# kernel-sign

*Release 4.3.0*

**Gene C**

**Jun 22, 2025**

# CONTENTS:

CHAPTER

ONE

# KERNEL-SIGN

## 1.1 Overview

Signed kernel modules provide a mechanism for the kernel to verify the integrity of a module. This provides the tools needed to build a kernel with support for signed modules.

- All git tags are signed with arch@sapience.com key which is available via WKD or download from https://www.sapience.com/tech.

### 1.1.1 Latest Changes

- Tidy and Improve code:
  - PEP-8, PEP-257, PEP-484 PEP-561
  - Refactor
- Add tests : run pytest in the *tests* directory.

  This will locate and use a kernel provided tool: /usr/lib/modules/xxx/build/scripts/sign-file
- key and hash types are now read from the kernel config file. Keeps everything consistent.
- Code re-org with supporing modules now moved to lib/xxx.py
- Code works with hash type sha3-xxx (e.g. *sha3-512*) available in kernel 6.7 and openssl 3.2 or later.

## 1.2 Available

Kernel-Sign Github repo:

- Kernel_Sign

These docs on signed kernel modules:

- Arch Wiki
- Kernel Docs

CHAPTER

TWO

# INTRODUCTION

## 2.1 Kernel Modules

The Linux kernel distinguishes and keeps separate the verification of modules from requiring or forcing modules to verify before allowing them to be loaded. Kernel modules fall into 2 classes:

Standard "in tree" modules which come with the kernel source code. They are compiled during the normal kernel build.

Out of tree modules which are not part of the kernel source distribution. They are built outside of the kernel tree, requiring the kernel headers package for each kernel they are to be built for. They can be built manually for a specific kernel and packaged, or they can be built whenever needed using DKMS DKMS

Examples of such packages, provided by Arch, include:

- Virtual gues modules virtualbox-guest-modules-arch

During a standard kernel compilation, the kernel build tools create a private/public key pair and sign every in tree module (using the private key). The public key is saved in the kernel itself. When a module is subsequently loaded, the public key can then be used to verify that the module is unchanged.

The kernel can be enabled to always verify modules and report any failures to standard logs. The choice to permit the loading and use of a module which could not be verified can be either compiled into kernel or turned on at run time using a kernel parameter as explained in Arch Wiki Kernel Parameters Arch Wiki Kernel Parameters.

## 2.2 How to sign kernel modules using a custom kernel

The starting point is based on building a custom kernel package outlined in Kernel/Arch Build System Kernel/Arch Build System,

We will adjust the build to:

- Sign the standard in tree kernel modules
- Provide what is needed to have signed out of tree modules and for the kernel to verify those modules.

Note: The goal is to have:

- In tree modules signed during the standard kernel build process.

  The standard kernel build creates a fresh public/private key pair on each build.

- Out of tree modules are signed and the associated public key is compiled in to the kernel. We will create a separate public/private key pair on each build.

## 2.3 Summary of what needs to be done

Each kernel build needs to made aware of the key/cert being used. Fresh keys are generated with each new kernel build.

A kernel config parameter is now used to make kernel aware of additional signing key:

```
CONFIG_SYSTEM_TRUSTED_KEYS="/path/to/oot-signing_keys.pem".
```

Keys and signing tools will be stored in current module build directory. Nothing needs to be done to clean this as removal is handled by the standard module cleanup.

Certs are thus installed in

```
/usr/lib/modules/<kernel-vers>-<build>/certs-local.
```

CHAPTER

THREE

# KERNEL CONFIGURATION

## 3.1 Kernel Config File

CONFIG_SYSTEM_TRUSTED_KEYS will be added automatically as explained below. In addition the following config options should be set by either manually editing the 'config' file, or via make menuconfig in the linux 'src' area and subsequently copying the updated *.config* file back to the build file *config*. It is preferable to use elliptic curve type keys and zstd compression.

