Bro Python Utilities Documentation

Release 0.2.5

Brian Wylie

Sep 02, 2017

Contents

Bro + Python = BroThon!

The BroThon package supports the ingestion, processing, and analysis of Bro IDS data with Python.

CHAPTER 1

Why BroThon?

Bro IDS already has a flexible, powerful scripting language why should I use BroThon?

Offloading: Running complex tasks (yara sigs on files, state machines, machine learning, etc..) should be offloaded from Bro IDS so that Bro can focus on the efficient processing of high volume network traffic.

Python: Pulling Bro data into Python allows us to leverage a large set of of Python modules for data analysis, statistics, machine learning and visualization.

Data Analysis: A growing set of notebooks/examples using statistics and machine learning on Bro data.

CHAPTER 2

Easy to Use

```
from brothon import bro_log_reader
...
    # Run the bro reader on a given log file
    reader = bro_log_reader.BroLogReader('dhcp.log')
    for row in reader.readrows():
        pprint(row)
```

Output: Each row is a nice Python Dictionary with timestamps and types properly converted.

```
{'assigned_ip': '192.168.84.10',
'id.orig_h': '192.168.84.10',
'id.orig_p': 68,
'id.resp_h': '192.168.84.1',
'id.resp_p': 67,
'lease_time': datetime.timedelta(49710, 23000),
'mac': '00:20:18:eb:ca:54',
'trans_id': 495764278,
'ts': datetime.datetime(2012, 7, 20, 3, 14, 12, 219654),
'uid': 'CJsdG95nCNF1RXuN5'}
...
```

CHAPTER 3

More Examples

- Easy ingestion of any Bro Log into Python (dynamic tailing and log rotations are handled)
- Bro Logs to Pandas Dataframes and Scikit-Learn
- Dynamically monitor files.log and make VirusTotal Queries
- Dynamically monitor http.log and show 'uncommon' User Agents
- Running Yara Signatures on Extracted Files
- Checking x509 Certificates
- Anomaly Detection
- See BroThon Examples for more details.

CHAPTER 4

Analysis Notebooks

BroThon enables the processing, analysis, and machine learning of realtime data coming from Bro IDS.

- Risky Domains Stats and Deployment: Risky Domains
- Bro to Scikit-Learn: Bro to Scikit
- Bro to Spark: Bro to Spark
- Anomaly Detection Exploration: Anomaly Detection

CHAPTER 5

Install

```
$ pip install brothon
or
$ pip install brothon[all]   # Includes additional dependencies to run all examples␣
↪(yara, etc)
```

CHAPTER 6

Documentation

BroThon.readthedocs.org

CHAPTER 7

Thanks

- The DummyEncoder is based on Tom Augspurger's great PyData Chicago 2016 Talk

CHAPTER 8

Contents:

Examples

To use Bro Python Utilities in a project:

```
import brothon
```

BroLog to Python

See brothon/examples/bro_pprint.py for full code listing.

```
from pprint import pprint
from brothon import bro_log_reader
...
    # Run the bro reader on a given log file
    reader = bro_log_reader.BroLogReader('dhcp.log')
    for row in reader.readrows():
        pprint(row)
```

Example Output: You get back a nice Python Dictionary with timestamps and types properly converted.

```
{'assigned_ip': '192.168.84.10',
'id.orig_h': '192.168.84.10',
'id.orig_p': 68,
'id.resp_h': '192.168.84.1',
'id.resp_p': 67,
'lease_time': datetime.timedelta(49710, 23000),
'mac': '00:20:18:eb:ca:54',
'trans_id': 495764278,
'ts': datetime.datetime(2012, 7, 20, 3, 14, 12, 219654),
'uid': 'CJsdG95nCNF1RXuN5'}
```

Bro to Pandas DataFrame

See brothon/examples/bro_to_pandas.py for full code listing. Notice that it's one line of code to convert to a Pandas DataFrame.

```
import pandas as pd
from brothon import bro_log_reader
...

    # Create a bro reader on a given log file
    reader = bro_log_reader.BroLogReader('http.log')

    # Create a Pandas dataframe from reader
    bro_df = pd.DataFrame(reader.readrows())

    # Print out the head of the dataframe
    print(bro_df.head())
```

