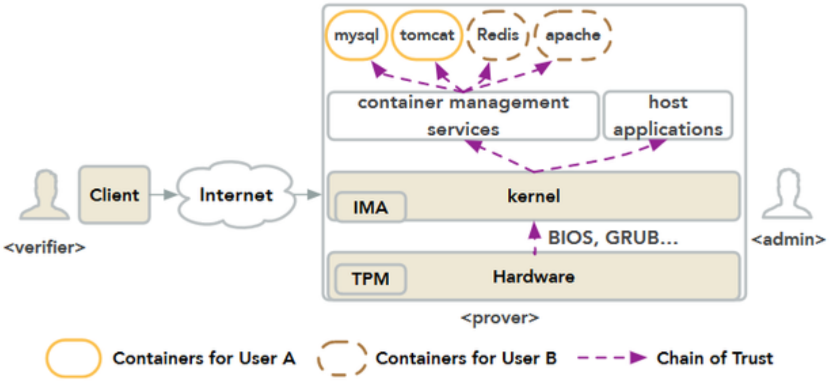
# Enabling Linux IMA with Policies Aligned to CIS Ubuntu 20.04 LTS v2.0.1 and Kubernetes v1.7.1 Benchmarks

Linux IMA is a subsystem within the Linux kernel that can be used to measure, store, and appraise the hashes of files before they are accessed. Linux IMA introduces hooks within the Linux kernel to support creating and collecting hashes of files when opened, before their contents are accessed for read or execute. Linux IMA uses measurements, default and custom policies, along with varying grub and configuration parameters to set filesystem policies. The IMA measurement subsystem can detect if a file has been altered accidentally or maliciously both remotely and or locally (Red Hat, 2020). These logged values can be accessed as part of an audit process. Additionally, access requests can be blocked based on configurable appraisal policies further limiting access if there is file tampering.

For this project, the build architecture is based on vCenter atop a physical Dell rack where a vSphere VM Kubernetes cluster has been created housing the control and worker nodes, each with vTPMs. The aim is to prove that IMA policies can be created with Ubuntu and Kubernetes benchmark assurance. The idea is to hand off the policy and implementation configurations to SBP or VMware colleagues who can develop capabilities for additional benchmarks.



IMA performs the measurements above the TPM at the kernel level. Container management services such as Kubernetes can manage various containers with different images such Ubuntu 20.04 in this case (Luo, et al., 2019, p. 489).

## Assuring Ubuntu 20.04 LTS Benchmark with IMA File Integrity Measurement and Appraisal

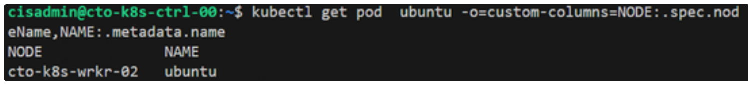
This part of the project will focus on the IMA measurements in the worker node (cto-k8s-wrkr-02) where the “ubuntu” pod is running. All nodes, including worker nodes are running the Ubuntu 20.04 LTS Server OS. The user is able to make changes to the kernel configuration file where IMA hooks are instantiated, influencing the extension upon runtime detected change to the vTPM’s PCR 10 hash value. The CIS benchmark for Ubuntu Linux 20.04 LTS indicates in Section 1.2, recommendation number 1.2.2 to “ensure filesystem integrity is regularly checked”. The benchmark recommends installing 3rd-party Github downloadable to “detect unauthorized changes to configuration files by altering when files are changed” and “AIDE is the file integrity checking tool, similar to tripwire” (Ubuntu 20.04 LTS v2.0.1).

Reflecting on SLTT capabilities and the want for out-of-the-box low-cost capabilities, it is asserted that built-in IMA filesystem integrity checking is superior to AIDE for several reasons. First, IMA is built in and “was introduced with the Linux 2.6.30 kernel is 2009” making it widely available (Stefen, 2012, p. 4). TPM PCR values are tied to IMA measurements and provide a level of machine integrity that enables expansion to remote attesting along with onboard key generation and protection capabilities.

This guide instructs the user to enable IMA, then offers proper filesystem integrity policy configurations on an Ubuntu 20.04 imaged Kubernetes worker node. Integrity logs can be compared with previous values to demonstrate changes along with audit log entries where enabled. Further, IMA appraise can enforce file checking, comparing the file hash against a known stored value. Automatic denial of access rules can be set therewith non-matching hash rules.

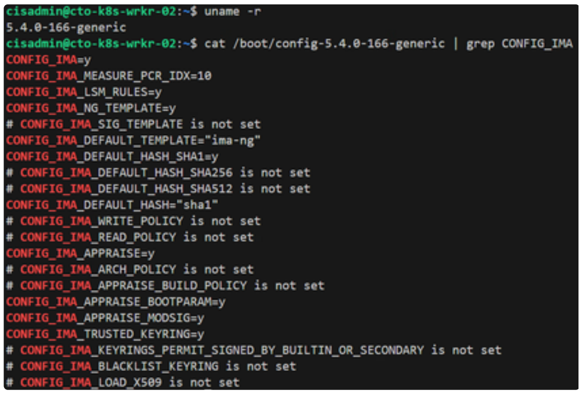
# Enabling IMA

1. Check which worker node the pod “ubuntu” is located for testing purposes. Do this by running “ kubectl get pod  ubuntu -o=custom-columns=NODE:.spec.nodeName,NAME:.metadata.name” inside the control plane.



```
cisadmin@cto-k8s-ctrl-00:~$ kubectl get pod  ubuntu -o=custom-columns=NODE:.spec.nod
eName,NAME:.metadata.name
NODE                 NAME
cto-k8s-wrkr-02      ubuntu
```

2. To determine if IMA is enabled, confirm the kernel version by running “uname -r” and then “cat /boot/config-5.4.0-166-generic | grep CONFIG_IMA”. If “CONFIG_IMA=y”, then IMA is enabled. The kernel name is used to locate the config file.

