dns_tools

Release 3.4.3

Gene C

Dec 31, 2024

CONTENTS:

CHAPTER

ONE

DNS_TOOLS

1.1 Overview

DNS server tools - aka DNSSEC made easy.

DNSSEC can be a little tricky especially rolling the keys. We provide the tools to simplify and automate this as much as possible.

1.2 New / Interesting

- Going forward all git tags will be signed by <arch@sapience.com>. Public key is available via WKD or download from website: https://www.sapience.com/tech After key is on keyring use the PKGBUILD source line ending with *?signed* or manually verify using *git tag -v <tag-name>
- Fix up restructred test formatting in README
- Use lockmgr package instead of local copy (also lockmgr AUR)
- pdf doc is now pre-build in Docs dir
- Use lockfile to enforce only one dns-tool runs at a time
- NB the locking code uses inotify to wait to acquire a lock if another process has it locked. This uses *inotify* from standard C-library and our python code depends on the size of *struct inotify_event* that is returned. To best of my knowledge this is same on all versions of linux. i.e. (int, uint32_t, uint32_t, uint32_t, ...). If anyone comes across anything different please let me know.

CHAPTER

TWO

GETTING STARTED

2.1 Installation

Available on * Github * Archlinux AUR

On Arch you can build using the PKGBUILD provided in packaging directory or from the AUR package.

To build manually, see Appendix *manual_build*.

CHAPTER

THREE

DNS_TOOLS APPLICATIONS

3.1 Testing Mode

For convenience each tool supports a test mode engaged using the *-t, –test* option. When run in test mode, actions are printed instead of actually being done.

When running in test mode nothing is done, which can lead to things seemingly being strange. For example, when testing rolling or generation of *next* keys, the code later checks for any missing keys. Now in test mode they can be missing since they were not actually created when they would normally be. So now you can see messages about keys being generated a second time. They wont be in non-test mode of course as nothing would be missing.

For testing, I also find it convenient to change the production dns zone directores to something like */tmp/dns* - and then run the tests without *-t*. This does everything as asked, but instead of pushing to the real dns servers, the files are pushed to the test production directory. When doing this, you can drop the *–dns_restart* option so as to skip restarting the dns servers - which is not needed obviously.

3.2 Kinds of DNSSEC keys

A quick reminder about DNSSEC keys.

- Zone Signing Key (ZSK)

  This is used to sign dns zone files. It is advisable to update this periodically, perhaps every 1 to 3 months. The mechanism to update the key requires some care and is known as *rolling* the keys. The tools make this straightforward. More on this later.

- Key Signing Key (KSK)

  This signs the zone signing key - and its this key that must be registered with the domain registrar for the root servers. The requirement ensures that there is an appropriate chain of trust from the root dns servers on down. Most, if not all registars now support DNSSEC - Google Domains does for example. This requirement means there is a manual step whenever the KSK changes, which is updating the root servers with the new information. KSK should be rolled occasionally, in spite of the manual step, perhaps every 1-3 years, and the corresponding DS (Delegation Signer) record for the new KSK should be uploaded to the domain registrar.

3.3 Key Rolling

The typical approach to doing this is accomplished in 2 basic steps.

- **Phase 1**

  Create a new key, called *next*, then sign the zone files using this key as well as the current key (we call this *curr* for short). With the records now double signed the existing key remains valid.

- **Phase 2**

  After a period sufficiently longer than the TTL for the zone, say 2 x TTL, then rename the *next* key to be the *curr* key. Resign using just this key. This gives time for DNS servers to catch up with the new key before the old one is removed.

Rolling KSK and ZSK is basically the same, but for KSK, the DS records must be uploaded to the domain registrar. In Phase 1 both old and new DS should be uploaded and in Phase 2 just the new (now current) KSK DS. The tool creates the DS records but uploading them to registrar must be done manually.

3.4 Using the tools

The following set of tools are provided.

- **dns-tool**

  This tool handles all DNSSEC related operations including key creation and rolling, and using those keys to sign the dns zone files.

- **dns-prod-push**

  This tool make it simple to push signed and/or unsigned dns zone files from the signing server to the production area for each primary dns server. the DNS primary server(s) should be on same machine or reachable via ssh. It also restarts those servers when appropriate.

- **dns-serial-bump**

  A standalone tool to check the validity and bump the serial number in the SOA of a dns zone file.

3.5 Example Usage

N.B. :

- Must run on signing server.

  The tools must be run on the signing server which is defined in the config file. To minimize chance of an accident, the code will refuse to run if that is not the case.

- Run as root.
  - operations require effective root user:
  - Changing the ownership permisions of staging zones to *dns_user* and *dns_group*.
  - Preserving ownership when files rsync –owner to dns server(s)

- Zone serial numbers should be in canonical format for serial bump to work properly.

  i.e. yyymmddnn where yyymmdd is date and nn is a 2 digit counter from 00 to 99 If not code will do best it can to migrate to canonical format if possible. It will warn of non-standard or invalid serials and replace them with valid serials. A valid serial is all numbers and must be expressable as 32 bits. You can use the *dns-serial-bump –check zonefile* to check for valid serial.

The tool supports 2 primary servers - an internal DNS server and an external server. The internal server may also serve additional unsigned zones, typically RFC1918 and their reverse zones. There can be unsigned zones for external server too of course and if there are, they will be pushed along with all the other signed zones.

The external primary is how the outside world views DNS for each domain. As usual once a primary dns server is updated, it's secondaries will get updated automatically via IXFR/AXFR.

The tool is driven by a straightforward config file which is first looked for in current directory under *./conf.d/config* and if not available there it should be in */etc/dns_tools/conf.d/config*.

The config file holds the information about where all the relevant files are kept and the command to use to restart the dns servers, the DNS server hosts and so on.

Copy the sample config file and edit it for your needs:

```
cd /etc/dns_tools
cp conf.d/config.sample conf.d/config
```

Edit the config file to suit your needs. Set the *work_dir* to wherever you want to keep the internal/external zone files and the keys. The sample config uses */etc/dns_tools* for the working directory. Relative directory names are always relative to the working directory.

The *work_dir* holds all the data and is the source for all key and zone information. Signed and unsigned zone files are pushed from the working dir to each of the DNS servers. Internal and external dns zone files are kept in their own directories. e.g.

```
<work_dir>/internal/staging/zones
```

The *ldns* package has standalone tools which used to handle key generation and to sign the zone files.

With that background information, and under the assumption that the domain registrar already has the ksk required information then to roll ZSK using dns_tools would be simply:

```
/usr/bin/dns-tool --zsk_roll_1
/usr/bin/dns-prod-push --dns_restart --to_production
```

and after couple hours or similar time, the second phase would be accomplished using:

```
/usr/bin/dns-tool --zsk_roll_2
/usr/bin/dns-prod-push --dns_restart --to_production
```

And of course in practice each of these would be run from cron - I run them monthly. A sample cron file is provided in */etc/dns_tools/cron/dnssec-roll.cron*. And for convenience, it uses the above commands wrapped by the shell scripts:

```
/etc/dns_tools/scripts/zsk-roll-1.sh
/etc/dns_tools/scripts/zsk-roll-2.sh
```

3.6 Create Keys

To get things started simply create the KSK and ZSK keys and then upload the DS key info to the domain registrar. To generate a new set of keys simply run:

```
/usr/bin/dns-tool --gen_ksk_curr --gen_zsk_curr
```

All the keys will be under the *keys* directory. For each domain, the info needed for the domain registrar will be found in the file:

```
<work_dir>/keys/<domain>/ksk/curr.all.ds
```

By default all the domains in the config are processed. To process a one or more specific domains just put them on the command line. Domains listed on command line will override the config file.

