Project Overview

- There is a need to make large amounts of NetFlow and IDS data available to the broader public, which can only be achieved once the data is anonymized
- What is the impact of using anonymized data for analysis?
- Does a researcher using “real” logs have an advantage over one using anonymized data?
- Can we create guidelines for anonymizing logs so analysis results are not impacted?
- Project Goals
  - Produce large amounts of anonymized datasets of network log data from the Zeek IDS
  - Evaluate the impact of the anonymization techniques on analysis using ***unsupervised*** machine learning techniques for anomaly detection

Experiment Design

- For two Zeek logs: conn.log (Netflow) and http.log (HTTP traffic), compare output of isolation forest anomaly detection for
  - Raw Data
  - Raw Data + encrypted IP address
  - Raw Data + encrypted IP address + dropped columns for sensitive fields
  - Raw Data + encrypted IP address + hashed columns for sensitive fields
- Ideally, our anonymization techniques will lead to similar clusters in the output of the isolation forest analysis

Raw Data: Connection Log

- Flow level statistics of each TCP connection logged by Zeek (~Netflow)
- https://docs.zeek.org/en/stable/scripts/base/protocols/conn/main.bro.html#type-Conn::Info
- 24 fields total
  - 3 fields to ***encrypt***, 2 fields to **salt+hash**, 1 field to drop

_node_name	ts	uid	***id.orig_h***	id.orig_p	***id.resp_h***
id.resp_p	proto	service	duration	orig_bytes	resp_bytes
conn_state	local_orig	local_resp	missed_bytes	history	orig_pkts
orig_ip_bytes	resp_pkts	resp_ip_bytes	***tunnel_parents***	**orig_l2_addr**	**resp_l2_addr**

Raw Data: HTTP Log

- HTTP requests and replies
- https://docs.zeek.org/en/stable/scripts/base/protocols/http/main.bro.html#type-HTTP::Info
- 30 fields total
  - 2 fields to ***encrypt***, 6 fields to **salt+hash,** 1 field to drop

_node_name	ts	uid	***id.orig_h***	id.orig_p	***id.resp_h***
id.resp_p	trans_depth	method	**host**	**uri**	referrer
version	user_agent	request_body_len	response_body_len	status_code	status_msg
info_code	info_msg	tags	**username**	**password**	proxied
orig_fuids	**orig_filenames**	orig_mime_types	resp_fluids	**resp_filenames**	resp_mime_types

Raw Data + Encrypted IP Address

- We choose to use Crypto-PAn algorithm to encrypt IP address. (Authored by Jinliang Fan, Jun Xu, Mostafa H. Ammar - Georgia Tech)
- Crypto-PAn properties:
  - One-to-one: mapping from original to encrypt IP address.
  - Prefix-preserving: if two original IP address share a k-bit prefix, their anonymized mapping also share a k-bit prefix.
  - Consistent across trace: the same IP address in different traces anonymized to the same address.
  - Cryptography-based: preserves the secrecy of the key and the randomness of the mapping from an original IP address to its anonymized counterpart.

Raw Data + Encrypted IP Address

- Implement python script to encrypt IP address locally.
- Able to encrypt IP addresses in logs using Crypto-PAN

Currently there are many implementations of the Crypto-PAn Algorithm:

- pycryptopan
- yacryptopan
- the original C++
- David Stott's Lucent Tech C++. This one is very fast since it uses the AES NI Intel Instruction.
- opencores implemenation. Requires hardware but is blazingly fast.
- ... (probably some more)...

```
Nanxins-MacBook-Pro:pycryptopan-0.01 jinnanxin$ python cryptopan.py
Line 1: 192.0.2.1
calculated: 2.90.93.17
Line 2: 192.0.3.2
calculated: 2.90.92.211
Line 3: 192.16.4.55
calculated: 2.77.195.201
Line 4: 192.16.4.10
calculated: 2.77.195.244
Line 5: 192.16.4.125
calculated: 2.77.195.133
Line 6: 192.16.4.12
calculated: 2.77.195.243
Line 7: 192.1.2.3
calculated: 2.91.249.220
```

```
(cryptopan) [goughes@scholar-fe03 ~]$ python
Python 3.6.4 |Anaconda, Inc.| (default, Jan 16 2018, 18:10:19)
[GCC 7.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from cryptopan import CryptoPan
>>> c=CryptoPan("thisKEYcanBEanySETof32characters")
>>> print(c.anonymize("128.211.143.32"))
192.211.63.31
>>> print(c.anonymize("128.211.143.33"))
192.211.63.30
>>> print(c.anonymize("128.211.10.233"))
192.211.221.150
>>> print(c.anonymize("10.11.234.5"))
49.240.150.254
>>>
```

