scanner.py

```
''' Scanner.py  submitted by Robert Blanchett 100639184
    v0.7        for Holmesglen CertIV in cybesecurity 22334VI
                Programming Assessment Task 2.

                A command line reporter from the Virus Total 

                find __main__ for notes on TODO to expand this

Developed on    Windows 10 Enterprise (Build 1904)
                Developer Evaluation Virtual Box VM (linux ho
                Python 3.9.6 (64 bit) from python.org
                VScode 1.59 with pylance installed

                All My own work. RDB

                Requires vt-py and validators from pypi

                ONLY urls are reported on at this stage to ke

                provided test datafiles contain IPs, URLs and
                spamhaus.de, URLhaus.de, iplists.FireHol.org 

                Please refer to the README for information de
                the distrubuted test files.
                cf README the one known BUG with usage printi
'''
import sys          # Python Runtime, exception tamer and base
import os           # path and file operations
import datetime     # stamping reports and filenames
import configparser # state persistence across executions
import argparse     # CLI from stdlib
import socket       # check the network
import time         # delay API calls
import vt           # virus Total API Python client Library (
import validators   # validators library (install with pip)
```

validators, vt-py modules from from pypi,

Keys and Essential Constants (Reserved from Configparser to aid readability)

```
VTAPIKEY = 'dd70d000e70408740bb90db27a8e9f4925a5868369ea6180f
install_directory = sys.argv[0][:-10]   # windows sets argv[0
now = datetime.datetime.now().strftime("-%Y-%m-%d-%H-%M-%S") 
scanrc = '.scanrc'                      # Config File
config = configparser.ConfigParser()
supplied =[]                            # processing buckets 
valid_ip = []                           #
valid_url = []                          #
valid_domain = []                       #


def init(args):
    ''' Reset the configuration file backing up an existing o

    if  os.path.isfile(install_directory+scanrc):
        print(f"\nBacking up Config File {install_directory+s
        os.rename(install_directory+scanrc, install_directory

    else:
       print(f"\nConfig File {install_directory+scanrc} not f

    config['DEFAULT'] = {'Runs': 0, 'URLScanCount': '0', 'Mal
    config['State'] = {'Runs': 0, 'UrlScanCount': 0, 'Maliciou
    config['LastRun'] = {}
    config.write(open(install_directory+scanrc, 'w'))


def check_network():
    '''Internet availability check. Cloudflare is always ther
    try:
        socket.create_connection(("1.1.1.1", 53))
        return True
    except OSError:
        return False


def scan(args):
    ''' Validate and submit to VirusTotal API for reports the
    if check_network():
        print("\nInternet available. Continuing.")

    else:
        print("\nInternet unavailable. Exiting.")
        sys.exit()

    print("\nprocessing supplied files.")
    for n in range(len(args.files)):
        print(args.files[n].name)
    print("\nplease wait. VirusTotal limits requests to 4/minu
    print("and so does this script!\n")

    for l in range(len(args.files)):
        for line in args.files[l]:
            supplied.append(line.rstrip())

    print((len(supplied)), "items to be validated before scanr

    print(supplied)
```

validate items

```
    to_validate = supplied.copy()
    valid_ip = [x for x in to_validate if validators.ip_addres
    valid_url = [x for x in to_validate if validators.url(x)]
    valid_domain = [x for x in to_validate if validators.domai
```

only scanning the URLs to keep the script within ~150 loc each returned object type has a different set of API endpoints and object members I'd have to code uniquely for

```
    vtGet = vt.Client(VTAPIKEY)
    urlResults = dict.fromkeys(valid_url)
    scanRuns =  config.getint('State', 'Runs')
    print("\nPrevious Runs", scanRuns)

    for i in range(len(valid_url)):

        print("\nSubmitting Url: ", valid_url[i])
        url_id = vt.url_id(valid_url[i])
        response = vtGet.get_object("/urls/{}", url_id)
        urlResults[valid_url[i]] = response.last_analysis_stat
        config.write(open(install_directory+scanrc, 'w'))
```

config to store results and increment of URLs scanned as subkeys in config file. future/excised work.

```
        time.sleep(13)

    vtGet.close()       # Cleanup http connection

    config.set('State', 'Runs', str(scanRuns +1))

    print("\nScanner run {} Report {}".format(config.getint('S
    print("The number of virus products and how the URL was re
    print("Results from The VirusTotal.com Pulic API")
    for url, results in urlResults.items():
        for type in results:
            print("{0:<11} : {1:<}".format(type, results[type]

    config.write(open(install_directory+scanrc, 'w'))


def main(args):
    '''  Framework logic and function dispatcher'''
```

Read config.

```
    if  os.path.isfile(install_directory+scanrc):
        config.read(install_directory+scanrc)

    else:
        init(args)
```

Command Dispatcher

```
    action = {'init': init, 'scan': scan}
    action[args.subcommand](args)

if __name__ == "__main__":
    '''File handle collection and CLI parsing by argparse'''
```

TODO: work removed to get minimum working code ~150 loc

include submission of IP addresses, domains and filehashes. functionality removed for LOC limitations add subparsers for unimplemented subcommands: list (previous runs etc), shutdown (handle KeyboardInterrupt Ctrl-C interrupt during scan) import hashlib to submit file hashes for checking import subprocess to do in-script installation of pypi on ModuleNotFoundError record detailed run information in .scanrc with configparser

```
    parser=argparse.ArgumentParser(description="scanner Regist
    scanner.py <command> [filenames ..]

    The currently implemented subcommands are:
    init                                Reset the Configuratic
    scan  [filename1 filename2 ..]      Submit one or more pla
                                        ONE IP address or ONE

    subparser = parser.add_subparsers(dest='subcommand', title
    subparser.required=True
    parser_init = subparser.add_parser('init', help='reset the
    parser_init.set_defaults(func=init)
    parser_scan = subparser.add_parser('scan', help='supply te
    parser_scan.add_argument('files', type=argparse.FileType('
    parser_scan.set_defaults(func=scan)

    args=parser.parse_args()

    main(args)
```