3[@]}"
   if [ "${#a_output2[@]}" -le 0 ]; then
      printf '%s\n' "" "- Audit Result:" "  ** PASS **" "${a_output[@]}"
   else
      printf '%s\n' "" "- Audit Result:" "  ** FAIL **" " - Reason(s) for
audit failure:" "${a_output2[@]}"
      [ "${#a_output[@]}" -gt 0 ] && printf '%s\n' "- Correctly set:"
"${a_output[@]}"
   fi
}
```

Remediation:

Run the following script to unload and disable the `tipc` module:
- IF - the `tipc` kernel module is available in ANY installed kernel:

- Create a file ending in `.conf` with `install tipc /bin/false` in the `/etc/modprobe.d/` directory
- Create a file ending in `.conf` with `blacklist tipc` in the `/etc/modprobe.d/` directory
- Run `modprobe -r tipc 2>/dev/null; rmmod tipc 2>/dev/null` to remove `tipc` from the kernel

- IF - the `tipc` kernel module is not available on the system, or pre-compiled into the kernel, no remediation is necessary

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

{
   a_output2=() a_output3=() l_dl="" l_mod_name="tipc" l_mod_type="net"
   l_mod_path="$(readlink -f /lib/modules/**/kernel/$l_mod_type | sort -u)"
   f_module_fix()
   {
      l_dl="y" a_showconfig=()
      while IFS= read -r l_showconfig; do
         a_showconfig+=("$l_showconfig")
      done < <(modprobe --showconfig | grep -P --
'\b(install|blacklist)\h+'"${l_mod_chk_name//-/_}"'\b')
      if  lsmod | grep "$l_mod_chk_name" &> /dev/null; then
         a_output2+=(" - unloading kernel module: \"$l_mod_name\"")
         modprobe -r "$l_mod_chk_name" 2>/dev/null; rmmod "$l_mod_name"
2>/dev/null
      fi
      if ! grep -Pq -- '\binstall\h+'"${l_mod_chk_name//-
/_}"'\h+(\/usr)?\/bin\/(true|false)\b' <<< "${a_showconfig[*]}"; then
         a_output2+=(" - setting kernel  module: \"$l_mod_name\" to
\"$(readlink -f /bin/false)\"")
         printf '%s\n' "install $l_mod_chk_name $(readlink -f /bin/false)" >>
/etc/modprobe.d/"$l_mod_name".conf
      fi
      if ! grep -Pq -- '\bblacklist\h+'"${l_mod_chk_name//-/_}"'\b' <<<
"${a_showconfig[*]}"; then
         a_output2+=(" - denylisting kernel module: \"$l_mod_name\"")
         printf '%s\n' "blacklist $l_mod_chk_name" >>
/etc/modprobe.d/"$l_mod_name".conf
      fi
   }
   for l_mod_base_directory in $l_mod_path; do # Check if the module exists
on the system
      if [ -d "$l_mod_base_directory/${l_mod_name/-/\/}" ] && [ -n "$(ls -A
"$l_mod_base_directory/${l_mod_name/-/\/}")" ]; then
         a_output3+=("  - \"$l_mod_base_directory\"")
         l_mod_chk_name="$l_mod_name"
         [[ "$l_mod_name" =~ overlay ]] && l_mod_chk_name="${l_mod_name::-2}"
         [ "$l_dl" != "y" ] && f_module_fix
      else
         printf '%s\n' " - kernel module: \"$l_mod_name\" doesn't exist in
\"$l_mod_base_directory\""
      fi
   done
   [ "${#a_output3[@]}" -gt 0 ] && printf '%s\n' "" " -- INFO --" " - module:
\"$l_mod_name\" exists in:" "${a_output3[@]}"
   [ "${#a_output2[@]}" -gt 0 ] && printf '%s\n' "" "${a_output2[@]}" ||
printf '%s\n' "" " - No changes needed"
   printf '%s\n' "" " - remediation of kernel module: \"$l_mod_name\"
complete" ""
}
```

References:

1. NIST SP 800-53 Rev. 5: SI-4, CM-7

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.8 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.		●	●
v7	9.2 Ensure Only Approved Ports, Protocols and Services Are Running Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1068, T1068.000, T1210, T1210.000	TA0008	M1042

3.2.3 Ensure rds kernel module is not available (Automated)

Profile Applicability:

- Level 2 - Server
- Level 2 - Workstation

Description:

The Reliable Datagram Sockets (RDS) protocol is a transport layer protocol designed to provide low-latency, high-bandwidth communications between cluster nodes. It was developed by the Oracle Corporation.

Rationale:

- IF - the protocol is not being used, it is recommended that kernel module not be loaded, disabling the service to reduce the potential attack surface.

Audit:

Run the following script to verify:
- IF - the `rds` kernel module is available in ANY installed kernel, verify:

- An entry including `/bin/true` or `/bin/false` exists in a file within the `/etc/modprobe.d/` directory
- The module is deny listed in a file within the `/etc/modprobe.d/` directory
- The module is not loaded in the running kernel

- IF - the `rds` kernel module is not available on the system, or pre-compiled into the kernel, no additional configuration is necessary

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

{
   a_output=() a_output2=() a_output3=() l_dl="" l_mod_name="rds"
l_mod_type="net"
   l_mod_path="$(readlink -f /lib/modules/**/kernel/$l_mod_type | sort -u)"
   f_module_chk()
   {
      l_dl="y" a_showconfig=()
      while IFS= read -r l_showconfig; do
         a_showconfig+=("$l_showconfig")
      done < <(modprobe --showconfig | grep -P --
'\b(install|blacklist)\h+'"${l_mod_chk_name//-/_}"'\b')
      if ! lsmod | grep "$l_mod_chk_name" &> /dev/null; then
         a_output+=("  - kernel module: \"$l_mod_name\" is not loaded")
      else
         a_output2+=("  - kernel module: \"$l_mod_name\" is loaded")
      fi
      if grep -Pq -- '\binstall\h+'"${l_mod_chk_name//-
/_}"'\h+(\/usr)?\/bin\/(true|false)\b' <<< "${a_showconfig[*]}"; then
         a_output+=("  - kernel module: \"$l_mod_name\" is not loadable")
      else
         a_output2+=("  - kernel module: \"$l_mod_name\" is loadable")
      fi
      if grep -Pq -- '\bblacklist\h+'"${l_mod_chk_name//-/_}"'\b' <<<
"${a_showconfig[*]}"; then
         a_output+=("  - kernel module: \"$l_mod_name\" is deny listed")
      else
         a_output2+=("  - kernel module: \"$l_mod_name\" is not deny listed")
      fi
   }
   for l_mod_base_directory in $l_mod_path; do
      if [ -d "$l_mod_base_directory/${l_mod_name/-/\/}" ] && [ -n "$(ls -A
"$l_mod_base_directory/${l_mod_name/-/\/}")" ]; then
         a_output3+=("  - \"$l_mod_base_directory\"")
         l_mod_chk_name="$l_mod_name"
         [[ "$l_mod_name" =~ overlay ]] && l_mod_chk_name="${l_mod_name::-2}"
         [ "$l_dl" != "y" ] && f_module_chk
      else
         a_output+=(" - kernel module: \"$l_mod_name\" doesn't exist in
\"$l_mod_base_directory\"")
      fi
   done
   [ "${#a_output3[@]}" -gt 0 ] && printf '%s\n' "" " -- INFO --" " - module:
\"$l_mod_name\" exists in:" "${a_output3[@]}"
   if [ "${#a_output2[@]}" -le 0 ]; then
      printf '%s\n' "" "- Audit Result:" "  ** PASS **" "${a_output[@]}"
   else
      printf '%s\n' "" "- Audit Result:" "  ** FAIL **" " - Reason(s) for
audit failure:" "${a_output2[@]}"
      [ "${#a_output[@]}" -gt 0 ] && printf '%s\n' "- Correctly set:"
"${a_output[@]}"
   fi
}
```

Remediation:

Run the following script to unload and disable the `rds` module:
- IF - the `rds` kernel module is available in ANY installed kernel:

- Create a file ending in `.conf` with `install rds /bin/false` in the `/etc/modprobe.d/` directory
- Create a file ending in `.conf` with `blacklist rds` in the `/etc/modprobe.d/` directory
- Run `modprobe -r rds 2>/dev/null; rmmod rds 2>/dev/null` to remove `rds` from the kernel

- IF - the `rds` kernel module is not available on the system, or pre-compiled into the kernel, no remediation is necessary

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

{
   a_output2=() a_output3=() l_dl="" l_mod_name="rds" l_mod_type="net"
   l_mod_path="$(readlink -f /lib/modules/**/kernel/$l_mod_type | sort -u)"
   f_module_fix()
   {
      l_dl="y" a_showconfig=()
      while IFS= read -r l_showconfig; do
         a_showconfig+=("$l_showconfig")
      done < <(modprobe --showconfig | grep -P --
'\b(install|blacklist)\h+'"${l_mod_chk_name//-/_}"'\b')
      if  lsmod | grep "$l_mod_chk_name" &> /dev/null; then
         a_output2+=(" - unloading kernel module: \"$l_mod_name\"")
         modprobe -r "$l_mod_chk_name" 2>/dev/null; rmmod "$l_mod_name"
2>/dev/null
      fi
      if ! grep -Pq -- '\binstall\h+'"${l_mod_chk_name//-
/_}"'\h+(\/usr)?\/bin\/(true|false)\b' <<< "${a_showconfig[*]}"; then
         a_output2+=(" - setting kernel  module: \"$l_mod_name\" to
\"$(readlink -f /bin/false)\"")
         printf '%s\n' "install $l_mod_chk_name $(readlink -f /bin/false)" >>
/etc/modprobe.d/"$l_mod_name".conf
      fi
      if ! grep -Pq -- '\bblacklist\h+'"${l_mod_chk_name//-/_}"'\b' <<<
"${a_showconfig[*]}"; then
         a_output2+=(" - denylisting kernel module: \"$l_mod_name\"")
         printf '%s\n' "blacklist $l_mod_chk_name" >>
/etc/modprobe.d/"$l_mod_name".conf
      fi
   }
   for l_mod_base_directory in $l_mod_path; do # Check if the module exists
on the system
      if [ -d "$l_mod_base_directory/${l_mod_name/-/\/}" ] && [ -n "$(ls -A
"$l_mod_base_directory/${l_mod_name/-/\/}")" ]; then
         a_output3+=("  - \"$l_mod_base_directory\"")
         l_mod_chk_name="$l_mod_name"
         [[ "$l_mod_name" =~ overlay ]] && l_mod_chk_name="${l_mod_name::-2}"
         [ "$l_dl" != "y" ] && f_module_fix
      else
         printf '%s\n' " - kernel module: \"$l_mod_name\" doesn't exist in
\"$l_mod_base_directory\""
      fi
   done
   [ "${#a_output3[@]}" -gt 0 ] && printf '%s\n' "" " -- INFO --" " - module:
\"$l_mod_name\" exists in:" "${a_output3[@]}"
   [ "${#a_output2[@]}" -gt 0 ] && printf '%s\n' "" "${a_output2[@]}" ||
printf '%s\n' "" " - No changes needed"
   printf '%s\n' "" " - remediation of kernel module: \"$l_mod_name\"
complete" ""
}
```

References:

1. NIST SP 800-53 Rev. 5: SI-4, CM-7

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.8 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.		●	●
v7	9.2 Ensure Only Approved Ports, Protocols and Services Are Running Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1068, T1068.000, T1210, T1210.000	TA0008	M1042

3.2.4 Ensure sctp kernel module is not available (Automated)

Profile Applicability:

- Level 2 - Server
- Level 2 - Workstation

Description:

The Stream Control Transmission Protocol (SCTP) is a transport layer protocol used to support message oriented communication, with several streams of messages in one connection. It serves a similar function as TCP and UDP, incorporating features of both. It is message-oriented like UDP, and ensures reliable in-sequence transport of messages with congestion control like TCP.

Rationale:

- IF - the protocol is not being used, it is recommended that kernel module not be loaded, disabling the service to reduce the potential attack surface.

