WiPal

IEEE 802.11 traces manipulation software

This manual is for WiPal (version 1.2, updated 26 March 2008.)

WiPal is a piece of software dedicated to IEEE 802.11 traces manipulation. It comes as a set of programs and a C++ library. A distinctive feature of WiPal is its merging tool, which enables merging multiple wireless traces into a unique global trace. WiPal's key features are flexibility, ease of use, and efficiency.

Table of Contents

1 The programs

This part documents the programs WiPal features. Looking for a specific command? See [Program index], page 14.

1.1 Invocation

WiPal's programs all use the same invocation scheme:

```
wipal-<command> [options] [inputs] [outputs]
```

The command line may include no options and, depending on the program, there may be no inputs or no outputs. Most programs expect at least one input however. See the specific documentation for each program in order to know how many inputs and outputs each program expects.

Inputs, outputs, and options may be mixed on the command line, e.g.:

```
wipal-merge -n -P input1.pcap input2.pcap output.pcap
wipal-merge input1.pcap input2.pcap output.pcap -P -n
wipal-merge input1.pcap -n input2.pcap -P output.pcap
...
```

are all equivalent.

WiPal's programs use `getopt`(3) to parse options, so they only have short options (no long options) composed of a dash followed by a letter (e.g. '`-a`', '`-t`', etc.) Option letters *always* have the same meaning whatever the program. All options are not available for all programs though (some options do not make sens with some programs). For instance, '`-P`' always means the invoked program should consider frames with non-zero Prism fields as invalid. In order to know which options a program accept, use the '`-h`' option.

Finally, some options expect an extra argument right after they are provided:

```
wipal-test-uniqueness -a hsh input.pcap
                         ^^^
                         This is not an input
```

1.1.1 Available options

'`-8`' When comparing two packets, only compare IEEE 802.11 frames. Do *not* compare Prism or PCAP headers.

'`-a`' See Section 1.7 [Unique frames], page 7. Specify which attributes the program must use to identify unique frames. An attribute specifier must follow this option on the command line. To see a list of valid attribute specifiers, use the '`-h`' option.

'`-b`' When comparing two packets, only compare packet bytes. Do *not* compare PCAP headers.

'`-c`' Do not print column headers. This is the default when standard output is not a TTY.

'`-C`' Do print column headers. This is the default when standard output is a TTY.

‘`-d`’ When comparing two packets, compare everything: PCAP headers and packet bytes. This is the default.

‘`-e`’ In table outputs, do *not* use a column to report error values. This is the default.

‘`-E`’ In table outputs, *do* use a column to report error values.

‘`-h`’ Help. Print a short summary describing how one should invoke the program, which options it accepts, and possibly which attribute specifiers are accepted for option ‘`-a`’.

‘`-i`’ In table outputs, do *not* print frame indices.

‘`-I`’ In table outputs, *do* print frame indices. This is the default.

‘`-m`’ Specify the address mapping file. An address mapping file maps 6 bytes MAC identifiers to 32 bit integers. The only purpose of such a mapping is to improve performances. The file is just a plaintext file with an integer and a MAC identifier on each line.

A filename should follow this option. The file might not exist (in which case it will be created). If it exist, it might be extended, but will not be truncated.

When not specified, the ‘`mapping`’ filename is used.

‘`-n`’ Consider Prism headers are little endian. This is the default when the corresponding PCAP file is little endian. Note that some broken traces are big endian yet have little endian Prism headers. Thus this option.

‘`-N`’ Consider Prism headers are big endian. This is the default when the corresponding PCAP file is big endian.

‘`-p`’ In Prism headers, do not consider noise fields have a special meaning. This is the default.

‘`-P`’ In Prism headers, consider non-null noise fields indicate a PHY error, and thus an invalid frame. Such frames will be ignored, e.g. with `wipal-cat` they will not appear in the output.

This option implicitly implies the input trace is composed of Prism headers (as PCAP link type).

‘`-q`’ Quiet. Produce minimal output.

‘`-r`’ Blacklist a given reference frame. The reference frame will then be ignored and will not be used during synchronization. See Section 1.6 [Synchronization], page 6.

A reference frame identifier must follow this option, e.g. `42-51` (indicating the reference frame composed of the unique frames 42 and 51).

You may use this option multiple times, e.g.

```
wipal-merge -r 42-51 -r 666-505 \
    input1.pcap input2.pcap output.pcap
```

will blacklist both references `42-51` and `666-505`.

