Tracer Documentation

Release recent

FrostyX

May 16, 2017

Contents

Tracer finds outdated running applications in your system

Tracer determines which applications use outdated files and prints them. For special kind of applications such as services or daemons, it suggests a standard command to restart it. Detecting whether file is outdated or not is based on a simple idea. If application has loaded in memory any version of a file which is provided by any package updated since system was booted up, tracer consider this application as outdated.

CHAPTER 1

Docs

User Guide

Installation

Please, see how to *Get Tracer*.

Standard usage

Standard way to use tracer is just running the `sudo tracer` command. I am working on user-only mode, but in this moment it runs only with root permissions (It needs access to your package manager's history). There you can see the output:

```
$[FrostyX  ~]-> sudo tracer
You should restart:
  * Some applications using:
      sudo service apache2 restart
      sudo service mpd restart

  * These applications manually:
      chromium
      dolphin
      gvim

Additionally to those process above, there are:
  - 6 processes requiring restarting your session (i.e. Logging out & Logging in␣
↪again)
  - 2 processes requiring reboot
```

As you can see, It hides few kinds of applications to not bothering you with such that you can't control. But of course, if you want, you can see them all, use the `-a` or `--all` parameter.

Helpers

You got list of applications, but what next. There is `-s` or `--show` parameter for showing some handy informations. It displays package owning this application, what user started it and when, its PID and recommended way, how to restart it. When you want to print helpers for all affected applications, you can use `--helpers` parameter.

```
$[FrostyX  ~]-> tracer -s apache2
* apache2
    Package:     www-servers/apache
    Description: The Apache Web Server.
    Type:        Daemon
    State:       apache2 has been started by root 34 minutes ago. PID - 18816

    How to restart:
         service apache2 restart
```

Helpers for custom application can be defined in `/etc/tracer/applications.xml` & `~/.config/tracer/applications.xml`. If you have any objections to the described way how to restart it, please create an issue.

Interactive mode

Printing helper for specific application is handy but not for every situation. For instance it can be little awkward to call `tracer -s app_name` a lot of specific applications. That would be lot of boring and senseless typing so I am introducing interactive mode to you.

When you use `-i` or `--interactive`, tracer will print number next to every application. Then you will be asked for input. That is simple way how to iterate through helpers for all applications.

```
$[FrostyX  ~]-> sudo tracer -i
[1] gvim
[2] mpd
[3] dolphin
[4] apache2

Press application number for help or 'q' to quit
--> 2
* mpd
    Package:     media-sound/mpd
    Description: The Music Player Daemon (mpd)
    Type:        Daemon
    State:       mpd has been started by frostyx 23 hours ago. PID - 3751

    How to restart:
         service mpd restart

-- Press enter to get list of applications --
```

Verbose

Like most of UNIX programs, even tracer has `verbose` mode. It provides three levels of chattiness.

Non-verbose mode

```
$[FrostyX  ~]-> sudo tracer -s gvim
* gvim
    Package:     app-editors/gvim
    Description: GUI version of the Vim text editor
    Type:        Application
    State:       gvim has been started by frostyx 2 hours ago. PID - 8431

    How to restart:
        Sorry, It's not known
```

First verbose level

```
$[FrostyX  ~]-> sudo tracer -s gvim -v
* gvim
    Package:     app-editors/gvim
    Description: GUI version of the Vim text editor
    Type:        Application
    State:       gvim has been started by frostyx 2 hours ago. PID - 8431

    Affected by:
        gnome-base/gvfs
        x11-libs/libX11

    How to restart:
        Sorry, It's not known
```

Second verbose level

```
$[FrostyX  ~]-> sudo tracer -s gvim -vv
* gvim
    Package:     app-editors/gvim
    Description: GUI version of the Vim text editor
    Type:        Application
    State:       gvim has been started by frostyx 2 hours ago. PID - 8431

    Affected by:
        gnome-base/gvfs
            /usr/lib/gvfs/libgvfscommon.so
            /usr/lib/gio/modules/libgioremote-volume-monitor.so
            /usr/lib/gio/modules/libgvfsdbus.so
        x11-libs/libX11
            /usr/lib/libX11.so
            /usr/lib/libX11-xcb.so

    How to restart:
        Sorry, It's not known
```

Distro-specific candy

Fedora - DNF plugin

There is plugin for new fedora package manager - DNF. It calls tracer after every successful transaction. Please note that it checks only packages in actual transaction, so if you run `tracer` from command line, you can actually get longer list.

If you want this feature, install the plugin package. Please notice that there are two of them. For F21 and higher install

the `dnf-plugins-extras-tracer`. If you are still using F20, please install `dnf-plugin-tracer`, but be aware that this package is obsoleted and will be no new versions of it.

```
$[FrostyX  ~]-> sudo dnf update vim-X11
...
Running transaction
  Upgrading    : vim-common-2:7.4.179-1.fc20.i686                              1/6
  Upgrading    : vim-X11-2:7.4.179-1.fc20.i686                                 2/6
  Upgrading    : vim-enhanced-2:7.4.179-1.fc20.i686                            3/6
  ...

Upgraded:
  vim-X11.i686 2:7.4.179-1.fc20           vim-common.i686 2:7.4.179-1.fc20
  vim-enhanced.i686 2:7.4.179-1.fc20

You should restart:
  gvim

Done!
```

If you can't see tracer section in your output, make sure that in your `/etc/dnf/dnf.conf` is not `plugins=0` or specified pluginpath to different than default directory.

Error occured

Some weird error occured! What should I do? Please keep calm and read it. There should be information what can be wrong and how you can deal with it. For instance

```
frostyx@kubuntu:~$ sudo tracer
You are running unsupported linux distribution

Please visit https://github.com/FrostyX/tracer/issues
and create new issue called 'Unknown or unsupported linux distribution: Ubuntu' if␣
↪there isn't such.

Don't you have an GitHub account? Please report this issue on frostyx@email.cz
```

There is little possibility that you can encounter different type of error. Something like this

```
Traceback (most recent call last):
  File "/usr/local/bin/tracer", line 169, in <module>
    main()
  File "/usr/local/bin/tracer", line 56, in main
    if args.interactive: _print_all_interactive(processes)
  File "/usr/local/bin/tracer", line 88, in _print_all_interactive
    answer = raw_input("--> ")
```

It is python traceback. My apologies, you shouldn't see it. The best thing you can do, is openning new issue in tracer's issue tracker. Please describe how can I reproduce this issue or what did you do when error occured. Please post complete error message too.

Troubleshooting

Only root can use this application

As I described above, tracer works only with root permissions so far.

You are running unsupported linux distribution

Please read rest of that message. It describes what you can do

Developer Guide

Python

Tracer is written in Python and it is compatible with its both 2.7 and 3.x versions. Besides standard python interpreter, tracer requires its few packages:

- psutil - For getting informations about processes, memory, etc
- beautifulsoup - For parsing user defined xml files such as `data/applications.xml` and `data/rules.xml`

Coding style

If you want to contribute and write some code, I will strictly insist on these rules

1. Use **tabs** for indentation
   - Tabs are meaningful - 1 tab = 1 logical level
   - Everyone can view the one same tab with different width
   - Easier to work with
2. Use **spaces** for aligning
   - There is no way how to correctly align with tabs
3. Use `camelCase` for naming files, `CamelCase` for classes, `underscore_case` for methods and variables
4. There are no access modifiers, so use prefix _ for private things

There is an example code following our coding style

```
class Rpm(IPackageManager):

    def packages_newer_than(self, unix_time):
        packages = []
        for t in self._transactions_newer_than(unix_time):
            # Append every package in transaction into `packages`
            ...
        return packages

    def _transactions_newer_than(self, unix_time):
        ...
```

Localization

The default language for the program is english. For translations to other languages, we need contributors. If you are using Tracer and want to help us, you could translate it to languages, that you know. There are only few strings in the code, so its the matter of minutes.

Please see the Tracer on transifex.