Audit:

Run the following script to verify:
- IF - the `sctp` kernel module is available in ANY installed kernel, verify:

- An entry including `/bin/true` or `/bin/false` exists in a file within the `/etc/modprobe.d/` directory
- The module is deny listed in a file within the `/etc/modprobe.d/` directory
- The module is not loaded in the running kernel

- IF - the `sctp` kernel module is not available on the system, or pre-compiled into the kernel, no additional configuration is necessary

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

{
   a_output=() a_output2=() a_output3=() l_dl="" l_mod_name="sctp"
l_mod_type="net"
   l_mod_path="$(readlink -f /lib/modules/**/kernel/$l_mod_type | sort -u)"
   f_module_chk()
   {
      l_dl="y" a_showconfig=()
      while IFS= read -r l_showconfig; do
         a_showconfig+=("$l_showconfig")
      done < <(modprobe --showconfig | grep -P --
'\b(install|blacklist)\h+'"${l_mod_chk_name//-/_}"'\b')
      if ! lsmod | grep "$l_mod_chk_name" &> /dev/null; then
         a_output+=("  - kernel module: \"$l_mod_name\" is not loaded")
      else
         a_output2+=("  - kernel module: \"$l_mod_name\" is loaded")
      fi
      if grep -Pq -- '\binstall\h+'"${l_mod_chk_name//-
/_}"'\h+(\/usr)?\/bin\/(true|false)\b' <<< "${a_showconfig[*]}"; then
         a_output+=("  - kernel module: \"$l_mod_name\" is not loadable")
      else
         a_output2+=("  - kernel module: \"$l_mod_name\" is loadable")
      fi
      if grep -Pq -- '\bblacklist\h+'"${l_mod_chk_name//-/_}"'\b' <<<
"${a_showconfig[*]}"; then
         a_output+=("  - kernel module: \"$l_mod_name\" is deny listed")
      else
         a_output2+=("  - kernel module: \"$l_mod_name\" is not deny listed")
      fi
   }
   for l_mod_base_directory in $l_mod_path; do
      if [ -d "$l_mod_base_directory/${l_mod_name/-/\/}" ] && [ -n "$(ls -A
"$l_mod_base_directory/${l_mod_name/-/\/}")" ]; then
         a_output3+=("  - \"$l_mod_base_directory\"")
         l_mod_chk_name="$l_mod_name"
         [[ "$l_mod_name" =~ overlay ]] && l_mod_chk_name="${l_mod_name::-2}"
         [ "$l_dl" != "y" ] && f_module_chk
      else
         a_output+=(" - kernel module: \"$l_mod_name\" doesn't exist in
\"$l_mod_base_directory\"")
      fi
   done
   [ "${#a_output3[@]}" -gt 0 ] && printf '%s\n' "" " -- INFO --" " - module:
\"$l_mod_name\" exists in:" "${a_output3[@]}"
   if [ "${#a_output2[@]}" -le 0 ]; then
      printf '%s\n' "" "- Audit Result:" "  ** PASS **" "${a_output[@]}"
   else
      printf '%s\n' "" "- Audit Result:" "  ** FAIL **" " - Reason(s) for
audit failure:" "${a_output2[@]}"
      [ "${#a_output[@]}" -gt 0 ] && printf '%s\n' "- Correctly set:"
"${a_output[@]}"
   fi
}
```

Remediation:

Run the following script to unload and disable the `sctp` module:
- IF - the `sctp` kernel module is available in ANY installed kernel:

- Create a file ending in `.conf` with `install sctp /bin/false` in the `/etc/modprobe.d/` directory
- Create a file ending in `.conf` with `blacklist sctp` in the `/etc/modprobe.d/` directory
- Run `modprobe -r sctp 2>/dev/null; rmmod sctp 2>/dev/null` to remove `sctp` from the kernel

- IF - the `sctp` kernel module is not available on the system, or pre-compiled into the kernel, no remediation is necessary

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

{
   a_output2=() a_output3=() l_dl="" l_mod_name="sctp" l_mod_type="net"
   l_mod_path="$(readlink -f /lib/modules/**/kernel/$l_mod_type | sort -u)"
   f_module_fix()
   {
      l_dl="y" a_showconfig=()
      while IFS= read -r l_showconfig; do
         a_showconfig+=("$l_showconfig")
      done < <(modprobe --showconfig | grep -P --
'\b(install|blacklist)\h+'"${l_mod_chk_name//-/_}"'\b')
      if  lsmod | grep "$l_mod_chk_name" &> /dev/null; then
         a_output2+=(" - unloading kernel module: \"$l_mod_name\"")
         modprobe -r "$l_mod_chk_name" 2>/dev/null; rmmod "$l_mod_name"
2>/dev/null
      fi
      if ! grep -Pq -- '\binstall\h+'"${l_mod_chk_name//-
/_}"'\h+(\/usr)?\/bin\/(true|false)\b' <<< "${a_showconfig[*]}"; then
         a_output2+=(" - setting kernel  module: \"$l_mod_name\" to
\"$(readlink -f /bin/false)\"")
         printf '%s\n' "install $l_mod_chk_name $(readlink -f /bin/false)" >>
/etc/modprobe.d/"$l_mod_name".conf
      fi
      if ! grep -Pq -- '\bblacklist\h+'"${l_mod_chk_name//-/_}"'\b' <<<
"${a_showconfig[*]}"; then
         a_output2+=(" - denylisting kernel module: \"$l_mod_name\"")
         printf '%s\n' "blacklist $l_mod_chk_name" >>
/etc/modprobe.d/"$l_mod_name".conf
      fi
   }
   for l_mod_base_directory in $l_mod_path; do # Check if the module exists
on the system
      if [ -d "$l_mod_base_directory/${l_mod_name/-/\/}" ] && [ -n "$(ls -A
"$l_mod_base_directory/${l_mod_name/-/\/}")" ]; then
         a_output3+=("  - \"$l_mod_base_directory\"")
         l_mod_chk_name="$l_mod_name"
         [[ "$l_mod_name" =~ overlay ]] && l_mod_chk_name="${l_mod_name::-2}"
         [ "$l_dl" != "y" ] && f_module_fix
      else
         printf '%s\n' " - kernel module: \"$l_mod_name\" doesn't exist in
\"$l_mod_base_directory\""
      fi
   done
   [ "${#a_output3[@]}" -gt 0 ] && printf '%s\n' "" " -- INFO --" " - module:
\"$l_mod_name\" exists in:" "${a_output3[@]}"
   [ "${#a_output2[@]}" -gt 0 ] && printf '%s\n' "" "${a_output2[@]}" ||
printf '%s\n' "" " - No changes needed"
   printf '%s\n' "" " - remediation of kernel module: \"$l_mod_name\"
complete" ""
}
```

References:

1. NIST SP 800-53 Rev. 5: SI-4, CM-7

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.8 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.		●	●
v7	9.2 Ensure Only Approved Ports, Protocols and Services Are Running Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1068, T1068.000, T1210, T1210.000	TA0008	M1042

3.3 Configure Network Kernel Parameters

The following network parameters are intended for use on both host only and router systems. A system acts as a router if it has at least two interfaces and is configured to perform routing functions.

Note:

- sysctl settings are defined through files in `/usr/local/lib`, `/usr/lib/`, `/lib/`, `/run/`, and `/etc/`
- Files are typically placed in the `sysctl.d` directory within the parent directory
- The paths where sysctl preload files usually exist
    - `/run/sysctl.d/*.conf`
    - `/etc/sysctl.d/*.conf`
    - `/usr/local/lib/sysctl.d/*.conf`
    - `/usr/lib/sysctl.d/*.conf`
    - `/lib/sysctl.d/*.conf`
    - `/etc/sysctl.conf`
- Files must have the "`.conf`" extension
- Vendors settings usually live in `/usr/lib/` or `/usr/local/lib/`
- To override a whole file, create a new file with the same name in `/etc/sysctl.d/` and put new settings there.
- To override only specific settings, add a file with a lexically later name in `/etc/sysctl.d/` and put new settings there.
- The command `/usr/lib/systemd/systemd-sysctl --cat-config` produces output containing The system's loaded kernel parameters and the files they're configured in:
    - Entries listed latter in the file take precedence over the same settings listed earlier in the file
    - Files containing kernel parameters that are over-ridden by other files with the same name will not be listed
    - On systems running UncomplicatedFirewall, the kernel parameters may be set or over-written. This will not be visible in the output of the command
- On systems with Uncomplicated Firewall, additional settings may be configured in `/etc/ufw/sysctl.conf`
    - The settings in `/etc/ufw/sysctl.conf` will override settings other settings and **will not** be visible in the output of the `/usr/lib/systemd/systemd-sysctl --cat-config` command
    - This behavior can be changed by updating the `IPT_SYSCTL` parameter in `/etc/default/ufw`

The system's loaded kernel parameters and the files they're configured in can be viewed by running the following command:

```
# /usr/lib/systemd/systemd-sysctl --cat-config
```

3.3.1 Ensure ip forwarding is disabled (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

The `net.ipv4.ip_forward` and `net.ipv6.conf.all.forwarding` flags are used to tell the system whether it can forward packets or not.

Rationale:

Setting `net.ipv4.ip_forward` and `net.ipv6.conf.all.forwarding` to `0` ensures that a system with multiple interfaces (for example, a hard proxy), will never be able to forward packets, and therefore, never serve as a router.

Impact:

IP forwarding is required on systems configured to act as a router. If these parameters are disabled, the system will not be able to perform as a router.

Many Cloud Service Provider (CSP) hosted systems require IP forwarding to be enabled. If the system is running on a CSP platform, this requirement should be reviewed before disabling IP forwarding.

Audit:

Run the following script to verify the following kernel parameters are set in the running configuration and correctly loaded from a kernel parameter configuration file:

- `net.ipv4.ip_forward` is set to `0`
- `net.ipv6.conf.all.forwarding` is set to `0`

Note:

- kernel parameters are loaded by file and parameter order precedence. The following script observes this precedence as part of the auditing procedure. The parameters being checked may be set correctly in a file. If that file is superseded, the parameter is overridden by an incorrect setting later in that file, or in a canonically later file, that "correct" setting will be ignored both by the script and by the system during a normal kernel parameter load sequence.
- IPv6 kernel parameters only apply to systems where IPv6 is enabled

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

{
   a_output=(); a_output2=(); l_ipv6_disabled=""
   a_parlist=("net.ipv4.ip_forward=0" "net.ipv6.conf.all.forwarding=0")
   l_ufwscf="$([ -f /etc/default/ufw ] && awk -F= '/^\s*IPT_SYSCTL=/ {print
$2}' /etc/default/ufw)"
   f_ipv6_chk()
   {
      l_ipv6_disabled="no"
      ! grep -Pqs -- '^\h*0\b' /sys/module/ipv6/parameters/disable &&
l_ipv6_disabled="yes"
      if sysctl net.ipv6.conf.all.disable_ipv6 | grep -Pqs --
"^\h*net\.ipv6\.conf\.all\.disable_ipv6\h*=\h*1\b" && \
         sysctl net.ipv6.conf.default.disable_ipv6 | grep -Pqs --
"^\h*net\.ipv6\.conf\.default\.disable_ipv6\h*=\h*1\b"; then
         l_ipv6_disabled="yes"
      fi
   }
   f_kernel_parameter_chk()
   {
      l_running_parameter_value="$(sysctl "$l_parameter_name" | awk -F=
'{print $2}' | xargs)" # Check running configuration
      if grep -Pq -- '\b'"$l_parameter_value"'\b' <<<
"$l_running_parameter_value"; then
         a_output+=(" - \"$l_parameter_name\" is correctly set to
\"$l_running_parameter_value\""
         "    in the running configuration")
      else
         a_output2+=(" - \"$l_parameter_name\" is incorrectly set to
\"$l_running_parameter_value\"" \
         "    in the running configuration" \
         "    and should have a value of: \"$l_value_out\"")
      fi
      unset A_out; declare -A A_out # Check durable setting (files)
      while read -r l_out; do
         if [ -n "$l_out" ]; then
            if [[ $l_out =~ ^\s*# ]]; then
               l_file="${l_out//# /}"
            else
               l_kpar="$(awk -F= '{print $1}' <<< "$l_out" | xargs)"
               [ "$l_kpar" = "$l_parameter_name" ] &&
A_out+=(["$l_kpar"]="$l_file")
            fi
         fi
      done < <("$l_systemdsysctl" --cat-config | grep -Po
'^\h*([^#\n\r]+|#\h*\/[^#\n\r\h]+\.conf\b)')
      if [ -n "$l_ufwscf" ]; then # Account for systems with UFW (Not covered
by systemd-sysctl --cat-config)
         l_kpar="$(grep -Po "^\h*$l_parameter_name\b" "$l_ufwscf" | xargs)"
         l_kpar="${l_kpar//\//.}"
         [ "$l_kpar" = "$l_parameter_name" ] &&
A_out+=(["$l_kpar"]="$l_ufwscf")
      fi
      if (( ${#A_out[@]} > 0 )); then # Assess output from files and generate
output
         while IFS="=" read -r l_fkpname l_file_parameter_value; do
```