Example Output

```
               host      id.orig_h  id.orig_p  response_body_len status_code          ␣
↪   uri
 hopraresidency.com  192.168.84.10       1030                372         200         /
↪foo.js
blogs.redheberg.com  192.168.84.10       1031               2111         200     /
↪mltools.js
    santiyesefi.com  192.168.84.10       1034                327         404     /
↪mltools.js
     tudespacho.net  192.168.84.10       1033              12350         200  /
↪32002245.html
     tudespacho.net  192.168.84.10       1033               5176         200      /
↪98765.pdf
```

Bro to Scikit-Learn

See brothon/examples/bro_to_scikit.py for full code listing, we've shortened the code listing here to demonstrate that it's literally just a few lines of code to get to Scikit-Learn.

```
# Create a bro reader on a given log file
reader = bro_log_reader.BroLogReader(args.bro_log)

# Create a Pandas dataframe from reader
bro_df = pd.DataFrame(reader.readrows())

# Use the Brothon DataframeToMatrix class (handles categorical data!)
to_matrix = dataframe_to_matrix.DataFrameToMatrix()
bro_matrix = to_matrix.fit_transform(bro_df)

# Now we're ready for scikit-learn!
kmeans = KMeans(n_clusters=5).fit_predict(bro_matrix)
pca = PCA(n_components=2).fit_transform(bro_matrix)
```

Example Output

```
Rows in Cluster: 42
                          query  Z proto qtype_name         x         y  cluster
```

```
0                     guyspy.com  0   udp          A -0.356148 -0.111347        0
1                 www.guyspy.com  0   udp          A -0.488648 -0.068594        0
2   devrubn8mli40.cloudfront.net  0   udp          A -0.471554 -0.110367        0
3  d31qbv1cthcecs.cloudfront.net  0   udp          A -0.454148 -0.165611        0
4                crl.entrust.net  0   udp          A -0.414992 -0.103959        0

...

Rows in Cluster: 4
            query  Z proto qtype_name         x         y  cluster
57  j.maxmind.com  1   udp          A -0.488136 -0.230034        3
58  j.maxmind.com  1   udp          A -0.461758 -0.235828        3
59  j.maxmind.com  1   udp          A -0.408193 -0.179723        3
60  j.maxmind.com  1   udp          A -0.460889 -0.217559        3

Rows in Cluster: 4
                                                query  Z proto qtype_name         x␣
↪       y  cluster
53  superlongcrazydnsqueryforanomalydetectionj.max...  0   udp          A -0.554213 -
↪0.206536        4
54  xyzsuperlongcrazydnsqueryforanomalydetectionj....  0   udp          A -0.559984 -
↪0.260327        4
55  abcsuperlongcrazydnsqueryforanomalydetectionj....  0   udp          A -0.622886 -
↪0.222030        4
56  qrssuperlongcrazydnsqueryforanomalydetectionj....  0   udp          A -0.571959 -
↪0.236560        4
```

Bro Files Log to VirusTotal Query

See brothon/examples/file_log_vtquery.py for full code listing (code simplified below)

```
from brothon import bro_log_reader
from brothon.utils import vt_query
...
    # Run the bro reader on on the files.log output
    reader = bro_log_reader.BroLogReader('files.log', tail=True) # This will␣
↪dynamically monitor this Bro log
    for row in reader.readrows():

        # Make the query with the file sha
        pprint(vtq.query(row['sha256']))
```

Example Output: Each file sha256/sha1 is queried against the VirusTotal Service.

```
{'file_sha': 'bdf941b7be6ba2a7a58b0aef9471342f8677b31c', 'not_found': True}
{'file_sha': '2283efe050a0a99e9a25ea9a12d6cf67d0efedfd', 'not_found': True}
{'file_sha': 'c73d93459563c1ade1f1d39fde2efb003a82ca4b',
    u'positives': 42,
    u'scan_date': u'2015-09-17 04:38:23',
    'scan_results': [(u'Gen:Variant.Symmi.205', 6),
        (u'Trojan.Win32.Generic!BT', 2),
        (u'Riskware ( 0015e4f01 )', 2),
        (u'Trojan.Inject', 2),
        (u'PAK_Generic.005', 2)]}

{'file_sha': '15728b433a058cce535557c9513de196d0cd7264',
```

```
    u'positives': 33,
    u'scan_date': u'2015-09-17 04:38:21',
    'scan_results': [(u'Java.Exploit.CVE-2012-1723.Gen.A', 6),
        (u'LooksLike.Java.CVE-2012-1723.a (v)', 2),
        (u'Trojan-Downloader ( 04c574821 )', 2),
        (u'Exploit:Java/CVE-2012-1723', 1),
        (u'UnclassifiedMalware', 1)]}
```

