



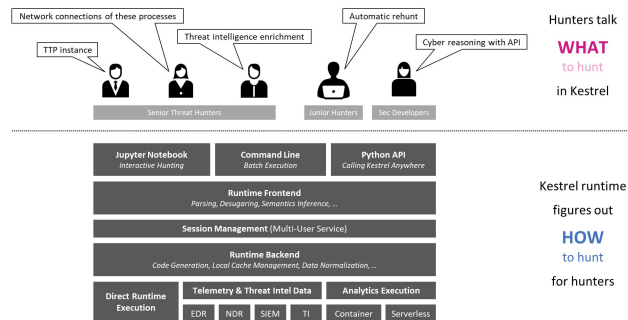
Kestrel

External Project Lead: Xiaokui Shu

Team Members: David Coletta, Megan Huang (& Jaisal Patel)

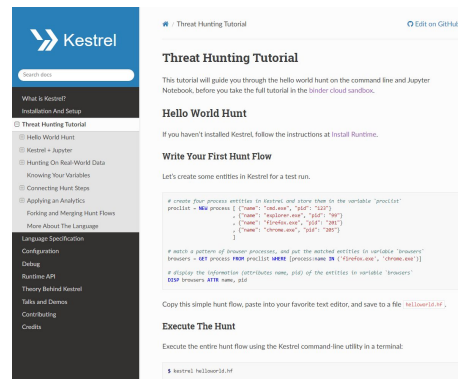
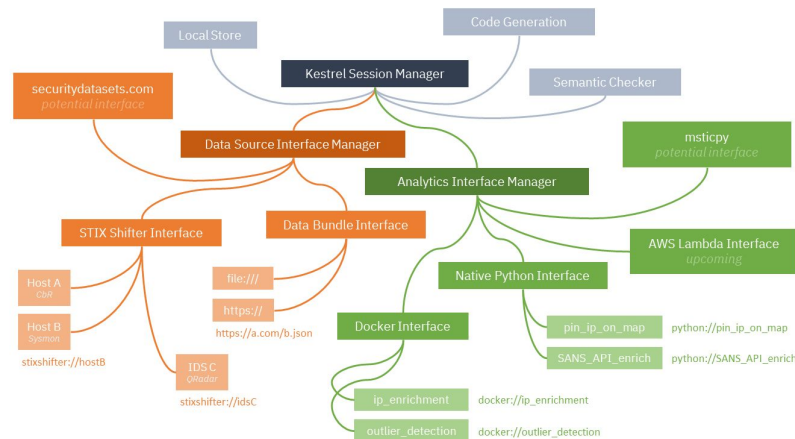
Background

- Open Cybersecurity Alliance (OCA)
- What is Kestrel? A loose explanation.
 - Cyber threat hunting language and runtime composed of Python (for the most part)
- Runtime environment requirements
 - Linux/Mac only, Python 3, SQLite (version 3.24 or newer)
 - Frontend: command line, Jupyter Notebook, or Python API



Kestrel

- Goals:
 - provide a layer of abstraction to build reusable, composable, and shareable hunt-flow
 - (eventually) train AI to the point where it can look for “what to hunt” on its own
- GitHub Repositories:
 - kestrel-lang (main)
 - kestrel-huntbook
 - kestrel-analytics
 - kestrel-jupyter
 - data-bucket-kestrel





VirusTotal API

(#124)

David Coletta (& Jaisal Patel)





Kestrel Analytics + VirusTotal API

I worked on connecting the VirusTotal API to the Kestrel language.

- Getting an API key from the virustotal website
- Write a python function that interfaces with the API
- Wrap this function as a Kestrel Analytics (Python module)

Docker Analytics for VirusTotal API #5



subbyte opened this issue on Oct 3, 2021 · 3 comments



subbyte commented on Oct 3, 2021 · edited ▾

Wanna create a Kestrel analytics that other hunters will clone and reuse? Try to do one for VirusTotal.

Useful information:

Attribute auto-complete (#79 & #264)

Megan Huang

Attribute auto-complete #79



subbyte opened this issue on Jul 7, 2021 · 14 comments



subbyte commented on Jul 7, 2021

Member



• edited

Is your feature request related to a problem? Please describe.

The current `auto-complete` function scans variable names (in the session), data source names, and analytics interface names. And it will be awesome if we can auto-complete entity attribute (and entity names also).

Advanced version: we may cache entity attributes in `VarStruct` for fast access.

autocomplete function doesn't behave correctly #264



vereimyst opened this issue 19 days ago · 0 comments



vereimyst commented 19 days ago



Description:

The `autocomplete` function doesn't properly address partially complete fields correctly. Fields with the same starter characters as commands also return commands matching `(last_word` as suggestions (which they should not be doing). No error messages are outputted, but the behavior does not match what is expected. Examples included below for clarity. A possible solution mentioned is completely revamping the logic of the `autocomplete()` function, looking at the parsing portion of the existing code in particular.