All zone files for both internal and external dns should be available as specified in the config file. See the sample config for more details.

3.7 KSK Keys and DS to root servers

When you create KSK keys a set of DS keys will be generated automatically. These actually come in different hash types:

- **1 : sha1** - deprecated and shouldn't be used
- **2 : sha256** - the default and saved in curr.ds
- **4 : sha512** - slower but somewhat more secure hash
- **g : gost**

We do not generate the type *4 gost* hash.

These are saved into the *<work_dir>/keys/<domain>/ksk/* directory.

In addition to *curr.ds*, *curr.all.ds* contains **sha1**, **sha256** and **sha512**. Choose one or more of these to upload to your domain registrar.

Its good to get this uploaded and available from the root servers soon as your KSK keys are ready and before you push any signed zones out. This is the only manual step. And if/when you roll your ksk, then it needs to be repeated with the new DS key info.

I recommend uploading both sha256 (type 2) and sha512 (type 4) keys. Associated with each the *curr.all.ds* file will also have a numerical Id, which you'll need to share with your registrar. Note that it can take some time for the root servers to get updated with your new KSK - which is fine. Just means that your DNS will be non-dnssec until they get the KSK pushed out to the world. Once that happens, then dns clients will see the KSK and dnssec will be operational.

Everthing else should be handled automatically by the tool.

3.8 Updating dns zone files

Whenever you update any zone files, they must be resigned. Make any zone file changes in the zone staging directories. i.e.

```
<work_dir>/internal/staging/zones
<work_dir>/external/staging/zones
```

You don't need to bump serial number, the tool will do it for you, though its benign to do so. When you're done with the changes then to resign and push just run:

or use the convenience wrapper script for these 2 commands by running:

```
/etc/dns_tool/resign.sh
```

This also takes optional arguments:

- –serial_bump
- list of domains. If none listed, then uses all domains in config file.

CHAPTER

FOUR

OVERVIEW OF OPTIONS

4.1 dns-tool options

Handles key generation, zone signing and key rolls.

While there are many options, majority are more for testing or speical needs. The main options are *test*, *print_keys*, *sign*, *zsk_toll_1*, *zsk_roll_2*

- positional arguments:

  one or more domains here will override config file.

- (*-h, –help*)

  show this help message and exit

- (*–theme*)

  Output color theme for tty. One of : dark, light or none

- (*-t, –test*)

  Test mode - print but dont do

- (*-v, –verb*)

  More verbosity

- (*–serial_bump*)

  Bump all serials. Not usually needed as happens auotmatically This implies *–sign* so that signed zones stay consistent.

- (*–keep_include*)

  Keep temp file which has $INCLUDE expanded

- (*–sign*)

  Short hand for sign with curr keys (ksk and zsk)

- (*–sign_ksk_next*)

  Sign with next ksk

- (*–sign_zsk_next*)

  Sign with next zsk

- (*–gen_zsk_curru*)

  Generate ZSK for curr

- (*–gen_zsk_next*)

  Generate ZSK for next

- (*–gen_ksk_curr*)

  Generate KSK for curr

- (*–gen_ksk_next*)

  Generate KSK for next

- (*–zsk_roll_1*)

  ZSK Phase 1 roll - old and new

- (*–zsk_roll_2*)

  ZSK Phase 2 roll - new only

- (*–ksk_roll_1*)

  KSK Phase 1 roll - old and new - NB must add to degistrar

- (*–ksk_roll_2*)

  KSK Phase 2 roll - new only

- (*–print_keys*)

  Print keys (curr and next)

4.2 dns-prod-push options

Tool to push signed and unsigned zones to the dns server(s)

- positional arguments:

  one or more domains here will override config file.

- (*-h, –help*)

  show help message and exit

- (*–theme*)

  Output color theme for tty. One of : dark, light or none

- (*–int_ext what*)

  What to push. One of : internal, external or both (default is both)

- (*–to_production*)

  Copy zone files from work staging area to live production area

- (*–dns_restart*)

  Restart the dns server after update zones using the config variable *dns_restart_cmd*.