Raw Data + encrypted IP address + dropped columns for sensitive fields

- Test with conn.log
- _node_name (drop), Ts, uid (drop), id.orig_h (encrypt), Id.orig_p, Id.resp_h (encrypt), Id.resp_p, Proto, Service, Duration, Orig_bytes, Resp_bytes, Conn_state, Local_orig, Local_resp, Missed_bytes, History, Orig_pkts, Orig_ip_bytes, Resp_pkts, Resp_ip_bytes, Tunnel_parents (drop) ,orig_l2_addr (drop),resp_l2_addr (drop)

```
pal-nat186-13-104:drop_field jinnanxin$ python drop_field_test.py
number_of_cols:  24
Original data set:
['worker-7-9', '1551675593.563488', 'CNvRcr2wtQ06zptvs1', '185.176.26.101', '44059', '128.211.149.157', '11911', 'tcp', '-', '-', '-', '-', 'S0', 'F', 'T', '0', 'S', '1', '40', '0', '0',
 '(empty)', 'd0:c7:89:a9:3e:c0', '00:2a:6a:8b:60:41\n']
['worker-7-9', '1551675593.788180', 'CBFYPPIsTeWB8SHHb', '112.10.242.140', '10175', '128.211.134.190', '5555', 'tcp', '-', '3.000977', '0', '0', 'S0', 'F', 'T', '0', 'S', '2', '104', '0'
, '0', '(empty)', '70:e4:22:73:df:f5', '00:2a:6a:8b:60:41\n']
['worker-2-2', '1551675593.967076', 'CzBEsO2juBuDwKrwlk', '107.170.202.251', '57368', '128.211.142.38', '4786', 'tcp', '-', '-', '-', '-', 'S0', 'F', 'T', '0', 'S', '1', '40', '0', '0',
'(empty)', 'd0:c7:89:a9:3e:c0', '00:2a:6a:8b:60:41\n']
['worker-2-2', '1551675589.112054', 'CX5Fukcp1gjVPUkda', '172.18.84.144', '59937', '128.210.11.57', '53', 'udp', 'dns', '0.000376', '46', '294', 'SF', 'T', 'F', '0', 'Dd', '1', '74', '1'
, '322', '(empty)', '00:2a:6a:8b:60:41', 'd0:c7:89:a9:58:c0\n']
['worker-2-2', '1551675539.622676', 'Cylcxm3FaOAjYUM36h', '128.211.146.30', '3', '107.170.202.251', '10', 'icmp', '-', '-', '-', '-', 'OTH', 'T', 'F', '0', '-', '1', '68', '0', '0', '(em
pty)', '00:2a:6a:8b:60:41', '70:e4:22:73:df:f5\n']
['worker-2-2', '1551675508.120088', 'Cf7x07430AS7MXDBii', '149.165.238.190', '39048', '128.211.136.42', '861', 'tcp', '-', '86.759797', '404', '224', 'SF', 'F', 'T', '0', 'ShAdDaFf', '14
', '1140', '13', '908', '(empty)', '70:e4:22:73:df:f5', '00:2a:6a:8b:60:41\n']
['worker-8-3', '1551675593.987926', 'CkYoFFVLexe0lMBJ', '5.188.206.134', '53106', '128.211.132.40', '9555', 'tcp', '-', '-', '-', '-', 'S0', 'F', 'T', '0', 'S', '1', '40', '0', '0', '(em
pty)', '70:e4:22:73:df:f5', '00:2a:6a:8b:60:41\n']
['worker-8-3', '1551675594.175753', 'Cb5bw6C4kbrHJ99a4', '208.100.26.228', '52846', '128.211.152.155', '5672', 'tcp', '-', '-', '-', '-', 'S0', 'F', 'T', '0', 'S', '1', '40', '0', '0', '
(empty)', '70:e4:22:73:df:f5', '00:2a:6a:8b:60:41\n']
['worker-8-3', '1551675594.242960', 'CW8az03tVVw9k4dwni', '89.248.174.3', '36235', '128.211.149.8', '143', 'tcp', '-', '-', '-', '-', 'S0', 'F', 'T', '0', 'S', '1', '40', '0', '0', '(emp
ty)', 'd0:c7:89:a9:3e:c0', '00:2a:6a:8b:60:41\n']
['worker-8-3', '1551675594.526526', 'CV3fnF3KlDsb6tz7Mk', '178.128.29.142', '36518', '128.211.136.164', '443', 'tcp', '-', '-', '-', '-', 'S0', 'F', 'T', '0', 'S', '1', '40', '0', '0', '
(empty)', '70:e4:22:73:df:f5', '00:2a:6a:8b:60:41\n']
['worker-8-3', '1551675594.716609', 'C4KiZ407qAhEMvhd8', '212.129.33.58', '48950', '128.211.147.137', '2000', 'tcp', '-', '-', '-', '-', 'S0', 'F', 'T', '0', 'S', '1', '40', '0', '0', '(
empty)', 'd0:c7:89:a9:3e:c0', '00:2a:6a:8b:7c:41\n']
['worker-3-5', '1551675593.499079', 'CLHJ67laBA1gK7D89', '185.176.27.62', '47037', '128.211.132.230', '568', 'tcp', '-', '-', '-', '-', 'S0', 'F', 'T', '0', 'S', '1', '40', '0', '0', '(e
mpty)', '70:e4:22:73:df:f5', '00:2a:6a:8b:60:41\n']
```