Bro HTTP Log User Agents

See brothon/examples/http_user_agents.py for full code listing (code simplified below)

```
from collections import Counter
from brothon import bro_log_reader
...
    # Run the bro reader on a given log file counting up user agents
    http_agents = Counter()
    reader = bro_log_reader.BroLogReader(args.bro_log, tail=True)
    for count, row in enumerate(reader.readrows()):
        # Track count
        http_agents[row['user_agent']] += 1

    print('\nLeast Common User Agents:')
    pprint(http_agents.most_common()[:-50:-1])
```

Example Output: Might be some interesting agents on this list...

```
Least Common User Agents:
[
 ('NetSupport Manager/1.0', 1),
 ('Mozilla/4.0 (Windows XP 5.1) Java/1.6.0_23', 1),
 ('Mozilla/5.0 (X11; Linux i686 on x86_64; rv:10.0.2) Gecko/20100101 Firefox/10.0.2',␣
↪1),
 ('oh sure', 2),
 ('Fastream NETFile Server', 2),
 ('Mozilla/5.0 (X11; Linux i686; rv:2.0.1) Gecko/20100101 Firefox/4.0.1', 3),
 ('Mozilla/5.0 (Windows NT 6.1; rv:7.0.1) Gecko/20100101 Firefox/7.0.1', 4),
 ('NESSUS::SOAP', 5),
 ('webmin', 6),
 ('Nessus SOAP v0.0.1 (Nessus.org)', 10),
 ('Mozilla/4.0 (compatible; gallery_203.nasl; Googlebot)', 31),
 ("mercuryboard_user_agent_sql_injection.nasl'", 31),
 ('Mozilla/5.0 (X11; Linux i686; rv:10.0.2) Gecko/20100101 Firefox/10.0.2', 46),
 ('*/*', 49),
 ('Nessus', 52),
 ...
 ('Mozilla/5.0 (compatible; Nmap Scripting Engine; http://nmap.org/book/nse.html)',␣
↪6166),
```

Yara rules on Bro extracted files

The example will dymancially monitor the extract_files directory and when a file is dropped by Bro IDS the code will run a set of Yara rules against that file. See brothon/examples/yara_matches.py for full code listing (code simplified below)

```
import yara
from brothon import dir_watcher
...

def yara_match(file_path, rules):
    """Callback for a newly extracted file"""
    print('New Extracted File: {:s}'.format(file_path))
    print('Mathes:')
    pprint(rules.match(file_path))

...
    # Load/compile the yara rules
    my_rules = yara.compile(args.rule_index)

    # Create DirWatcher and start watching the Bro extract_files directory
    print('Watching Extract Files Directory: {:s}'.format(args.extract_dir))
    dir_watcher.DirWatcher(args.extract_dir, callback=yara_match, rules=my_rules)
```

Example Output:

```
Loading Yara Rules from ../brothon/utils/yara_test/index.yar
Watching Extract Files Directory: /home/ubuntu/software/bro/extract_files
New Extracted File: /home/ubuntu/software/bro/extract_files/test.tmp
Mathes:
[AURIGA_driver_APT1]
```

Risky Domains

The example will use the analysis in our Risky Domains notebook to flag domains that are 'at risk' and conduct a Virus Total query on those domains. See brothon/examples/risky_dns.py for full code listing (code simplified below)

```
from brothon import bro_log_reader
from brothon.utils import vt_query
...

    # Create a VirusTotal Query Class
    vtq = vt_query.VTQuery()

    # See our 'Risky Domains' Notebook for the analysis and
    # statistical methods used to compute this risky set of TLDs
    risky_tlds = set(['info', 'tk', 'xyz', 'online', 'club', 'ru', 'website', 'in',
↪'ws', 'top', 'site', 'work', 'biz', 'name', 'tech'])

    # Run the bro reader on the dns.log file looking for risky TLDs
    reader = bro_log_reader.BroLogReader(args.bro_log, tail=True)
    for row in reader.readrows():

        # Pull out the TLD
        query = row['query']
        tld = tldextract.extract(query).suffix

        # Check if the TLD is in the risky group
        if tld in risky_tlds:
            # Make the query with the full query
            results = vtq.query_url(query)
            if results.get('positives'):
```

```
                print('\nOMG the Network is on Fire!!!')
                pprint(results)
```

Example Output: To test this example simply do a "$ping uni10.tk" on a machine being monitored by your Bro IDS.

