6.3 Configure Integrity Checking

AIDE is a file integrity checking tool, similar in nature to Tripwire. While it cannot prevent intrusions, it can detect unauthorized changes to configuration files by alerting when the files are changed. When setting up AIDE, decide internally what the site policy will be concerning integrity checking. Review the AIDE quick start guide and AIDE documentation before proceeding.

6.3.1 Ensure AIDE is installed (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

AIDE takes a snapshot of filesystem state including modification times, permissions, and file hashes which can then be used to compare against the current state of the filesystem to detect modifications to the system.

Rationale:

By monitoring the filesystem state compromised files can be detected to prevent or limit the exposure of accidental or malicious misconfigurations or modified binaries.

Audit:

Run the following command to verify `aide` is installed:

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

aide is installed
```

Run the following command to verify `aide-common` is installed:

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

aide-common is installed
```

Remediation:

Install AIDE using the appropriate package manager or manual installation:

```
# apt install aide aide-common
```

Configure AIDE as appropriate for your environment. Consult the AIDE documentation for options.
Run the following commands to initialize AIDE:

```
# aideinit
# mv /var/lib/aide/aide.db.new /var/lib/aide/aide.db
```

References:

1. NIST SP 800-53 Rev. 5: AU-2

Additional Information:

The prelinking feature can interfere with AIDE because it alters binaries to speed up their start up times. Run `prelink -ua` to restore the binaries to their prelinked state, thus avoiding false positives from AIDE.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.14 Log Sensitive Data Access Log sensitive data access, including modification and disposal.			●
v7	14.9 Enforce Detail Logging for Access or Changes to Sensitive Data Enforce detailed audit logging for access to sensitive data or changes to sensitive data (utilizing tools such as File Integrity Monitoring or Security Information and Event Monitoring).			●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1565, T1565.001	TA0001	M1022

6.3.2 Ensure filesystem integrity is regularly checked (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

Periodic checking of the filesystem integrity is needed to detect changes to the filesystem.

Rationale:

Periodic file checking allows the system administrator to determine on a regular basis if critical files have been changed in an unauthorized fashion.

Audit:

Run the following command:

```
# systemctl list-unit-files | awk
'$1~/^dailyaidecheck\.(timer|service)$/{print $1 "\t" $2}'
```

Example output:

```
dailyaidecheck.service  static
dailyaidecheck.timer    enabled
```

Verify `dailyaidecheck.timer` is `enabled` and `dailyaidecheck.service` is either `static` or `enabled`.
Run the following command to verify `dailyaidecheck.timer` is `active`:

```
# systemctl is-active dailyaidecheck.timer

active
```

Remediation:

Run the following command to unmask `dailyaidecheck.timer` and `dailyaidecheck.service`:

```
# systemctl unmask dailyaidecheck.timer dailyaidecheck.service
```

Run the following command to enable and start `dailyaidecheck.timer`:

```
# systemctl --now enable dailyaidecheck.timer
```

References:

1. https://github.com/konstruktoid/hardening/blob/master/config/aidecheck.service
2. https://github.com/konstruktoid/hardening/blob/master/config/aidecheck.timer
3. NIST SP 800-53 Rev. 5: AU-2

Additional Information:

The checking in this recommendation occurs every day at 5am. Alter the frequency and time of the checks in compliance with site policy

systemd timers, timer file `aidecheck.timer` and service file `aidecheck.service`, have been included as an optional alternative to using `cron`

Ubuntu advises using `/usr/bin/aide.wrapper` rather than calling `/usr/bin/aide` directly in order to protect the database and prevent conflicts

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	14.9 Enforce Detail Logging for Access or Changes to Sensitive Data Enforce detailed audit logging for access to sensitive data or changes to sensitive data (utilizing tools such as File Integrity Monitoring or Security Information and Event Monitoring).			●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1036, T1036.002, T1036.003, T1036.004, T1036.005, T1565, T1565.001	TA0040	M1022

6.3.3 Ensure cryptographic mechanisms are used to protect the integrity of audit tools (Automated)

Profile Applicability:

- Level 2 - Server
- Level 2 - Workstation

Description:

Audit tools include, but are not limited to, vendor-provided and open source audit tools needed to successfully view and manipulate audit information system activity and records. Audit tools include custom queries and report generators.

`aide.conf` is case-sensitive. Leading and trailing white spaces are ignored. Each config lines must end with new line.

`AIDE` uses the backslash character `\` as escape character for ' ' (space), '@' and '' (backslash) (e.g. '\ ' or '@'). To literally match a '' in a file path with a regular expression you have to escape the backslash twice (i.e. '\\').

There are three types of lines in `aide.conf`:

- The configuration options which are used to set configuration parameters and define groups.
- (restricted) rules that are used to indicate which files are added to the database.
- Macro lines define or undefine variables within the config file.

Note: Lines beginning with `#` are ignored as comments.
`@@include` <*FILE*> - Include <*FILE*>.

- The content of the file is used as if it were inserted in this part of the config file.
- The maximum depth of nested includes is 16.

`@@include <*DIRECTORY*> <*REGEX*> - [RULE_PREFIX] (added in AIDE v0.17)

- Include all (regular) files found in <DIRECTORY> matching regular expression <REGEX> (sub-directories are ignored).
- The file are included in lexical sort order.
- If `RULE_PREFIX` is set, all rules included by the statement are prefixed with given <RULE_PREFIX> (added in AIDE v0.18). Prefixes from nested include statements are concatenated.
- The content of the files is used as if it were inserted in this part of the config file.