‘`-t`’ When comparing two packets, only compare IEEE 802.11 frames, along with some timestamps (e.g. PCAP time, Prism MAC time, etc). Which timestamps

are used deped on the traces' link types. Compare time values with a precision of 106 microseconds (that is, assume two values are equal when they are spaced by less than 106 microseconds).

'`-u`' In table outputs, do *not* print microsecond timestamps. This is the default.

'`-U`' In table outputs, *do* print microsecond timestamps.

'`-v`' Display the program's version (actually the version of the WiPal's package the program come from).

1.1.2 Input syntax

Basic usage

You may provide the name of a PCAP trace file as input.

```
wipal-cat input.pcap output.pcap
```

Advanced usage

You may provide the name of several PCAP traces separated with columns (do not include any space). This tells the program to consider the concatenation of each trace as a single input.

```
wipal-cat input1.pcap:input2.pcap:input3.pcap output.pcap
```

will put into '`output.pcap`' the content of '`input1.pcap`', followed by the content of '`input2.pcap`' and then '`input3.pcap`'.

Every programs understand this syntax. Note that specifying multiple traces with columns makes no sense for outputs:

```
wipal-cat input1.pcap:input2.pcap output1.pcap:output2.pcap
```

will concatenate '`input1.pcap`' and '`input2.pcap`' into a single file named '`output1.pcap:output2.pcap`'!

1.2 Concatenation (and Prism noise filtering)

One may concatenate traces using the `wipal-cat` command. It takes exactly one input and one output. It may be useful to recombine a trace that was split, or filter out frames with Prism noise (using the '`-P`' option).

```
wipal-cat in.pcap out.pcap
wipal-cat foo.pcap.0:foo.pcap.1 foo.pcap
wipal-cat -P in.pcap out.pcap
wipal-cat -P bar.pcap.0:bar.pcap.1:bar.pcap.2 bar.pcap
```

The first example just copies '`in.pcap`' into '`out.pcap`'. Note that the two files might be different at the byte level, e.g. if '`in.pcap`' is big endian and the program is run on a little endian machine.

The second example concatenate '`foo.pcap.0`' and '`foo.pcap.1`' and put the result into '`foo.pcap`'.

The third example copies '`in.pcap`' into '`out.pcap`' but removes frames that have a non-zero noise field in their Prism headers.

The fourth example both concatenates traces while filtering noisy frames out.

1.3 Comparisons

One may test two PCAP traces for equivalence using the `wipal-cmp` command. The default is to compare every bits of information (PCAP headers plus packet bytes) but you may change this behavior using the '`-8`', '`-b`', or '`-t`' options. Note that this is different however to using `diff` or `cmp` since traces with different endianness may contain the same packets.

By default `wipal-cmp` produces a report indicating either that traces are equal, either which packet is the first to mismatch. Use '`-q`' if you are only interested in the program's exit status and do not want to produce any output.

e.g.:

```
wipal-cmp foo.pcap bar.pcap
wipal-cmp -q foo.pcap bar.pcap
wipal-cmp -q -8 in1.pcap.0:in1.pcap.1 in2.pcap
...
```

1.4 Sub-traces

One may extract sub-traces of PCAP traces using `wipal-extract-subtrace`, `wipal-extract-transmitter`, or `wipal-extract-bssid`.

`wipal-extract-subtrace`
: takes two dates and a PCAP trace as inputs, and produces one output. Unfortunately, it does not support any option currently.

`wipal-extract-transmitter`
: takes a MAC address and a PCAP trace as input, and produces one output. Its output contains the frames from its input that were transmitted by the given address. Note that the command looks at *transmitters*, not *originators*, e.g. the transmitter of a data frame that crossed the distribution system is the output access point, not the original sender. Also note that some frames do not contain information regarding their transmitters (e.g. MAC acknoledgements) and therefore cannot appear in the output, even if they were effectively sent by the given address.

`wipal-extract-bssid`
: works as `wipal-extract-transmitter`, but the MAC address represents a BSSID and the command extracts frames that belong to the corresponding BSS. Again, note that some frames do not contain information regarding their BSS. These frames therefore cannot appear in the output, even if they were effectively belonging to the given BSS.

e.g.:

```
wipal-extract-subtrace 2007-01-01 2008-01-01 \
        in.pcap.0:in.pcap.1 out.pcap

wipal-extract-subtrace \
        "2004-Aug-30 16:59:39.789221" "2004-Aug-30 16:59:39.929872" \
        kalahari-ath2 subtrace.pcap
```

```
wipal-extract-transmitter 71:19:9f:6f:71:33 in.pcap out.pcap
wipal-extract-bssid       9b:d2:d7:7f:aa:63 in.pcap out.pcap
```

1.5 Merging

One may merge two IEEE 802.11 traces into one using the `wipal-merge` command.