```
            l_fkpname="${l_fkpname// /}";
l_file_parameter_value="${l_file_parameter_value// /}"
            if grep -Pq -- '\b'"$l_parameter_value"'\b' <<<
"$l_file_parameter_value"; then
               a_output+=(" - \"$l_parameter_name\" is correctly set to
\"$l_file_parameter_value\"" \
               "    in \"$(printf '%s' "${A_out[@]}")\"")
            else
               a_output2+=(" - \"$l_parameter_name\" is incorrectly set to
\"$l_file_parameter_value\""
               "    in \"$(printf '%s' "${A_out[@]}")\"" \
               "    and should have a value of: \"$l_value_out\"")
            fi
         done < <(grep -Po -- "^\h*$l_parameter_name\h*=\h*\H+"
"${A_out[@]}")
      else
         a_output2+=(" - \"$l_parameter_name\" is not set in an included
file" \
         "    ** Note: \"$l_parameter_name\" May be set in a file that's
ignored by load procedure **")
      fi
   }
   l_systemdsysctl="$(readlink -f /lib/systemd/systemd-sysctl)"
   while IFS="=" read -r l_parameter_name l_parameter_value; do # Assess and
check parameters
      l_parameter_name="${l_parameter_name// /}";
l_parameter_value="${l_parameter_value// /}"
      l_value_out="${l_parameter_value//-/ through }";
l_value_out="${l_value_out//|/ or }"
      l_value_out="$(tr -d '(){}' <<< "$l_value_out")"
      if grep -q '^net.ipv6.' <<< "$l_parameter_name"; then
         [ -z "$l_ipv6_disabled" ] && f_ipv6_chk
         if [ "$l_ipv6_disabled" = "yes" ]; then
            a_output+=(" - IPv6 is disabled on the system,
\"$l_parameter_name\" is not applicable")
         else
            f_kernel_parameter_chk
         fi
      else
         f_kernel_parameter_chk
      fi
   done < <(printf '%s\n' "${a_parlist[@]}")
   if [ "${#a_output2[@]}" -le 0 ]; then
      printf '%s\n' "" "- Audit Result:" "  ** PASS **" "${a_output[@]}" ""
   else
      printf '%s\n' "" "- Audit Result:" "  ** FAIL **" " - Reason(s) for
audit failure:" "${a_output2[@]}"
      [ "${#a_output[@]}" -gt 0 ] && printf '%s\n' "" "- Correctly set:"
"${a_output[@]}" ""
   fi
}
```

Remediation:

Set the following parameter in `/etc/sysctl.conf` or a file in `/etc/sysctl.d/` ending in `.conf`:

- `net.ipv4.ip_forward = 0`

Example:

```
# printf '%s\n' "net.ipv4.ip_forward = 0" >> /etc/sysctl.d/60-
netipv4_sysctl.conf
```

Run the following script to set the active kernel parameters:

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

{
   sysctl -w net.ipv4.ip_forward=0
   sysctl -w net.ipv4.route.flush=1
}
```

- IF - IPv6 is enabled on the system:
Set the following parameter in `/etc/sysctl.conf` or a file in `/etc/sysctl.d/` ending in `.conf`:

- `net.ipv6.conf.all.forwarding = 0`

Example:

```
# printf '%s\n' "net.ipv6.conf.all.forwarding = 0" >> /etc/sysctl.d/60-
netipv6_sysctl.conf
```

Run the following script to set the active kernel parameters:

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

{
   sysctl -w net.ipv6.conf.all.forwarding=0
   sysctl -w net.ipv6.route.flush=1
}
```

Note: If these settings appear in a canonically later file, or later in the same file, these settings will be overwritten

Default Value:

net.ipv4.ip_forward = 0

net.ipv6.conf.all.forwarding = 0

References:

1. NIST SP 800-53 Rev. 5: CM-1, CM-2, CM-6, CM-7, IA-5

Additional Information:

On systems with Uncomplicated Firewall, additional settings may be configured in `/etc/ufw/sysctl.conf`

- The settings in `/etc/ufw/sysctl.conf` will override settings in `/etc/sysctl.conf`
- This behavior can be changed by updating the `IPT_SYSCTL` parameter in `/etc/default/ufw`

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.8 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.		●	●
v7	9.2 Ensure Only Approved Ports, Protocols and Services Are Running Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1557, T1557.000	TA0006, TA0009	M1030, M1042

3.3.2 Ensure packet redirect sending is disabled (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

ICMP Redirects are used to send routing information to other hosts. As a host itself does not act as a router (in a host only configuration), there is no need to send redirects.

Rationale:

An attacker could use a compromised host to send invalid ICMP redirects to other router devices in an attempt to corrupt routing and have users access a system set up by the attacker as opposed to a valid system.

Impact:

IP forwarding is required on systems configured to act as a router. If these parameters are disabled, the system will not be able to perform as a router.

Audit:

Run the following script to verify the following kernel parameters are set in the running configuration and correctly loaded from a kernel parameter configuration file:

- `net.ipv4.conf.all.send_redirects` is set to `0`
- `net.ipv4.conf.default.send_redirects` is set to `0`

Note: kernel parameters are loaded by file and parameter order precedence. The following script observes this precedence as part of the auditing procedure. The parameters being checked may be set correctly in a file. If that file is superseded, the parameter is overridden by an incorrect setting later in that file, or in a canonically later file, that "correct" setting will be ignored both by the script and by the system during a normal kernel parameter load sequence.

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

{
   a_output=(); a_output2=(); l_ipv6_disabled=""
   a_parlist=("net.ipv4.conf.all.send_redirects=0"
"net.ipv4.conf.default.send_redirects=0")
   l_ufwscf="$([ -f /etc/default/ufw ] && awk -F= '/^\s*IPT_SYSCTL=/ {print
$2}' /etc/default/ufw)"
   f_ipv6_chk()
   {
      l_ipv6_disabled="no"
      ! grep -Pqs -- '^\h*0\b' /sys/module/ipv6/parameters/disable &&
l_ipv6_disabled="yes"
      if sysctl net.ipv6.conf.all.disable_ipv6 | grep -Pqs --
"^\h*net\.ipv6\.conf\.all\.disable_ipv6\h*=\h*1\b" && \
         sysctl net.ipv6.conf.default.disable_ipv6 | grep -Pqs --
"^\h*net\.ipv6\.conf\.default\.disable_ipv6\h*=\h*1\b"; then
         l_ipv6_disabled="yes"
      fi
   }
   f_kernel_parameter_chk()
   {
      l_running_parameter_value="$(sysctl "$l_parameter_name" | awk -F=
'{print $2}' | xargs)" # Check running configuration
      if grep -Pq -- '\b'"$l_parameter_value"'\b' <<<
"$l_running_parameter_value"; then
         a_output+=(" - \"$l_parameter_name\" is correctly set to
\"$l_running_parameter_value\""
         "    in the running configuration")
      else
         a_output2+=(" - \"$l_parameter_name\" is incorrectly set to
\"$l_running_parameter_value\"" \
         "    in the running configuration" \
         "    and should have a value of: \"$l_value_out\"")
      fi
      unset A_out; declare -A A_out # Check durable setting (files)
      while read -r l_out; do
         if [ -n "$l_out" ]; then
            if [[ $l_out =~ ^\s*# ]]; then
               l_file="${l_out//# /}"
            else
               l_kpar="$(awk -F= '{print $1}' <<< "$l_out" | xargs)"
               [ "$l_kpar" = "$l_parameter_name" ] &&
A_out+=(["$l_kpar"]="$l_file")
            fi
         fi
      done < <("$l_systemdsysctl" --cat-config | grep -Po
'^\h*([^#\n\r]+|#\h*\/[^#\n\r\h]+\.conf\b)')
      if [ -n "$l_ufwscf" ]; then # Account for systems with UFW (Not covered
by systemd-sysctl --cat-config)
         l_kpar="$(grep -Po "^\h*$l_parameter_name\b" "$l_ufwscf" | xargs)"
         l_kpar="${l_kpar//\//.}"
         [ "$l_kpar" = "$l_parameter_name" ] &&
A_out+=(["$l_kpar"]="$l_ufwscf")
      fi
      if (( ${#A_out[@]} > 0 )); then # Assess output from files and generate
output
```

```
         while IFS="=" read -r l_fkpname l_file_parameter_value; do
            l_fkpname="${l_fkpname// /}";
l_file_parameter_value="${l_file_parameter_value// /}"
            if grep -Pq -- '\b'"$l_parameter_value"'\b' <<<
"$l_file_parameter_value"; then
               a_output+=(" - \"$l_parameter_name\" is correctly set to
\"$l_file_parameter_value\"" \
               "    in \"$(printf '%s' "${A_out[@]}")\"")
            else
               a_output2+=(" - \"$l_parameter_name\" is incorrectly set to
\"$l_file_parameter_value\""
               "    in \"$(printf '%s' "${A_out[@]}")\"" \
               "    and should have a value of: \"$l_value_out\"")
            fi
         done < <(grep -Po -- "^\h*$l_parameter_name\h*=\h*\H+"
"${A_out[@]}")
      else
         a_output2+=(" - \"$l_parameter_name\" is not set in an included
file" \
         "    ** Note: \"$l_parameter_name\" May be set in a file that's
ignored by load procedure **")
      fi
   }
   l_systemdsysctl="$(readlink -f /lib/systemd/systemd-sysctl)"
   while IFS="=" read -r l_parameter_name l_parameter_value; do # Assess and
check parameters
      l_parameter_name="${l_parameter_name// /}";
l_parameter_value="${l_parameter_value// /}"
      l_value_out="${l_parameter_value//-/ through }";
l_value_out="${l_value_out//|/ or }"
      l_value_out="$(tr -d '(){}' <<< "$l_value_out")"
      if grep -q '^net.ipv6.' <<< "$l_parameter_name"; then
         [ -z "$l_ipv6_disabled" ] && f_ipv6_chk
         if [ "$l_ipv6_disabled" = "yes" ]; then
            a_output+=(" - IPv6 is disabled on the system,
\"$l_parameter_name\" is not applicable")
         else
            f_kernel_parameter_chk
         fi
      else
         f_kernel_parameter_chk
      fi
   done < <(printf '%s\n' "${a_parlist[@]}")
   if [ "${#a_output2[@]}" -le 0 ]; then
      printf '%s\n' "" "- Audit Result:" "  ** PASS **" "${a_output[@]}" ""
   else
      printf '%s\n' "" "- Audit Result:" "  ** FAIL **" " - Reason(s) for
audit failure:" "${a_output2[@]}"
      [ "${#a_output[@]}" -gt 0 ] && printf '%s\n' "" "- Correctly set:"
"${a_output[@]}" ""
   fi
}
```

Remediation:

Set the following parameters in `/etc/sysctl.conf` or a file in `/etc/sysctl.d/` ending in `.conf`:

- `net.ipv4.conf.all.send_redirects = 0`
- `net.ipv4.conf.default.send_redirects = 0`

Example:

```
# printf '%s\n' "net.ipv4.conf.all.send_redirects = 0"
"net.ipv4.conf.default.send_redirects = 0" >> /etc/sysctl.d/60-
netipv4_sysctl.conf
```

Run the following script to set the active kernel parameters:

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

{
   sysctl -w net.ipv4.conf.all.send_redirects=0
   sysctl -w net.ipv4.conf.default.send_redirects=0
   sysctl -w net.ipv4.route.flush=1
}
```

Note: If these settings appear in a canonically later file, or later in the same file, these settings will be overwritten

Default Value:

net.ipv4.conf.all.send_redirects = 1

net.ipv4.conf.default.send_redirects = 1

References:

1. NIST SP 800-53 Rev. 5: CM-1, CM-2, CM-6, CM-7, IA-5

Additional Information:

On systems with Uncomplicated Firewall, additional settings may be configured in `/etc/ufw/sysctl.conf`

- The settings in `/etc/ufw/sysctl.conf` will override settings in `/etc/sysctl.conf`
- This behavior can be changed by updating the `IPT_SYSCTL` parameter in `/etc/default/ufw`

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.8 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.		●	●
v7	9.2 Ensure Only Approved Ports, Protocols and Services Are Running Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1557, T1557.000	TA0006, TA0009	M1030, M1042

3.3.3 Ensure bogus icmp responses are ignored (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

Setting `net.ipv4.icmp_ignore_bogus_error_responses` to `1` prevents the kernel from logging bogus responses (RFC-1122 non-compliant) from broadcast reframes, keeping file systems from filling up with useless log messages.

Rationale:

Some routers (and some attackers) will send responses that violate RFC-1122 and attempt to fill up a log file system with many useless error messages.

Audit:

Run the following script to verify the following kernel parameter is set in the running configuration and correctly loaded from a kernel parameter configuration file:

- `net.ipv4.icmp_ignore_bogus_error_responses` is set to `1`

Note: kernel parameters are loaded by file and parameter order precedence. The following script observes this precedence as part of the auditing procedure. The parameters being checked may be set correctly in a file. If that file is superseded, the parameter is overridden by an incorrect setting later in that file, or in a canonically later file, that "correct" setting will be ignored both by the script and by the system during a normal kernel parameter load sequence.

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

{
   a_output=(); a_output2=(); l_ipv6_disabled=""
   a_parlist=("net.ipv4.icmp_ignore_bogus_error_responses=1")
   l_ufwscf="$([ -f /etc/default/ufw ] && awk -F= '/^\s*IPT_SYSCTL=/ {print
$2}' /etc/default/ufw)"
   f_ipv6_chk()
   {
      l_ipv6_disabled="no"
      ! grep -Pqs -- '^\h*0\b' /sys/module/ipv6/parameters/disable &&
l_ipv6_disabled="yes"
      if sysctl net.ipv6.conf.all.disable_ipv6 | grep -Pqs --
"^\h*net\.ipv6\.conf\.all\.disable_ipv6\h*=\h*1\b" && \
         sysctl net.ipv6.conf.default.disable_ipv6 | grep -Pqs --
"^\h*net\.ipv6\.conf\.default\.disable_ipv6\h*=\h*1\b"; then
         l_ipv6_disabled="yes"
      fi
   }
   f_kernel_parameter_chk()
   {
      l_running_parameter_value="$(sysctl "$l_parameter_name" | awk -F=
'{print $2}' | xargs)" # Check running configuration
      if grep -Pq -- '\b'"$l_parameter_value"'\b' <<<
"$l_running_parameter_value"; then
         a_output+=(" - \"$l_parameter_name\" is correctly set to
\"$l_running_parameter_value\""
         "    in the running configuration")
      else
         a_output2+=(" - \"$l_parameter_name\" is incorrectly set to
\"$l_running_parameter_value\"" \
         "    in the running configuration" \
         "    and should have a value of: \"$l_value_out\"")
      fi
      unset A_out; declare -A A_out # Check durable setting (files)
      while read -r l_out; do
         if [ -n "$l_out" ]; then
            if [[ $l_out =~ ^\s*# ]]; then
               l_file="${l_out//# /}"
            else
               l_kpar="$(awk -F= '{print $1}' <<< "$l_out" | xargs)"
               [ "$l_kpar" = "$l_parameter_name" ] &&
A_out+=(["$l_kpar"]="$l_file")
            fi
         fi
      done < <("$l_systemdsysctl" --cat-config | grep -Po
'^\h*([^#\n\r]+|#\h*\/[^#\n\r\h]+\.conf\b)')
      if [ -n "$l_ufwscf" ]; then # Account for systems with UFW (Not covered
by systemd-sysctl --cat-config)
         l_kpar="$(grep -Po "^\h*$l_parameter_name\b" "$l_ufwscf" | xargs)"
         l_kpar="${l_kpar//\//.}"
         [ "$l_kpar" = "$l_parameter_name" ] &&
A_out+=(["$l_kpar"]="$l_ufwscf")
      fi
      if (( ${#A_out[@]} > 0 )); then # Assess output from files and generate
output
         while IFS="=" read -r l_fkpname l_file_parameter_value; do
```

```
            l_fkpname="${l_fkpname// /}";
l_file_parameter_value="${l_file_parameter_value// /}"
            if grep -Pq -- '\b'"$l_parameter_value"'\b' <<<
"$l_file_parameter_value"; then
               a_output+=(" - \"$l_parameter_name\" is correctly set to
\"$l_file_parameter_value\"" \
               "    in \"$(printf '%s' "${A_out[@]}")\"")
            else
               a_output2+=(" - \"$l_parameter_name\" is incorrectly set to
\"$l_file_parameter_value\""
               "    in \"$(printf '%s' "${A_out[@]}")\"" \
               "    and should have a value of: \"$l_value_out\"")
            fi
         done < <(grep -Po -- "^\h*$l_parameter_name\h*=\h*\H+"
"${A_out[@]}")
      else
         a_output2+=(" - \"$l_parameter_name\" is not set in an included
file" \
         "    ** Note: \"$l_parameter_name\" May be set in a file that's
ignored by load procedure **")
      fi
   }
   l_systemdsysctl="$(readlink -f /lib/systemd/systemd-sysctl)"
   while IFS="=" read -r l_parameter_name l_parameter_value; do # Assess and
check parameters
      l_parameter_name="${l_parameter_name// /}";
l_parameter_value="${l_parameter_value// /}"
      l_value_out="${l_parameter_value//-/ through }";
l_value_out="${l_value_out//|/ or }"
      l_value_out="$(tr -d '(){}' <<< "$l_value_out")"
      if grep -q '^net.ipv6.' <<< "$l_parameter_name"; then
         [ -z "$l_ipv6_disabled" ] && f_ipv6_chk
         if [ "$l_ipv6_disabled" = "yes" ]; then
            a_output+=(" - IPv6 is disabled on the system,
\"$l_parameter_name\" is not applicable")
         else
            f_kernel_parameter_chk
         fi
      else
         f_kernel_parameter_chk
      fi
   done < <(printf '%s\n' "${a_parlist[@]}")
   if [ "${#a_output2[@]}" -le 0 ]; then
      printf '%s\n' "" "- Audit Result:" "  ** PASS **" "${a_output[@]}" ""
   else
      printf '%s\n' "" "- Audit Result:" "  ** FAIL **" " - Reason(s) for
audit failure:" "${a_output2[@]}"
      [ "${#a_output[@]}" -gt 0 ] && printf '%s\n' "" "- Correctly set:"
"${a_output[@]}" ""
   fi
}
```

Remediation:

Set the following parameter in `/etc/sysctl.conf` or a file in `/etc/sysctl.d/` ending in `.conf`:

- `net.ipv4.icmp_ignore_bogus_error_responses = 1`

Example:

```
# printf '%s\n' "net.ipv4.icmp_ignore_bogus_error_responses = 1" >>
/etc/sysctl.d/60-netipv4_sysctl.conf
```

Run the following script to set the active kernel parameters:

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

{
   sysctl -w net.ipv4.icmp_ignore_bogus_error_responses=1
   sysctl -w net.ipv4.route.flush=1
}
```

Note: If these settings appear in a canonically later file, or later in the same file, these settings will be overwritten

Default Value:

net.ipv4.icmp_ignore_bogus_error_responses = 1

References:

1. NIST SP 800-53 Rev. 5: CM-1,CM-2, CM-6, CM-7, IA-5

Additional Information:

On systems with Uncomplicated Firewall, additional settings may be configured in `/etc/ufw/sysctl.conf`

- The settings in `/etc/ufw/sysctl.conf` will override settings in `/etc/sysctl.conf`
- This behavior can be changed by updating the `IPT_SYSCTL` parameter in `/etc/default/ufw`

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.8 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.		●	●
v7	9.2 Ensure Only Approved Ports, Protocols and Services Are Running Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1562, T1562.006	TA0040	M1053

3.3.4 Ensure broadcast icmp requests are ignored (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

Setting `net.ipv4.icmp_echo_ignore_broadcasts` to `1` will cause the system to ignore all ICMP echo and timestamp requests to broadcast and multicast addresses.

Rationale:

Accepting ICMP echo and timestamp requests with broadcast or multicast destinations for your network could be used to trick your host into starting (or participating) in a Smurf attack. A Smurf attack relies on an attacker sending large amounts of ICMP broadcast messages with a spoofed source address. All hosts receiving this message and responding would send echo-reply messages back to the spoofed address, which is probably not routable. If many hosts respond to the packets, the amount of traffic on the network could be significantly multiplied.

Audit:

Run the following script to verify the following kernel parameter is set in the running configuration and correctly loaded from a kernel parameter configuration file:

- `net.ipv4.icmp_echo_ignore_broadcasts` is set to `1`

Note: kernel parameters are loaded by file and parameter order precedence. The following script observes this precedence as part of the auditing procedure. The parameters being checked may be set correctly in a file. If that file is superseded, the parameter is overridden by an incorrect setting later in that file, or in a canonically later file, that "correct" setting will be ignored both by the script and by the system during a normal kernel parameter load sequence.

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

{
   a_output=(); a_output2=(); l_ipv6_disabled=""
   a_parlist=("net.ipv4.icmp_echo_ignore_broadcasts=1")
   l_ufwscf="$([ -f /etc/default/ufw ] && awk -F= '/^\s*IPT_SYSCTL=/ {print
$2}' /etc/default/ufw)"
   f_ipv6_chk()
   {
      l_ipv6_disabled="no"
      ! grep -Pqs -- '^\h*0\b' /sys/module/ipv6/parameters/disable &&
l_ipv6_disabled="yes"
      if sysctl net.ipv6.conf.all.disable_ipv6 | grep -Pqs --
"^\h*net\.ipv6\.conf\.all\.disable_ipv6\h*=\h*1\b" && \
         sysctl net.ipv6.conf.default.disable_ipv6 | grep -Pqs --
"^\h*net\.ipv6\.conf\.default\.disable_ipv6\h*=\h*1\b"; then
         l_ipv6_disabled="yes"
      fi
   }
   f_kernel_parameter_chk()
   {
      l_running_parameter_value="$(sysctl "$l_parameter_name" | awk -F=
'{print $2}' | xargs)" # Check running configuration
      if grep -Pq -- '\b'"$l_parameter_value"'\b' <<<
"$l_running_parameter_value"; then
         a_output+=(" - \"$l_parameter_name\" is correctly set to
\"$l_running_parameter_value\""
         "    in the running configuration")
      else
         a_output2+=(" - \"$l_parameter_name\" is incorrectly set to
\"$l_running_parameter_value\"" \
         "    in the running configuration" \
         "    and should have a value of: \"$l_value_out\"")
      fi
      unset A_out; declare -A A_out # Check durable setting (files)
      while read -r l_out; do
         if [ -n "$l_out" ]; then
            if [[ $l_out =~ ^\s*# ]]; then
               l_file="${l_out//# /}"
            else
               l_kpar="$(awk -F= '{print $1}' <<< "$l_out" | xargs)"
               [ "$l_kpar" = "$l_parameter_name" ] &&
A_out+=(["$l_kpar"]="$l_file")
            fi
         fi
      done < <("$l_systemdsysctl" --cat-config | grep -Po
'^\h*([^#\n\r]+|#\h*\/[^#\n\r\h]+\.conf\b)')
      if [ -n "$l_ufwscf" ]; then # Account for systems with UFW (Not covered
by systemd-sysctl --cat-config)
         l_kpar="$(grep -Po "^\h*$l_parameter_name\b" "$l_ufwscf" | xargs)"
         l_kpar="${l_kpar//\//.}"
         [ "$l_kpar" = "$l_parameter_name" ] &&
A_out+=(["$l_kpar"]="$l_ufwscf")
      fi
      if (( ${#A_out[@]} > 0 )); then # Assess output from files and generate
output
         while IFS="=" read -r l_fkpname l_file_parameter_value; do
```

```
            l_fkpname="${l_fkpname// /}";
l_file_parameter_value="${l_file_parameter_value// /}"
            if grep -Pq -- '\b'"$l_parameter_value"'\b' <<<
"$l_file_parameter_value"; then
               a_output+=(" - \"$l_parameter_name\" is correctly set to
\"$l_file_parameter_value\"" \
               "    in \"$(printf '%s' "${A_out[@]}")\"")
            else
               a_output2+=(" - \"$l_parameter_name\" is incorrectly set to
\"$l_file_parameter_value\""
               "    in \"$(printf '%s' "${A_out[@]}")\"" \
               "    and should have a value of: \"$l_value_out\"")
            fi
         done < <(grep -Po -- "^\h*$l_parameter_name\h*=\h*\H+"
"${A_out[@]}")
      else
         a_output2+=(" - \"$l_parameter_name\" is not set in an included
file" \
         "    ** Note: \"$l_parameter_name\" May be set in a file that's
ignored by load procedure **")
      fi
   }
   l_systemdsysctl="$(readlink -f /lib/systemd/systemd-sysctl)"
   while IFS="=" read -r l_parameter_name l_parameter_value; do # Assess and
check parameters
      l_parameter_name="${l_parameter_name// /}";
l_parameter_value="${l_parameter_value// /}"
      l_value_out="${l_parameter_value//-/ through }";
l_value_out="${l_value_out//|/ or }"
      l_value_out="$(tr -d '(){}' <<< "$l_value_out")"
      if grep -q '^net.ipv6.' <<< "$l_parameter_name"; then
         [ -z "$l_ipv6_disabled" ] && f_ipv6_chk
         if [ "$l_ipv6_disabled" = "yes" ]; then
            a_output+=(" - IPv6 is disabled on the system,
\"$l_parameter_name\" is not applicable")
         else
            f_kernel_parameter_chk
         fi
      else
         f_kernel_parameter_chk
      fi
   done < <(printf '%s\n' "${a_parlist[@]}")
   if [ "${#a_output2[@]}" -le 0 ]; then
      printf '%s\n' "" "- Audit Result:" "  ** PASS **" "${a_output[@]}" ""
   else
      printf '%s\n' "" "- Audit Result:" "  ** FAIL **" " - Reason(s) for
audit failure:" "${a_output2[@]}"
      [ "${#a_output[@]}" -gt 0 ] && printf '%s\n' "" "- Correctly set:"
"${a_output[@]}" ""
   fi
}
```

Remediation:

Set the following parameter in `/etc/sysctl.conf` or a file in `/etc/sysctl.d/` ending in `.conf`:

- `net.ipv4.icmp_echo_ignore_broadcasts = 1`

Example:

```
# printf '%s\n' "net.ipv4.icmp_echo_ignore_broadcasts = 1" >>
/etc/sysctl.d/60-netipv4_sysctl.conf
```

Run the following script to set the active kernel parameters:

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

{
   sysctl -w net.ipv4.icmp_echo_ignore_broadcasts=1
   sysctl -w net.ipv4.route.flush=1
}
```

Note: If these settings appear in a canonically later file, or later in the same file, these settings will be overwritten

Default Value:

net.ipv4.icmp_echo_ignore_broadcasts = 1

References:

1. NIST SP 800-53 Rev. 5: CM-1, CM-2, CM-6, CM-7, IA-5

Additional Information:

On systems with Uncomplicated Firewall, additional settings may be configured in `/etc/ufw/sysctl.conf`

- The settings in `/etc/ufw/sysctl.conf` will override settings in `/etc/sysctl.conf`
- This behavior can be changed by updating the `IPT_SYSCTL` parameter in `/etc/default/ufw`

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.8 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.		●	●
v7	9.2 Ensure Only Approved Ports, Protocols and Services Are Running Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1498, T1498.001	TA0040	M1037

3.3.5 Ensure icmp redirects are not accepted (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

ICMP redirect messages are packets that convey routing information and tell your host (acting as a router) to send packets via an alternate path. It is a way of allowing an outside routing device to update your system routing tables.

Rationale:

ICMP redirect messages are packets that convey routing information and tell your host (acting as a router) to send packets via an alternate path. It is a way of allowing an outside routing device to update your system routing tables. By setting `net.ipv4.conf.all.accept_redirects`, `net.ipv4.conf.default.accept_redirects`, `net.ipv6.conf.all.accept_redirects`, and `net.ipv6.conf.default.accept_redirects` to `0`, the system will not accept any ICMP redirect messages, and therefore, won't allow outsiders to update the system's routing tables.

Audit:

Run the following script to verify the following kernel parameters are set in the running configuration and correctly loaded from a kernel parameter configuration file:

- `net.ipv4.conf.all.accept_redirects` is set to `0`
- `net.ipv4.conf.default.accept_redirects` is set to `0`
- `net.ipv6.conf.all.accept_redirects` is set to `0`
- `net.ipv6.conf.default.accept_redirects` is set to `0`

Note:

- kernel parameters are loaded by file and parameter order precedence. The following script observes this precedence as part of the auditing procedure. The parameters being checked may be set correctly in a file. If that file is superseded, the parameter is overridden by an incorrect setting later in that file, or in a canonically later file, that "correct" setting will be ignored both by the script and by the system during a normal kernel parameter load sequence.
- IPv6 kernel parameters only apply to systems where IPv6 is enabled

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

{
   a_output=(); a_output2=(); l_ipv6_disabled=""
   a_parlist=("net.ipv4.conf.all.accept_redirects=0"
"net.ipv4.conf.default.accept_redirects=0"
"net.ipv6.conf.all.accept_redirects=0"
"net.ipv6.conf.default.accept_redirects=0")
   l_ufwscf="$([ -f /etc/default/ufw ] && awk -F= '/^\s*IPT_SYSCTL=/ {print
$2}' /etc/default/ufw)"
   f_ipv6_chk()
   {
      l_ipv6_disabled="no"
      ! grep -Pqs -- '^\h*0\b' /sys/module/ipv6/parameters/disable &&
l_ipv6_disabled="yes"
      if sysctl net.ipv6.conf.all.disable_ipv6 | grep -Pqs --
"^\h*net\.ipv6\.conf\.all\.disable_ipv6\h*=\h*1\b" && \
         sysctl net.ipv6.conf.default.disable_ipv6 | grep -Pqs --
"^\h*net\.ipv6\.conf\.default\.disable_ipv6\h*=\h*1\b"; then
         l_ipv6_disabled="yes"
      fi
   }
   f_kernel_parameter_chk()
   {
      l_running_parameter_value="$(sysctl "$l_parameter_name" | awk -F=
'{print $2}' | xargs)" # Check running configuration
      if grep -Pq -- '\b'"$l_parameter_value"'\b' <<<
"$l_running_parameter_value"; then
         a_output+=(" - \"$l_parameter_name\" is correctly set to
\"$l_running_parameter_value\""
         "    in the running configuration")
      else
         a_output2+=(" - \"$l_parameter_name\" is incorrectly set to
\"$l_running_parameter_value\"" \
         "    in the running configuration" \
         "    and should have a value of: \"$l_value_out\"")
      fi
      unset A_out; declare -A A_out # Check durable setting (files)
      while read -r l_out; do
         if [ -n "$l_out" ]; then
            if [[ $l_out =~ ^\s*# ]]; then
               l_file="${l_out//# /}"
            else
               l_kpar="$(awk -F= '{print $1}' <<< "$l_out" | xargs)"
               [ "$l_kpar" = "$l_parameter_name" ] &&
A_out+=(["$l_kpar"]="$l_file")
            fi
         fi
      done < <("$l_systemdsysctl" --cat-config | grep -Po
'^\h*([^#\n\r]+|#\h*\/[^#\n\r\h]+\.conf\b)')
      if [ -n "$l_ufwscf" ]; then # Account for systems with UFW (Not covered
by systemd-sysctl --cat-config)
         l_kpar="$(grep -Po "^\h*$l_parameter_name\b" "$l_ufwscf" | xargs)"
         l_kpar="${l_kpar//\//.}"
         [ "$l_kpar" = "$l_parameter_name" ] &&
A_out+=(["$l_kpar"]="$l_ufwscf")
      fi
```

```
      if (( ${#A_out[@]} > 0 )); then # Assess output from files and generate
output
         while IFS="=" read -r l_fkpname l_file_parameter_value; do
            l_fkpname="${l_fkpname// /}";
l_file_parameter_value="${l_file_parameter_value// /}"
            if grep -Pq -- '\b'"$l_parameter_value"'\b' <<<
"$l_file_parameter_value"; then
               a_output+=(" - \"$l_parameter_name\" is correctly set to
\"$l_file_parameter_value\"" \
               "    in \"$(printf '%s' "${A_out[@]}")\"")
            else
               a_output2+=(" - \"$l_parameter_name\" is incorrectly set to
\"$l_file_parameter_value\""
               "    in \"$(printf '%s' "${A_out[@]}")\"" \
               "    and should have a value of: \"$l_value_out\"")
            fi
         done < <(grep -Po -- "^\h*$l_parameter_name\h*=\h*\H+"
"${A_out[@]}")
      else
         a_output2+=(" - \"$l_parameter_name\" is not set in an included
file" \
         "    ** Note: \"$l_parameter_name\" May be set in a file that's
ignored by load procedure **")
      fi
   }
   l_systemdsysctl="$(readlink -f /lib/systemd/systemd-sysctl)"
   while IFS="=" read -r l_parameter_name l_parameter_value; do # Assess and
check parameters
      l_parameter_name="${l_parameter_name// /}";
l_parameter_value="${l_parameter_value// /}"
      l_value_out="${l_parameter_value//-/ through }";
l_value_out="${l_value_out//|/ or }"
      l_value_out="$(tr -d '(){}' <<< "$l_value_out")"
      if grep -q '^net.ipv6.' <<< "$l_parameter_name"; then
         [ -z "$l_ipv6_disabled" ] && f_ipv6_chk
         if [ "$l_ipv6_disabled" = "yes" ]; then
            a_output+=(" - IPv6 is disabled on the system,
\"$l_parameter_name\" is not applicable")
         else
            f_kernel_parameter_chk
         fi
      else
         f_kernel_parameter_chk
      fi
   done < <(printf '%s\n' "${a_parlist[@]}")
   if [ "${#a_output2[@]}" -le 0 ]; then
      printf '%s\n' "" "- Audit Result:" "  ** PASS **" "${a_output[@]}" ""
   else
      printf '%s\n' "" "- Audit Result:" "  ** FAIL **" " - Reason(s) for
audit failure:" "${a_output2[@]}"
      [ "${#a_output[@]}" -gt 0 ] && printf '%s\n' "" "- Correctly set:"
"${a_output[@]}" ""
   fi
}
```

Remediation:

Set the following parameters in `/etc/sysctl.conf` or a file in `/etc/sysctl.d/` ending in `.conf`:

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

Example:

```
# printf '%s\n' "net.ipv4.conf.all.accept_redirects = 0"
"net.ipv4.conf.default.accept_redirects = 0" >> /etc/sysctl.d/60-
netipv4_sysctl.conf
```

Run the following script to set the active kernel parameters:

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

{
   sysctl -w net.ipv4.conf.all.accept_redirects=0
   sysctl -w net.ipv4.conf.default.accept_redirects=0
   sysctl -w net.ipv4.route.flush=1
}
```

- IF - IPv6 is enabled on the system:
Set the following parameters in `/etc/sysctl.conf` or a file in `/etc/sysctl.d/` ending in `.conf`:

- `net.ipv6.conf.all.accept_redirects = 0`
- `net.ipv6.conf.default.accept_redirects = 0`

Example:

```
# printf '%s\n' "net.ipv6.conf.all.accept_redirects = 0"
"net.ipv6.conf.default.accept_redirects = 0" >> /etc/sysctl.d/60-
netipv6_sysctl.conf
```

Run the following script to set the active kernel parameters:

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

{
   sysctl -w net.ipv6.conf.all.accept_redirects=0
   sysctl -w net.ipv6.conf.default.accept_redirects=0
   sysctl -w net.ipv6.route.flush=1
}
```

Note: If these settings appear in a canonically later file, or later in the same file, these settings will be overwritten

Default Value:

net.ipv4.conf.all.accept_redirects = 1

net.ipv4.conf.default.accept_redirects = 1

net.ipv6.conf.all.accept_redirects = 1

net.ipv6.conf.default.accept_redirects = 1

References:

1. NIST SP 800-53 Rev. 5: CM-1, CM-2, CM-6, CM-7, IA-5

Additional Information:

On systems with Uncomplicated Firewall, additional settings may be configured in `/etc/ufw/sysctl.conf`

- The settings in `/etc/ufw/sysctl.conf` will override settings in `/etc/sysctl.conf`
- This behavior can be changed by updating the `IPT_SYSCTL` parameter in `/etc/default/ufw`

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.8 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.		●	●
v7	9.2 Ensure Only Approved Ports, Protocols and Services Are Running Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1557, T1557.000	TA0006, TA0009	M1030, M1042

3.3.6 Ensure secure icmp redirects are not accepted (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

Secure ICMP redirects are the same as ICMP redirects, except they come from gateways listed on the default gateway list. It is assumed that these gateways are known to your system, and that they are likely to be secure.

Rationale:

It is still possible for even known gateways to be compromised. Setting `net.ipv4.conf.all.secure_redirects` and `net.ipv4.conf.default.secure_redirects` to `0` protects the system from routing table updates by possibly compromised known gateways.

Audit:

Run the following script to verify the following kernel parameters are set in the running configuration and correctly loaded from a kernel parameter configuration file:

- `net.ipv4.conf.all.secure_redirects` is set to `0`
- `net.ipv4.conf.default.secure_redirects` is set to `0`

Note: kernel parameters are loaded by file and parameter order precedence. The following script observes this precedence as part of the auditing procedure. The parameters being checked may be set correctly in a file. If that file is superseded, the parameter is overridden by an incorrect setting later in that file, or in a canonically later file, that "correct" setting will be ignored both by the script and by the system during a normal kernel parameter load sequence.