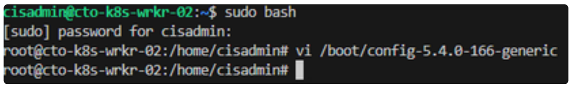


```
cisadmin@cto-k8s-wrkr-02:~$ uname -r
5.4.0-166-generic
cisadmin@cto-k8s-wrkr-02:~$ cat /boot/config-5.4.0-166-generic | grep CONFIG_IMA
CONFIG_IMA=y
CONFIG_IMA_MEASURE_PCR_IDX=10
CONFIG_IMA_LSM_RULES=y
CONFIG_IMA_NG_TEMPLATE=y
# CONFIG_IMA_SIG_TEMPLATE is not set
CONFIG_IMA_DEFAULT_TEMPLATE="ima-ng"
CONFIG_IMA_DEFAULT_HASH_SHA1=y
# CONFIG_IMA_DEFAULT_HASH_SHA256 is not set
# CONFIG_IMA_DEFAULT_HASH_SHA512 is not set
CONFIG_IMA_DEFAULT_HASH="sha1"
# CONFIG_IMA_WRITE_POLICY is not set
# CONFIG_IMA_READ_POLICY is not set
CONFIG_IMA_APPRAISE=y
# CONFIG_IMA_ARCH_POLICY is not set
# CONFIG_IMA_APPRAISE_BUILD_POLICY is not set
CONFIG_IMA_APPRAISE_BOOTPARAM=y
CONFIG_IMA_APPRAISE_MODSIG=y
CONFIG_IMA_TRUSTED_KEYRING=y
# CONFIG_IMA_KEYRINGS_PERMIT_SIGNED_BY_BUILTIN_OR_SECONDARY is not set
# CONFIG_IMA_BLACKLIST_KEYRING is not set
# CONFIG_IMA_LOAD_X509 is not set
```

# Updating Configurations Needed for IMA Support

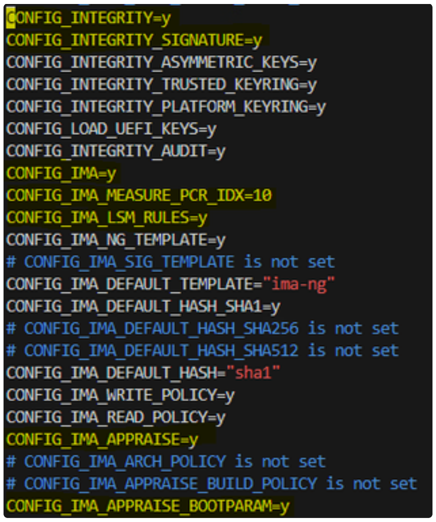
## Kernel Configuration File

1. Replace the lines in the kernel configuration file as such. However, many systems may already have some configurations set. Note that the user must be root to write to the file.



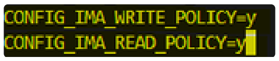
```
cisadmin@cto-k8s-wrkr-02:~$ sudo bash
[sudo] password for cisadmin:
root@cto-k8s-wrkr-02:/home/cisadmin# vi /boot/config-5.4.0-166-generic
root@cto-k8s-wrkr-02:/home/cisadmin# 
```

```
vi /boot/config-5.4.0-166-generic
CONFIG_INTEGRITY=y
CONFIG_IMA=y
CONFIG_IMA_MEASURE_PCR_IDX=10
CONFIG_IMA_LSM_RULES=y
CONFIG_INTEGRITY_SIGNATURE=y
CONFIG_IMA_APPRAISE=y
--
# Since 4.13
IMA_APPRAISE_BOOTPARAM=y
--
```



```
CONFIG_INTEGRITY=y
CONFIG_INTEGRITY_SIGNATURE=y
CONFIG_INTEGRITY_ASYMMETRIC_KEYS=y
CONFIG_INTEGRITY_TRUSTED_KEYRING=y
CONFIG_INTEGRITY_PLATFORM_KEYRING=y
CONFIG_LOAD_UEFI_KEYS=y
CONFIG_INTEGRITY_AUDIT=y
CONFIG_IMA=y
CONFIG_IMA_MEASURE_PCR_IDX=10
CONFIG_IMA_LSM_RULES=y
CONFIG_IMA_NG_TEMPLATE=y
# CONFIG_IMA_SIG_TEMPLATE is not set
CONFIG_IMA_DEFAULT_TEMPLATE="ima-ng"
CONFIG_IMA_DEFAULT_HASH_SHA1=y
# CONFIG_IMA_DEFAULT_HASH_SHA256 is not set
# CONFIG_IMA_DEFAULT_HASH_SHA512 is not set
CONFIG_IMA_DEFAULT_HASH="sha1"
CONFIG_IMA_WRITE_POLICY=y
CONFIG_IMA_READ_POLICY=y
CONFIG_IMA_APPRAISE=y
# CONFIG_IMA_ARCH_POLICY is not set
# CONFIG_IMA_APPRAISE_BUILD_POLICY is not set
CONFIG_IMA_APPRAISE_BOOTPARAM=y
```