  For example for nsd, set this to:

  dns_restart_cmd = “/usr/bin/systemctl restart nsd”

- (*-t, –test*)

  Test mode - print but dont do

- (*-v, –verb*)

  More verbosity

4.3 dns-serial-bump options

Tool to bump the serial number of a DNS zone file.:

    dns-serial-bump [-c] <zonefile>

Arguments:

- positional arguments One or more zonefiles with SOA containing a serial number.
- (*-h, –help*)

  show help message and exit

- (*-c, –check*)

  Check and show current and updated serial number for each zonefile. When check is enabled zonefiles do not have their serial number updated. Without *check* option each zonefile will also be updated with new serial.

4.4 Update your DNS to use signed zone file

When you're ready to switch your dns to dnssec then all that's needed is change the primary server config to point to the signed zone file rather than the unsigned.

For nsd this would be of the form:

```
zone:
    name:       example.com
    #zonefile:  %s                      # unsigned
    zonefile:   %s.signed/zone          # signed
    include-pattern: "tosecondary"      # notify all secondary servers
```

CHAPTER

FIVE

FAQ

5.1 Why is name not dnssec_tools?

This is a good question. I did give some thought to this and ended up with the more generic name.

My thinking is this. Since the tool is really about managing DNS zones in one place and not just about keys/signing I went with the more generic name combined with the addition of DNSSEC keyword.

There are three basic parts to the tools:

- Check the validity and increment the serial number in the SOA section of zonefile.
- Push zone files to primary DNS servers (internal and external facing servers) and restart them.
- Generate and manage KSK and ZSK keys and use them to sign zones.

While all of them are needed to provide automation of key rolls, the first two items above are not specific to DNSSEC. That said the bulk of the code deals with the more complex DNSSEC tasks.

CHAPTER

SIX

APPENDIX

6.1 Dependencies

Run Time :

- python (3.9 or later)
- ldns
- *tomli* if python < 3.11 (aka python-tomli)

Building Package:

- git
- hatch (aka python-hatch)
- wheel (aka python-wheel)
- build (aka python-build)
- installer (aka python-installer)
- rsync

Optional to build docs: * sphinx * myst-parser * texlive-latexextra (archlinux packaging of texlive tools)

CHAPTER

SEVEN

BUILD MANUALLY

To build it manually, clone the repo and do:

```
rm -f dist/*
python -m build --wheel --no-isolation
root_dest="/"
./scripts/do-install $root_dest
```

When running as non-root then set root_dest a user writable directory

7.1 Philosophy

We follow the *live at head commit* philosophy. This means we recommend using the latest commit on git master branch. We also provide git tags.

This approach is also taken by Google[1][2].

7.2 License

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

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

[1] https://github.com/google/googletest
[2] https://abseil.io/about/philosophy#upgrade-support

CHAPTER

EIGHT

CHANGELOG

[3.4.3] —— 2024-12-31

```
Git tags are now signed.
Update SPDX tags
update Docs/Changelog.rst Docs/dns_tools.pdf
```

[3.4.2] —— 2024-09-07

```
More rst tweaks
```

[3.4.1] —— 2024-09-07

```
update Docs/Changelog.rst Docs/dns_tools.pdf
Fix up restructred test formatting in README
update Docs/Changelog.rst Docs/dns_tools.pdf
```

[3.4.0] —— 2024-06-02

```
Improve exception handling with setting file permissions
update requirements.txt to show dep on lockmgr
update Docs/Changelog.rst Docs/dns_tools.pdf
```

[3.3.0] —— 2024-03-30

```
Lockfile now attached to uid
update readme
update Docs/Changelog.rst
update project version
use max(a,b) instead of if(a>b) etc
Now uses separate lockmgr package instead of local copy
update Docs/Changelog.rst Docs/dns_tools.pdf
```