Note: You can also ping something like 'isaftaho.tk' which is not on any of the blacklist but will still hit. The script will obviously cast a much wider net than just the blacklists.

```
$ python risky_dns.py -f /usr/local/var/spool/bro/dns.log
  Successfully monitoring /usr/local/var/spool/bro/dns.log...

  OMG the Network is on Fire!!!
  {'filescan_id': None,
   'positives': 9,
   'query': 'uni10.tk',
   'scan_date': '2016-12-19 23:49:04',
   'scan_results': [('clean site', 55),
                    ('malicious site', 5),
                    ('unrated site', 4),
                    ('malware site', 4),
                    ('suspicious site', 1)],
   'total': 69,
   'url': 'http://uni10.tk/'}
```

Cert Checker

There's been discussion about Let's Encrypt issuing certficates to possible phishing/malicious site owners. This example will quickly check and dynamically monitor your Bro IDS x509 logs for certificates that may be from malicious sites.

See brothon/examples/cert_checker.py for full code listing (code simplified below)

```
from brothon import bro_log_reader
from brothon.utils import vt_query
...

    # These domains may be spoofed with a certificate issued by 'Let's Encrypt'
    spoofed_domains = set(['paypal', 'gmail', 'google', 'apple','ebay', 'amazon'])

    # Run the bro reader on the x509.log file looking for spoofed domains
    reader = bro_log_reader.BroLogReader(args.bro_log, tail=True)
    for row in reader.readrows():

        # Pull out the Certificate Issuer
        issuer = row['certificate.issuer']
        if "Let's Encrypt" in issuer:

            # Check if the certificate subject has any spoofed domains
            subject = row['certificate.subject']
            domain = subject[3:] # Just chopping off the 'CN=' part
            if any([domain in subject for domain in spoofed_domains]):
                print('\n<<< Suspicious Certificate Found >>>')
                pprint(row)

                # Make a Virus Total query with the spoofed domain (just for fun)
                results = vtq.query_url(domain)
                if results.get('positives', 0) >= 2: # At least two hits
```

```
                    print('\n<<< Virus Total Query >>>')
                    pprint(results)
```

Example Output: Simply run this example script on your Bro IDS x509.log.

```
$ python cert_checker.py -f ../data/x509.log
  Successfully monitoring ../data/x509.log...

  <<< Suspicious Certificate Found >>>
  {'basic_constraints.ca': True,
   'certificate.issuer': "CN=Let's Encrypt Authority X3,O=Let's Encrypt,C=US",
   'certificate.key_alg': 'rsaEncryption',
   'certificate.key_length': 4096,
   'certificate.key_type': 'rsa',
   'certificate.sig_alg': 'sha256WithRSAEncryption',
   'certificate.subject': 'CN=paypal.migems.com',
   ...}

  <<< Virus Total Query >>>
  {'filescan_id': None,
   'positives': 8,
   'query': 'paypal.migems.com',
   'scan_date': '2017-04-16 09:39:52',
   'scan_results': [('clean site', 50),
                    ('phishing site', 6),
                    ('unrated site', 6),
                    ('malware site', 1),
                    ('malicious site', 1)],
   'total': 64,
   'url': 'http://paypal.migems.com/'}
```

Anomaly Detection

Here we're demonstrating anomaly detection using the Isolated Forest algorithm. Once anomalies are identified we then use clustering to group our anomalies into organized segments that allow an analyst to 'skim' the output groups instead of looking at each row.

See brothon/examples/anomaly_detection.py for full code listing (code simplified below)

```
# Create a Bro IDS log reader
reader = bro_log_reader.BroLogReader(args.bro_log)

# Create a Pandas dataframe from reader
bro_df = pd.DataFrame(reader.readrows())

# Using Pandas we can easily and efficiently compute additional data metrics
bro_df['query_length'] = bro_df['query'].str.len()

# Use the BroThon DataframeToMatrix class
features = ['Z', 'rejected', 'proto', 'query', 'qclass_name', 'qtype_name', 'rcode_
↪name', 'query_length']
to_matrix = dataframe_to_matrix.DataFrameToMatrix()
bro_matrix = to_matrix.fit_transform(bro_df[features])

# Train/fit and Predict anomalous instances using the Isolation Forest model
odd_clf = IsolationForest(contamination=0.35) # Marking 35% as odd
```

```
odd_clf.fit(bro_matrix)

# Add clustering to our anomalies
bro_df['cluster'] = KMeans(n_clusters=4).fit_predict(bro_matrix)

# Now we create a new dataframe using the prediction from our classifier
odd_df = bro_df[features+['cluster']][odd_clf.predict(bro_matrix) == -1]

# Now group the dataframe by cluster
cluster_groups = bro_df[features+['cluster']].groupby('cluster')

# Now print out the details for each cluster
print('<<< Outliers Detected! >>>')
for key, group in cluster_groups:
    print('\nCluster {:d}: {:d} observations'.format(key, len(group)))
    print(group.head())
```