Raw Data + encrypted IP address + dropped columns for sensitive fields + hashed columns for sensitive fields

ML Models

K Means (Unsupervised learning)

- User has to specify k (number of clusters) in the beginning.
- K-means can only handle numerical data.

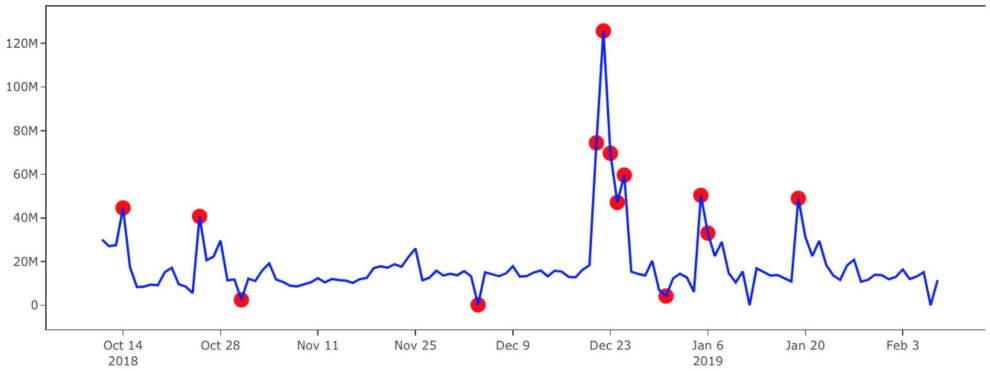
Time series forecasting Models, LSTM Models

- adds the complexity of a sequence dependence among the input variables.
- Each metric needs to be validated with parameters to detect anomalies.
- Metrics with different distribution of data needs different approach.

Isolation Forest for anomaly detection

- works for multiple metrics at a time and can be drilled down to anomalies on individual metrics in them.