[3.1.1] —— 2023-12-26

```
Remove tomli from depends() in PKGBUILD as not needed for python >= 3.11
update Docs/Changelog.rst Docs/dns_tools.pdf
```

[3.1.0] —— 2023-11-26

```
Switch python backend build to hatch
update Docs/Changelog.rst Docs/dns_tools.pdf
```

[3.0.0] —— 2023-11-16

```
Some lint cleanups.
Add lock to ensure only one dns-tool runs at a time.
NB The inotify code, used to wait on lock, uses inotify in libc
  This returns a struct inotify_event and the size of this struct is important.
  Best I know on every (linux) system the struct size is:
    (int, uint_32_t, uint_32_t, uint_32_t, ...)
  If you find a system where they are different (see man inotify) let me know!
update Docs/Changelog.rst Docs/dns_tools.pdf
```

[2.6.0] —— 2023-11-12

```
resign.sh accept --serial-bump, -s, --serial_bump
Do not expand $INCLUDE when in a comment line before signing
update Docs/Changelog.rst
```

[2.5.0] —— 2023-11-06

```
scripts/resign.sh now has optional argument --serial-bump
resign.sh now takes optional domain list.
  If none provided then does all domeains in /etc/dns_tool/conf.d/config as previously
update readme for resign.sh changes
fix typo in comment
update Docs/Changelog.rst
```

[2.4.0] —— 2023-09-27

```
Reorg and rework documentation.
Now simple to generate html and pdf docs using sphinx
update CHANGELOG.md
```

[2.3.2] —— 2023-05-18

```
Update build info in README
update CHANGELOG.md
```

[2.3.1] —— 2023-05-18

```
PKGBUILD - add dependency on python installer module
update CHANGELOG.md
```

[2.3.0] —— 2023-05-18

```
install: switch from pip to python installer package. This adds optimized bytecode
update CHANGELOG.md
```

[2.2.4] —— 2023-05-18

```
PKGBUILD: add python-build to makedepends
update CHANGELOG.md
```

[2.2.3] —— 2023-05-18

```
PKGBUILD: build wheel back to using python -m build instead of poetry
update CHANGELOG.md
```

[2.2.2] — 2023-05-17

```
Simplify Arch PKGBUILD and more closely follow arch guidelines
update CHANGELOG.md
```

[2.2.1] — 2023-04-16

```
update 2.2.1 with few more notes about KSK and root servers
Add few more lines about root servers and KSK
update CHANGELOG.md
```

[2.2.0] — 2023-02-10

```
Fix typo in rsync - this case is not used here
update CHANGELOG.md
```

[2.1.0] — 2023-02-04

```
rel_from_abs_path now uses os.path.relpath() instead of our own function
Improve message about checking to ensure required keys are available
Small readme changes
more readme changes
readme tweaks
update CHANGELOG.md
```

[2.0.2] — 2023-01-24

```
Add note to change primary to point to signed zone files
readme whitespace markdown fix
more polishing of readme
tweak readme
Add FAQ to readme
update CHANGELOG.md
```

[2.0.1] — 2023-01-22

```
Remove "coming soon" from readme
fix PKGBUILD
update CHANGELOG.md
```

[2.0.0] — 2023-01-22

```
Initial release
updated readme
improve readme
updated readme
readme update
Initial Commit
```

CHAPTER

NINE

MIT LICENSE

CHAPTER

TEN

HOW TO HELP WITH THIS PROJECT

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

10.1 Important resources

- Git Repo

10.2 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.

10.3 Code Changes

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

CHAPTER

ELEVEN

CONTRIBUTOR COVENANT CODE OF CONDUCT

11.1 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.

11.2 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

11.3 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.

11.4 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.

11.5 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.

11.6 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

11.7 Interpretation

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

CHAPTER

TWELVE

INDICES AND TABLES

- genindex
- modindex
- search