2. While in the config file, verify the value “CONFIG_IMA_READ_POLICY=y” as this parameter is needed to for root to read the policy rules and verify “CONFIG_IMA_WRITE_POLICY=y”, enabling multiple appends to the custom IMA policy.

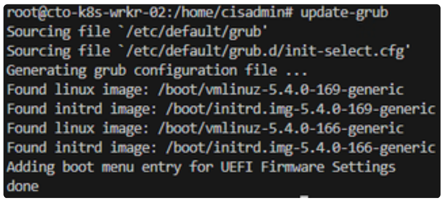


```
CONFIG_IMA_WRITE_POLICY=y
CONFIG_IMA_READ_POLICY=y
```

3. Save file and exit.

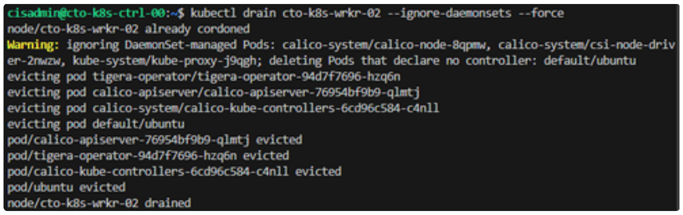
## Bootloader Configuration

1. Next, update the bootloader configuration. Add the following line to the etc/default/grub file. `GRUB_CMDLINE_LINUX="ima_tcb lsm=integrity ima_appraise=fix ima_appraise_tcb"`
2. Update grub with `update-grub`.



```
root@cto-k8s-wrkr-02:/home/cisadmin# update-grub
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.4.0-169-generic
Found initrd image: /boot/initrd.img-5.4.0-169-generic
Found linux image: /boot/vmlinuz-5.4.0-166-generic
Found initrd image: /boot/initrd.img-5.4.0-166-generic
Adding boot menu entry for UEFI Firmware Settings
done
```

3. Prepare kubernetes nodes for reboot.
   - From the control node (cto-k8s-ctrl-00), “cordon” and “drain” the Kubernetes worker node (cto-k8s-wrkr-02) by running “kubectl cordon cto-k8s-wrkr-02”. Then run “kubectl drain node”. Note that --ignore-daemonsets may be needed for daemon managed pods as in this example. Also, use --force if needed to override deleting pods.



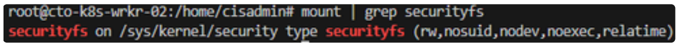
```
cisadmin@cto-k8s-ctrl-00:~$ kubectl drain cto-k8s-wrkr-02 --ignore-daemonsets --force
node/cto-k8s-wrkr-02 already cordoned
Warning: ignoring DaemonSet-managed Pods: calico-system/calico-node-8qpmw, calico-system/csi-node-driv
er-2nwzw, kube-system/kube-proxy-j9qgh; deleting Pods that declare no controller: default/ubuntu
evicting pod tigera-operator/tigera-operator-94d7f7696-hzq6n
evicting pod calico-apiserver/calico-apiserver-76954bf9b9-qlmtj
evicting pod calico-system/calico-kube-controllers-6cd96c584-c4nll
evicting pod default/ubuntu
pod/calico-apiserver-76954bf9b9-qlmtj evicted
pod/tigera-operator-94d7f7696-hzq6n evicted
pod/calico-kube-controllers-6cd96c584-c4nll evicted
pod/ubuntu evicted
node/cto-k8s-wrkr-02 drained
```

4. Reboot
5. Return to the command line for the worker node (cto-k8s-wrkr-02) and run “kubectl uncordon cto-k8s-wrkr-02”. Also, run “swapoff -a” if the node is persisting in NotReady after restart and executing uncordon.

At this point, IMA is enabled with `ima_tcb lsm=integrity ima_appraise=fix ima_appraise_tcb`. Complete the following section if `ima_appraise=enforce` is desired, causing the system to validate the hash against a stored value before using it ( Integrity Measureme nt Architecture - Gentoo wiki ).