Isolation Forests

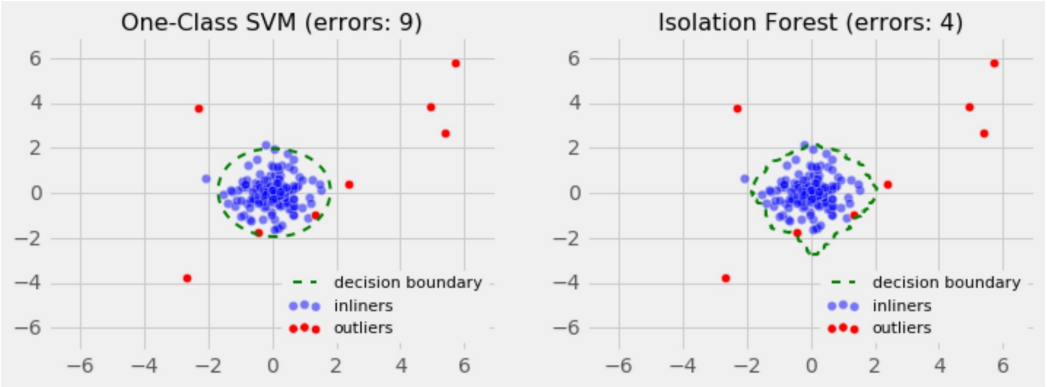


A sudden spike or dip in a metric is an **anomalous** behavior and both the cases needs attention. (Sample diagram)

- unsupervised
- The data we used is a **use case for network traffic** with 5 features.
- Identify first if there is an anomaly at a use case level. Then drill down to specific metrics (if required).
- Isolation forest tries to **separate each point** in the data.
- An anomalous point could be separated in a few steps while normal points which are closer could take significantly more steps to be segregated.

ML Models

- Set up proper python environments and Jupyter notebooks to perform isolation forest analysis with Scikit-Learn
- Performed isolation forest analysis on some test datasets (tried random forest, RNN LSTM approach). IF deals with the effects of swamping and masking effectively.



HTTP Log Analysis - Raw vs. Encrypted IP

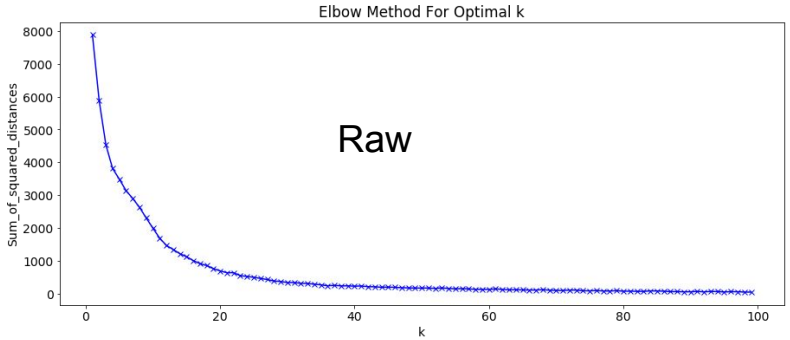
- Zeek HTTP log with ~35k rows was analyzed using 5 fields
  - id.resp_p - Responding web server port
  - method - HTTP request method (GET/POST/etc)
  - resp_mime_types - Type of file returned
  - request_body_len - Length of the request
  - id.resp_h - IP Address of responding web server
- Bro Analysis Toolkit was used to convert logs to Pandas dataframes
- IsolationForest model from SciKit-Learn
- The categorical variables (IP Address, Mime Type, Method) were encoded using dummy encoding via Pandas get_dummies()
  - Input - (34455, 5)
  - MIME/Method - (34455, 25)
  - IP Address - (34455, 269)
- All analysis python code written in JupyterHub

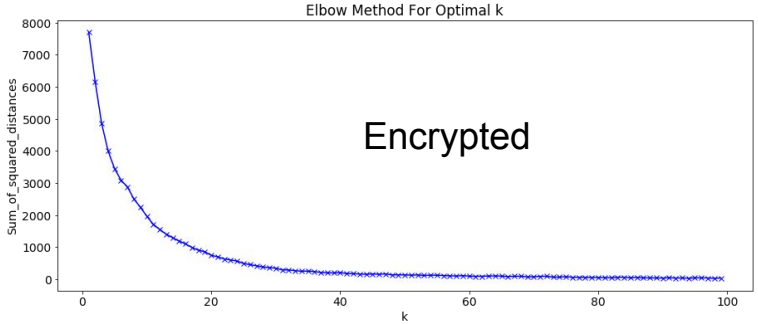
ts	104.31.90.195	104.31.91.195	111.68.96.165	119.40.121.16
2019-03-04 00:00:00.291222	0	0	0	0
2019-03-04 00:00:01.725721	0	0	0	0
2019-03-04 00:00:02.307902	0	0	0	0
2019-03-04 00:00:02.847030	0	1	0	0
2019-03-04 00:00:02.019046	1	0	0	0

