pySim - Feature #5114
Introduce proper TLV parser + constructor
04/11/2021 01:07 PM - laforge

Status: New
Priority: High
Assignee:
Category:
Target version:
Spec Reference:

Description
With 'construct' we now have the tools to deal with various hand-coded binary formats.

What we're still missing is a proper TLV parser/encoder that does what we need:

- support single-byte tag + single-byte value
- support BER-TLV with variable-length length fields
- discovering unknown TLVs is not an error and should just work
- a way to specify human-readable description text for each tag
- a way to specify a brief name/identifier which is to be used in the parse result to describe the field
- ability to call a sub-decoder
  - for nested TLVs, or
  - to further decode the value part in whatever format it may be

Once we have this in place, we should migrate any existing encoders/decoders over to this new TLV codebase.

We've so far explored

- pytlv
  - lacks support for BER-TLV
  - doesn't support parsing unknown tags
  - name/id mapping needs to be glued on top
- utlv
  - lacks support for BER-TLV
  - natively supports tag names via its tag_map concept

History

#1 - 04/11/2021 01:13 PM - laforge

There's also mitshell (https://github.com/mitshell/card/blob/master/card/utils.py) which

- supports BER-TLV
- supports multiple occurrence of the same tag
- doesn't have a concept of names or sub-parsers
- dates back to python2 days, before there were bytes() or bytearray()

Furthermore, there's cyberflex_shell (https://github.com/henryk/cyberflex-shell/blob/master/TLV_utils.py), which

- supports BER-TLV
- supports multiple occurrence of the same tag
- supports tag-specific sub-parsers, see around line 291
- doesn't have the notion of names/identifiers for each tag

Neither really do what we'd want, but they each have some nice features.

#2 - 04/11/2021 01:21 PM - laforge

There's also https://github.com/philipschoemig/BER-TLV, which

- only supports BERTLV
- is quite new
doesn't have a lot of docs/examples

#3 - 04/11/2021 11:06 PM - fixeria

With 'construct' we now have the tools to deal with various hand-coded binary formats. I am pretty sure you can quickly implement a TLV parser with python-construct, see:

https://construct.readthedocs.io/en/latest/basics.html#sequences

Is there a reason to go for some other TLV parsing tool?

#4 - 04/12/2021 10:07 AM - laforge

On Sun, Apr 11, 2021 at 11:06:17PM +0000, fixeria [REDMINE] wrote:

I am pretty sure you can quickly implement a TLV parser with python-construct, see:
https://construct.readthedocs.io/en/latest/basics.html#sequences

will it fulfill the requirements I've stated in this ticket? I seriously doubt it.

#5 - 04/18/2021 10:16 PM - fixeria

Updated by laforge 6 days ago:

will it fulfill the requirements I've stated in this ticket? I seriously doubt it.

Well, I don't see why wouldn't it fulfill the requirements. I am not really familiar with python-construct's API, but I see no serious difficulties implementing the parser even on top of the lightweight codec.py that I introduced for trx_toolkit, which is a lot simpler than python-construct. I think I could implement it on top of python-construct, but not any time soon - still busy with VAMOS. JFYI.

#6 - 04/19/2021 07:00 AM - laforge

On Sun, Apr 18, 2021 at 10:16:52PM +0000, fixeria [REDMINE] wrote:

Updated by laforge 6 days ago:

will it fulfill the requirements I've stated in this ticket? I seriously doubt it.

Well, I don't see why wouldn't it fulfill the requirements.

As stated, I seriously doubt it.