Packaging

Tracer uses tito for managing versions and doing all RPM stuff. There is quote what tito is:

> "Tito is a tool **for managing RPM based projects** using git for their source code repository"

That means if you are fedora user, you can use prepared `tito` package, but else you probably would have to install it by your own.

Next, there is Makefile describing most of actions you might want to do.

In general, managing versions

```
make release

# For specific version use
tito tag --use-version X.Y.Z
```

Fedora

```
# Develop
make rpm-test                       # [1]
make rpm-try                        # [2]

# Create official package
make rpm                            # [3]
```

- [1] Create RPM package from last commit.
- [2] Same as [1] but additionally install it.
- [3] Create SRPM from last tag and ask copr to build RPM packages and distribute it through frostyx/tracer repository. There is `.repo` files for F19 and F20

Get Tracer

There are few ways how to get tracer. Recommended is installing it through linux distribution package, but so far there are only few supported ditributions and even less of them has tracer packaged. If you are unlucky and can't find your system in the list, you can do three things.

1. Create and maintain tracer package for your ditribution
2. Request me to do it for you (but I don't promise that I will)
3. Use tracer from git

Fedora

Fedora is intended as the primary system, so there shouldn't be a problem. You can simply install tracer using

```
dnf install tracer
```

Please note that for DNF also exists plugin which calls tracer after every successful transaction. You can install it using

```
dnf install dnf-plugins-extras-tracer
```

Take a look into User Guide at *Fedora - DNF plugin*.

Enterprise Linux 7 (CentOS, ect.)

Tracer can be install from EPEL

```
yum install epel-release
yum install tracer
```

Gentoo

So far I have `tracer.ebuild` in my personal overlay. Please take note that *is not* properly packaged - it just clone the git repository. However it has one relevant advantage and that is that it takes care about dependencies.

```
layman -a frostyx
emerge tracer
```

ArchLinux

An unofficial tracer package can be found in the AUR: https://aur.archlinux.org/packages/tracer

Git

You can download the code by running this command:

```
git clone git@github.com:FrostyX/tracer.git
```

1) Manuall installation

First of all check tracer's requirements and install them.

Now you should be able to run it by `tracer/bin/tracer.py`, but becaue it is so unhandy I will recommend you to make symlink into `$PATH` directory. For instance

```
sudo ln -s tracer/bin/tracer.py /usr/local/bin/tracer
```

and then run it just by `tracer` command.

2) Automated installation

Previous paragraph can be done by running

```
sudo pip install ./tracer/
# or
pip install --user ./tracer/
```

Tracer

DESCRIPTION

Tracer determines which applications use outdated files and prints them. For special kind of applications such as services or daemons, it suggests a standard command to restart it. Detecting whether file is outdated or not is based on a simple idea. If application has loaded in memory any version of a file which is provided by any package updated since system was booted up, tracer consider this application as outdated.

OPTIONS

GENERAL

```
--version             print program version

-h, --help            show this help message and exit

-q, --quiet           do not print additional information

-v, --verbose         print more informations. Use -v or -vv
```

MODES

```
--helpers             not list applications, but list their helpers

-i, --interactive     run tracer in interactive mode. Print numbered
                      applications and give helpers based on numbers

-s app_name [app_name ...], --show app_name [app_name ...]
                      show helper for given application

-a, --all             list even session and unrestartable applications

--daemons-only, --services-only
                      list only daemons/services

--hooks-only          do not print traced applications, only run their hooks

-t TIMESTAMP, --timestamp TIMESTAMP
                      since when the updates should be

-n, --now             when there are specified packages, dont look for time
                      of their update. Use "now" instead
```

USERS

```
-u username, --user username

-r, --root

-e, --everyone
```

DEBUG

```
--show-resource=<option>
                      options: packages | processes | rules | applications | system
                      dump informations that tracer can use
```

EXAMPLES

```
Show your applications which needs restarting (basic usage)
    tracer

Show informations about application
    tracer --show mysqld

Show even affected files of the application
    tracer --show mysqld -vv

In interactive mode show all applications modified only through packages changed␣
↪since timestamp
    tracer -iat 1414248388.04
```

Tracer API

Tracer provides a small interface that you can use in your application. It consist from `Query` class which is able to ask Tracer for various informations and provide you a results. And from data structues from which results consists.