- CONFIG_MODULE_SIG=y

  Enable Loadable module suppot —>

  Module Signature Verification -> activate

- CONFIG_MODULE_SIG_FORCE=n

  Require modules to be validly signed -> leave off

  This allows the decision to enforce verified modules only as boot command line. If you are comfortable all is working then by all means change this to 'y' Command line version of this is : module.sig_enforce=1

- CONFIG_MODULE_SIG_HASH=sha512

  Automatically sign all modules -> activate Which hash algorithm -> SHA-512

  kernel 6.7 and later support sha3 hashes. The preferred hash choice is then sha3-512. This also requires openssl version 3.2 or newer.

- CONFIG_MODULE_COMPRESS_ZSTD=y

  Compress modules on installation -> activate Compression algorithm (ZSTD)

- CONFIG_MODULE_SIG_KEY_TYPE_ECDSA=y

  Cryptographic API —> Certificates for Signature Checking —> Type of module signing key to be generated -> ECDSA

- CONFIG_MODULE_ALLOW_MISSING_NAMESPACE_IMPORTS=n

  Enable Loadable module support —> Allow loading of modules with missing namespace imports -> set off

## 3.2 Kernel command line

After you are comfortable things are working well you can enable the kernel parameter to require that the kernel only permit verified modules to be loaded:

```
module.sig_enforce=1
```

CHAPTER

# FOUR

# TOOLS NEEDED

## 4.1 kernel build package

In the directory where the kernel package is built:

```
mkdir certs-local
```

This directory will provide the tools to create the keys, as well as signing kernel modules.

- Copy these files into certs-local directory:

```
genkeys.py
install-certs.py
sign_module.py
lib/*.py
x509.oot.genkey
```

## 4.2 genkey.py & x509.oot.genkey

genkey.py along with its configuration file x509.oot.genkey are used to create key pairs. It also provides the kernel with the key to sign out of tree modules by updating the config file used to build the kernel.

genkeys.py will create the key pairs in a directory named by date-time. It defaults to refreshing the keys every 7 days but this can be changed with the *–refresh* command line option.

It also creates a soft link named 'current' which points to the newly created directory with the 'current' keys. The actual key directory is named by date and time.

genkeys will check and update kernel configs given by the –config config(s) option. This takes either a single config file, or a shell glob for mulitple files. e.g. –config 'conf/config.*'. Remember to quote any wildcard characters to prevent the shell from expanding them.

All configs will be updated with the same key. The default keytype and hash are taken from the kernel config (from CONFIG_MODULE_SIG_HASH and CONFIG_MODULE_SIG_KEY_TYPE_xxx)[1].

If multiple kernel configs are being used, all must use same key and hash types.

## 4.3 sign_module.py

signs out of tree kernel modules. It can be run manually but is typically invoked by dkms/kernel-sign.sh. It handles modules compressed with zstd, xz and gzip and depends on python-zstandard package to help with those compressed with zstd.

[1] In earlier versions these defaulted to elliptic curve and sha512 and could be set from the command line.

## 4.4 install-certs.py

is called from the package_headers() function of PKGBUILD to install the signing keys. Example is given below.

These files are all provided.

## 4.5 dkms support

**Important**

DKMS a mechanism for out-of-tree modules to be compiled against the kernel headers. It is one thing to use signed modules provided in the kernel source but it is quite another to use modules, signed or not, that are out-of-tree. Any such module will *taint* the kernel. See kernel docs tainted_kernel for more information.

```
mkdir certs-local/dkms
```

and add 2 files to the dkms dir:

```
kernel-sign.conf
kernel-sign.sh
```

These will be installed in /etc/dkms and provide a mechanism for dkms to automatically sign modules using the local key discussed above - this is the reccommended way to sign kernel modules. As explained, below - once this is installed - all that is needed to have dkms automatically sign modules is to make a soft link:

```
cd /etc/dkms
ln -s kernel-sign.conf <module-name>.conf
```

For example: .. code-block:: bash

    ln -s kernel-sign.conf vboxdrv.conf

The link creation can easily be added to an arch package to simplify further if desired.