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

{
   a_output=(); a_output2=(); l_ipv6_disabled=""
   a_parlist=("net.ipv4.conf.all.secure_redirects=0"
"net.ipv4.conf.default.secure_redirects=0")
   l_ufwscf="$([ -f /etc/default/ufw ] && awk -F= '/^\s*IPT_SYSCTL=/ {print
$2}' /etc/default/ufw)"
   f_ipv6_chk()
   {
      l_ipv6_disabled="no"
      ! grep -Pqs -- '^\h*0\b' /sys/module/ipv6/parameters/disable &&
l_ipv6_disabled="yes"
      if sysctl net.ipv6.conf.all.disable_ipv6 | grep -Pqs --
"^\h*net\.ipv6\.conf\.all\.disable_ipv6\h*=\h*1\b" && \
         sysctl net.ipv6.conf.default.disable_ipv6 | grep -Pqs --
"^\h*net\.ipv6\.conf\.default\.disable_ipv6\h*=\h*1\b"; then
         l_ipv6_disabled="yes"
      fi
   }
   f_kernel_parameter_chk()
   {
      l_running_parameter_value="$(sysctl "$l_parameter_name" | awk -F=
'{print $2}' | xargs)" # Check running configuration
      if grep -Pq -- '\b'"$l_parameter_value"'\b' <<<
"$l_running_parameter_value"; then
         a_output+=(" - \"$l_parameter_name\" is correctly set to
\"$l_running_parameter_value\""
         "    in the running configuration")
      else
         a_output2+=(" - \"$l_parameter_name\" is incorrectly set to
\"$l_running_parameter_value\"" \
         "    in the running configuration" \
         "    and should have a value of: \"$l_value_out\"")
      fi
      unset A_out; declare -A A_out # Check durable setting (files)
      while read -r l_out; do
         if [ -n "$l_out" ]; then
            if [[ $l_out =~ ^\s*# ]]; then
               l_file="${l_out//# /}"
            else
               l_kpar="$(awk -F= '{print $1}' <<< "$l_out" | xargs)"
               [ "$l_kpar" = "$l_parameter_name" ] &&
A_out+=(["$l_kpar"]="$l_file")
            fi
         fi
      done < <("$l_systemdsysctl" --cat-config | grep -Po
'^\h*([^#\n\r]+|#\h*\/[^#\n\r\h]+\.conf\b)')
      if [ -n "$l_ufwscf" ]; then # Account for systems with UFW (Not covered
by systemd-sysctl --cat-config)
         l_kpar="$(grep -Po "^\h*$l_parameter_name\b" "$l_ufwscf" | xargs)"
         l_kpar="${l_kpar//\//.}"
         [ "$l_kpar" = "$l_parameter_name" ] &&
A_out+=(["$l_kpar"]="$l_ufwscf")
      fi
      if (( ${#A_out[@]} > 0 )); then # Assess output from files and generate
output
```

```
         while IFS="=" read -r l_fkpname l_file_parameter_value; do
            l_fkpname="${l_fkpname// /}";
l_file_parameter_value="${l_file_parameter_value// /}"
            if grep -Pq -- '\b'"$l_parameter_value"'\b' <<<
"$l_file_parameter_value"; then
               a_output+=(" - \"$l_parameter_name\" is correctly set to
\"$l_file_parameter_value\"" \
               "    in \"$(printf '%s' "${A_out[@]}")\"")
            else
               a_output2+=(" - \"$l_parameter_name\" is incorrectly set to
\"$l_file_parameter_value\""
               "    in \"$(printf '%s' "${A_out[@]}")\"" \
               "    and should have a value of: \"$l_value_out\"")
            fi
         done < <(grep -Po -- "^\h*$l_parameter_name\h*=\h*\H+"
"${A_out[@]}")
      else
         a_output2+=(" - \"$l_parameter_name\" is not set in an included
file" \
         "    ** Note: \"$l_parameter_name\" May be set in a file that's
ignored by load procedure **")
      fi
   }
   l_systemdsysctl="$(readlink -f /lib/systemd/systemd-sysctl)"
   while IFS="=" read -r l_parameter_name l_parameter_value; do # Assess and
check parameters
      l_parameter_name="${l_parameter_name// /}";
l_parameter_value="${l_parameter_value// /}"
      l_value_out="${l_parameter_value//-/ through }";
l_value_out="${l_value_out//|/ or }"
      l_value_out="$(tr -d '(){}' <<< "$l_value_out")"
      if grep -q '^net.ipv6.' <<< "$l_parameter_name"; then
         [ -z "$l_ipv6_disabled" ] && f_ipv6_chk
         if [ "$l_ipv6_disabled" = "yes" ]; then
            a_output+=(" - IPv6 is disabled on the system,
\"$l_parameter_name\" is not applicable")
         else
            f_kernel_parameter_chk
         fi
      else
         f_kernel_parameter_chk
      fi
   done < <(printf '%s\n' "${a_parlist[@]}")
   if [ "${#a_output2[@]}" -le 0 ]; then
      printf '%s\n' "" "- Audit Result:" "  ** PASS **" "${a_output[@]}" ""
   else
      printf '%s\n' "" "- Audit Result:" "  ** FAIL **" " - Reason(s) for
audit failure:" "${a_output2[@]}"
      [ "${#a_output[@]}" -gt 0 ] && printf '%s\n' "" "- Correctly set:"
"${a_output[@]}" ""
   fi
}
```

Remediation:

Set the following parameters in `/etc/sysctl.conf` or a file in `/etc/sysctl.d/` ending in `.conf`:

- `net.ipv4.conf.all.secure_redirects = 0`
- `net.ipv4.conf.default.secure_redirects = 0`

Example:

```
# printf '%s\n' "net.ipv4.conf.all.secure_redirects = 0"
"net.ipv4.conf.default.secure_redirects = 0" >> /etc/sysctl.d/60-
netipv4_sysctl.conf
```

Run the following script to set the active kernel parameters:

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

{
   sysctl -w net.ipv4.conf.all.secure_redirects=0
   sysctl -w net.ipv4.conf.default.secure_redirects=0
   sysctl -w net.ipv4.route.flush=1
}
```

Note: If these settings appear in a canonically later file, or later in the same file, these settings will be overwritten

Default Value:

net.ipv4.conf.all.secure_redirects = 1

net.ipv4.conf.default.secure_redirects = 1

References:

1. NIST SP 800-53 Rev. 5: CM-1, CM-2, CM-6, CM-7, IA-5

Additional Information:

On systems with Uncomplicated Firewall, additional settings may be configured in `/etc/ufw/sysctl.conf`

- The settings in `/etc/ufw/sysctl.conf` will override settings in `/etc/sysctl.conf`
- This behavior can be changed by updating the `IPT_SYSCTL` parameter in `/etc/default/ufw`

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.8 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.		●	●
v7	9.2 Ensure Only Approved Ports, Protocols and Services Are Running Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1557, T1557.000	TA0006, TA0009	M1030, M1042

3.3.7 Ensure reverse path filtering is enabled (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

Setting `net.ipv4.conf.all.rp_filter` and `net.ipv4.conf.default.rp_filter` to `1` forces the Linux kernel to utilize reverse path filtering on a received packet to determine if the packet was valid. Essentially, with reverse path filtering, if the return packet does not go out the same interface that the corresponding source packet came from, the packet is dropped (and logged if `log_martians` is set).

Rationale:

Setting `net.ipv4.conf.all.rp_filter` and `net.ipv4.conf.default.rp_filter` to `1` is a good way to deter attackers from sending your system bogus packets that cannot be responded to. One instance where this feature breaks down is if asymmetrical routing is employed. This would occur when using dynamic routing protocols (bgp, ospf, etc) on your system. If you are using asymmetrical routing on your system, you will not be able to enable this feature without breaking the routing.

Impact:

If you are using asymmetrical routing on your system, you will not be able to enable this feature without breaking the routing.

Audit:

Run the following script to verify the following kernel parameters are set in the running configuration and correctly loaded from a kernel parameter configuration file:

- `net.ipv4.conf.all.rp_filter` is set to `1`
- `net.ipv4.conf.default.rp_filter` is set to `1`

Note: kernel parameters are loaded by file and parameter order precedence. The following script observes this precedence as part of the auditing procedure. The parameters being checked may be set correctly in a file. If that file is superseded, the parameter is overridden by an incorrect setting later in that file, or in a canonically later file, that "correct" setting will be ignored both by the script and by the system during a normal kernel parameter load sequence.

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

{
   a_output=(); a_output2=(); l_ipv6_disabled=""
   a_parlist=("net.ipv4.conf.all.rp_filter=1"
"net.ipv4.conf.default.rp_filter=1")
   l_ufwscf="$([ -f /etc/default/ufw ] && awk -F= '/^\s*IPT_SYSCTL=/ {print
$2}' /etc/default/ufw)"
   f_ipv6_chk()
   {
      l_ipv6_disabled="no"
      ! grep -Pqs -- '^\h*0\b' /sys/module/ipv6/parameters/disable &&
l_ipv6_disabled="yes"
      if sysctl net.ipv6.conf.all.disable_ipv6 | grep -Pqs --
"^\h*net\.ipv6\.conf\.all\.disable_ipv6\h*=\h*1\b" && \
         sysctl net.ipv6.conf.default.disable_ipv6 | grep -Pqs --
"^\h*net\.ipv6\.conf\.default\.disable_ipv6\h*=\h*1\b"; then
         l_ipv6_disabled="yes"
      fi
   }
   f_kernel_parameter_chk()
   {
      l_running_parameter_value="$(sysctl "$l_parameter_name" | awk -F=
'{print $2}' | xargs)" # Check running configuration
      if grep -Pq -- '\b'"$l_parameter_value"'\b' <<<
"$l_running_parameter_value"; then
         a_output+=(" - \"$l_parameter_name\" is correctly set to
\"$l_running_parameter_value\""
         "    in the running configuration")
      else
         a_output2+=(" - \"$l_parameter_name\" is incorrectly set to
\"$l_running_parameter_value\"" \
         "    in the running configuration" \
         "    and should have a value of: \"$l_value_out\"")
      fi
      unset A_out; declare -A A_out # Check durable setting (files)
      while read -r l_out; do
         if [ -n "$l_out" ]; then
            if [[ $l_out =~ ^\s*# ]]; then
               l_file="${l_out//# /}"
            else
               l_kpar="$(awk -F= '{print $1}' <<< "$l_out" | xargs)"
               [ "$l_kpar" = "$l_parameter_name" ] &&
A_out+=(["$l_kpar"]="$l_file")
            fi
         fi
      done < <("$l_systemdsysctl" --cat-config | grep -Po
'^\h*([^#\n\r]+|#\h*\/[^#\n\r\h]+\.conf\b)')
      if [ -n "$l_ufwscf" ]; then # Account for systems with UFW (Not covered
by systemd-sysctl --cat-config)
         l_kpar="$(grep -Po "^\h*$l_parameter_name\b" "$l_ufwscf" | xargs)"
         l_kpar="${l_kpar//\//.}"
         [ "$l_kpar" = "$l_parameter_name" ] &&
A_out+=(["$l_kpar"]="$l_ufwscf")
      fi
      if (( ${#A_out[@]} > 0 )); then # Assess output from files and generate
output
```

```
         while IFS="=" read -r l_fkpname l_file_parameter_value; do
            l_fkpname="${l_fkpname// /}";
l_file_parameter_value="${l_file_parameter_value// /}"
            if grep -Pq -- '\b'"$l_parameter_value"'\b' <<<
"$l_file_parameter_value"; then
               a_output+=(" - \"$l_parameter_name\" is correctly set to
\"$l_file_parameter_value\"" \
               "    in \"$(printf '%s' "${A_out[@]}")\"")
            else
               a_output2+=(" - \"$l_parameter_name\" is incorrectly set to
\"$l_file_parameter_value\""
               "    in \"$(printf '%s' "${A_out[@]}")\"" \
               "    and should have a value of: \"$l_value_out\"")
            fi
         done < <(grep -Po -- "^\h*$l_parameter_name\h*=\h*\H+"
"${A_out[@]}")
      else
         a_output2+=(" - \"$l_parameter_name\" is not set in an included
file" \
         "    ** Note: \"$l_parameter_name\" May be set in a file that's
ignored by load procedure **")
      fi
   }
   l_systemdsysctl="$(readlink -f /lib/systemd/systemd-sysctl)"
   while IFS="=" read -r l_parameter_name l_parameter_value; do # Assess and
check parameters
      l_parameter_name="${l_parameter_name// /}";
l_parameter_value="${l_parameter_value// /}"
      l_value_out="${l_parameter_value//-/ through }";
l_value_out="${l_value_out//|/ or }"
      l_value_out="$(tr -d '(){}' <<< "$l_value_out")"
      if grep -q '^net.ipv6.' <<< "$l_parameter_name"; then
         [ -z "$l_ipv6_disabled" ] && f_ipv6_chk
         if [ "$l_ipv6_disabled" = "yes" ]; then
            a_output+=(" - IPv6 is disabled on the system,
\"$l_parameter_name\" is not applicable")
         else
            f_kernel_parameter_chk
         fi
      else
         f_kernel_parameter_chk
      fi
   done < <(printf '%s\n' "${a_parlist[@]}")
   if [ "${#a_output2[@]}" -le 0 ]; then
      printf '%s\n' "" "- Audit Result:" "  ** PASS **" "${a_output[@]}" ""
   else
      printf '%s\n' "" "- Audit Result:" "  ** FAIL **" " - Reason(s) for
audit failure:" "${a_output2[@]}"
      [ "${#a_output[@]}" -gt 0 ] && printf '%s\n' "" "- Correctly set:"
"${a_output[@]}" ""
   fi
}
```

Remediation:

Set the following parameters in `/etc/sysctl.conf` or a file in `/etc/sysctl.d/` ending in `.conf`:

- `net.ipv4.conf.all.rp_filter = 1`
- `net.ipv4.conf.default.rp_filter = 1`

Example:

```
# printf '%s\n' "net.ipv4.conf.all.rp_filter = 1"
"net.ipv4.conf.default.rp_filter = 1" >> /etc/sysctl.d/60-netipv4_sysctl.conf
```

Run the following script to set the active kernel parameters:

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

{
   sysctl -w net.ipv4.conf.all.rp_filter=1
   sysctl -w net.ipv4.conf.default.rp_filter=1
   sysctl -w net.ipv4.route.flush=1
}
```

Note: If these settings appear in a canonically later file, or later in the same file, these settings will be overwritten

Default Value:

net.ipv4.conf.all.rp_filter = 2

net.ipv4.conf.default.rp_filter = 1

References:

1. NIST SP 800-53 Rev. 5: CM-1,CM-2, CM-6, CM-7, IA-5

Additional Information:

On systems with Uncomplicated Firewall, additional settings may be configured in `/etc/ufw/sysctl.conf`

- The settings in `/etc/ufw/sysctl.conf` will override settings in `/etc/sysctl.conf`
- This behavior can be changed by updating the `IPT_SYSCTL` parameter in `/etc/default/ufw`

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.8 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.		●	●
v7	9.2 Ensure Only Approved Ports, Protocols and Services Are Running Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1498, T1498.001	TA0006, TA0040	M1030, M1042

3.3.8 Ensure source routed packets are not accepted (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

In networking, source routing allows a sender to partially or fully specify the route packets take through a network. In contrast, non-source routed packets travel a path determined by routers in the network. In some cases, systems may not be routable or reachable from some locations (e.g. private addresses vs. Internet routable), and so source routed packets would need to be used.