### Enable Security File System

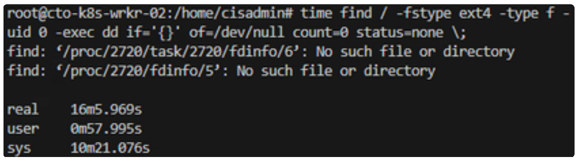
1. To confirm the security file system is mounted, run `mount | grep securityfs`



```
root@cto-k8s-wrkr-02:/home/cisadmin# mount | grep securityfs
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
```

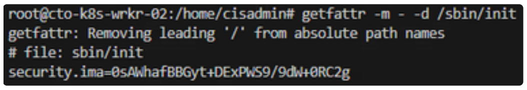
## Registering the file hashes for the system

This is only needed if `ima_appraise=enforce`



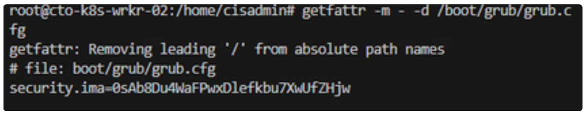
```
root@cto-k8s-wrkr-02:/home/cisadmin# time find / -fstype ext4 -type f -
uid 0 -exec dd if='{}' of=/dev/null count=0 status=none \;
find: '/proc/2720/task/2720/fdinfo/6': No such file or directory
find: '/proc/2720/fdinfo/5': No such file or directory

real    16m5.969s
user    0m57.995s
sys     10m21.076s
```



```
root@cto-k8s-wrkr-02:/home/cisadmin# getfattr -m - -d /sbin/init
getfattr: Removing leading '/' from absolute path names
# file: sbin/init
security.ima=0sAWhafBBGyt+DExPWS9/9dW+0RC2g
```

Example 1 of an IMA registered file hash value



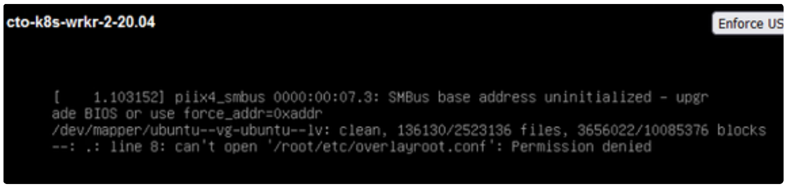
```
root@cto-k8s-wrkr-02:/home/cisadmin# getfattr -m - -d /boot/grub/grub.c
fg
getfattr: Removing leading '/' from absolute path names
# file: boot/grub/grub.cfg
security.ima=0sAb8Du4WaFPwxDlefkbu7XwUfZHjw
```

Example 2 of an IMA registered file hash

After completing this section, reboot with `ima_appraise=enforce` added to kernel `GRUB_CMDLINE_LINUX` in /etc/default/grub.

# IMA Appraisal in Action

Upon startup with `ima_appraise=enforce`, IMA will be enforcing the appraisal policy to check file hashes against stored hash value. Access is denied to the appraised file if the hash is missing or does not match. The default external IMA policy enforces appraising all the executables, shared libraries, kernel modules and firmwares with the digital signature in the effective root identity, or euid=0.



```
[    1.103152] piix4_smbus 0000:00:07.3: SMBus base address uninitialized - upgr
ade BIOS or use force_addr=0xaddr
/dev/mapper/ubuntu--vg-ubuntu--lv: clean, 136130/2523136 files, 3656022/10085376 blocks
--: .: line 8: can't open '/root/etc/overlayroot.conf': Permission denied
```

Example 1 showing permission denied error when access was requested

## Assurance of Kubernetes 1.7.1 and Ubuntu 20.04 Benchmarks using IMA Appraisal

IMA Appraisal provides the filesystem integrity checking recommended in both CIS Ubuntu and Kubernetes benchmarks. The Kubernetes Benchmark 1.71 recommends 11 times for the user in “level 1 - worker node” to maintain ownership and file permissions. This could be achieved with IMA with **appraise_tcb** which would check all files owned by root. If a file changes unexpectedly, the file will not be allowed to be opened and considered untrusted. This process is also automatic once set up.

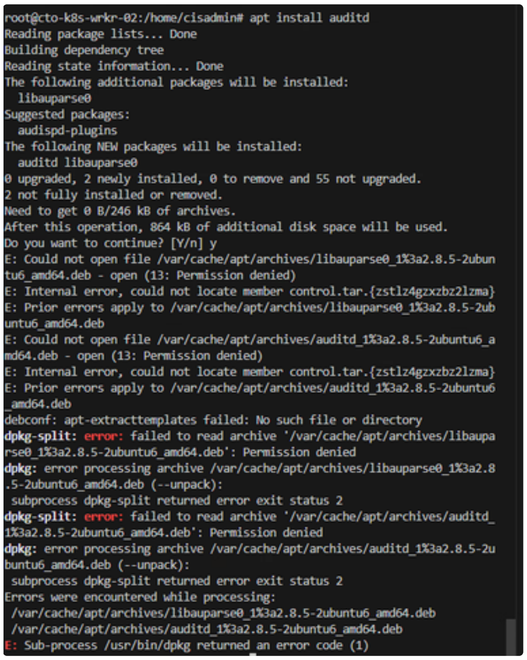
### Enabling File Appraisals

To appraise file changes, you can use IMA policies that provide logic to detect runtime changes to files and consequently updates calculated hashes. The following IMA policies can be used to appraise file changes and **ima_policy=** can take 1 of 3 values “tcb, appraise_tcb, secure_boot”:

- **tcb**: Measures all executables run, all mmap’d files for execution (such as shared libraries), all kernel modules loaded, and all firmware loaded. Additionally, all files read by root are measured as well.
- **appraise_tcb**: Appraises all files owned by root.
- **secure_boot**: Appraises all loaded modules, firmware, kexec’d kernel, and IMA policies. It also requires them to have an IMA signature as well. This is normally used with the CONFIG_INTEGRITY_TRUSTED_KEYRING option in the kernel in “secure boot” scenario, with the public key obtained from the OEM in firmware or via the MOK (Machine Owner Key) in shim.