HTTP Log Analysis - Raw vs. Encrypted IP

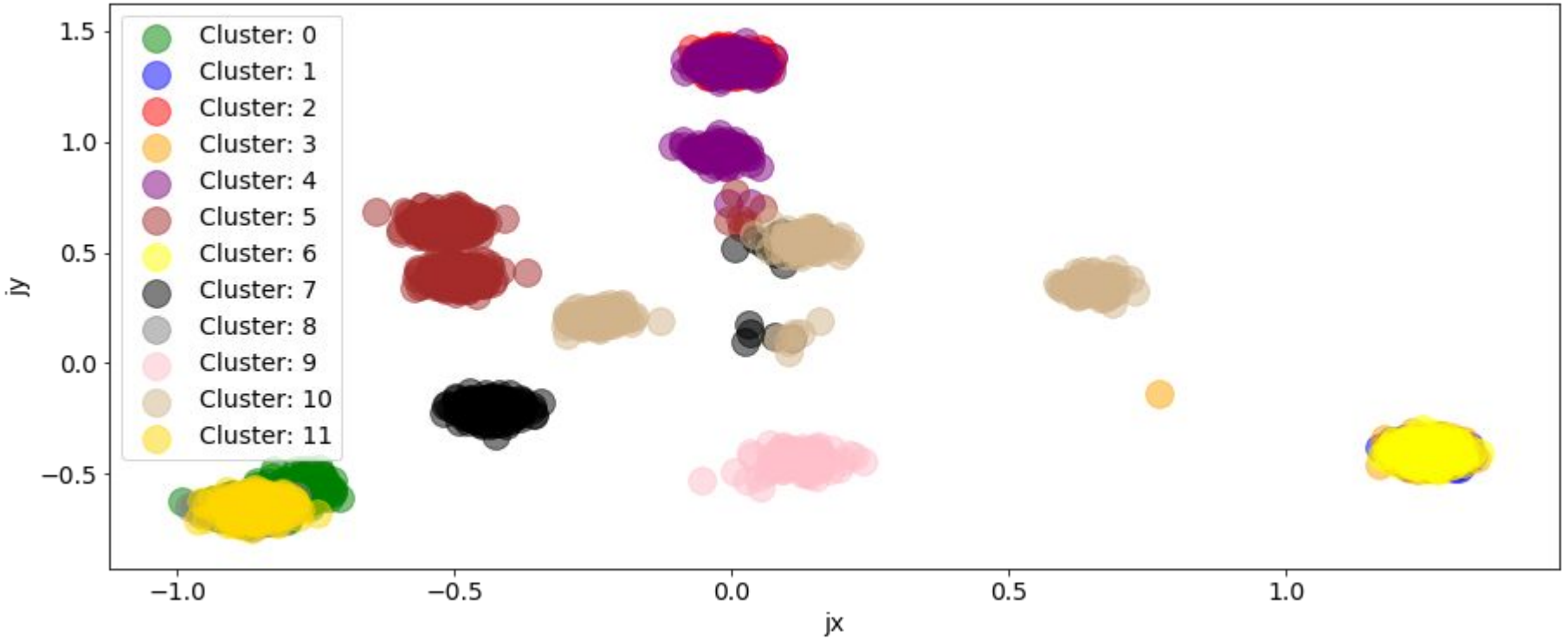
- Trained two Isolation Forest models, 10% contamination
  - Raw Data
  - Raw Data with Encrypted IP
- Predicted anomalous rows for each dataset
  - Raw Data produced 3408 outliers
  - Raw Data with Encrypted IP produced 3434 outliers
- Used PCA and KMeans clustering to visualize the anomalous HTTP requests
  - Elbow method for determining # of clusters on each dataset