CHAPTER

FIVE

# MODIFY PKGBUILD

## 5.1 What to change

We need to make changes to kernel build as follows:

## 5.2 prepare()

Add the following to the top of the prepare() function:

```
prepare() {
 ...
    echo "Rebuilding local signing key..."
    # adjust cerdir as needed
    certdir='../certs-local'
    $certdir/genkeys.py -v --config ../config --refresh 30d
  ...
}
```

The default key regeneration refresh period is 7 days, but this can be changed on the command line. If you want to create new keys monthly, then use "–refresh 30days" as an option to genekeys.py. You can refresh on every build by using "–refresh always". Refresh units can be seconds,minutes,hours,days or weeks.

## 5.3 _package-headers()

Add the following to the bottom of the _package-headers() function:

```
_package-headers() {

...

#
# Out of Tree Module signing
# This is run in the kernel source / build directory
#
echo "Local Signing certs for out of tree modules..."

certs_local_src="../../certs-local"
certs_local_dst="${builddir}/certs-local"

${certs_local_src}/install-certs.py $certs_local_dst
```

(continues on next page)

(continued from previous page)

```
# install dkms tools if needed
dkms_src="$certs_local_src/dkms"
dkms_dst="${pkgdir}/etc/dkms"
mkdir -p $dkms_dst

rsync -a $dkms_src/{kernel-sign.conf,kernel-sign.sh} $dkms_dst/
}
```

CHAPTER

SIX

# REQUIRED FILES

This is the list of files referenced above. Remember to make scripts executable.

- certs-local/genkeys.py
- certs-local/install-certs.py
- certs-local/x509.oot.genkey
- certs-local/sign_module.py
- certs-local/lib/arg_parse.py
- certs-local/lib/refresh_needed.py
- certs-local/lib/class_genkeys.py
- certs-local/lib/get_key_hash.py
- certs-local/lib/make_keys.py
- certs-local/lib/signer_class.py
- certs-local/lib/update_config.py
- certs-local/lib/utils.py
- certs-local/dkms/kernel-sign.conf
- certs-local/dkms/kernel-sign.sh

CHAPTER

SEVEN

# ARCH AUR PACKAGS

## 7.1 AUR Packages

There is an Arch Sign Modules package in the AUR along with its companion github repo Arch-SKM which make use of Kernel_Sign

arch-sign-modules reduces the manual steps for building a fully signed custom kernel to 3 commands to *Update*, *Build* and *Install* a kernel.

```
abk -u kernel-name
abk -b kernel-name
abk -i kernel-name
```

For more information see Arch-SKM-README and example Arch-SKM-PKGBUIILD

CHAPTER

EIGHT

# LICENSE

Created by Gene C. and licensed under the terms of the MIT license.

- SPDX-License-Identifier: MIT
- Copyright (c) 2020-2023, Gene C

CHAPTER

NINE

# CHANGELOG

## 9.1 Tags

```
0.1.0 (2019-11-10) -> 4.3.0 (2025-06-22)
110 commits.
```

## 9.2 Commits

- 2025-06-22 : **4.3.0**

```
              run_prog: sync with latest from pyconcurrent
2025-06-20    update Docs/Changelogs Docs/kernel-sign.pdf
```

- 2025-06-20 : **4.2.0**

```
              Pull local copy of latest run_prog from pyconcurrent
2025-05-21    update Docs/Changelogs Docs/kernel-sign.pdf
```

- 2025-05-21 : **4.1.0**

```
Use builtin types where possible. e.g. typing.List -> list
update Docs/Changelogs Docs/kernel-sign.pdf
```

- 2025-05-21 : **4.0.0**

```
              Tidy and Improve code:
                PEP-8, PEP-257, PEP-484 PEP-561
                Refactor
              Add tests : run pytest in the *tests* directory.
                This will locate and use a kernel provided tool:
                /usr/lib/modules/xxx/build/scripts/sign-file
2024-12-31    update Docs/Changelog.rst Docs/kernel-sign.pdf
```

- 2024-12-31 : **3.0.2**

```
              Git tags are now signed.
              Update SPDX tags
2023-11-05    update Docs/Changelog.rst Docs/kernel-sign.pdf
```

- 2023-11-05 : **3.0.1**

```
fix readme.rst
update Docs/Changelog.rst Docs/kernel-sign.pdf
```