## Confirming IMA with appraisal is working

There are several ways. The first is with a diff tool such as diffchecker.com where the user can put the runtime measurement files from 2 time periods and assess the changes. If rebooting into “enforce mode”, there are some issues accessing files, that is because the appraise policy is being enforced. Once set to “fix”, the enforcement of the appraise policy will cease.



```
root@cto-k8s-wrkr-02:/home/cisadmin# apt install auditd
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libauparse0
Suggested packages:
  audispd-plugins
The following NEW packages will be installed:
  auditd libauparse0
0 upgraded, 2 newly installed, 0 to remove and 55 not upgraded.
2 not fully installed or removed.
Need to get 0 B/246 kB of archives.
After this operation, 864 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
E: Could not open file /var/cache/apt/archives/libauparse0_1%3a2.8.5-2ubun
tu6_amd64.deb - open (13: Permission denied)
E: Internal error, could not locate member control.tar.{zstlz4gzxzbz2lzma}
E: Prior errors apply to /var/cache/apt/archives/libauparse0_1%3a2.8.5-2ub
untu6_amd64.deb
E: Could not open file /var/cache/apt/archives/auditd_1%3a2.8.5-2ubuntu6_a
md64.deb - open (13: Permission denied)
E: Internal error, could not locate member control.tar.{zstlz4gzxzbz2lzma}
E: Prior errors apply to /var/cache/apt/archives/auditd_1%3a2.8.5-2ubuntu6
_amd64.deb
debconf: apt-extracttemplates failed: No such file or directory
dpkg-split: error: failed to read archive '/var/cache/apt/archives/libaupa
rse0_1%3a2.8.5-2ubuntu6_amd64.deb': Permission denied
dpkg: error processing archive /var/cache/apt/archives/libauparse0_1%3a2.8
.5-2ubuntu6_amd64.deb (--unpack):
 subprocess dpkg-split returned error exit status 2
dpkg-split: error: failed to read archive '/var/cache/apt/archives/auditd_
1%3a2.8.5-2ubuntu6_amd64.deb': Permission denied
dpkg: error processing archive /var/cache/apt/archives/auditd_1%3a2.8.5-2u
buntu6_amd64.deb (--unpack):
 subprocess dpkg-split returned error exit status 2
Errors were encountered while processing:
 /var/cache/apt/archives/libauparse0_1%3a2.8.5-2ubuntu6_amd64.deb
 /var/cache/apt/archives/auditd_1%3a2.8.5-2ubuntu6_amd64.deb
E: Sub-process /usr/bin/dpkg returned an error code (1)
```

with ima_appraise=enforce when attempting to install auditd

## Using Auditd Policies to Ensure File Integrity

CIS recommends that Kubernetes logs events while ensuring filesystem integrity. Auditd is a userspace component and receives and log information from the underlying auditing system and uses information from IMA. Install by running `sudo apt-get install auditd audispd-plugins`. Auditd allows one to create and customize rules relating to permission access of a file. They should be added to `/etc/audit/rules.d/audit.rules` or use the `auditctl` command.

Once the packages are installed, you can start and enable the service with:

```
$ sudo systemctl start auditd
$ sudo systemctl enable auditd
```

All auditd events are located in:

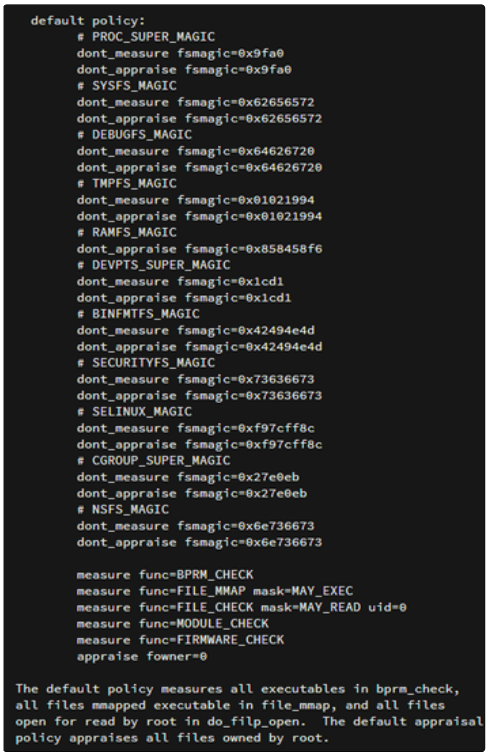
```
/var/log/audit/audit.log
```

## Create an IMA Policy Configuration File to Load Custom Policies

A policy configuration file is needed to load custom policies. Custom and built in policies can be found here.