* All models used the same random seed for consistency across runs

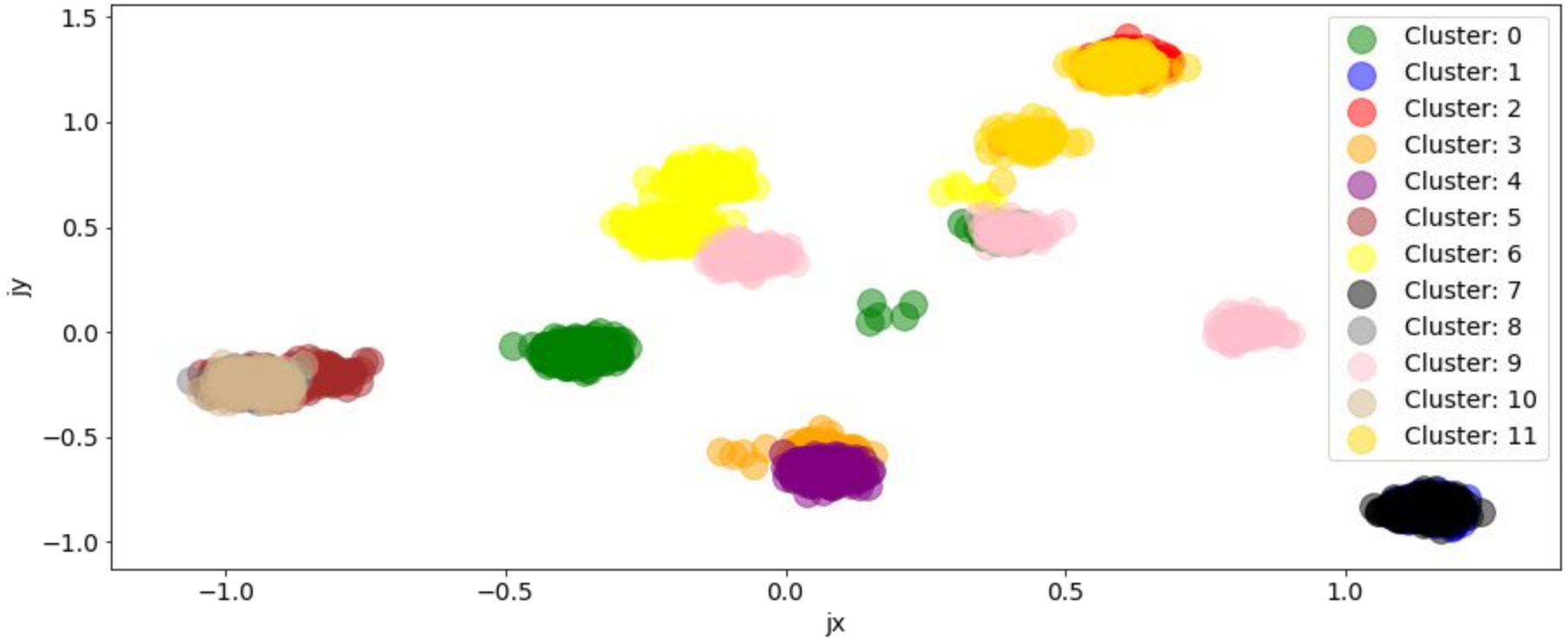




PCA and KMeans for Anomalous Raw Data

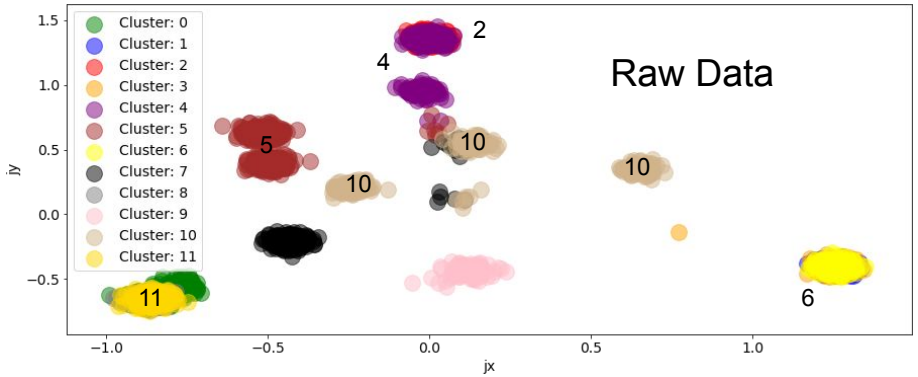


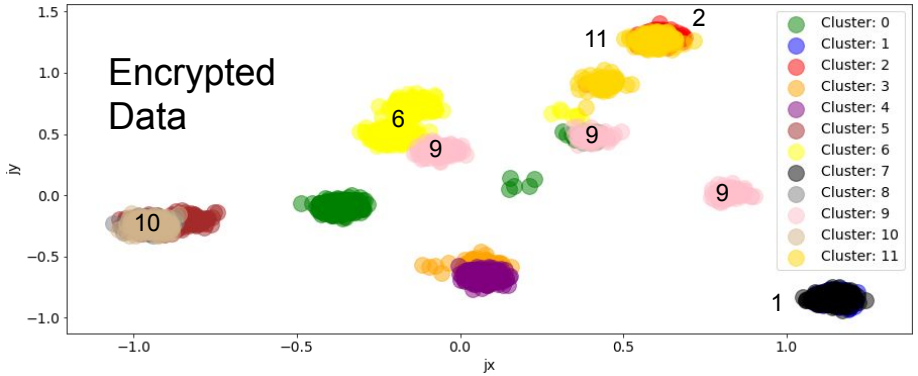
PCA and KMeans for Anomalous Encrypted Data



Comparing the Isolation Forest Clusters

Raw (# obs)	Encrypted (# obs)	What is it?
Cluster 11 (275)	Cluster 10 (275)	Traffic to Science Gateway at NERSC
Cluster 4 (268)	Cluster 11 (267)	External PHP GET Scan on a Web Server
Cluster 2 (252)	Cluster 2 (252)	External PHP POST Scan on a Web Server
Cluster 5 (467)	Cluster 6 (468)	External PHP GET Scan on a Web Server
Cluster 6 (282)	**Not found!**	Cluster node updating CRL
Not found!	Cluster 1 (283)	Cluster node updating CRL
Cluster 10 (291)	Cluster 9 (220)	Various web traffic





Cluster Details

- Scanner running a bunch of POST requests to random PHP pages

Cluster 2: 252 observations

ts	id.resp_p	method	resp_mime_types	request_body_len	id.resp_h
2019-03-04 00:18:08.708365	80	PROPFIND	text/html	0	216.16.111.30
2019-03-04 00:18:14.745586	80	POST	text/html	19	216.16.111.30
2019-03-04 00:18:15.335620	80	POST	text/html	21	216.16.111.30
2019-03-04 00:18:15.097184	80	POST	text/html	19	216.16.111.30
2019-03-04 00:18:14.867296	80	POST	text/html	19	216.16.111.30
2019-03-04 00:18:15.919097	80	POST	text/html	22	216.16.111.30