- 2023-11-05 : **3.0.0**

```
* key and hash types are now read from the kernel config file. Keeps
everything consistent.
* Code re-org with supporing modules now moved to lib/xxx.py
* Confirm code works with hash type *sha3-512* introduced in kernel 6.7
  Requires openssl 3.2+ / kernel 6.7+
```

- 2023-10-04 : **2.6.1**

```
              update Docs/Changelog.rst
              Fix from Author: Expertcoderz <expertcoderzx@gmail.com>
              Merge pull request #8 from Expertcoderz/patch-1
              Fix -v argument help text f-string
              Fix -v argument help text f-string
2023-09-27    update Docs/Changelog.rst
```

- 2023-09-27 : **2.6.0**

```
              Reorganize documrnts under Docs.
              Migrate to restructured text
              Now easy to build html and pdf docs using sphinx
2023-03-31    update CHANGELOG.md
```

- 2023-03-31 : **2.5.1**

```
              update changelog
              small README wordsmithing
2023-03-30    update CHANGELOG.md
```

- 2023-03-30 : **2.5.0**

```
              Use Full path to openssl executable /usr/bin/openssl
2023-01-06    Rename Changelog to CHANGELOG following our current release tools. Update
```

- 2023-01-06 : **2.4.0**

```
              Add SPDX licensing lines
              Update kernel doc reference link
2022-07-07    Update kernel.org doc link to more recent kernel
```

- 2022-05-25 : **2.3.9**

```
suppress a few silly pylint warnings
```

- 2022-05-21 : **2.3.8**

```
Remove uneeded exception we just added. It is a subclass of OSError and so
not needed.
```

- 2022-05-21 : **2.3.7**

```
utils open_file catches (OSError,FileNotFoundError) exceoptions now
```

- 2022-05-18 : **2.3.6**

```
update changelog
Fix open_file to only apply encoding to text files (thanks  pylint)
```

- 2022-05-18 : **2.3.5**

```
 - More tidy up - try keep pylint noise down so dont miss things it finds
delete old stuff
comment tweak
```

- 2022-05-18 : **2.3.4**

```
               Update Changelog
               More code tidying in genkeys
               little more tidyup - no functional changes
2022-05-12     Add missing date to Changelog
```

- 2022-05-09 : **2.3.3**

```
update Changelog
Use OSError exception which has replaced IOError
Catch OSError when file open fails
```

- 2022-05-08 : **2.3.2**

```
Ack and Tested by by @itoffshore
update Changelog
trivial tidy
```

- 2022-05-08 : **2.3.1**

```
more code tidying
Update Changelog
fix typo for refresh check
tidy and improve exception handling
tidy
more cleaning
more tidy
more tidy ups
some code tidying
another typo!
typo
fix file to name to avoid module conflict
```

- 2022-05-08 : **2.3.0**

```
- Code re-org to be more robust and easier to read.
- Introduce KernelModSigner class and ModuleTool class to help organize
- Functionality is unchanged.
```

- 2022-05-04 : **2.2.1**

```
Update Changelog and README to reflect sign_module.py replacing
sign_manual.sh
Changelog - add date for 2.2.0
```

- 2022-05-04 : **2.2.0**

```
               update changelog
               archive sign_manual.sh
               turn off dev to ready for production
               Improve module signing scripts:
                - sign_module.py replaces sign_manual.sh
                - dkms/kernel_sign.sh updated accordingly
                - install-certs updated accordingly
                - adds dependency : python-zstandard for handling zst compressed modules
2022-05-03     README - small markdown tweaks
```

- 2022-05-03 : **2.1.1**

```
update changelog
typo
```

- 2022-05-03 : **2.1.0**

```
               update Changelog
               The key type and hash are now saved in files along side the keys. This
               allows the signing script to read them, and this means it no longer has
               hardcoded hash.  the sign script falls back on sha512 in case of previous␣
↪key
               directory without a saved hash
2022-05-02     remove extraneous |
```

- 2022-05-02 : **2.0.0**

```
update changelog
word smith README
fix markdown on last addition
```