Environment:

Assign

No o

Label

bug

Project

None

Mile

Attribute Autocompletion

- understanding the issue
- issue breakdown
- points of confusion
 - `do_complete()` function flow
 - structure of code base
 - variable names
 - how + where to get information
- other troubles

- Case 1: `token == ATTRIBUTES` (old `STIXPATHS`)
 - e.g. `DISP <variable> ATTR <autocomplete>`
- Case 2: `token == STIXPATTERNBODY`
 - e.g. `WHERE <autocomplete>`

```
prefix = code[cursor_pos]
words = prefix.split(" ")
last_word = words[-1]
last_char = prefix[-1]
_logger.debug('code="%s" prefix="%s" last_word="%s"', code, prefix, last_word)

if "START" in prefix or "STOP" in prefix:
    return self._get_complete_timestamp(last_word)
elif "://" in last_word:
    scheme, path = last_word.split("://")
    # If scheme is self data source manager scheme():
else:
    _logger.debug("standard auto-complete")

    try:
        stmt = self.parse(prefix)
        _logger.debug("first parse: %s", stmt)
        last_stmt = stmt[-1]
        if last_stmt["command"] == "assign" and last_stmt["output"] == "_":
            # Special case for a varname alone on a line
            allnames = [
                v for v in self.get_variable_names() if v.startswith(prefix)
            ]
            if not allnames:
                return ["-", "+"] if prefix.endswith(" ") else []

            # If it parses successfully, add something so it will fail
            self.parse(prefix + " @autocompletions@")
        except KestrelSyntaxError as e:
            _logger.debug("exception: %s", e)
            varnames = self.get_variable_names()
            keywords = set(get_keywords())
            _logger.debug("keywords: %s", keywords)
            tmp = []
            for token in e.expected:
                _logger.debug("token: %s", token)
                if token == "VARIABLE":
                    tmp.extend(varnames)
```

Debugging

```
20:38:40 DEBUG kestrel.session varname: browsers
20:38:40 DEBUG kestrel.session BEFORE attribute autocompletion: []
20:38:40 DEBUG firepit.sqlitestorage Executing query: PRAGMA table_info("browsers")
20:38:40 DEBUG firepit.sqlitestorage browsers columns = ['name', 'pid', 'id']
20:38:40 DEBUG firepit.sqlitestorage Executing query: SELECT DISTINCT "pid" FROM "browsers"
20:38:40 DEBUG kestrel.session pid
```

- function return values
 - `get_entity_id_attribute()` → `session.store.columns()`
- attribute subvalues
 - `_ref.id/value` under `ipv4/ipv6-addr` tables
- new bug uncovered

```
17:39:38 DEBUG kestrel.session code="DISP conns ATTR p" prefix="DISP conns ATTR p"
last_word="p"
17:39:38 DEBUG kestrel.session standard auto-complete
17:39:39 DEBUG kestrel.session first parse: [{ 'command': 'disp', 'input': 'conns',
'transform': None, 'attrs': 'p' }]
17:39:39 DEBUG kestrel.session exception: [ERROR] KestrelSyntaxError: invalid character "@"
at line 1 column 18, expects one of ['GET', 'FIND', 'SAVE', 'LIMIT', 'TRANSFORM', 'NEW',
'GROUP', 'JOIN', 'LOAD', 'VARIABLE', 'SORT', 'DISP', 'APPLY', 'INFO', 'OFFSET']
rewrite the failed statement.
-----
17:39:39 DEBUG kestrel.session ['TIMESTAMPED', '_', 'apply', 'conns', 'disp', 'find', 'get',
'group', 'info', 'join', 'limit', 'load', 'new', 'offset', 'save', 'sort'] -> []
```

DISP conns ATTR

tmp = NEW process
DISP tmp ATTR

create four p
proclist = NEW

dst_port
dst_ref
end
id
protocols
src_port
src_ref
start



What we learned

- Forking repositories
- How to use Kestrel
- Working on large existing codebases
- Getting comfortable with looking at previously written code
- Working with
 - APIs & API keys
 - syntax parsers (Lark)
- Running test scripts

Thank you for watching!

(links to additional resources can be found after this slide)

Any questions?



Citations

All pictures are linked to their respective sources (exceptions are local testing and debug logs).

Information taken from [Kestrel's Official Documentation](#), the [kestrel-lang](#) repository, [IBM Researcher Profiles](#), and the [OCA GitHub Profile](#).



More about Xiaokui Shu

[OCA Articles](#)

[ACM Interview](#) (recommended for beginners interested in working on Kestrel)

[GitHub](#)

[IBM Research](#)

[RSA Conference](#)

[Google Scholar](#)



Kestrel Resources

[Kestrel documentation](#)

[GitHub repository for kestrel-lang](#)

[IBM article on Kestrel](#)

[Black Hat 2022 Demo](#)

[InfoSec Jupyterthon 2021 Kestrel Demo](#)

[Kestrel Interactive Tutorial](#)

[Black Hat Hunting Lab](#)