2019-03-04 00:18:15.216321	80	POST	text/html	19	216.16.111.30
2019-03-04 00:18:14.983059	80	POST	text/html	19	216.16.111.30
2019-03-04 00:18:16.269337	80	POST	text/html	36	216.16.111.30
2019-03-04 00:18:15.804674	80	POST	text/html	18	216.16.111.30

How similar was the Isolation Forest output?

Conclusions

- Scripts were created to anonymize the connection and HTTP logs from Zeek
- Results of using ML models was different when using encrypted data
- Using encrypted IP addresses changed the anomalous traffic returned by the Isolation Forest analysis
  - Most likely due to differences in dummy coding the address
  - The same types of traffic, but from different hosts
- PCA and KMeans results were also skewed, but clusters were similar
- Generally speaking, anomaly detection only gets you so far
  - Anomalous != Malicious
  - Some anomalous clusters were normal traffic, some were not!
  - Need domain expert review to determine malicious traffic

Future Work

- Deeper understanding of why Isolation Forest results were different
  - Find optimal decision function threshold
- Alternative ways to use IP address as a feature
  - 32 features for each bit of an IP address
- Explore alternative anomaly detection techniques
  - LocalOutlierFactor (LoF) for spark (on much larger datasets)
- Group anomalous behaviour that occurs continuously
- Experimental setup and analysis for real-time network traffic.

Questions?