sdvalidator Documentation

Release 2.1.11

Calum Baird

Feb 04, 2019

CONTENTS:

CHAPTER

ONE

INSTALLATION

1.1 From pip

```
$ pip3 install sdvalidator
```

1.2 From source

```
$ git clone https://github.com/calumsbaird/sdvalidator.git
$ cd sdvalidator
$ python3 setup.py install
```

CHAPTER

TWO

USAGE

2.1 From command line

2.1.1 Command line arguments

```
$ sdvalidate google.com
google.com SPF: VALID DMARC: VALID
```

2.1.2 File IO

```
$ cat domains.txt
google.com
csbaird.com
github.com

$ cat domains.txt | sdvalidate
google.com SPF: VALID  DMARC: VALID
csbaird.com SPF: VALID  DMARC: VALID
github.com SPF: VALID  DMARC: VALID
```

2.2 Python Usage

Given a list of domains find which domains have valid spf and dmarc records and cache the results. Optionally pickle the output.:

```
from sdvalidator import *
domains = ['csbaird.com','google.com','github.com']
cache = validate_sd(domains,verbose=True)
print(cache['csbaird.com']['spf_validity']) # VALID

import pickle
with open('backup.pickle','wb') as f:
    pickle.dump(cache, f)
```

CHAPTER

THREE

FUNCTIONS

sdvalidator.**resolves**(*domain*)

```
>>> resolves('csbaird.com')
True
```

If domain resolves to an A, AAAA or CNAME return True

Parameters **domain** (*str*) – A domain such as 'example.com'

Returns True if resolves to an address

Return type int

sdvalidator.**filter_resolving_domains**(*domains*, *verbose=False*)

```
>>> filter_resolving_domains(['csbaird.com','fake.csbaird.com'])
['csbaird.com']
```

Filters out any non-resolving domain from a list.

Parameters

- **domains** (*list*) – A list of str domains
- **verbose** (*bool*) – Optionally print progress of filter

Returns List of resolving domains

List

sdvalidator.**pull_spf**(*domain*)

```
>>> pull_spf('csbaird.com')
['v=spf1 mx -all']
```

Get all txt records that look like they are an spf record

Parameters **domain** (*str*) – domain like example.com

Returns List of records that look like they are spf records.

Return type list

sdvalidator.**pull_dmarc**(*domain*)

```
>>> pull_dmarc('csbaird.com')
['v=DMARC1;  p=reject; pct=100; rua=mailto:root@csbaird.com;␣
↪ruf=mailto:root@csbaird.com']
```

Get all txt records that look like they are an dmarc record

Parameters **domain** (*str*) – domain like example.com

Returns List of records that look like they are dmarc records.

Return type list

sdvalidator.**resolve_record**(*domain*)

```
>>> resolve_record('csbaird.com')
['ca3-0eb269d493c84687a2b27e8bea13ca55', 'v=spf1 mx -all']
```

Get all txt records associated with a domain. If a CNAME gets record of the domain it redirects to.

Parameters **domain** (*str*) – domain like example.com

Returns List of txt records associated with the domain

Return type list

sdvalidator.**pull_sd**(*domains*, *cache={}*)

```
>>> dns_grab.pull_sd(['csbaird.com'])
{'csbaird.com': {'spf': ['v=spf1 mx -all'], 'dmarc': ['v=DMARC1;  p=reject;␣
↪pct=100; rua=mailto:root@csbaird.com; ruf=mailto:root@csbaird.com']}}
```

Get the SPF and DMARC records and store them in a dictionary.

Parameters

- **domains** (*list*) – list of domains.
- **cache** (*dict*) – dictionary containing domains mapped to dicts containing spf and dmarc records.

Returns cache

Return type dict

sdvalidator.**validate_spf**(*domain*, *cache={}*, *__depth=0*)

```
>>> validate_spf('csbaird.com')
'VALID'
```

Check validity of a domain's spf record. Uses a regex and checks recursive lookup depth.

Parameters

- **domain** (*str*) – A domain such as example.com
- **cache** (*dict*) – Dictionary for storing spf/dmarc records

Returns 'VALID'|'INVALID'|'MISSING'

Return type str

sdvalidator.**validate_dmarc**(*domain*, *cache={}*)

```
>>> validate_dmarc('csbaird.com')
'VALID'
```

Check validity of a domain's dmarc record. Uses a regex and checks recursive lookup depth.

Parameters

- **domain** (*str*) – A domain such as example.com
- **cache** (*dict*) – Dictionary for storing spf/dmarc records

Returns 'VALID'|'INVALID'|'MISSING'

Return type str

sdvalidator.**validate_sd**(*domains*, *cache={}*, *verbose=False*)

```
>>> validate_sd(['csbaird.com'])
{'csbaird.com': {'spf': ['v=spf1 mx -all'], 'dmarc': ['v=DMARC1;  p=reject;␣
↪pct=100; rua=mailto:root@csbaird.com; ruf=mailto:root@csbaird.com'], 'spf_
↪validity': 'VALID', 'dmarc_validity': 'VALID'}, 'fake.csbaird.com': {'spf': [
↪'v=spf1 -all'], 'dmarc': [], 'spf_validity': 'VALID', 'dmarc_validity':
↪'VALID'}}
>>> validate_sd(['csbaird.com'])['csbaird.com']['spf_validity']
'VALID'
```

Check the validity of SPF and DMARC records for a list of domains. **cache[<domain>]** keys:

- '*spf*': SPF record for the domain
- '*spf_validity*': validity of the SPF record
- '*dmarc*': DMARC record for the domain
- '*dmarc_validity*': validity of the DMARC record for a domain

Parameters

- **domains** (*list*) – A list of domains to test
- **cache** (*list*) – A dict to store results
- **verbose** (*bool*) – Optionally print the progress to the stdout

Returns cache

Return type dict

CHAPTER

FOUR

QUICK START

4.1 Command line

```
$ pip3 install sdvalidator
$ sdvalidate csbaird.com
VALID
```

4.2 Script

test.py:

```
from sdvalidator import *
domains = ['csbaird.com','google.com','github.com']
cache = validate_sd(domains,verbose=True)
print(cache['csbaird.com']['spf_validity']) # VALID
```

```
$ python3 test.py
VALID
```

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX