Anomaly-based Network Intrusion Detection

Master Thesis Project #75

Philippe Partarrieu <ppartarrieu@os3.nl>
Philipp Mieden <pmieden@os3.nl>

Supervisors
Joao Novaismarques <joao.novaismarques@kpn.com>
Jordi Scharloo <jordi.scharloo@kpn.com>
Giovanni Sileno <g.sileno@uva.nl>

UNIVERSITEIT VAN AMSTERDAM

The problems with signature-based solutions

- Signatures can be easily bypassed by updating key attack characteristics
- Boundless growth of signature DB over time
- Requires Up-to-date signature DB

Anomaly detection can help here

- System learns the baseline of normal network behavior
- Alerts can be generated once there are deviations
- Many different algorithms to choose from
- Known to work well for many similar data science problems

Research Question: Which algorithms are best suited for the task of intrusion detection in computer networks and why?

Dataset: CIC IDS 2018

Large scale, modern dataset with traffic from over 400 different machines

- 450GB pcaps
- 80+ network flow based features provided as CSV
- 6 types of attacks (brute-force, DoS, DDoS, bot, injection, infiltration)

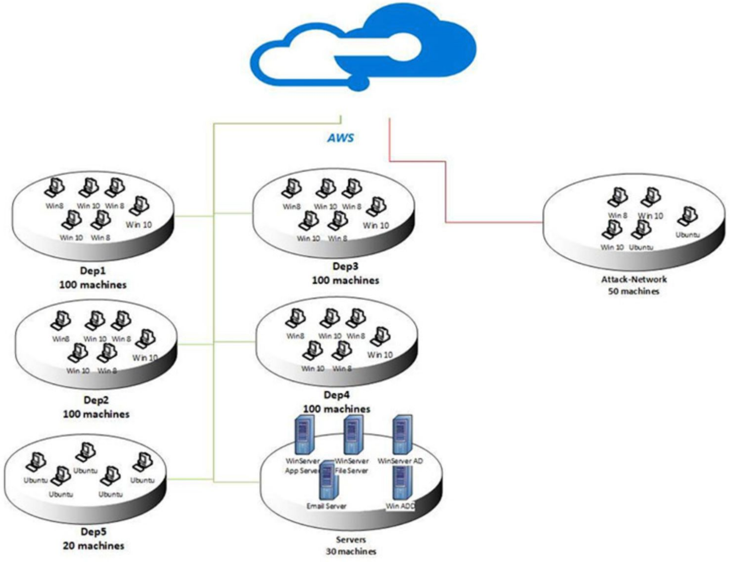
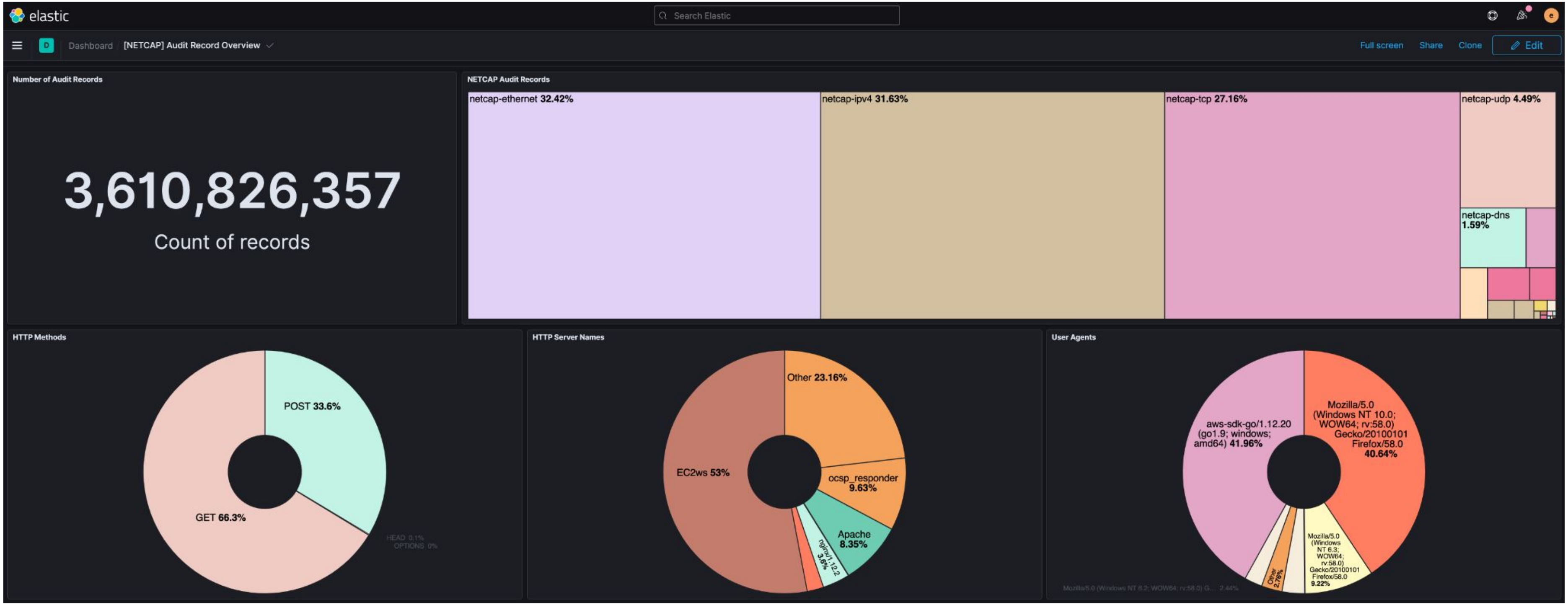


Figure 1: Network Topology

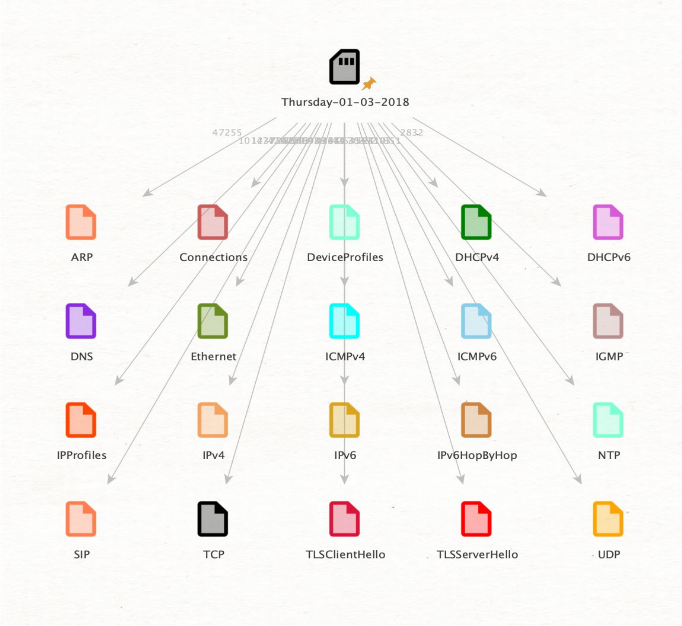
https://www.unb.ca/cic/datasets/ids-2018.html

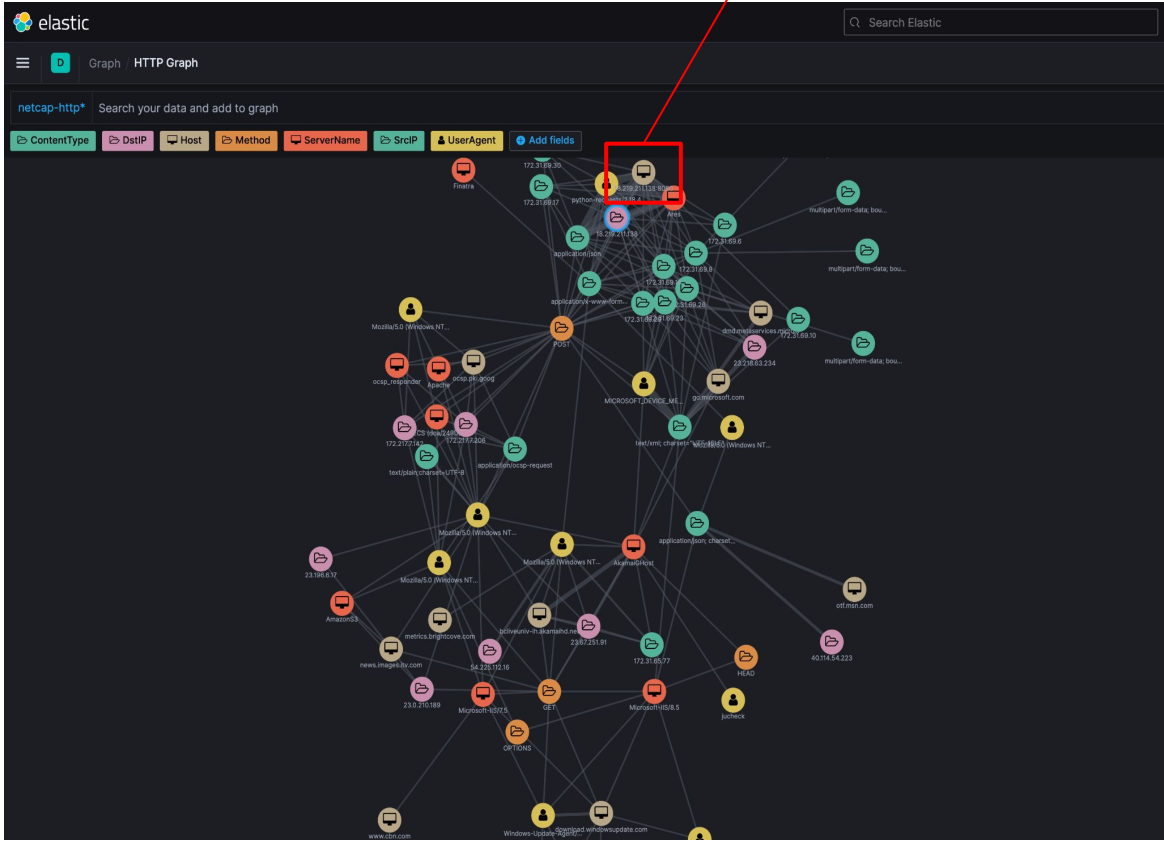
Dataset: Exploration



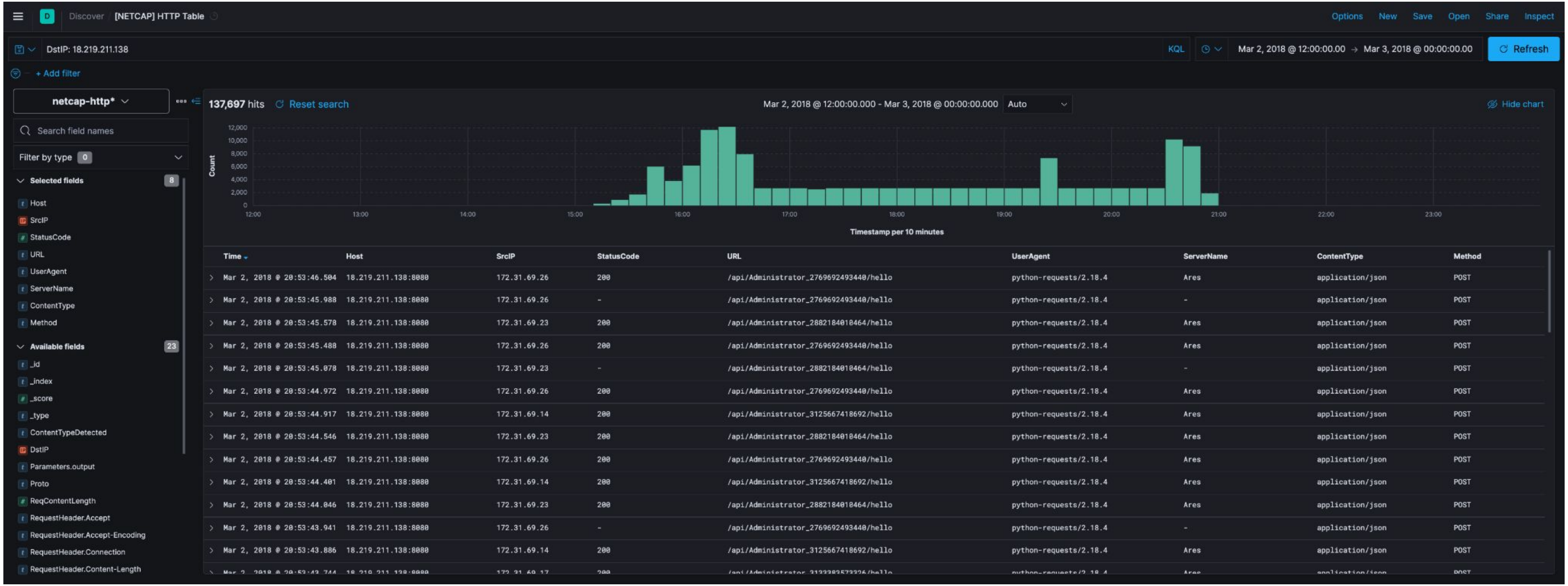
HTTP request to IP 18.219.211.138:8080
instead to domain name

Dataset: Exploration





Dataset: Exploration - Botnet activity



Dataset: Issues

- Original network flow CSV was missing information
  - Flow ID, Src IP, Dst IP, Src Port
- Labeling tool not open source
- One provided pcap was missing attack traffic
  - one day (Thursday-15-02-2018) contains no attack traffic at all

Dataset: Imbalance

An issue for some network intrusion detection algorithms

- Some algorithms require an equal distribution of positive and negative classes
- Some algorithms require a high ratio of anomalies

Different strategies to deal with dataset imbalance exist:

- Class weights
- Oversampling
- Outlier exposure

Metrics: Confusion Matrix

True Positive	False Positive
False Negative	True Negative

Correct prediction

Incorrect prediction

Metrics: Accuracy

True Positive 0	False Positive 0
False Negative 1	True Negative 99

Accuracy is never suitable as a metrics when dealing with **imbalanced data**

A simple example: assume a dataset with 99 benign and 1 malicious sample

An algorithm that **always predicts** that the sample is **normal** would score **99% accuracy**. It has identifies 99 benign events but missed the anomaly

Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$ = $\frac{99}{100}$

Metrics: F1 score

True Positive 0	False Positive 0
False Negative 1	True Negative 99

For the same dataset, let's calculate the F1 score

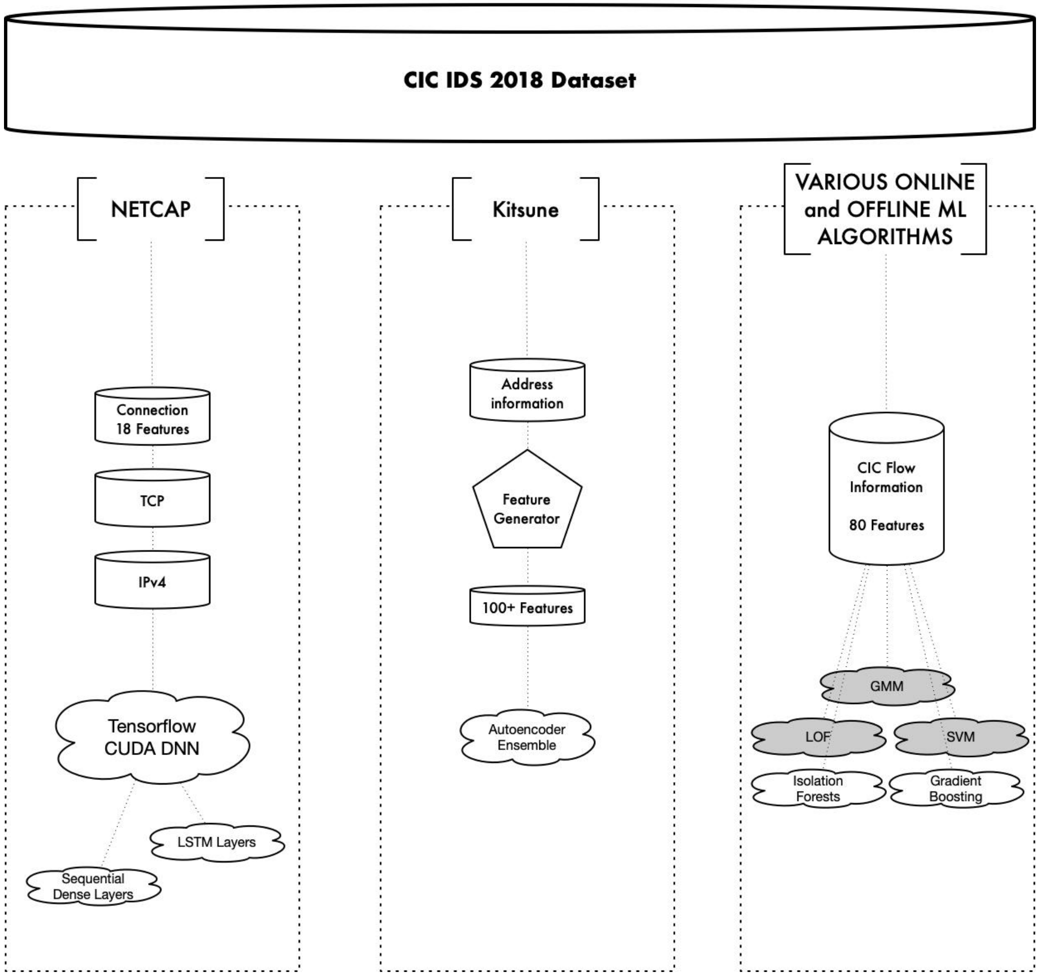
F1 score = $2 * \frac{precision * recall}{precision + recall}$

Where *precision* = $\frac{TP}{TP + FP}$ and *recall* = $\frac{TP}{TP + FN}$

We obtain F1 = 2 * 0 = **0**

The F1 score takes the relevance of the different error types into account!

Experiment Design



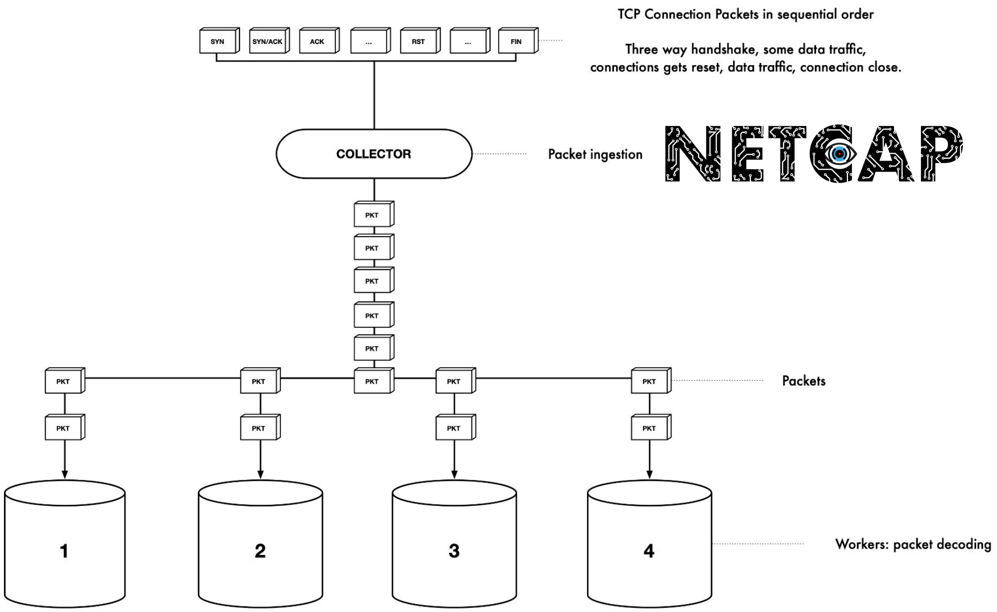
Experiment Design: Model Choice

Model	**Online**	**Supervised**	**Deep Learning**
Deep Neural Network	✓	✓	✓
Auto Encoders	✓	X	✓
Gradient Boosting	X	✓	X
Isolation Forest	X	X	X

Feature Extraction

Ideally parallelized to take advantage

of multi-core processors.



Concurrency causes problems though:

- Race conditions
- Shared state

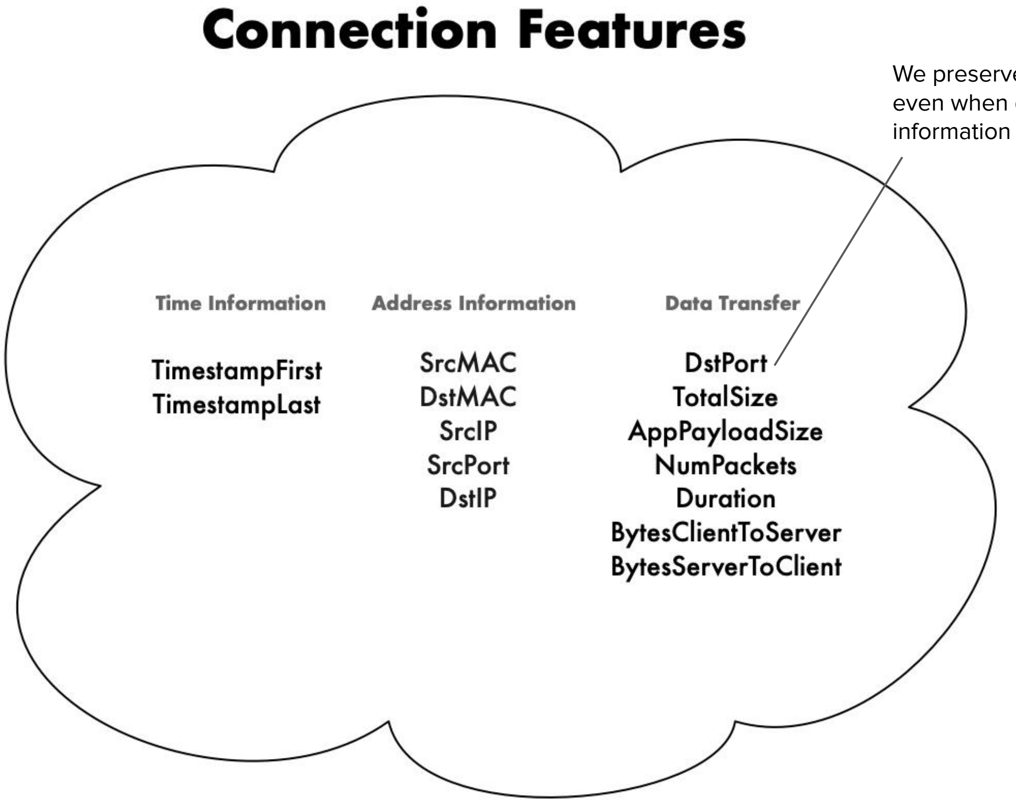
Ideally something lightweight to calculate that is still expressive enough to capture network trends

- Solution: bi-directional network flow summaries (Connection Audit Records)
  - Requires keeping only minimal state, aggregate subflows
  - Processing rate: 300K pkts/second on a Ryzen 9, 16 core processor

Connection Audit Records

We chose a deliberately simple set of features for establishing a baseline.

Additional features can always be added later and their effect on prediction quality measured.

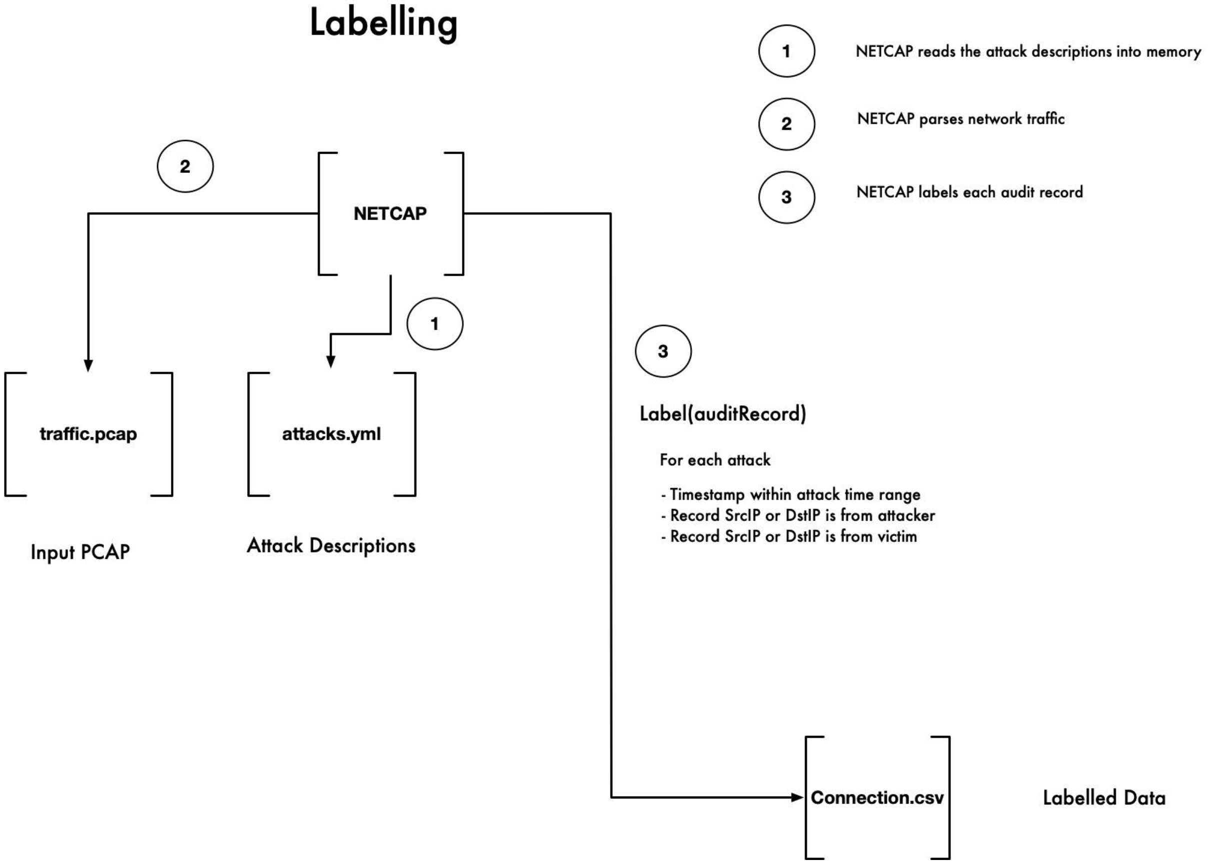


We preserved the DstPort even when dropping address information

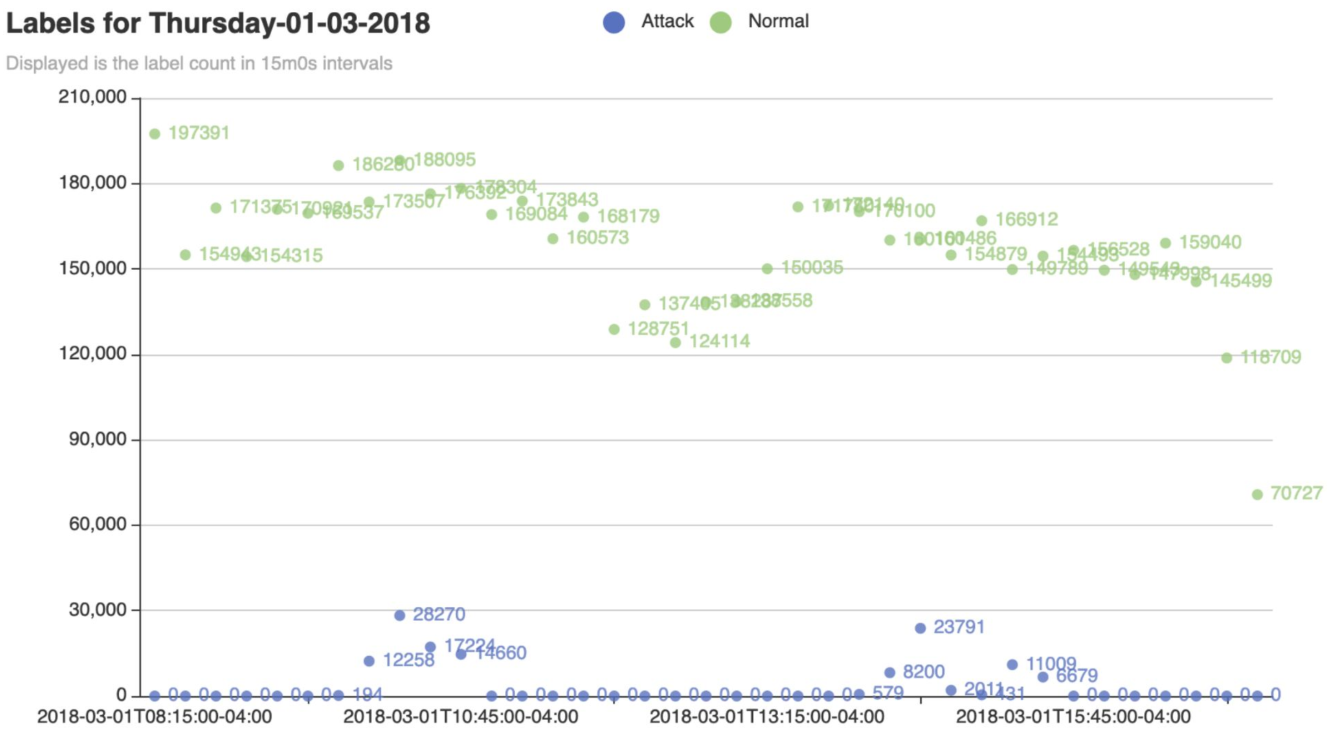
Labelling: adding attack information

We implemented unit tests to ensure labelling works as expected

Labelling can be applied to any audit record from NETCAP



Labelling: adding attack information



Data Encoding

Categorical data (eg: strings) must be transformed into numeric values

Multiple approaches for encoding categorical data:

- Enumeration
- ~~One Hot~~
- ~~Learned Embedding~~

We chose enumeration because it does not alter the feature dimension!

Strings: Enumeration

map[string]int

```
{
"TCP": 0,
"UDP": 1,
"DNS": 2,
"ARP": 3,
...
}
```

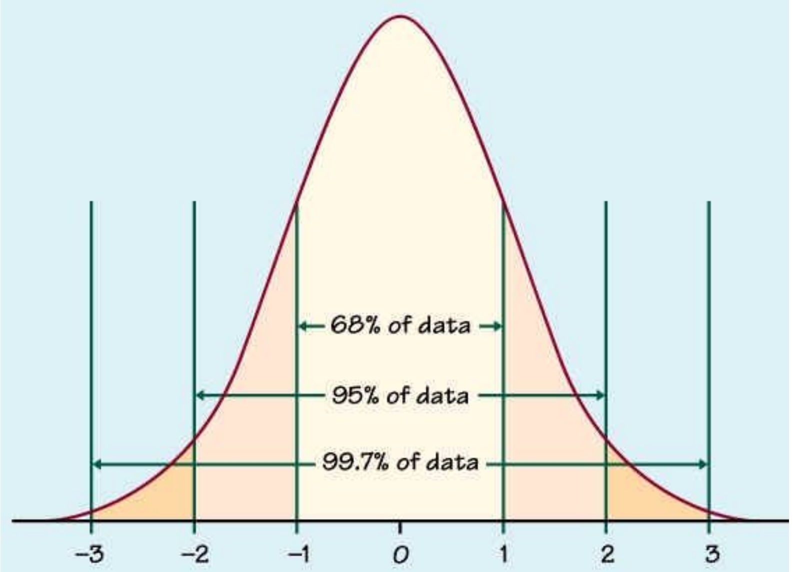
The only relation the numeric representation has to each other, is the time of first appearance.

Data Normalization

Numeric values must be normalized to reside within a certain threshold

We used:

- Zscore: The Standard Normal Distribution



http://www.ltcconline.net/greenl/courses/201/probdist/zScore.htm

Tensorflow

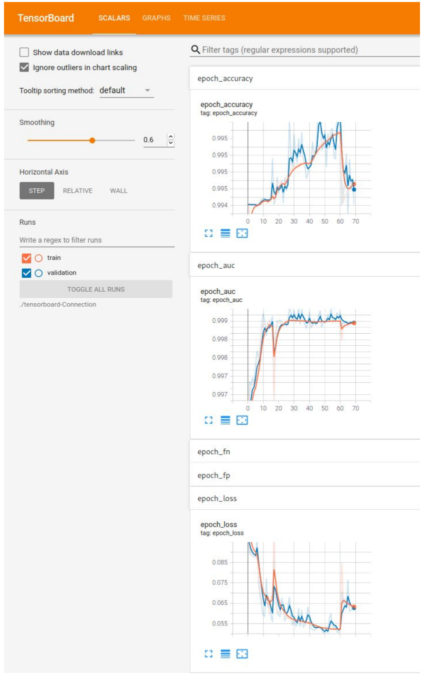
Free and open source software library for machine learning from Google

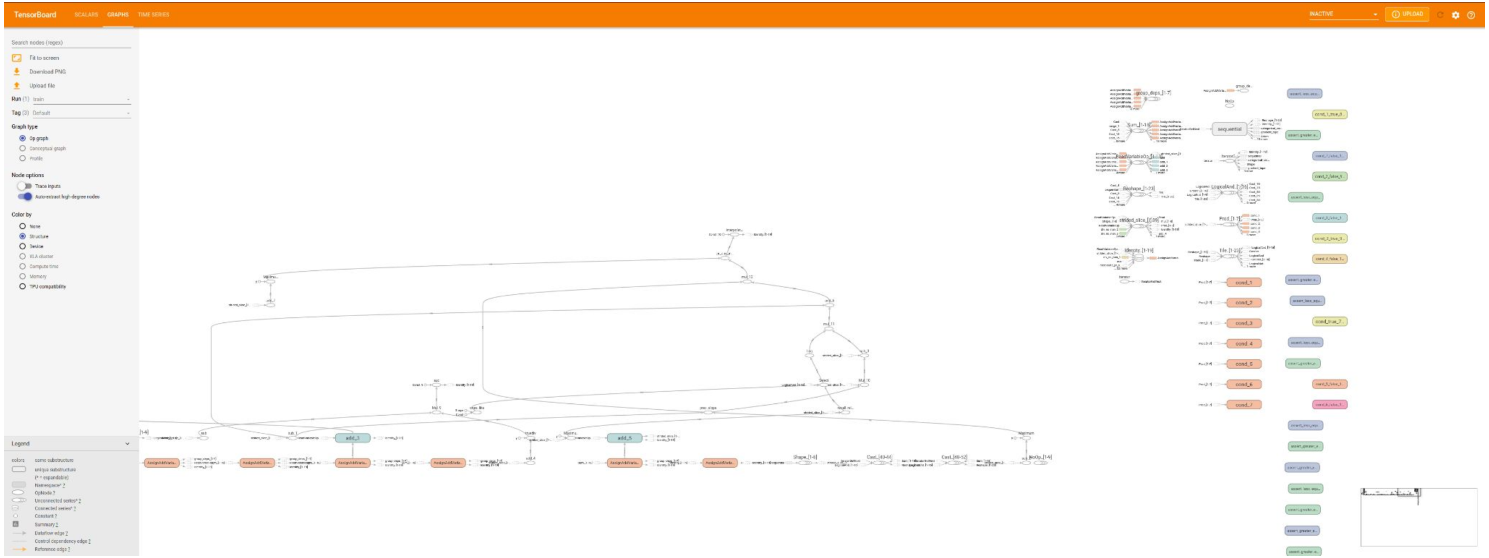
Supports different backends for computations: CPU, GPU, FPGA etc

Can be run in a cluster mode to run processing jobs on multiple hardware devices

https://www.tensorflow.org

Tensorboard

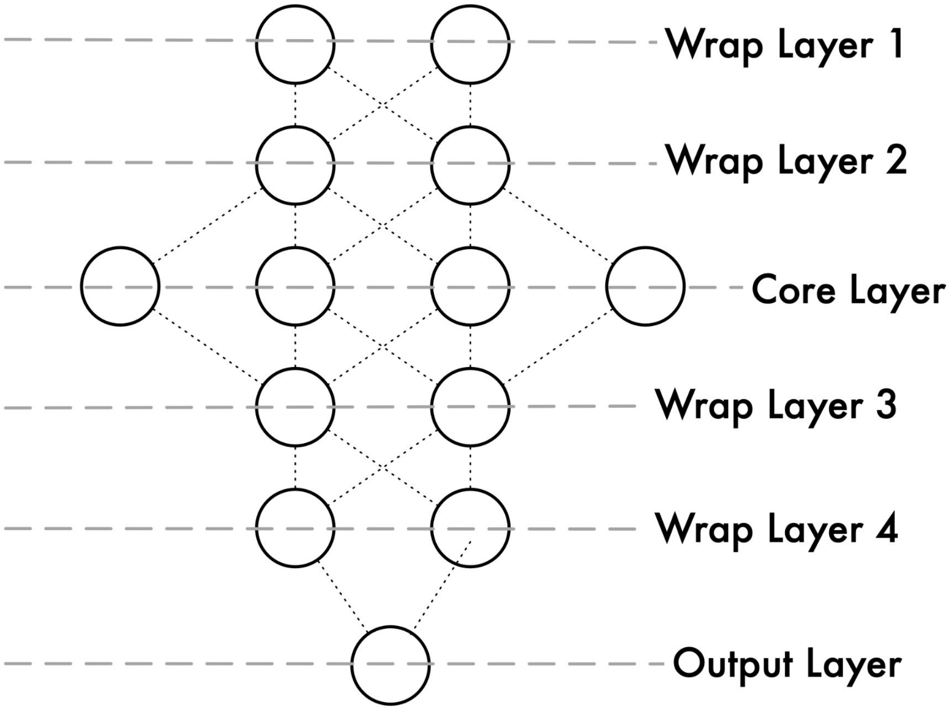




Deep Neural Network: Baseline Model

We chose a deliberately small network for the baseline experiments

Bigger does not always mean better, as later experiment results confirmed



Deep Neural Network: GPU acceleration

GEFORCE RTX 3090

- coreClock: 1.74GHz
- coreCount: 82
- deviceMemorySize: 23.67GiB
- deviceMemoryBandwidth: 871.81GiB/s

Processing 6m Connection audit records, during training and testing
~2s per Epoch (= one run over the entire data).

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.84       Driver Version: 460.84       CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  GeForce RTX 3090    Off  | 00000000:2D:00.0  On |                  N/A |
| 30%   48C    P2   122W / 370W |  23118MiB / 24243MiB |     19%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A      1182      G   /usr/lib/xorg/Xorg                767MiB |
|    0   N/A  N/A      1529      G   xfwm4                               5MiB |
|    0   N/A  N/A   3513694      G   ...AAAAAAAAA= --shared-files      121MiB |
|    0   N/A  N/A   3518857      G   ...b/firefox-esr/firefox-esr        4MiB |
|    0   N/A  N/A   3519006      G   ...b/firefox-esr/firefox-esr       42MiB |
|    0   N/A  N/A   3519763      C   python3                         22171MiB |
+-----------------------------------------------------------------------------+
```

Results: DNN* with address information

Day	14/02	15/02	16/02	20/02	21/02	22/02	23/02	28/02	01/03	02/03
Attack Labels	Brute-force	DoS	DoS	DDoS	DDoS	Brute-force / Injection	Brute-force / Injection	Infiltration	Infiltration	Bot
Attack Ratio (%)	0.48	Pcaps contain no attack traffic	0.59	4.61	2.74	0.000035	0.000048	1.06	2.1	1.6
F1	0.98	-	0.97	0.93	0.93	0	0	0.94	0.90	0.78

*one model trained per day

Results: DNN* without address information

Day	14/02	15/02	16/02	20/02	21/02	22/02	23/02	28/02	01/03	02/03
Attack Labels	Brute-force	DoS	DoS	DDoS	DDoS	Brute-force / Injection	Brute-force / Injection	Infiltration	Infiltration	Bot
Attack Ratio (%)	0.48	Pcaps contain no attack traffic	0.59	4.61	2.74	0.000035	0.000048	1.06	2.1	1.6
F1	0	-	0.99	0.82	0.94	0	0	0.28	0	0.77

*one model trained per day

Results: Isolation Forest

Day	14/02	15/02	16/02	20/02	21/02	22/02	23/02	28/02	01/03	02/03
Attack Labels	Brute-force	DoS	DoS	DDoS	DDoS	Brute-force / Injection	Brute-force / Injection	Infiltration	Infiltration	Bot
Attack Ratio (%)	5.38	0.8	12.99	7.29	12.9	0.01	0.79	11.24	28.11	3.48
F1	0.95	0.99	0.91	0.95	0.88	0.99	0.99	0.77	0.56	0.99

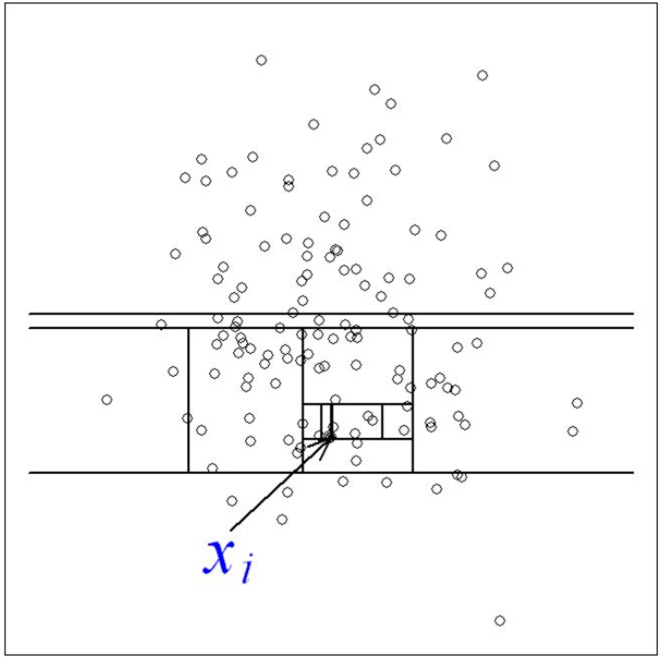
Run on enriched network flow data
With IP address information

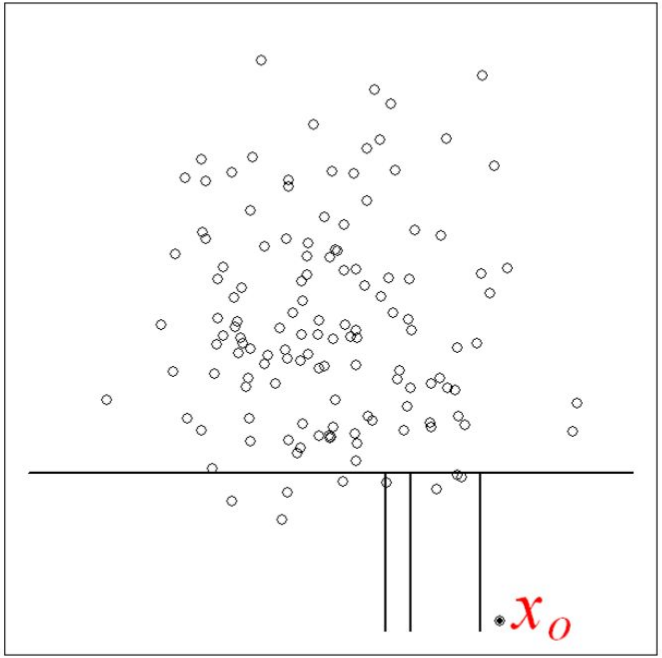
Discussion: Isolation Forest

Based on the idea that anomalies are more susceptible to isolation under random partitioning

Doesn't perform well when anomaly clusters are large and dense

To get the best results, it requires a "contamination rate"





https://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/icdm08b.pdf

Results: Gradient Boosting

Day	14/02	15/02	16/02	20/02	21/02	22/02	23/02	28/02	01/03	02/03
Attack Labels	Brute-force	DoS	DoS	DDoS	DDoS	Brute-force / Injection	Brute-force / Injection	Infiltration	Infiltration	Bot
F1	0.95	1	0.99	0.99	1	0.99	0.99	0.68	0.73	0.99

Without IP address information
Run on the first 1 million lines of each file

Discussion: Gradient Boosting

Likely overfitting the dataset

We used the first 1 million lines of each network flow file instead of the full day because the scikit learn implementation is slow

Like DNN, loops over the dataset N times

Results: Ensemble of Auto Encoders (Kitsune)

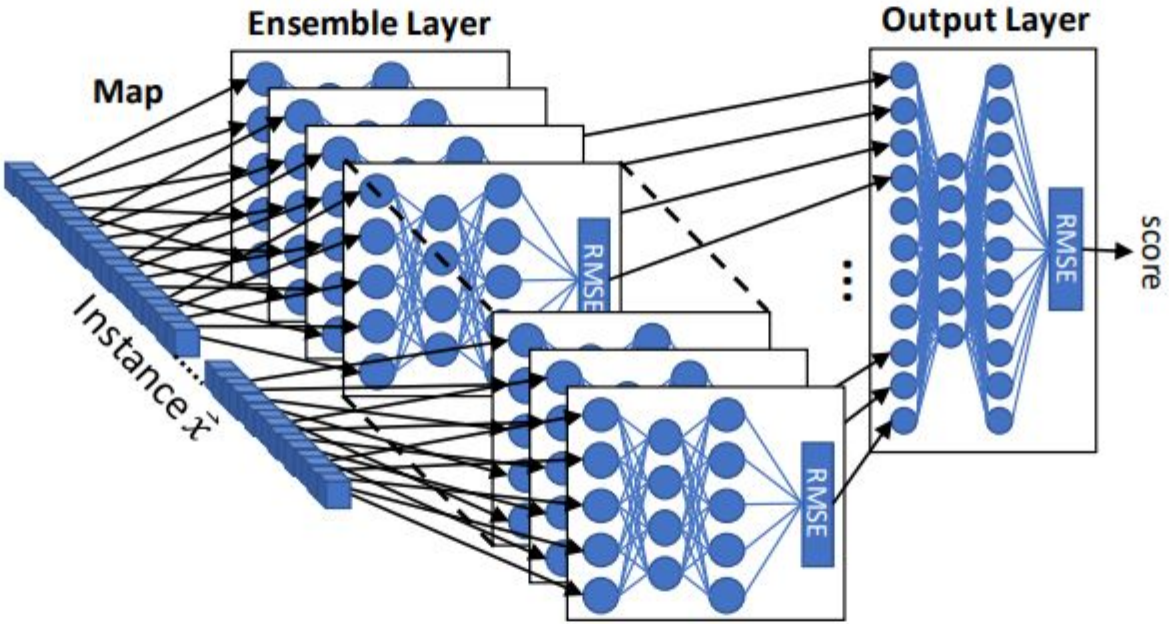
Day	14/02	15/02	16/02	20/02	21/02	22/02	23/02	28/02	01/03	02/03
Attack Labels	Brute-force	DoS	DoS	DDoS	DDoS	Brute-force	Brute-force	Infiltration	Infiltration	Bot
Attack Ratio (%)	0.0048	0.54	0.0059	0.046	0.027	0.000037	0.000049	0.011	0.21	0.016
F1	0.65	0.51	0.59	0.65	X	0.68	0.68	0.53	0.45	0.43

Run on connection audit records
With IP address information
On the first 1 million lines of each file

Discussion: Ensemble of Auto Encoders (Kitsune)

Results are poor because of under-exposure to anomalies

Takes > 24h to run on a single day with 6 million samples



https://arxiv.org/pdf/1802.09089.pdf

Results Recap

Day	14/02	15/02	16/02	20/02	21/02	22/02	23/02	28/02	01/03	02/03	**Training Time**
Attack Labels	Brute-force	DoS	DoS	DDoS	DDoS	Brute-force / Injection	Brute-force / Injection	Infiltration	Infiltration	Bot	
DNN	0.98	-	0.97	0.93	0.93	0	0	0.94	0.90	0.78	2 min*
iForest	0.95	0.99	0.91	0.95	0.88	0.99	0.99	0.77	0.56	0.99	3 min⁺
GBoost	0.95	1	0.99	0.99	1	0.99	0.99	0.68	0.73	0.99	30 min⁺
Kitsune	0.65	0.51	0.59	0.65	X	0.68	0.68	0.53	0.45	0.43	4 hours⁺

* run on GPU: GEFORCE RTX 3090
⁺ run on CPU: AMD Ryzen 5 3600 6-Core @ 3.6GHz

Conclusion

High success ratio for the supervised strategies, even without address information

- Knowledge transfer between networks should be possible

GPU or parallelisation are essential for processing large amounts of data

Overfitting of certain models can be mitigated to make them generalisable

Future Work

Complete alert pipeline and test analysis in Maltego / Elastic

Further research and more experiments with unsupervised algorithms

Recap and contributions

Analyzed a modern dataset for network intrusion detection using state of the art algorithms for anomaly detection

Found numerous errors in the dataset and reported them back to authors

Created our own feature extraction and labelling logic and open sourced it

Created a DNN using tensorflow and evaluated its performance

Created a generic analyzer with support for many other online and offline models, including isolation forests, gradient boosting, kitsune and more

Recap and contributions

Bootstrapped a pipeline for feeding the generated alerts into a modern analytics platform, Elastic / Kibana or Maltego

Open sourced our entire experiment testbed and internal documentation for reproducibility

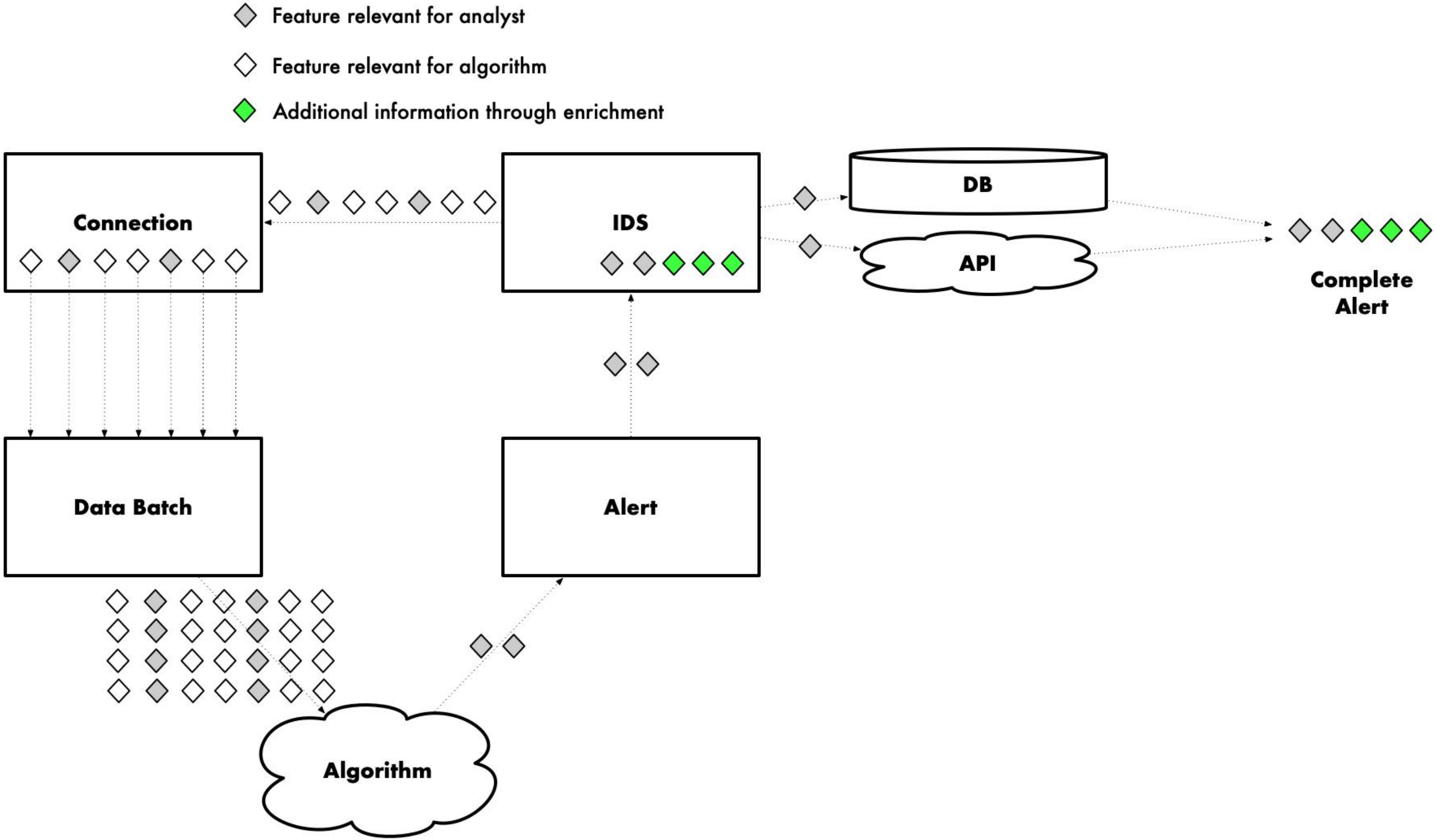
Evaluated the novel autoencoder ensemble Kitsune framework on the CIC IDS 2018 dataset

Questions?

Links:

https://github.com/dreadl0ck/netcap
https://github.com/ppartarr/anomaly

Data Flow



Feature relevant for analyst
Feature relevant for algorithm
Additional information through enrichment
Connection
IDS
DB
API
Complete Alert
Data Batch
Alert
Algorithm

DNN Train / Test Split

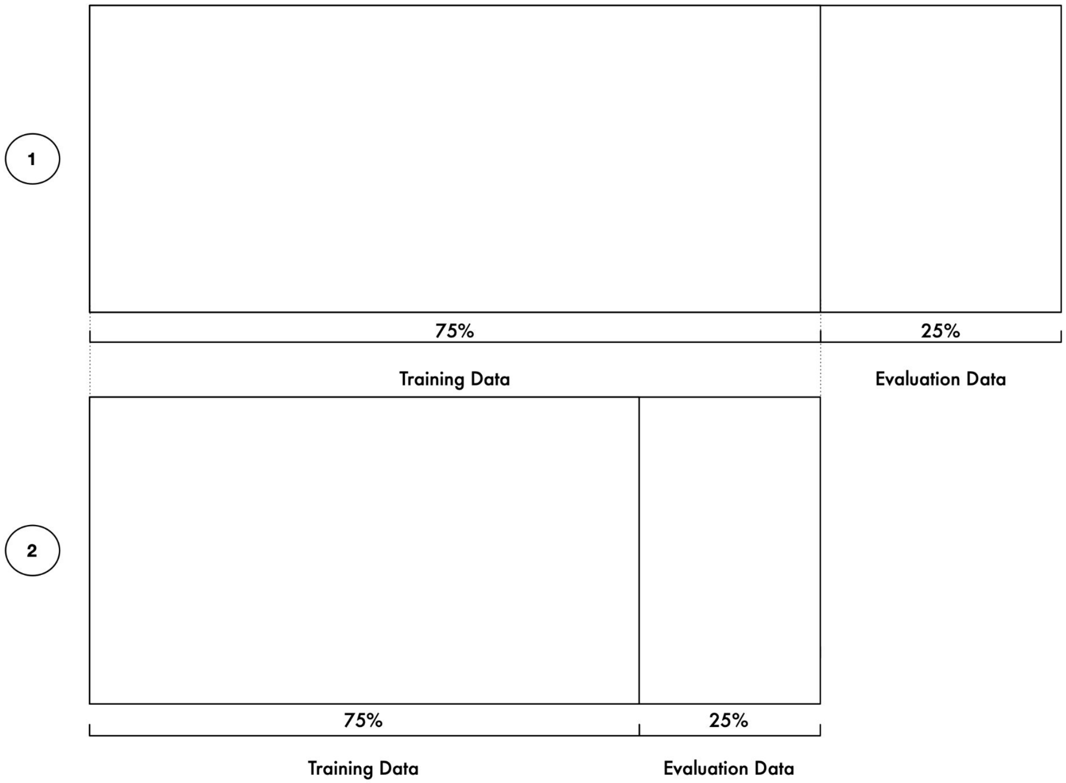
DNN Train / Test Split

The DNN should never be evaluated on data it has seen already in the training phase.
Therefore the data will be split into a training and evaluation portion initially.

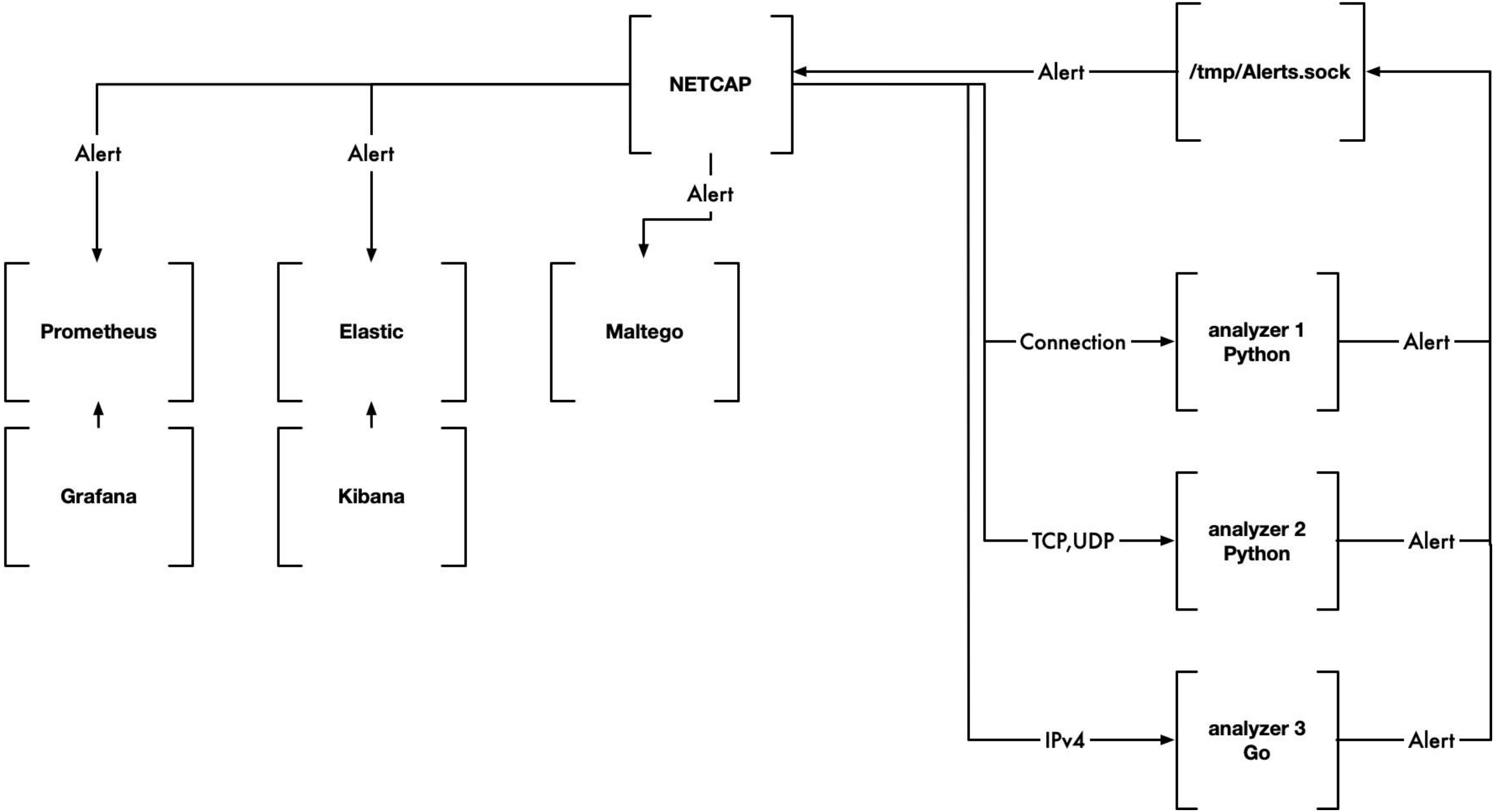
The ratio for this is configurable, the baseline experiments use 75% of the data for training and 25% for evaluation.

1. Training / Evaluation Split is created
2. For the training phase, the data is split again for training and testing according to configuration

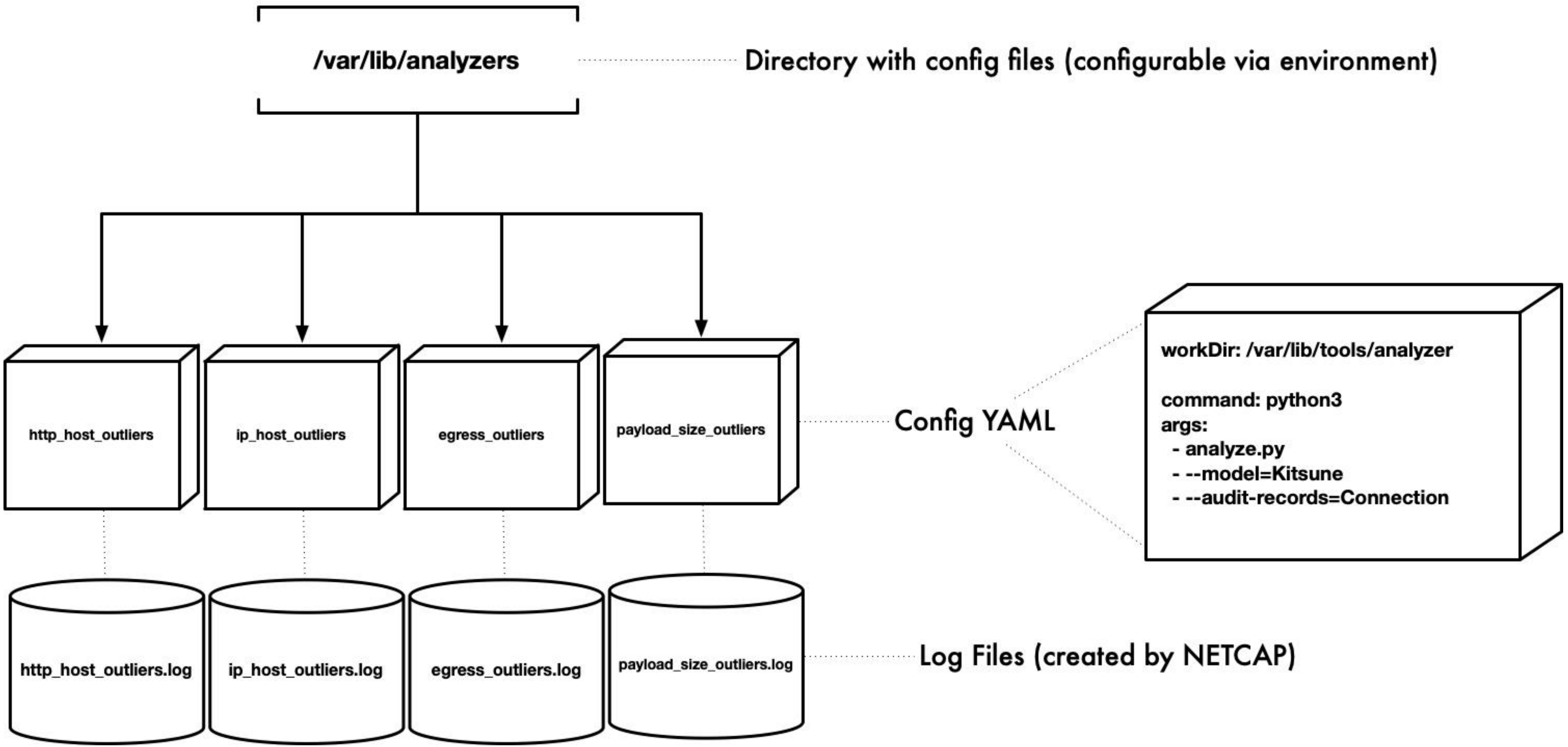
DNN Train / Test Split



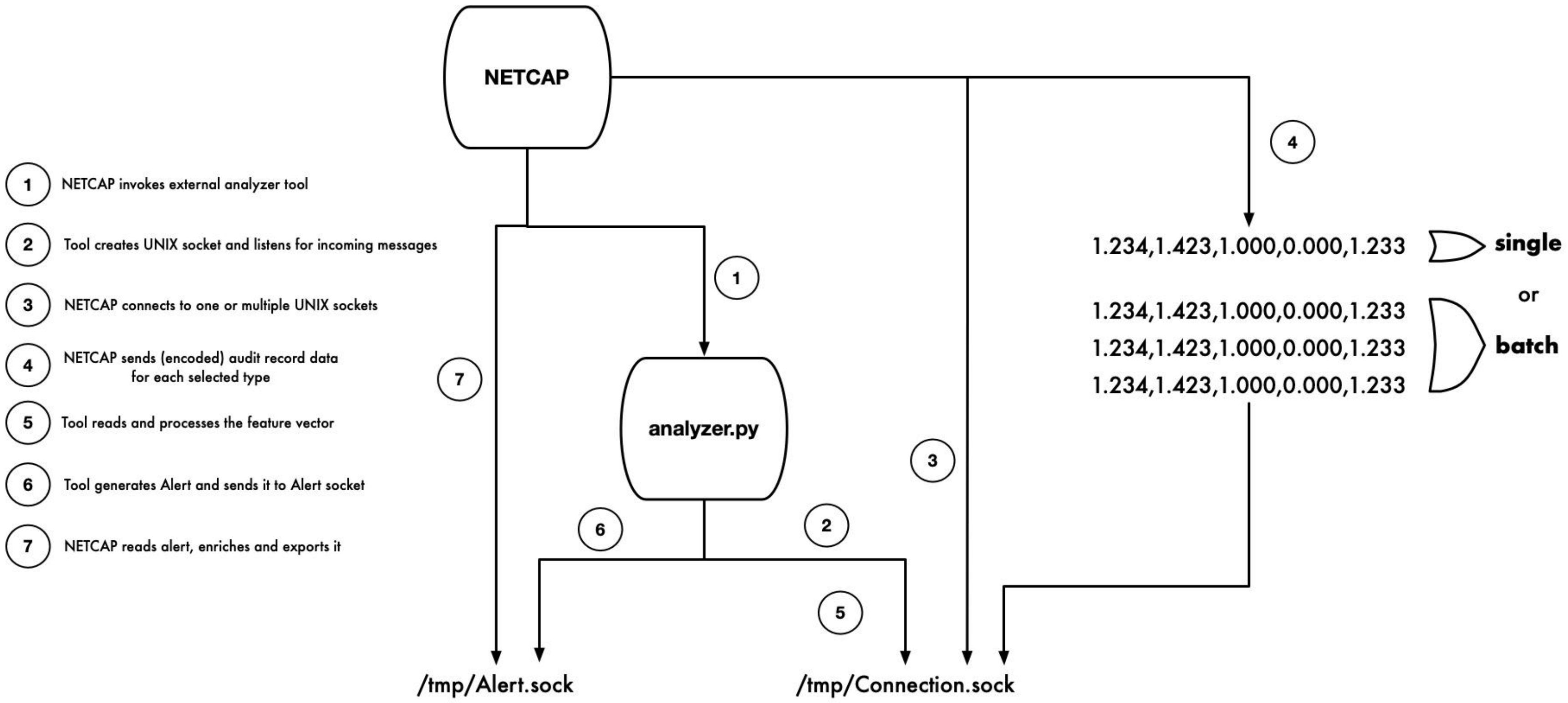
Analyzer Plugins for NETCAP



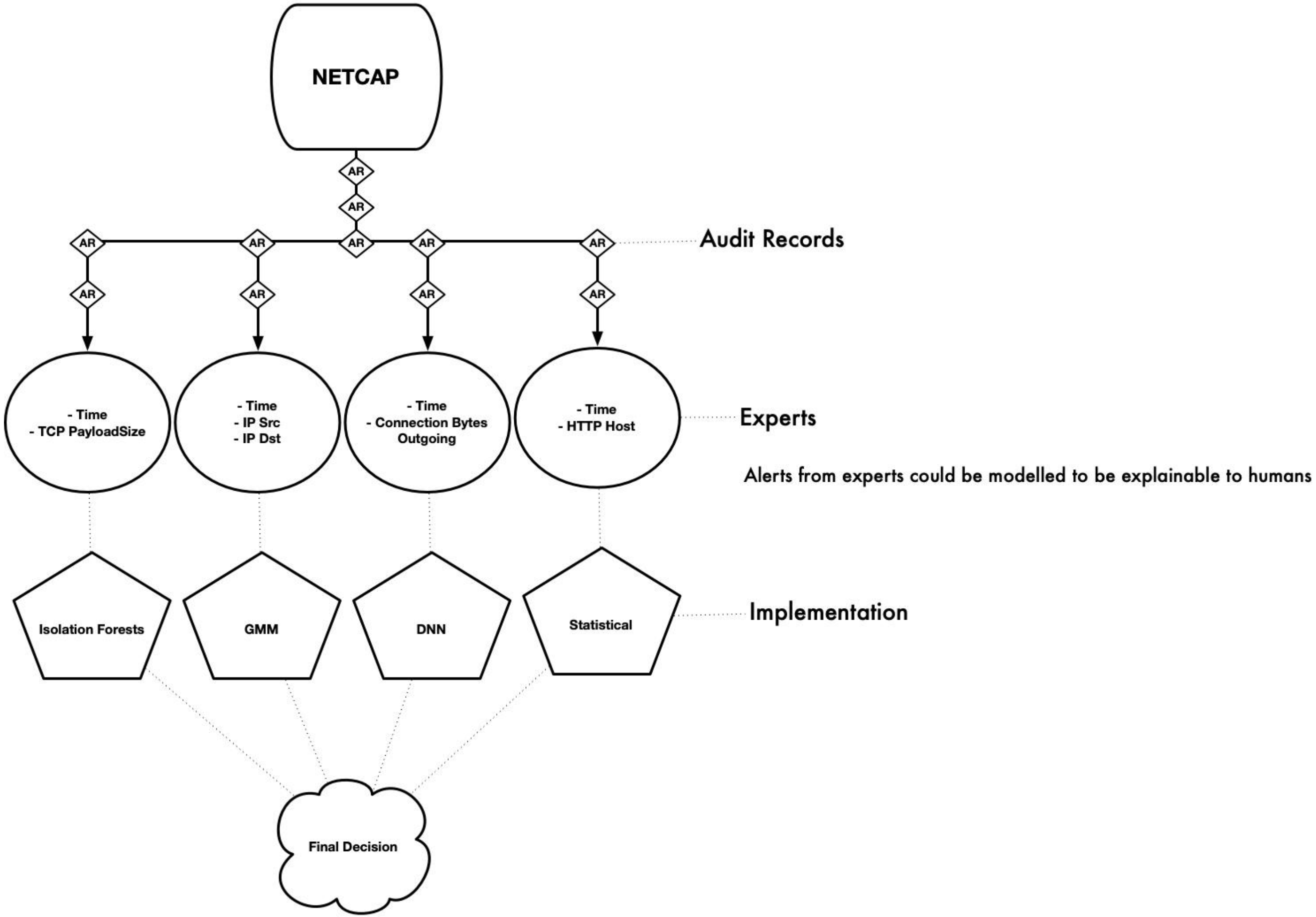
Analyzer Configuration

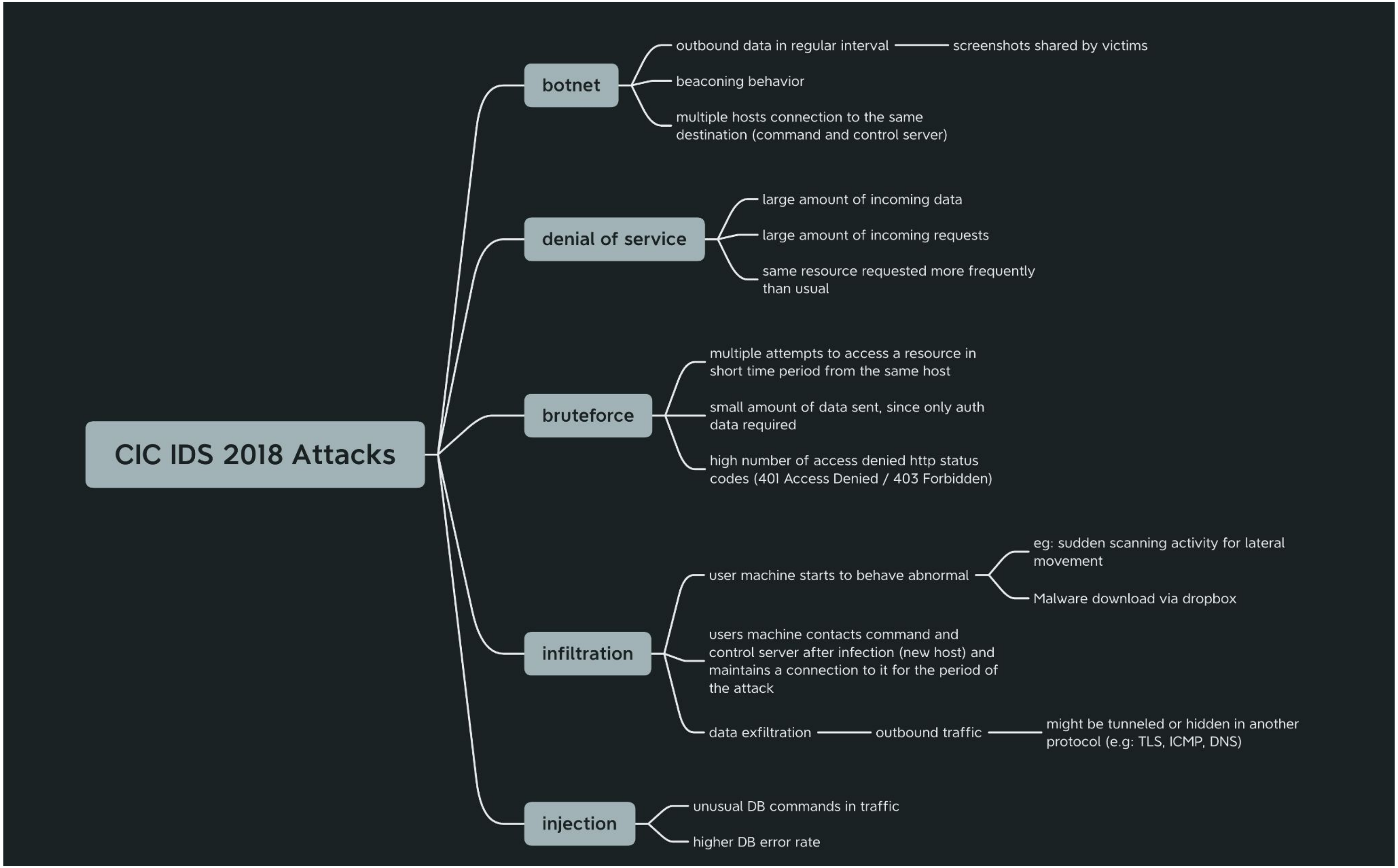


UNIX socket processing



Expert Model




CIC IDS 2018 Attacks
botnet
outbound data in regular interval
screenshots shared by victims
beaconing behavior
multiple hosts connection to the same destination (command and control server)
denial of service
large amount of incoming data
large amount of incoming requests
same resource requested more frequently than usual
bruteforce
multiple attempts to access a resource in short time period from the same host
small amount of data sent, since only auth data required
high number of access denied http status codes (401 Access Denied / 403 Forbidden)
infiltration
user machine starts to behave abnormal
eg: sudden scanning activity for lateral movement
Malware download via dropbox
users machine contacts command and control server after infection (new host) and maintains a connection to it for the period of the attack
data exfiltration
outbound traffic
might be tunneled or hidden in another protocol (e.g: TLS, ICMP, DNS)
injection
unusual DB commands in traffic
higher DB error rate