1. Create an IMA policy configuration file in the /etc/ima/policy.conf by running `vi /etc/ima/policy.conf`
2. Add the following function. It is used to audit the execution of binary files: `audit func=BPRM_CHECK mask=MAY_EXEC`.
3. To load the IMA policy, use `cat /etc/ima/policy.conf > /sys/kernel/security/ima/policy`
4. Restart for changes to take effect.

Custom policies can be created to further enhance file integrity. By default, the following appraisal policies are loaded.



```
  default policy:
        # PROC_SUPER_MAGIC
        dont_measure fsmagic=0x9fa0
        dont_appraise fsmagic=0x9fa0
        # SYSFS_MAGIC
        dont_measure fsmagic=0x62656572
        dont_appraise fsmagic=0x62656572
        # DEBUGFS_MAGIC
        dont_measure fsmagic=0x64626720
        dont_appraise fsmagic=0x64626720
        # TMPFS_MAGIC
        dont_measure fsmagic=0x01021994
        dont_appraise fsmagic=0x01021994
        # RAMFS_MAGIC
        dont_appraise fsmagic=0x858458f6
        # DEVPTS_SUPER_MAGIC
        dont_measure fsmagic=0x1cd1
        dont_appraise fsmagic=0x1cd1
        # BINFMTFS_MAGIC
        dont_measure fsmagic=0x42494e4d
        dont_appraise fsmagic=0x42494e4d
        # SECURITYFS_MAGIC
        dont_measure fsmagic=0x73636673
        dont_appraise fsmagic=0x73636673
        # SELINUX_MAGIC
        dont_measure fsmagic=0xf97cff8c
        dont_appraise fsmagic=0xf97cff8c
        # CGROUP_SUPER_MAGIC
        dont_measure fsmagic=0x27e0eb
        dont_appraise fsmagic=0x27e0eb
        # NSFS_MAGIC
        dont_measure fsmagic=0x6e736673
        dont_appraise fsmagic=0x6e736673

        measure func=BPRM_CHECK
        measure func=FILE_MMAP mask=MAY_EXEC
        measure func=FILE_CHECK mask=MAY_READ uid=0
        measure func=MODULE_CHECK
        measure func=FIRMWARE_CHECK
        appraise fowner=0

The default policy measures all executables in bprm_check,
all files mmapped executable in file_mmap, and all files
open for read by root in do_filp_open.  The default appraisal
policy appraises all files owned by root.
```

https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/ABI/testing/ima_policy

## Policy Rule Examples

Each policy rule must start with the following actions:

- measure: Perform IMA measurement.
- dont_measure: Exclude from IMA measurement.
- appraise: Perform IMA appraisal.
- dont_appraise: Exclude from IMA appraisal.
- audit: Adds the messages to the audit log.

See policy grammar on writing policies here.

An example could look similar to this policy `measure func=FILE_CHECK mask=MAY_READ euid=0`. The policy indicates that for both executing or memory mapping files, measurement action would be performed. The appraisal of root files would also occur.

## Continuous Monitoring

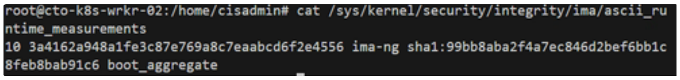
In order to see file changes as they happen, the user must enable i_version mount.

1. To enable i_version on all mounts, add as an example `/dev/vda1  /  ext4  noatime,iversion  1 2` to /etc/fstab.
2. Use the following rootflags in the bootloader configuration
3. Restart
4. Upon restart, due to the new mount, files will be logged as they change providing the following kernel bootloader configuration. The configuration uses rootflags to ensure immediate mounting and i_version support. `GRUB_CMDLINE_LINUX="rootflags=i_version dolvm lsm=integrity ima_appraise=enforce ima_policy=tcb ima_policy=appraise_tcb"`

## Reading Integrity Logs

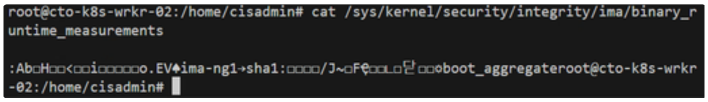
After setting the desired policies, IMA will take measurements based on those policies. Inspecting logs before and after bootloader changes can help one to detect drift and troubleshoot any issues. There are two measurements applicable to comparing runtime measurements detecting drift, runtime measurements and binary measurements. If there has been a change to a file under policy, there will be an entry and a corresponding PCR extend operation.

1. Inspect the logs by catting the ascii_runtime_measurements. These values represent by default PCR 10 measurements. The binary measurements are also available. Both are located in /sys/kernel/security/integrity/ima/ directory.
   Initial ASCII Log taken before IMA setup:



```
root@cto-k8s-wrkr-02:/home/cisadmin# cat /sys/kernel/security/integrity/ima/ascii_ru
ntime_measurements
10 3a4162a948a1fe3c87e769a8c7eaabcd6f2e4556 ima-ng sha1:99bb8aba2f4a7ec846d2bef6bb1c
8feb8bab91c6 boot_aggregate
```

