[redacted]

Notes on path

Table of Contents

With the recent (2019-09-01) changes and creation of `make install` command it is easy to get lost.

If you install the Proxmark tools with `make install`, they will go under the prefix `/usr/local/` but if you install the tools from your distro, there are chances the path is `/usr` so you'll have to adapt the paths presented here.

Installed elements

^Top

Binaries

^Top

The main Proxmark3 executables / shellscripts will be copied to

```
/usr/local/bin/
```

- executables: `proxmark3`
- scripts: `pm3`, `pm3-flash`, `pm3-flash-all`, `pm3-flash-bootloader`,

`pm3-flash-fullimage`

Some more executable / scripts will be copied to

```
/usr/local/share/proxmark3/tools
```

- executables: `mfkey32`, `mfkey32v2`, `mfkey64`, `nonce2key`
- scripts: `pm3_eml2lower.sh`, `pm3_eml2upper.sh`, `pm3_mfdread.py`, `pm3_mfd2eml.py`, `pm3_eml2mfd.py`, `findbits.py`, `rfidtest.pl`, `xorcheck.py`

Firmwares

^Top

The recovery / firmware files will be copied to

```
/usr/local/share/proxmark3/firmware
```

- Proxmark3 firmware: `bootrom.elf`, `fullimage.elf`, `proxmark3_recovery.bin` (used for JTAG)
- SIM firmware: `sim013.bin`, `sim013.sha512.txt`

Traces

^Top

Proxmark3 client has a lot of sample trace files for many different low frequency tags. They will be copied to

279 lines (188 sloc) | 9.1 KB Raw Blame

JTAG-related stuff

^Top

JTAG configurations and helper scripts for OpenOCD will be copied to

`/usr/local/share/proxmark3/jtag_openocd`

Proxmark3 client files: dictionaries

^Top

Dictionaries used by the client will be copied to

```
/usr/local/share/proxmark3/dictionaries
```

Here you find the default dictionaries used for commands like `hf mf chk`, `hf mf fchk`, `lf t55xx chk` A dictionary file is a text based file with one key per line in hexadecimal form. The length of the key is decided by the Proxmark3 client for the different commands. All chars afterwards on line is ignored. if key isn't a hex number, the key is ignored.

- t55xx, Mifare Ultralight/NTAG - uses 4 hexbytes (11223344)
- Mifare classic uses 6 hexbytes (112233445566)
- iClass uses 8 hexbytes (1122334455667788)

See here how to add your own dictionaries.

Proxmark3 client files: cmd scripts

^Top

Cmd scripts used by the client will be copied to

```
/usr/local/share/proxmark3/cmdscripts
```

See here how to add your own cmd scripts.

Proxmark3 client files: Lua libraries and scripts

^Top

Lua libraries and scripts used by the client will be copied to

```
/usr/local/share/proxmark3/lualibs
/usr/local/share/proxmark3/luascripts
```

`lualibs` contains the supporting lua libraries used for lua scripts. Basically reused functions in a lua file like converting string to hex etc.

See here how to add your own Lua scripts.

Proxmark3 client files: various resources

^Top

Various resources used by the client will be copied to

```
/usr/local/share/proxmark3/resources
```

It comprises the needed files for commands like hardnested, fido, EMV, iClass.

See here how to add your own resources.

Documentation

^Top

Documentation will be copied to

```
/usr/local/share/doc/proxmark3
```

User files

^Top

The client will make use of a personal directory `~/.proxmark3` (or more precisely `$HOME/.proxmark3`)

.history / log files

^Top

We have now a rolling log file, created new per day. All these logfiles and the history file are now located at

```
~/.proxmark3/history.txt
~/.proxmark3/log_YYYYMMDD.txt
```

Proxmark3 client files and traces

^Top

If you wants to add scripts, dictionaries or other resources, you can use the same structure as the installed directory structure and add your own files there, e.g.

```
~/.proxmark3/cmdscripts/mycmdscript.cmd
~/.proxmark3/dictionaries/mydict.dic
~/.proxmark3/luascripts/myluascript.lua
~/.proxmark3/resources/oids.json
~/.proxmark3/traces/mylftrace.pm3
```

If you add a file with the same name as the file provided with the Proxmark3 installation, it will take precedence.

See also Scripts on how to write your own scripts.

Searching files

^Top

With the directory structure explained above, the client applies some heuristics to find its files or the files you specified in command line.

TL;DR

^Top

It adds the expected suffix if you didn't provide it yet, then it looks (by order of precedence):

1. in the current directory, or in the path if you provided also a path, so it works with autocompletion
2. in the `~/.proxmark3` directory structure as seen above, so it works with your stuffs
3. in the repo directory structure, so it works as usual if used from the Git repo
4. in the installed directory structure, so it works when installed

Gory details

^Top

The client is using *searchFile* (in *client/fileutils.c*) when calling a Proxmark3 command with a filename or when the client needs to find its files. *searchFile* takes as argument a relative path *pm3dir*, a file to search and possibly a *suffix*.

So for example when using *searchFile* over a filename supposed to be a dictionary file, it's called with *pm3dir=dictionaries/* and *suffix=.dic*. When a user provides a filename (including possibly a path), *searchFile* will search different locations and return as soon as a file is found:

- Add the suffix if the suffix is not yet present, so: *foo -> foo.dic* and *foo.dic -> foo.dic*
- If the filename is an absolute path (*/tmp/foo.dic*), take it as it is, try to access the file and return.
- If the filename is an explicit relative path (*./foo.dic*), take it as it is, try to access the file from the current directory and return.
- Try to find the filename as relative path (*foo.dic -> ./foo.dic*), so filenames provided by CLI autocompletion work as expected.
- Try to find the filename in the *pm3dir* relative to the user directory *$HOME/.proxmark3* (*foo.dic -> ~/.proxmark3/dictionaries/foo.dic*)
- Try to find the filename in the *pm3dir* relative to where the client binary is when in the repo configuration (*foo.dic -> $(path_to_dir_of_proxmark3_bin)/dictionaries/foo.dic*), so when the client is executed from a repo workdir, filenames are searched in the expected location.
- Try to find the filename in the *pm3dir* relative to where the client binary is when in the installed configuration (*foo.dic -> $(path_to_dir_of_proxmark3_bin)/../share/proxmark3/dictionaries/foo.dic* which resolves to e.g. */usr/share/proxmark3/dictionaries/foo.dic* or */usr/local/share/proxmark3/dictionaries/foo.dic*), so when the client is executed from a repo workdir, filenames are searched in the expected location.

Scripts

^Top

You can provide your own lua or cmd scripts. Look at existing scripts for ideas how to create your own scripts.

Proxmark command script (.cmd)

^Top

For cmd scripts, the command line scripts, the client can run a text file containing Proxmark3 commands.

A samplefile could be like this.

```
$> cat myscript.cmd

rem running some HF-based info commands
hf 14a info
hf mfu info
rem done
```

You call it with:

`$> pm3 -s myscript.cmd`

The client will execute each one of the commands in order and then exit. There is also a possibility to remain in the client afterwards with the -i parameter:

`$> pm3 -s myscript.cmd -i`

You can place it in `~/.proxmark3/cmdscripts/` and it will be found automatically.

You can skip the script file extension, it works equally well with.

`pm3 -s myscript`

Shebangs (on *nix)

^Top

You can also use the magic of shebangs to make an executable script, e.g. taking the example above, we can write:

```
$> cat myscript.cmd

#!/usr/bin/env -S pm3 -s
hf 14a info
hf mfu info

$> chmod +x myscript.cmd
$> ./myscript.cmd
```

And it will be executed invoking the `pm3` script.

Use the following if your script is intended to work offline.

```
#!/usr/bin/env -S proxmark3 -s
```

Beware the shebang trick is not available on all the platforms, it requires your `env` to accept the `-S` switch.