`@x_include`:

- is identical to `@@include`, except that if a config file is executable is is run and the output is used as config.
- If the executable file exits with status greater than zero or writes to stderr aide stops with an error.
- For security reasons <*DIRECTORY*> and each executable config file must be owned by the current user or root. They must not be group- or world-writable.
- `@@x_include _<FILE>_` (added in AIDE v0.17):

- `@@x_include <*DIRECTORY*> <*REGEX*> [RULE_PREFIX] (added in AIDE v0.17)

`@@x_include_setenv` <*VAR*> <*VALUE*> (added in AIDE v0.17)

- Adds the variable <*VAR*> with the value <*VALUE*> to the environment used for config file execution.
- Environment variable names are limited to alphanumeric characters (A-Za-z0-9) and the underscore '_' and must not begin with a digit.

Rationale:

Protecting the integrity of the tools used for auditing purposes is a critical step toward ensuring the integrity of audit information. Audit information includes all information (e.g., audit records, audit settings, and audit reports) needed to successfully audit information system activity.

Attackers may replace the audit tools or inject code into the existing tools with the purpose of providing the capability to hide or erase system activity from the audit logs.

Audit tools should be cryptographically signed in order to provide the capability to identify when the audit tools have been modified, manipulated, or replaced. An example is a checksum hash of the file or files.

Audit:

Verify that Advanced Intrusion Detection Environment (AIDE) is properly configured. Run the following script to verify:

- AIDE is configured to use cryptographic mechanisms to protect the integrity of audit tools:
- The following audit tool files include the options "p, i, n, u, g, s, b, acl, xattrs and sha512"
    - auditctl
    - auditd
    - ausearch
    - aureport
    - autrace
    - augenrules

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

{
   a_output=() a_output2=() l_tool_dir="$(readlink -f /sbin)"
   a_items=("p" "i" "n" "u" "g" "s" "b" "acl" "xattrs" "sha512")
   l_aide_cmd="$(whereis aide | awk '{print $2}')"
   a_audit_files=("auditctl" "auditd" "ausearch" "aureport" "autrace"
"augenrules")
   if [ -f "$l_aide_cmd" ] && command -v "$l_aide_cmd" &>/dev/null; then
      a_aide_conf_files=("$(find -L /etc -type f -name 'aide.conf')")
      f_file_par_chk()
      {
         a_out2=()
         for l_item in "${a_items[@]}"; do
            ! grep -Psiq -- '(\h+|\+)'"$l_item"'(\h+|\+)' <<< "$l_out" && \
            a_out2+=("  - Missing the \"$l_item\" option")
         done
         if [ "${#a_out2[@]}" -gt "0" ]; then
            a_output2+=(" - Audit tool file: \"$l_file\"" "${a_out2[@]}")
         else
            a_output+=(" - Audit tool file: \"$l_file\" includes:" "
\"${a_items[*]}\"")
         fi
      }
      for l_file in "${a_audit_files[@]}"; do
         if [ -f "$l_tool_dir/$l_file" ]; then
            l_out="$("$l_aide_cmd" --config "${a_aide_conf_files[@]}" -p
f:"$l_tool_dir/$l_file")"
            f_file_par_chk
         else
            a_output+=("  - Audit tool file \"$l_file\" doesn't exist")
         fi
      done
   else
      a_output2+=("  - The command \"aide\" was not found"  "    Please
install AIDE")
   fi
   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
}
```

Note: The script is written to read the "winning" configuration setting, to include any configuration settings in files included as part of the `@@x_include` setting.

Remediation:

Run the following command to determine the absolute path to the non-symlinked version on the audit tools:

```
# readlink -f /sbin
```

The output will be either `/usr/sbin` **- OR -** `/sbin`. Ensure the correct path is used. Edit `/etc/aide/aide.conf` and add or update the following selection lines replacing `<PATH>` with the correct path returned in the command above:

```
# Audit Tools
<PATH>/auditctl p+i+n+u+g+s+b+acl+xattrs+sha512
<PATH>/auditd p+i+n+u+g+s+b+acl+xattrs+sha512
<PATH>/ausearch p+i+n+u+g+s+b+acl+xattrs+sha512
<PATH>/aureport p+i+n+u+g+s+b+acl+xattrs+sha512
<PATH>/autrace p+i+n+u+g+s+b+acl+xattrs+sha512
<PATH>/augenrules p+i+n+u+g+s+b+acl+xattrs+sha512
```

Example

```
# printf '%s\n' "" "# Audit Tools" "$(readlink -f /sbin/auditctl)
p+i+n+u+g+s+b+acl+xattrs+sha512" "$(readlink -f /sbin/auditd)
p+i+n+u+g+s+b+acl+xattrs+sha512" "$(readlink -f /sbin/ausearch)
p+i+n+u+g+s+b+acl+xattrs+sha512" "$(readlink -f /sbin/aureport)
p+i+n+u+g+s+b+acl+xattrs+sha512" "$(readlink -f /sbin/autrace)
p+i+n+u+g+s+b+acl+xattrs+sha512" "$(readlink -f /sbin/augenrules)
p+i+n+u+g+s+b+acl+xattrs+sha512" >> /etc/aide/aide.conf
```

Note: - IF - `/etc/aide/aide.conf` includes a `@@x_include` statement:

- *<DIRECTORY>* and each executable config file must be owned by the current user or root
- They must not be group or world-writable

Example:

```
@@x_include /etc/aide.conf.d ^[a-zA-Z0-9_-]+$
```

References:

1. AIDE.CONF(5)

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1070, T1070.002, T1083, T1083.000	TA0007	