Rationale:

Setting `net.ipv4.conf.all.accept_source_route`, `net.ipv4.conf.default.accept_source_route`, `net.ipv6.conf.all.accept_source_route` and `net.ipv6.conf.default.accept_source_route` to `0` disables the system from accepting source routed packets. Assume this system was capable of routing packets to Internet routable addresses on one interface and private addresses on another interface. Assume that the private addresses were not routable to the Internet routable addresses and vice versa. Under normal routing circumstances, an attacker from the Internet routable addresses could not use the system as a way to reach the private address systems. If, however, source routed packets were allowed, they could be used to gain access to the private address systems as the route could be specified, rather than rely on routing protocols that did not allow this routing.

Audit:
Run the following script to verify the following kernel parameters are set in the running configuration and correctly loaded from a kernel parameter configuration file:

- `net.ipv4.conf.all.accept_source_route` is set to `0`
- `net.ipv4.conf.default.accept_source_route` is set to `0`
- `net.ipv6.conf.all.accept_source_route` is set to `0`
- `net.ipv6.conf.default.accept_source_route` is set to `0`

Note:

- kernel parameters are loaded by file and parameter order precedence. The following script observes this precedence as part of the auditing procedure. The parameters being checked may be set correctly in a file. If that file is superseded, the parameter is overridden by an incorrect setting later in that file, or in a canonically later file, that "correct" setting will be ignored both by the script and by the system during a normal kernel parameter load sequence.
- IPv6 kernel parameters only apply to systems where IPv6 is enabled

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

{
   a_output=(); a_output2=(); l_ipv6_disabled=""
   a_parlist=("net.ipv4.conf.all.accept_source_route=0"
"net.ipv4.conf.default.accept_source_route=0"
"net.ipv6.conf.all.accept_source_route=0"
"net.ipv6.conf.default.accept_source_route=0")
   l_ufwscf="$([ -f /etc/default/ufw ] && awk -F= '/^\s*IPT_SYSCTL=/ {print
$2}' /etc/default/ufw)"
   f_ipv6_chk()
   {
      l_ipv6_disabled="no"
      ! grep -Pqs -- '^\h*0\b' /sys/module/ipv6/parameters/disable &&
l_ipv6_disabled="yes"
      if sysctl net.ipv6.conf.all.disable_ipv6 | grep -Pqs --
"^\h*net\.ipv6\.conf\.all\.disable_ipv6\h*=\h*1\b" && \
         sysctl net.ipv6.conf.default.disable_ipv6 | grep -Pqs --
"^\h*net\.ipv6\.conf\.default\.disable_ipv6\h*=\h*1\b"; then
         l_ipv6_disabled="yes"
      fi
   }
   f_kernel_parameter_chk()
   {
      l_running_parameter_value="$(sysctl "$l_parameter_name" | awk -F=
'{print $2}' | xargs)" # Check running configuration
      if grep -Pq -- '\b'"$l_parameter_value"'\b' <<<
"$l_running_parameter_value"; then
         a_output+=(" - \"$l_parameter_name\" is correctly set to
\"$l_running_parameter_value\""
         "    in the running configuration")
      else
         a_output2+=(" - \"$l_parameter_name\" is incorrectly set to
\"$l_running_parameter_value\"" \
         "    in the running configuration" \
         "    and should have a value of: \"$l_value_out\"")
      fi
      unset A_out; declare -A A_out # Check durable setting (files)
      while read -r l_out; do
         if [ -n "$l_out" ]; then
            if [[ $l_out =~ ^\s*# ]]; then
               l_file="${l_out//# /}"
            else
               l_kpar="$(awk -F= '{print $1}' <<< "$l_out" | xargs)"
               [ "$l_kpar" = "$l_parameter_name" ] &&
A_out+=(["$l_kpar"]="$l_file")
            fi
         fi
      done < <("$l_systemdsysctl" --cat-config | grep -Po
'^\h*([^#\n\r]+|#\h*\/[^#\n\r\h]+\.conf\b)')
      if [ -n "$l_ufwscf" ]; then # Account for systems with UFW (Not covered
by systemd-sysctl --cat-config)
         l_kpar="$(grep -Po "^\h*$l_parameter_name\b" "$l_ufwscf" | xargs)"
         l_kpar="${l_kpar//\//.}"
         [ "$l_kpar" = "$l_parameter_name" ] &&
A_out+=(["$l_kpar"]="$l_ufwscf")
      fi
```

```
      if (( ${#A_out[@]} > 0 )); then # Assess output from files and generate
output
         while IFS="=" read -r l_fkpname l_file_parameter_value; do
            l_fkpname="${l_fkpname// /}";
l_file_parameter_value="${l_file_parameter_value// /}"
            if grep -Pq -- '\b'"$l_parameter_value"'\b' <<<
"$l_file_parameter_value"; then
               a_output+=(" - \"$l_parameter_name\" is correctly set to
\"$l_file_parameter_value\"" \
               "    in \"$(printf '%s' "${A_out[@]}")\"")
            else
               a_output2+=(" - \"$l_parameter_name\" is incorrectly set to
\"$l_file_parameter_value\""
               "    in \"$(printf '%s' "${A_out[@]}")\"" \
               "    and should have a value of: \"$l_value_out\"")
            fi
         done < <(grep -Po -- "^\h*$l_parameter_name\h*=\h*\H+"
"${A_out[@]}")
      else
         a_output2+=(" - \"$l_parameter_name\" is not set in an included
file" \
         "    ** Note: \"$l_parameter_name\" May be set in a file that's
ignored by load procedure **")
      fi
   }
   l_systemdsysctl="$(readlink -f /lib/systemd/systemd-sysctl)"
   while IFS="=" read -r l_parameter_name l_parameter_value; do # Assess and
check parameters
      l_parameter_name="${l_parameter_name// /}";
l_parameter_value="${l_parameter_value// /}"
      l_value_out="${l_parameter_value//-/ through }";
l_value_out="${l_value_out//|/ or }"
      l_value_out="$(tr -d '(){}' <<< "$l_value_out")"
      if grep -q '^net.ipv6.' <<< "$l_parameter_name"; then
         [ -z "$l_ipv6_disabled" ] && f_ipv6_chk
         if [ "$l_ipv6_disabled" = "yes" ]; then
            a_output+=(" - IPv6 is disabled on the system,
\"$l_parameter_name\" is not applicable")
         else
            f_kernel_parameter_chk
         fi
      else
         f_kernel_parameter_chk
      fi
   done < <(printf '%s\n' "${a_parlist[@]}")
   if [ "${#a_output2[@]}" -le 0 ]; then
      printf '%s\n' "" "- Audit Result:" "  ** PASS **" "${a_output[@]}" ""
   else
      printf '%s\n' "" "- Audit Result:" "  ** FAIL **" " - Reason(s) for
audit failure:" "${a_output2[@]}"
      [ "${#a_output[@]}" -gt 0 ] && printf '%s\n' "" "- Correctly set:"
"${a_output[@]}" ""
   fi
}
```

Remediation:

Set the following parameters in `/etc/sysctl.conf` or a file in `/etc/sysctl.d/` ending in `.conf`:

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

Example:

```
# printf '%s\n' "net.ipv4.conf.all.accept_source_route = 0"
"net.ipv4.conf.default.accept_source_route = 0" >> /etc/sysctl.d/60-
netipv4_sysctl.conf
```

Run the following script to set the active kernel parameters:

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

{
   sysctl -w net.ipv4.conf.all.accept_source_route=0
   sysctl -w net.ipv4.conf.default.accept_source_route=0
   sysctl -w net.ipv4.route.flush=1
}
```

- IF - IPv6 is enabled on the system:
Set the following parameters in `/etc/sysctl.conf` or a file in `/etc/sysctl.d/` ending in `.conf`:

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

Example:

```
# printf '%s\n' "net.ipv6.conf.all.accept_source_route = 0"
"net.ipv6.conf.default.accept_source_route = 0" >> /etc/sysctl.d/60-
netipv6_sysctl.conf
```

Run the following command to set the active kernel parameters:

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

{
   sysctl -w net.ipv6.conf.all.accept_source_route=0
   sysctl -w net.ipv6.conf.default.accept_source_route=0
   sysctl -w net.ipv6.route.flush=1
}
```

Note: If these settings appear in a canonically later file, or later in the same file, these settings will be overwritten

Default Value:

net.ipv4.conf.all.accept_source_route = 0

net.ipv4.conf.default.accept_source_route = 0

net.ipv6.conf.all.accept_source_route = 0

net.ipv6.conf.default.accept_source_route = 0

References:

1. NIST SP 800-53 Rev. 5: CM-1, CM-2, CM-6, CM-7, IA-5

Additional Information:

On systems with Uncomplicated Firewall, additional settings may be configured in `/etc/ufw/sysctl.conf`

- The settings in `/etc/ufw/sysctl.conf` will override settings in `/etc/sysctl.conf`
- This behavior can be changed by updating the `IPT_SYSCTL` parameter in `/etc/default/ufw`

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.8 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.		●	●
v7	9.2 Ensure Only Approved Ports, Protocols and Services Are Running Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1590, T1590.005	TA0007	

3.3.9 Ensure suspicious packets are logged (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

When enabled, this feature logs packets with un-routable source addresses to the kernel log.

Rationale:

Setting `net.ipv4.conf.all.log_martians` and `net.ipv4.conf.default.log_martians` to `1` enables this feature. Logging these packets allows an administrator to investigate the possibility that an attacker is sending spoofed packets to their system.

Audit:

Run the following script to verify the following kernel parameters are set in the running configuration and correctly loaded from a kernel parameter configuration file:

- `net.ipv4.conf.all.log_martians` is set to `1`
- `net.ipv4.conf.default.log_martians` is set to `1`

Note: kernel parameters are loaded by file and parameter order precedence. The following script observes this precedence as part of the auditing procedure. The parameters being checked may be set correctly in a file. If that file is superseded, the parameter is overridden by an incorrect setting later in that file, or in a canonically later file, that "correct" setting will be ignored both by the script and by the system during a normal kernel parameter load sequence.

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

{
   a_output=(); a_output2=(); l_ipv6_disabled=""
   a_parlist=("net.ipv4.conf.all.log_martians=1"
"net.ipv4.conf.default.log_martians=1")
   l_ufwscf="$([ -f /etc/default/ufw ] && awk -F= '/^\s*IPT_SYSCTL=/ {print
$2}' /etc/default/ufw)"
   f_ipv6_chk()
   {
      l_ipv6_disabled="no"
      ! grep -Pqs -- '^\h*0\b' /sys/module/ipv6/parameters/disable &&
l_ipv6_disabled="yes"
      if sysctl net.ipv6.conf.all.disable_ipv6 | grep -Pqs --
"^\h*net\.ipv6\.conf\.all\.disable_ipv6\h*=\h*1\b" && \
         sysctl net.ipv6.conf.default.disable_ipv6 | grep -Pqs --
"^\h*net\.ipv6\.conf\.default\.disable_ipv6\h*=\h*1\b"; then
         l_ipv6_disabled="yes"
      fi
   }
   f_kernel_parameter_chk()
   {
      l_running_parameter_value="$(sysctl "$l_parameter_name" | awk -F=
'{print $2}' | xargs)" # Check running configuration
      if grep -Pq -- '\b'"$l_parameter_value"'\b' <<<
"$l_running_parameter_value"; then
         a_output+=(" - \"$l_parameter_name\" is correctly set to
\"$l_running_parameter_value\""
         "    in the running configuration")
      else
         a_output2+=(" - \"$l_parameter_name\" is incorrectly set to
\"$l_running_parameter_value\"" \
         "    in the running configuration" \
         "    and should have a value of: \"$l_value_out\"")
      fi
      unset A_out; declare -A A_out # Check durable setting (files)
      while read -r l_out; do
         if [ -n "$l_out" ]; then
            if [[ $l_out =~ ^\s*# ]]; then
               l_file="${l_out//# /}"
            else
               l_kpar="$(awk -F= '{print $1}' <<< "$l_out" | xargs)"
               [ "$l_kpar" = "$l_parameter_name" ] &&
A_out+=(["$l_kpar"]="$l_file")
            fi
         fi
      done < <("$l_systemdsysctl" --cat-config | grep -Po
'^\h*([^#\n\r]+|#\h*\/[^#\n\r\h]+\.conf\b)')
      if [ -n "$l_ufwscf" ]; then # Account for systems with UFW (Not covered
by systemd-sysctl --cat-config)
         l_kpar="$(grep -Po "^\h*$l_parameter_name\b" "$l_ufwscf" | xargs)"
         l_kpar="${l_kpar//\//.}"
         [ "$l_kpar" = "$l_parameter_name" ] &&
A_out+=(["$l_kpar"]="$l_ufwscf")
      fi
      if (( ${#A_out[@]} > 0 )); then # Assess output from files and generate
output
```

```
         while IFS="=" read -r l_fkpname l_file_parameter_value; do
            l_fkpname="${l_fkpname// /}";
l_file_parameter_value="${l_file_parameter_value// /}"
            if grep -Pq -- '\b'"$l_parameter_value"'\b' <<<
"$l_file_parameter_value"; then
               a_output+=(" - \"$l_parameter_name\" is correctly set to
\"$l_file_parameter_value\"" \
               "    in \"$(printf '%s' "${A_out[@]}")\"")
            else
               a_output2+=(" - \"$l_parameter_name\" is incorrectly set to
\"$l_file_parameter_value\""
               "    in \"$(printf '%s' "${A_out[@]}")\"" \
               "    and should have a value of: \"$l_value_out\"")
            fi
         done < <(grep -Po -- "^\h*$l_parameter_name\h*=\h*\H+"
"${A_out[@]}")
      else
         a_output2+=(" - \"$l_parameter_name\" is not set in an included
file" \
         "    ** Note: \"$l_parameter_name\" May be set in a file that's
ignored by load procedure **")
      fi
   }
   l_systemdsysctl="$(readlink -f /lib/systemd/systemd-sysctl)"
   while IFS="=" read -r l_parameter_name l_parameter_value; do # Assess and
check parameters
      l_parameter_name="${l_parameter_name// /}";
l_parameter_value="${l_parameter_value// /}"
      l_value_out="${l_parameter_value//-/ through }";
l_value_out="${l_value_out//|/ or }"
      l_value_out="$(tr -d '(){}' <<< "$l_value_out")"
      if grep -q '^net.ipv6.' <<< "$l_parameter_name"; then
         [ -z "$l_ipv6_disabled" ] && f_ipv6_chk
         if [ "$l_ipv6_disabled" = "yes" ]; then
            a_output+=(" - IPv6 is disabled on the system,
\"$l_parameter_name\" is not applicable")
         else
            f_kernel_parameter_chk
         fi
      else
         f_kernel_parameter_chk
      fi
   done < <(printf '%s\n' "${a_parlist[@]}")
   if [ "${#a_output2[@]}" -le 0 ]; then
      printf '%s\n' "" "- Audit Result:" "  ** PASS **" "${a_output[@]}" ""
   else
      printf '%s\n' "" "- Audit Result:" "  ** FAIL **" " - Reason(s) for
audit failure:" "${a_output2[@]}"
      [ "${#a_output[@]}" -gt 0 ] && printf '%s\n' "" "- Correctly set:"
"${a_output[@]}" ""
   fi
}
```