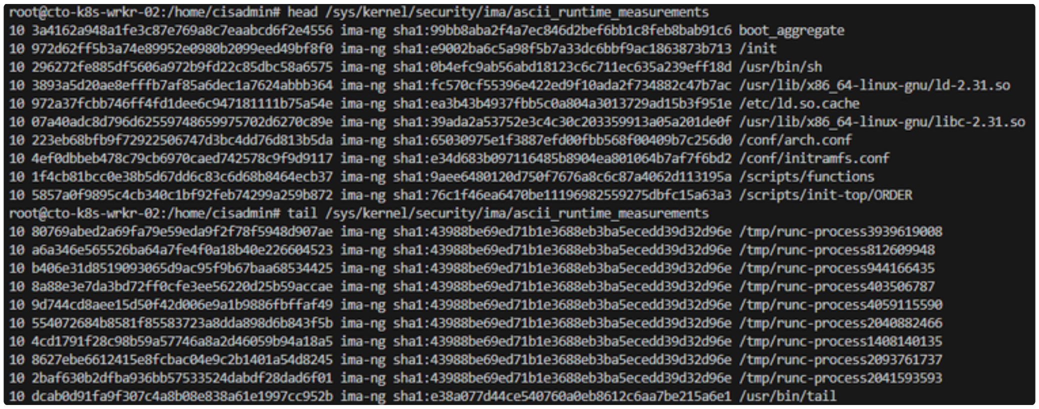
   Initial Binary taken before IMA setup:



```
root@cto-k8s-wrkr-02:/home/cisadmin# cat /sys/kernel/security/integrity/ima/binary_r
untime_measurements

:Ab□H□□<□□i□□□□□□o.EV♠ima-ng1→sha1:□□□□/J~□F₠□□L□ᄅ□□○boot_aggregateroot@cto-k8s-wrkr
-02:/home/cisadmin# █
```

2. After enabling IMA, setting policies and turning on the appraisal of files, the new ASCII log looks like this. They are typically used over binary for readability and preferred when noting changes in PCR values. The change in PCR 10 is recorded as an extended hash where file access and execution is logged via TPM hardware.



```
root@cto-k8s-wrkr-02:/home/cisadmin# head /sys/kernel/security/ima/ascii_runtime_measurements
10 3a4162a948a1fe3c87e769a8c7eaabcd6f2e4556 ima-ng sha1:99bb8aba2f4a7ec846d2bef6bb1c8feb8bab91c6 boot_aggregate
10 972d62ff5b3a74e89952e0980b2099eed49bf8f0 ima-ng sha1:e9002ba6c5a98f5b7a33dc6bbf9ac1863873b713 /init
10 296272fe885df5606a972b9fd22c85dbc58a6575 ima-ng sha1:0b4efc9ab56abd18123c6c711ec635a239eff18d /usr/bin/sh
10 3893a5d20ae8efffb7af85a6dec1a7624abbb364 ima-ng sha1:fc570cf55396e422ed9f10ada2f734882c47b7ac /usr/lib/x86_64-linux-gnu/ld-2.31.so
10 972a37fcbb746ff4fd1dee6c947181111b75a54e ima-ng sha1:ea3b43b4937fbb5c0a804a3013729ad15b3f951e /etc/ld.so.cache
10 07a40adc8d796d62559748659975702d6270c89e ima-ng sha1:39ada2a53752e3c4c30c203359913a05a201de0f /usr/lib/x86_64-linux-gnu/libc-2.31.so
10 223eb68bfb9f72922506747d3bc4dd76d813b5da ima-ng sha1:65030975e1f3887efd00fbb568f00409b7c256d0 /conf/arch.conf
10 4ef0dbbeb478c79cb6970caed742578c9f9d9117 ima-ng sha1:e34d683b097116485b8904ea801064b7af7f6bd2 /conf/initramfs.conf
10 1f4cb81bcc0e38b5d67dd6c83c6d68b8464ecb37 ima-ng sha1:9aee6480120d750f7676a8c6c87a4062d113195a /scripts/functions
10 5857a0f9895c4cb340c1bf92feb74299a259b872 ima-ng sha1:76c1f46ea6470be11196982559275dbfc15a63a3 /scripts/init-top/ORDER
root@cto-k8s-wrkr-02:/home/cisadmin# tail /sys/kernel/security/ima/ascii_runtime_measurements
10 80769abed2a69fa79e59eda9f2f78f5948d907ae ima-ng sha1:43988be69ed71b1e3688eb3ba5ecedd39d32d96e /tmp/runc-process3939619008
10 a6a346e565526ba64a7fe4f0a18b40e226604523 ima-ng sha1:43988be69ed71b1e3688eb3ba5ecedd39d32d96e /tmp/runc-process812609948
10 b406e31d8519093065d9ac95f9b67baa68534425 ima-ng sha1:43988be69ed71b1e3688eb3ba5ecedd39d32d96e /tmp/runc-process944166435
10 8a88e3e7da3bd72ff0cfe3ee56220d25b59accae ima-ng sha1:43988be69ed71b1e3688eb3ba5ecedd39d32d96e /tmp/runc-process403506787
10 9d744cd8aee15d50f42d006e9a1b9886fbffaf49 ima-ng sha1:43988be69ed71b1e3688eb3ba5ecedd39d32d96e /tmp/runc-process4059115590
10 554072684b8581f85583723a8dda898d6b843f5b ima-ng sha1:43988be69ed71b1e3688eb3ba5ecedd39d32d96e /tmp/runc-process2040882466
10 4cd1791f28c98b59a57746a8a2d46059b94a18a5 ima-ng sha1:43988be69ed71b1e3688eb3ba5ecedd39d32d96e /tmp/runc-process1408140135
10 8627ebe6612415e8fcbac04e9c2b1401a54d8245 ima-ng sha1:43988be69ed71b1e3688eb3ba5ecedd39d32d96e /tmp/runc-process2093761737
10 2baf630b2dfba936bb57533524dabdf28dad6f01 ima-ng sha1:43988be69ed71b1e3688eb3ba5ecedd39d32d96e /tmp/runc-process2041593593
10 dcab0d91fa9f307c4a8b08e838a61e1997cc952b ima-ng sha1:e38a077d44ce540760a0eb8612c6aa7be215a6e1 /usr/bin/tail
```