Use the '`-h`' option to have a description of the command's syntax. It takes two inputs and produce one output. When ran, the merging process starts by synchronizing precisely both inputs (see Section 1.6 [Synchronization], page 6). Then both traces are merged and special care is given not to re-order packets or account duplicate packets twice in the output (that is, packets that are present in both traces appear only once in the output).

This command expects PCAP traces with either Prism headers, AVS headers, Radiotap headers, or raw IEEE 802.11 frames as link type. The '`-p`' and '`-P`' options only work with Prism headers. The following timestamps are used:

IEEE 802.11 frames
: PCAP timestamps,

Radiotap headers
: Radiotap headers' `tsft` fields. The command will fail with Radiotap headers that do not contain such fields,

AVS headers
: AVS headers' `mactime` fields,

Prism headers
: Prism headers' `mactime` fields.

e.g.:

```
wipal-merge a.pcap b.pcap output.pcap
wipal-merge -P -n foo-ath2.0:foo-ath2.1 bar-ath2 foo-bar-ath2
...
```

1.6 Synchronization

In order to merge two IEEE 802.11 traces WiPal needs to synchronize them precisely. In order to do so, it first identifies some frames that appear in both inputs. These are reference frames. It uses these frames to model clock desynchronization among the traces. It then update the first trace's timestamps so they are synchronized with the second trace.

One may use the `wipal-synchronize` command to synchronize two traces. It takes two inputs and produce one output. The output contains the same packets as the first input, but with synchronized timestamps.

To extract reference frames WiPal extract some specific frames called *unique frames* (see Section 1.7 [Unique frames], page 7) from both input traces and then intersect the two obtained sets. One may use the `wipal-intersect-unique-frames` command to get the result of this operation (i.e. the list of reference frames used for synchronization of two traces).

WiPal's synchronization process synchronizes reference frames before it synchronizes other frames. One may get the result of this operation using the `wipal-synchronize-unique-frames` command.

e.g.:

```
wipal-intersect-unique-frames -n -P foo.0:foo.1:foo.2 bar.0:bar.1
wipal-synchronize-unique-frames -n -P foo.0:foo.1:foo.2 bar.0:bar.1
wipal-synchronize -n -P foo.0:foo.1:foo.2 bar.0:bar.1 foo-sync
```

1.7 Unique frames

A frame is said to be unique when it appears in the air once and only once for the whole duration of a trace. WiPal's unique frame extraction process is an important stage of its trace synchronization process. WiPal considers all beacon frames and all non-retransmitted probe responses as unique frames.

One may use the `wipal-extract-unique-frames` command to get a list of the unique frames that compose a trace. Run `wipal-extract-unique-frames -h` to get its invocation syntax.

In practice, WiPal does not extract and load full unique frames into memory. This would slow the process down and require an excessive amount of memory. The default is to work on MD5 frame hashes when WiPal was compiled using OpenSSL. When compiled without OpenSSL, WiPal only extracts a subset of frame fields. We call the pieces of information WiPal extracts to identify unique frames "frame attributes", or sometimes "frame identifiers".

You may specify frame attributes to use with the '`-a`' option. In practice, the difference in speed and memory consumption between attributes is negligible. There is an important difference between attributes, though. With some attributes, different unique frames may yield identical attributes (collisions). This is of course an undesirable behavior.

One may check that a given trace's unique frames are really unique w.r.t. unique frame attributes using the `wipal-test-uniqueness` command. This command finds collisions inside its input traces. You might specify different frame attributes using the '`-a`' option.

e.g.:

```
wipal-test-uniqueness -P -a timestamp foo.pcap.1:foo.pcap.2
wipal-extract-unique-frames -P foo.pcap.1:foo.pcap.2 > foo-unique.txt
```

1.8 Duplicate data frames

One may use the `wipal-find-data-dups` command to search some invalid data frames. It looks into traces on a per-sender basis for successive duplicate data frames (it only considers non-retransmitted frames). Such cases should not occur in theory - as it ignores retransmissions, successive data frames from the same sender should at least show variations in their sequence numbers. Surprisingly, some traces contain such anomalies: identical data frames that are not retransmissions and are only spaced by a few milliseconds. We have no explanations why some datasets exhibit those phenomena.

e.g.:

```
wipal-find-data-dups foo.pcap.0:foo.pcap.1:foo.pcap.2
```

1.9 Statistics

`wipal-stats` produces statistics concerning its given input PCAP traces. For the moment, this program is not finalized and its interfaces are subject to changes. Things will get more stable in future releases of WiPal. Consequently, there is no documentation for this command yet.