Remediation:

Set the following parameters in `/etc/sysctl.conf` or a file in `/etc/sysctl.d/` ending in `.conf`:

- `net.ipv4.conf.all.log_martians = 1`
- `net.ipv4.conf.default.log_martians = 1`

Example:

```
# printf '%s\n' "net.ipv4.conf.all.log_martians = 1"
"net.ipv4.conf.default.log_martians = 1" >> /etc/sysctl.d/60-
netipv4_sysctl.conf
```

Run the following script to set the active kernel parameters:

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

{
   sysctl -w net.ipv4.conf.all.log_martians=1
   sysctl -w net.ipv4.conf.default.log_martians=1
   sysctl -w net.ipv4.route.flush=1
}
```

Note: If these settings appear in a canonically later file, or later in the same file, these settings will be overwritten

Default Value:

net.ipv4.conf.all.log_martians = 0

net.ipv4.conf.default.log_martians = 0

References:

1. NIST SP 800-53 Rev. 5: AU-3

Additional Information:

On systems with Uncomplicated Firewall, additional settings may be configured in `/etc/ufw/sysctl.conf`

- The settings in `/etc/ufw/sysctl.conf` will override settings in `/etc/sysctl.conf`
- This behavior can be changed by updating the `IPT_SYSCTL` parameter in `/etc/default/ufw`

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	8.5 Collect Detailed Audit Logs Configure detailed audit logging for enterprise assets containing sensitive data. Include event source, date, username, timestamp, source addresses, destination addresses, and other useful elements that could assist in a forensic investigation.		●	●
v7	6.2 Activate audit logging Ensure that local logging has been enabled on all systems and networking devices.	●	●	●
v7	6.3 Enable Detailed Logging Enable system logging to include detailed information such as an event source, date, user, timestamp, source addresses, destination addresses, and other useful elements.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1562, T1562.006	TA0005	

3.3.10 Ensure tcp syn cookies is enabled (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

When `tcp_syncookies` is set, the kernel will handle TCP SYN packets normally until the half-open connection queue is full, at which time, the SYN cookie functionality kicks in. SYN cookies work by not using the SYN queue at all. Instead, the kernel simply replies to the SYN with a SYN/ACK, but will include a specially crafted TCP sequence number that encodes the source and destination IP address and port number and the time the packet was sent. A legitimate connection would send the ACK packet of the three way handshake with the specially crafted sequence number. This allows the system to verify that it has received a valid response to a SYN cookie and allow the connection, even though there is no corresponding SYN in the queue.

Rationale:

Attackers use SYN flood attacks to perform a denial of service attacked on a system by sending many SYN packets without completing the three way handshake. This will quickly use up slots in the kernel's half-open connection queue and prevent legitimate connections from succeeding. Setting `net.ipv4.tcp_syncookies` to `1` enables SYN cookies, allowing the system to keep accepting valid connections, even if under a denial of service attack.

Audit:

Run the following script to verify the following kernel parameter is set in the running configuration and correctly loaded from a kernel parameter configuration file:

- `net.ipv4.tcp_syncookies` is set to `1`

Note: kernel parameters are loaded by file and parameter order precedence. The following script observes this precedence as part of the auditing procedure. The parameters being checked may be set correctly in a file. If that file is superseded, the parameter is overridden by an incorrect setting later in that file, or in a canonically later file, that "correct" setting will be ignored both by the script and by the system during a normal kernel parameter load sequence.

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

{
   a_output=(); a_output2=(); l_ipv6_disabled=""
   a_parlist=("net.ipv4.tcp_syncookies=1")
   l_ufwscf="$([ -f /etc/default/ufw ] && awk -F= '/^\s*IPT_SYSCTL=/ {print
$2}' /etc/default/ufw)"
   f_ipv6_chk()
   {
      l_ipv6_disabled="no"
      ! grep -Pqs -- '^\h*0\b' /sys/module/ipv6/parameters/disable &&
l_ipv6_disabled="yes"
      if sysctl net.ipv6.conf.all.disable_ipv6 | grep -Pqs --
"^\h*net\.ipv6\.conf\.all\.disable_ipv6\h*=\h*1\b" && \
         sysctl net.ipv6.conf.default.disable_ipv6 | grep -Pqs --
"^\h*net\.ipv6\.conf\.default\.disable_ipv6\h*=\h*1\b"; then
         l_ipv6_disabled="yes"
      fi
   }
   f_kernel_parameter_chk()
   {
      l_running_parameter_value="$(sysctl "$l_parameter_name" | awk -F=
'{print $2}' | xargs)" # Check running configuration
      if grep -Pq -- '\b'"$l_parameter_value"'\b' <<<
"$l_running_parameter_value"; then
         a_output+=(" - \"$l_parameter_name\" is correctly set to
\"$l_running_parameter_value\""
         "    in the running configuration")
      else
         a_output2+=(" - \"$l_parameter_name\" is incorrectly set to
\"$l_running_parameter_value\"" \
         "    in the running configuration" \
         "    and should have a value of: \"$l_value_out\"")
      fi
      unset A_out; declare -A A_out # Check durable setting (files)
      while read -r l_out; do
         if [ -n "$l_out" ]; then
            if [[ $l_out =~ ^\s*# ]]; then
               l_file="${l_out//# /}"
            else
               l_kpar="$(awk -F= '{print $1}' <<< "$l_out" | xargs)"
               [ "$l_kpar" = "$l_parameter_name" ] &&
A_out+=(["$l_kpar"]="$l_file")
            fi
         fi
      done < <("$l_systemdsysctl" --cat-config | grep -Po
'^\h*([^#\n\r]+|#\h*\/[^#\n\r\h]+\.conf\b)')
      if [ -n "$l_ufwscf" ]; then # Account for systems with UFW (Not covered
by systemd-sysctl --cat-config)
         l_kpar="$(grep -Po "^\h*$l_parameter_name\b" "$l_ufwscf" | xargs)"
         l_kpar="${l_kpar//\//.}"
         [ "$l_kpar" = "$l_parameter_name" ] &&
A_out+=(["$l_kpar"]="$l_ufwscf")
      fi
      if (( ${#A_out[@]} > 0 )); then # Assess output from files and generate
output
         while IFS="=" read -r l_fkpname l_file_parameter_value; do
```

```
            l_fkpname="${l_fkpname// /}";
l_file_parameter_value="${l_file_parameter_value// /}"
            if grep -Pq -- '\b'"$l_parameter_value"'\b' <<<
"$l_file_parameter_value"; then
               a_output+=(" - \"$l_parameter_name\" is correctly set to
\"$l_file_parameter_value\"" \
               "    in \"$(printf '%s' "${A_out[@]}")\"")
            else
               a_output2+=(" - \"$l_parameter_name\" is incorrectly set to
\"$l_file_parameter_value\""
               "    in \"$(printf '%s' "${A_out[@]}")\"" \
               "    and should have a value of: \"$l_value_out\"")
            fi
         done < <(grep -Po -- "^\h*$l_parameter_name\h*=\h*\H+"
"${A_out[@]}")
      else
         a_output2+=(" - \"$l_parameter_name\" is not set in an included
file" \
         "    ** Note: \"$l_parameter_name\" May be set in a file that's
ignored by load procedure **")
      fi
   }
   l_systemdsysctl="$(readlink -f /lib/systemd/systemd-sysctl)"
   while IFS="=" read -r l_parameter_name l_parameter_value; do # Assess and
check parameters
      l_parameter_name="${l_parameter_name// /}";
l_parameter_value="${l_parameter_value// /}"
      l_value_out="${l_parameter_value//-/ through }";
l_value_out="${l_value_out//|/ or }"
      l_value_out="$(tr -d '(){}' <<< "$l_value_out")"
      if grep -q '^net.ipv6.' <<< "$l_parameter_name"; then
         [ -z "$l_ipv6_disabled" ] && f_ipv6_chk
         if [ "$l_ipv6_disabled" = "yes" ]; then
            a_output+=(" - IPv6 is disabled on the system,
\"$l_parameter_name\" is not applicable")
         else
            f_kernel_parameter_chk
         fi
      else
         f_kernel_parameter_chk
      fi
   done < <(printf '%s\n' "${a_parlist[@]}")
   if [ "${#a_output2[@]}" -le 0 ]; then
      printf '%s\n' "" "- Audit Result:" "  ** PASS **" "${a_output[@]}" ""
   else
      printf '%s\n' "" "- Audit Result:" "  ** FAIL **" " - Reason(s) for
audit failure:" "${a_output2[@]}"
      [ "${#a_output[@]}" -gt 0 ] && printf '%s\n' "" "- Correctly set:"
"${a_output[@]}" ""
   fi
}
```

Remediation:

Set the following parameter in `/etc/sysctl.conf` or a file in `/etc/sysctl.d/` ending in `.conf`:

- `net.ipv4.tcp_syncookies = 1`

Example:

```
# printf '%s\n' "net.ipv4.tcp_syncookies = 1" >> /etc/sysctl.d/60-
netipv4_sysctl.conf
```

Run the following script to set the active kernel parameters:

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

{
   sysctl -w net.ipv4.tcp_syncookies=1
   sysctl -w net.ipv4.route.flush=1
}
```

Note: If these settings appear in a canonically later file, or later in the same file, these settings will be overwritten

Default Value:

net.ipv4.tcp_syncookies = 1

References:

1. NIST SP 800-53 Rev. 5: CM-1,CM-2, CM-6, CM-7, IA-5

Additional Information:

On systems with Uncomplicated Firewall, additional settings may be configured in `/etc/ufw/sysctl.conf`

- The settings in `/etc/ufw/sysctl.conf` will override settings in `/etc/sysctl.conf`
- This behavior can be changed by updating the `IPT_SYSCTL` parameter in `/etc/default/ufw`

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.8 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.		●	●
v7	9.2 Ensure Only Approved Ports, Protocols and Services Are Running Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1499, T1499.001	TA0040	M1037