The columns are from right to left:

- PCR number
- Template hash: Hash that combines the length and values of the file content and the pathname.
- Template used to register the integrity value (ima-ng). IMA templates can be selected and the log output may vary depending on the selection IMA Template Management Mechanism — The Linux Kernel documentation
- File Content: Hash of the file itself.

## Assuring Kubernetes Benchmark 1.71 with IMA File Integrity Policies

IMA allows for at least 12 recommended configuration files to be monitored for drift based on a stored hash comparison. This goes beyond AIDE recommend in the Ubuntu benchmark as it is built in and very cost effective. Its architecture is governed by TCG specifications whereas a 3rd party software could introduce more complications to an already questionable system.

Even though there are limitations in namespacing with IMA where Kubernetes has not released the ability to namespace containers, the worker nodes can be examined for further hardening as it shares a vTPM with pods created on it. Kubernetes.io aims to add functionality to differentiate kernel processes via namespaces soon. There is a pull request in progress to do so. In the meantime, one can create a custom forked version published by the same requestor here. For worker node benchmark recommendations, 12 are addressed via IMA or 54.5% of the worker node benchmark component. Those recommendations surround file permissions and ownership including the entire section 1.1. Although the master node was not measured for the sake of this project, 21 of 97 master node recommendations involving file integrity and associated permissions and ownership would be satisfied by IMA policies at 21.65%.

## Conclusions

For both benchmarks, Kubernetes and Ubuntu, Linux IMA can help most with issues surrounding file integrity and access management as well as file appraisal and auditing services. Linux IMA is included with most Linux distributions and offers a policy based integrity system at low-to-no cost to SLTTs. IMA anchors the aggregate integrity value in the TPM and that measurement list is difficult to compromise by a software attack without being detectable. Further, IMA's ability to satisfy greater than half of the Kubernetes worker node recommendations speaks to its capability to deliver applicable filesystem integrity without more than default policies. With more custom policy development, the adoption of IMA to cover a greater number of benchmarks is realistic.

References

Sharefile Links to Benchmarks:

- Kubernetes v1.7.1
- Ubuntu 20.04 LTS v2.0.1

Ask Ubuntu. (n.d.). Where are my kernels? Ask Ubuntu. Where are my kernels?

Burgess, M. (2021, March 22). How to Check the Linux Kernel and Operating System Version. How-To Geek. How to Check the Linux Kernel and Operating System Version

Das, A. (2021, May 5). How to Find Which Kernel Version is Running in Ubuntu. It's FOSS. Check Linux Kernel Version in Command Line [3 Ways]

De Benedictis, Marco & Lioy, Antonio. (2019). Integrity verification of Docker containers for a lightweight cloud environment. Future Generation Computer Systems. 97. 10.1016/j.future.2019.02.026.

Gentoo Wiki. (n.d.). Integrity Measurement Architecture. Gentoo Wiki. Integrity Measurement Architecture - Gentoo wiki

Add support for IMA namespaces by asierHuawei · Pull Request #3703 · kubernetes/enhancements

ima-doc. (n.d.). IMA Configuration [Website]. Read the Docs. IMA Configuration — IMA 1.0 documentation

ima-doc. (n.d.). IMA Policy [Website]. Read the Docs. IMA Policy — IMA 1.0 documentation

Kukuk, T. (2021, January 22). Integrity Measurement Architecture. Linux Magazine. Gotcha » Linux Magazine

Luo, W., Shen, Q., Xia, Y., & Wu, Z. (2019). Container-IMA: A privacy-preserving Integrity Measurement Architecture for Containers. In 22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019). https://www.usenix.org/system/files/raid2019-luo.pdf

openSUSE. (2022). SDB:Ima evm. openSUSE Wiki. SDB:Ima evm - openSUSE Wiki

Red Hat. (2020, October 22). How to use the Linux kernel’s Integrity Measurement Architecture. Red Hat. How to use the Linux kernel's Integrity Measurement Architecture

Steffen, A. (2012). The Linux Integrity Measurement Architecture and TPM-Based Network Endpoint Assessment. In *Linux Security Summit* (pp. 1-16). https://www.strongswan.org/lss2012.pdf

The Linux Foundation. (2023). *IMA Namespaces for Containers - Asier Gutierrez, Huawei* . Retrieved from IMA Namespaces for Containers - Asier Gutierrez, Huawei.