1.10 Undocumented programs

WiPal's `configure` script has two options '`--enable-probe-stats`' and '`--enable-wit-import`'. These options enable the build of several programs, namely `wipal-probe-stats`, `wit-create-datafiles`, `wit-create-tables-and-load-data`, and `wit-import`. By default the build of those programs is *disabled*.

Those are legacy programs that were useful to somebody once, yet are incomplete and flawed. They will *not* be updated later, and are *not* documented here. Build and use at your own risks!

2 The library

A C++ library also compose WiPal. WiPal programs all use this library. At a low level it provides various convenience tools (PCAP file input/output, random access to PCAP traces, support for various static C++ techniques, etc.) At an upper level it provides a generic IEEE 802.11 frame parser that is easy to customize and re-use. Finally, it provides various mechanisms to synchronize and merge PCAP traces directly from C++ code.

The library is called `libwipal` and its headers are located in `$(prefix)/include/wipal`. You should be able to include them as follows:

```
#include <wipal/pcap/descriptor.hh>
// ...
```

You will then need to provide the '`-lwipal`' option to your compiling/linking tools.

The main documentation for this library is provided as a Doxygen documentation. It should be installed into WiPal's package data directory, into the '`doxygen`' subdirectory. By default this gives '`/usr/local/share/wipal/doxygen/`'. This documentation is however a bit messy, and lacks some parts. The best entry point to learn how to use the library is to look at some of WiPal's tools' source code (e.g. into '`src/misc/wipal-find-data-dups.cc`'). You may also want to have a look at WScout which is another program that uses WiPal (some versions of WScout embeds WiPal under the name *trace-tools*).

3 FAQ

3.1 What systems does WiPal support?

WiPal was mostly designed using standard C++ and portable libraries. It however uses a few GCC extensions. Yet WiPal should run fine on most systems (e.g. GNU/Linux, WhateverBSD, Mac OS, Windows, ...).

WiPal is however exclusively tested on Debian GNU/Linux (amd64 and, to a lower extent, powerpc). Which means you might experience problems on other systems, which the developers might not be aware of. In this case, please give feedback to them so they can fix it. Anyway, there should be no major obstacle to WiPal's portability.

3.2 What are WiPal's requirements?

WiPal needs:

- GCC.
- The Boost C++ libraries. More specifically:
  - array,
  - date_time,
  - foreach,
  - format,
  - conversion/lexical_cast,
  - optional,
  - preprocessor
  - smart_ptr,
  - tokenizer,
  - tuple,
  - variant.
- The GNU MP Bignum Library,
- OpenSSL.

3.3 How do I install WiPal?

WiPal's packaging follows the GNU conventions. An installation documentation is provided in the '`INSTALL`' file in the package's root directory. However, with a standard system, the following commands should do the trick:

```
mkdir _build
cd _build
../configure
make
make install-strip
make check
```

On some systems, you might have to customize the '`configure`' script's invocation. e.g.:

```
mkdir _build
cd _build
../configure CPPFLAGS=-I/foo/bar/libgmp
make
make install-strip
make check
```

3.4 Are there any options to optimize WiPal when building it?

You might want to compile WiPal with the `NDEBUG` preprocessor symbol defined. If you use GCC you might also want to use its `-O3` option. You can do that by running '`configure`' with the following options:

```
./configure CPPFLAGS=-DNDEBUG CXXFLAGS=-O3
```

3.5 Gee! WiPal's compilation takes long and requires a lot of memory!

WiPal heavily uses static C++ mechanisms and a full build requires instanciating many templates. This results in a long build process that requires much memory. You may disable some template instanciations to have a faster and lighter build process. This will however remove some features at the end. You may invoke `configure` with the following options:

'`--enable-linktypes=LT1:LT2:...`'
: will only enable the listed PCAP link types when compiling WiPal. The available link types are:

  `IEEE802_11`
  : raw IEEE 802.11 frames,

  `IEEE802_11_RADIO`
  : Radiotap headers,

  `IEEE802_11_RADIO_AVS`
  : AVS headers,

  `PRISM_HEADER`
  : Prism headers.