Example Output: Run this example script on your Bro IDS dns.log...

```
<<< Outliers Detected! >>>

Cluster 0: 4 observations
    Z rejected proto                                              query qclass_name␣
↪qtype_name rcode_name  query_length  cluster
53  0    False   udp  superlongcrazydnsqueryforanomalydetectionj.max...  C_INTERNET  ␣
↪       A    NOERROR            54        0
54  0    False   udp  xyzsuperlongcrazydnsqueryforanomalydetectionj....  C_INTERNET  ␣
↪       A    NOERROR            57        0
55  0    False   udp  abcsuperlongcrazydnsqueryforanomalydetectionj....  C_INTERNET  ␣
↪       A    NOERROR            57        0
56  0    False   udp  qrssuperlongcrazydnsqueryforanomalydetectionj....  C_INTERNET  ␣
↪       A    NOERROR            57        0

Cluster 1: 11 observations
    Z rejected proto query qclass_name qtype_name rcode_name  query_length  cluster
39  0    False   udp     -           -          -          -             1        1
40  0    False   udp     -           -          -          -             1        1
41  0    False   udp     -           -          -          -             1        1
42  0    False   udp     -           -          -          -             1        1
43  0    False   udp     -           -          -          -             1        1

Cluster 2: 6 observations
    Z rejected proto          query qclass_name qtype_name rcode_name  query_length ␣
↪cluster
61  0    False   tcp  j.maxmind.com  C_INTERNET          A    NOERROR            13  ␣
↪     2
62  0    False   tcp  j.maxmind.com  C_INTERNET          A    NOERROR            13  ␣
↪     2
63  0    False   tcp  j.maxmind.com  C_INTERNET          A    NOERROR            13  ␣
↪     2
64  0    False   tcp  j.maxmind.com  C_INTERNET          A    NOERROR            13  ␣
↪     2
65  0    False   tcp  j.maxmind.com  C_INTERNET          A    NOERROR            13  ␣
↪     2

Cluster 3: 4 observations
    Z rejected proto          query qclass_name qtype_name rcode_name  query_length ␣
↪cluster
```

```
57  1    False   udp  j.maxmind.com  C_INTERNET          A    NOERROR            13  ␣
↪      3
58  1    False   udp  j.maxmind.com  C_INTERNET          A    NOERROR            13  ␣
↪      3
59  1    False   udp  j.maxmind.com  C_INTERNET          A    NOERROR            13  ␣
↪      3
60  1    False   udp  j.maxmind.com  C_INTERNET          A    NOERROR            13  ␣
↪      3
```

Streaming Outlier Detector

Here we're demonstrating a streaming anomaly detection to show the use of the dataframe_cache class. The dataframe_cache allows us to stream data from Bro IDS into a 'time-windowed' dataframe. In this example we blah blah..

- Every 5 seconds we run anomaly detection
- The dataframe contains a window of data (30 seconds in this example)

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

Types of Contributions

Report Bugs

Report bugs at https://github.com/kitware/BroThon/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

Write Documentation

Bro Python Utilities could always use more documentation, whether as part of the official Bro Python Utilities docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/kitware/BroThon/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

Get Started!

Ready to contribute? Here's how to set up *BroThon* for local development.

1. **Fork_** the *BroThon* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/BroThon.git
```

3. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

4. When you're done making changes, check that your changes pass style and unit tests, including testing other Python versions with tox:

```
$ tox
```

To get tox, just pip install it.

5. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

6. Submit a pull request through the GitHub website.

Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.5, 3.6, and PyPy. Check https://travis-ci.org/kitware/BroThon under pull requests for active pull requests or run the `tox` command and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ py.test brothon
```

Credits

Development Lead

- Brian Wylie <brian.wylie@kitware.com>

Contributors

None yet. Why not be the first?

CHAPTER 9

Feedback

If you have any suggestions or questions about **Bro IDS Python Utilities** feel free to email me at brian.wylie@kitware.com.

If you encounter any errors or problems with **Bro IDS Python Utilities**, please let me know! Open an Issue at the GitHub https://github.com/kitware/BroThon main repository.