Surely you could access internal classes and methods on your own, but there is a risk that they will change in the future. In API could be a little changes too, but they will be documented

Quering

class `tracer.`**`Query`**(*tracer=<class 'tracer.resources.tracer.Tracer'>*)

Provide API for Tracer quering operations. They are executed kind of lazily, so running the operation will return just an wrapper class with `get()` method.

Example:

```
from tracer.query import Query
q = Query()
q.affected_applications().get()
```

Note: Some quering methods can require root permissions

affected_applications(*user=None*)
Return list of applications which use some outdated files

from_packages(*packages*)
List of `Package` that only should be traced

now()
Pretend that specified packages have been updated just now. Benefit of this is absolutely no need for openning the package history database

Hooks

Tracer also provides API for user-defined hooks. They can be defined as a simple functions decorated by `@hooks.match("app_name")`. Tracer will search for them in directories `~/.config/tracer/hooks/` and `/etc/tracer/hooks/`. Such hook will be called when tracer determines, that linked application needs restarting.

`tracer.hooks.`**match**(*apps*)
Decorator for tracer hooks.

Example:

```
from tracer import hooks

@hooks.match("foo")
def hook_app():
    print("Hey, application foo was found")
```

Note: You can match multiple applications by calling `@hooks.match` with list of them.

Note: If you want to run tracer's hooks and print no other output, use `tracer --hooks-only`

Exit codes

In some use-cases you may want to examine Tracer's results through exit codes (also known as status codes). See their meanings:

1-99	Error exit codes
0	No affected applications
101	Found some affected applications
102	Found some affected daemons
103	Session restart needed
104	Reboot needed

Data structures

Follows list of classes which quering results may consist from. Not all their properties are covered within API. If your use case requires some which are not listed below, please let me know to cover them in API too.

Packages

class `tracer.`**`Package`**(*name*, *modified=None*)
Represents linux package

`modified` = None
UNIX timestamp of the modification

`name` = None
Name of the package

Applications

class `tracer.`**`Application`**(*attributes_dict*)
Represent the application defined in `applications.xml`

Parameters

- **`name`** (*`str`*) – The name of the application
- **`type`** (*`str`*) – See `Applications.TYPES` for possible values
- **`helper`** (*`str`*) – Describes how to restart the applications
- **`note`** (*`bool`*) – Provides additional informations to the `helper`
- **`ignore`** (*`bool`*) – If `True`, the application won't be printed
- **`processes_factory`** (*`Processes`*) – Class providing list of running processes

`helpers`
Return the list of helpers which describes how to restart the application. When no `helper_format` was described, empty list will be returned. If `helper_format` contains process specific arguments such a `{PID}`, etc. list will contain helper for every application instance. In other cases, there will be just one helper in the list.

`instances`
Return collection of processes with same name as application. I.e. running instances of the application

class `tracer.`**`Process`**(*pid=None*)
Represent the process instance uniquely identifiable through PID

For all class properties and methods, please see http://pythonhosted.org/psutil/#process-class

Bellow listed are only reimplemented ones.

`children`(*recursive=False*)
The collection of process's children. Each of them casted from `psutil.Process` to tracer `Process`.

`exe`
The absolute path to process executable. Cleaned from arbitrary strings which appears on the end.

`parent`()
The parent process casted from `psutil.Process` to tracer `Process`

`username`()
The user who owns the process. If user was deleted in the meantime, `None` is returned instead.

CHAPTER 2

Feedback

Please report any bugs or feature requests to issues on this repository. Pull requests are also welcome, but please visit *Developer Guide* first. If you rather want a talk or something, you can find me on `#gentoo.cs` or `#fedora-cs @freenode` or you can mail me to frostyx@email.cz.

Index