'`--enable-attributes=A1:A2:...`'
: will only enable the listed unique frame attributes (see Section 1.7 [Unique frames], page 7) when compiling WiPal. The list's first attribute is the default one (when '`-a`' is not provided on the command line). Available attributes are:

  - tmp
  - seq_tmp
  - dst_tmp
  - src_tmp
  - bss_tmp
  - src_bss_tmp

- seq_bss_tmp
- seq_dst_bss_tmp
- seq_src_bss_tmp
- hsh (requires OpenSSL)

If you know you are going to need only one PCAP link type (e.g. Prism headers), and you do not want to test various attributes, a good choice might be:

```
./configure --enable-linktypes=PRISM_HEADER --enable-attributes=seq_bss_tmp
```

which will only instanciate one template configuration for each WiPal utility.

3.6 Do WiPal's tools have a verbose mode to report extra information about their operation?

There is no such options that can be activated dynamically. You might want however to compile WiPal with the `WP_ENABLE_INFO` preprocessor symbol defined. This will enable the printing of some extra information in some tools as they run (e.g. number of processed frames, synchronization error, etc.). Invoke the '`configure`' script with the following option:

```
./configure CPPFLAGS=-DWP_ENABLE_INFO
```

Note however that this may slow some tools down and may require more memory.

3.7 You say WiPal is flexible and customizable. Is there a way to customize WiPal's tools beyond the options they propose?

Yes! But this requires recompiling WiPal's tools, and sometimes modifying a few lines of their source code.

- You may change WiPal's linear regression window (for trace synchronization) by defining the `WP_LRSYNC_WINDOW_SIZE` macro symbol. Use the `CPPFLAGS` environment variable for this. The default value is 3.

  e.g.:

  ```
  ./configure CPPFLAGS='-DWP_LRSYNC_WINDOW_SIZE=42'
  ```

- You may change the windowed merging algorithm's window size by defining the `WP_WMERGE_WINDOW_SIZE` macro symbol. Use the `CPPFLAGS` environment variable for this. The default value is 3.

  e.g.:

  ```
  ./configure CPPFLAGS='-DWP_WMERGE_WINDOW_SIZE=42'
  ```

- You may change the frame attributes (i.e. frame identifiers) to use in tools that do not support the '`-a`' option by modifying a few lines of their source code. This generally needs changing an include and a typedef, e.g.:

  ```
  -#include <wipal/wifi/frame/unique_id/seqctl_bssid_timestamp.hh>
  +#include <wipal/wifi/frame/unique_id/seqctl_source_bssid_timestamp.hh>

  // ...

  -      typedef wifi::frame::seq_bss_tmp_id              unique_id;
  +      typedef wifi::frame::seq_src_bss_tmp_id          unique_id;
  ```

3.8 'configure' complains it did not find library *X*?

Either library *X* is not installed on your system, either your system is not properly configured, so the library cannot be found.

You may use the `CPPFLAGS` and `LDFLAGS` variables to correct this behavior.

e.g., run

```
./configure CPPFLAGS=-I/custom/path/include \
            LDFLAGS=-L/custom/path/lib
```

3.9 'configure' complains it found library *X*'s headers, but is unable to link?

Most probably library *X* is installed but its binaries are in a non-standard place. Use the `LDFLAGS` variable as described previously.

3.10 'configure' complains library *X*'s headers are unusable, despite successful linking?

Most probably library *X* is installed but its headers are in a non-standard place. Use the `CPPFLAGS` variable as described previously.

3.11 Do you have a list of WiPal's bugs?

No. We are not aware of any serious bug in WiPal. We take a special care at testing WiPal with an automated test suite. Do not hesitate to report unknown bugs to the package's maintainers. We will hunt them.

With some tools, you might however encounter some strange behaviors when providing invalid inputs (e.g. running `wipal-find-data-dups a:b` with '`b`' having a link type different from '`a`'). Consider that as a "feature"! ;-)

3.12 I have found a bug, what should I do?

Report it to the package's maintainers.

3.13 I would really love having feature *X* implemented!

Give feedback to the package's maintainers about the features you want. We might not have the time to implement them, yet it is important for us to know when important features are missing.

Regarding features you miss, you are greatly encouraged to contribute to WiPal. Again, contact the package's maintainers so they can help you implement new features.

3.14 I have a question this file did not answer!

Mail the package's maintainers.

Index

Program index

Concept index