- 2022-05-02 : **1.3.5**

```
Update README and Changelog
Add few more words about some available tooks by @itoffshore
```

- 2022-05-02 : **1.3.4**

```
Update Changelog
White space patches from @itoffshore
```

- 2022-05-02 : **1.3.3**

```
Update Changelog
Typo in echo found by @itoffshore
Changelog udpate
Add reference to @itoffshore aur package and github repo
```

- 2022-05-02 : **1.3.2**

```
Fix hexdump typo "--e" to "-e"
Changelog update
Mindor markdown tweaks
```

- 2022-05-02 : **1.3.1**

```
typo fix
Update Changelog
```

- 2022-05-02 : **1.3.0**

```
Per @ittoffshore, add comment about quoting wildcard characters
Fixes from @itoffshore
1. For manual signing
   zstd modules use .zst instead of .zsrd
   support for gzip
2. For dkms
   Add gzip support
```

- 2022-05-01 : **1.2.0**

```
Expand help with reminder wildcards must be quoted
```

- 2022-05-01 : **1.1.0**

```
tweak the prepare() example
small word smithing
```

- 2022-05-01 : **1.0.1**

```
remove debugging
```

- 2022-05-01 : **1.0.0**

```
              Update readme and changelog
              genkeys now handles multiple configs using shell glob with --config
              support utilities
              Rename tools to utils
              Share coupld functions via tools.py
              Add install-certs.py for use by package_headers() to simplify PKGBUILD
2022-04-30    Update package_headers() to remove reference to file no longer being
              created. Part of issue #3
              Add a little markdown to Changlelog.md
              Update changes for 0.8.0 and 0.8.1
```

- 2022-04-30 : **0.8.1**

```
Remove references to now unused scripts
```

- 2022-04-30 : **0.8.0**

```
fix typo
Tidy up README
As per itoffshore check for key exists prior to getting mtime. Fixes bug in
check_refresh()
```

- 2022-04-30 : **0.7.0**

```
version [0.7.0]                                                    -
20220430
  - Add genkeys.py (replaces both genkeys.sh and fix_config.sh)
    This supports refresh key frequency (default is 7 days)
    PKGBUILD use: ./genkeys.py -v
    Creates new keys as needed and updates kernel config.
version [0.6.0]                                                    -
20220430
  - Support zstd module compression in sign_manual.sh
  - Improve hexdump for signed module detection in sign_manual.sh
  - Has hardcoded sha512 hash - needs updating/replacing
version [0.5.0]                                                    -
20220420
  - Switch to using elliptic curve
```

- 2021-10-20 : **0.4.0**

```
Update kernel-sign.sh for compressed modules
```

- 2019-11-15 : **0.3.0**

```
Tidy Readme
```

- 2019-11-10 : **0.2.0**

```
tidy up readme
```

- 2019-11-10 : **0.1.0**

```
Initial revision
```

CHAPTER

TEN

# MIT LICENSE

CHAPTER

# ELEVEN

# HOW TO HELP WITH THIS PROJECT

Thank you for your interest in improving this project. This project is open-source under the MIT license.

## 11.1 Important resources

- Git Repo

## 11.2 Reporting Bugs or feature requests

Please report bugs on the issue tracker in the git repo. To make the report as useful as possible, please include

- operating system used
- version of python
- explanation of the problem or enhancement request.

## 11.3 Code Changes

If you make code changes, please update the documentation if it's appropriate.

CHAPTER

TWELVE

# CONTRIBUTOR COVENANT CODE OF CONDUCT

## 12.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, sex characteristics, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, religion, or sexual identity and orientation.

## 12.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

## 12.3 Our Responsibilities

Maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

## 12.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

## 12.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at <arch@sapience.com>. All complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The Code of Conduct Committee is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

## 12.6 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available at https://www.contributor-covenant.org/version/1/4/code-of-conduct.html

## 12.7 Interpretation

The interpretation of this document is at the discretion of the project team.

CHAPTER

THIRTEEN

# INDICES AND TABLES

- genindex
- modindex
- search