Discriminating reflective DDoS attack tools at the reflector

Fons Mijnen

fons.mijnen@os3.nl

Max Grim

max.grim@os3.nl

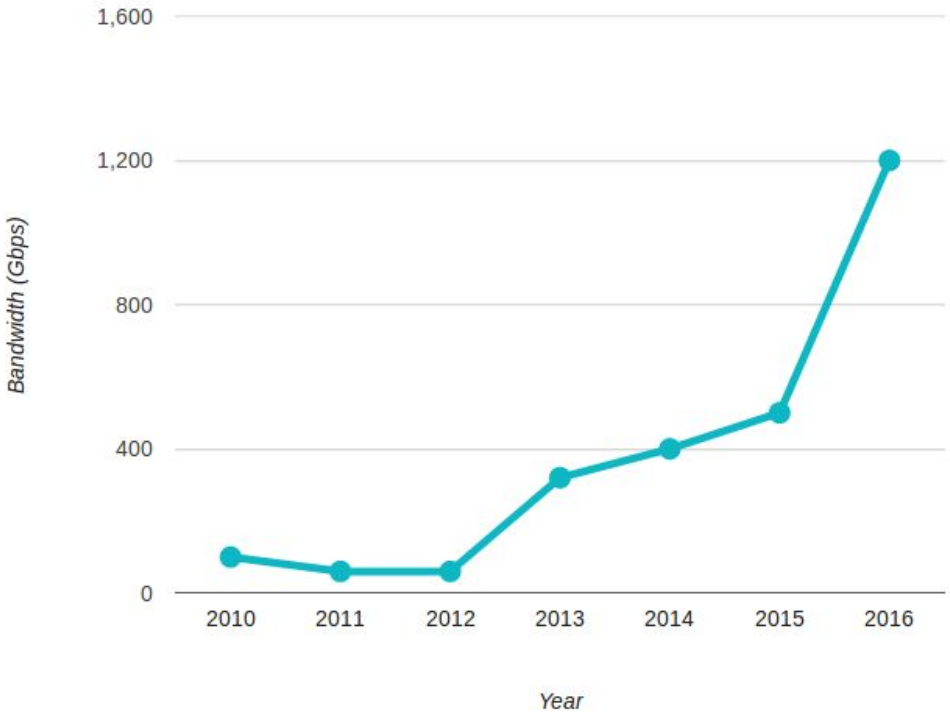
DDoS attacks

DDoS attacks are a problem internet users have faced for many years, and is still relevant today.

DDoS attacks

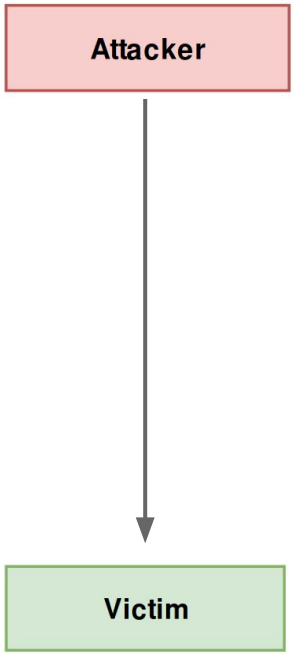
DDoS attacks are a problem internet users have faced for many years, and is still relevant today.

IoT and **booter services** have increased the bandwidth of DDoS attacks



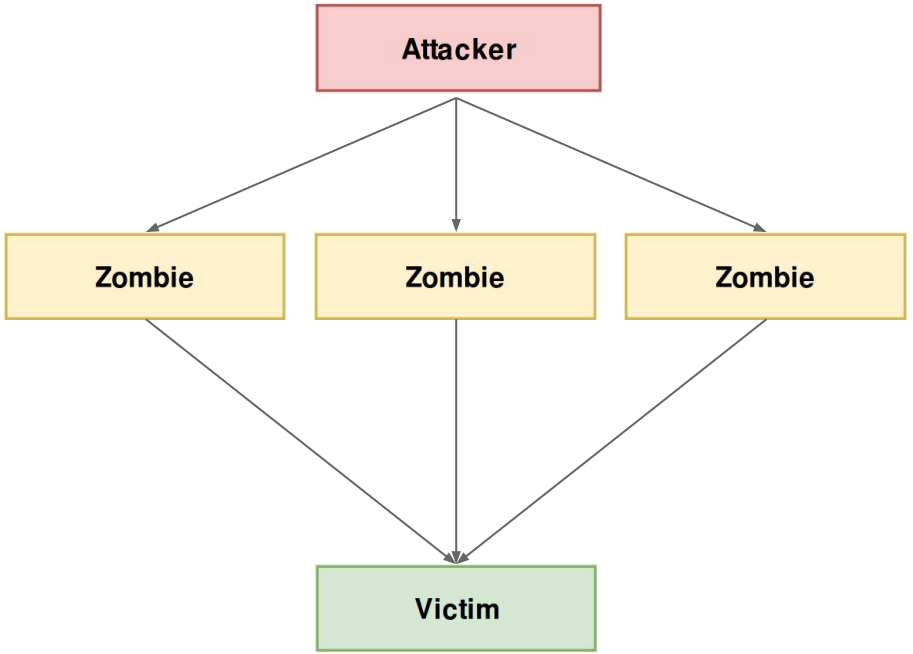
DoS

- One attacker
- One DoS machine
- Bandwidth depletion



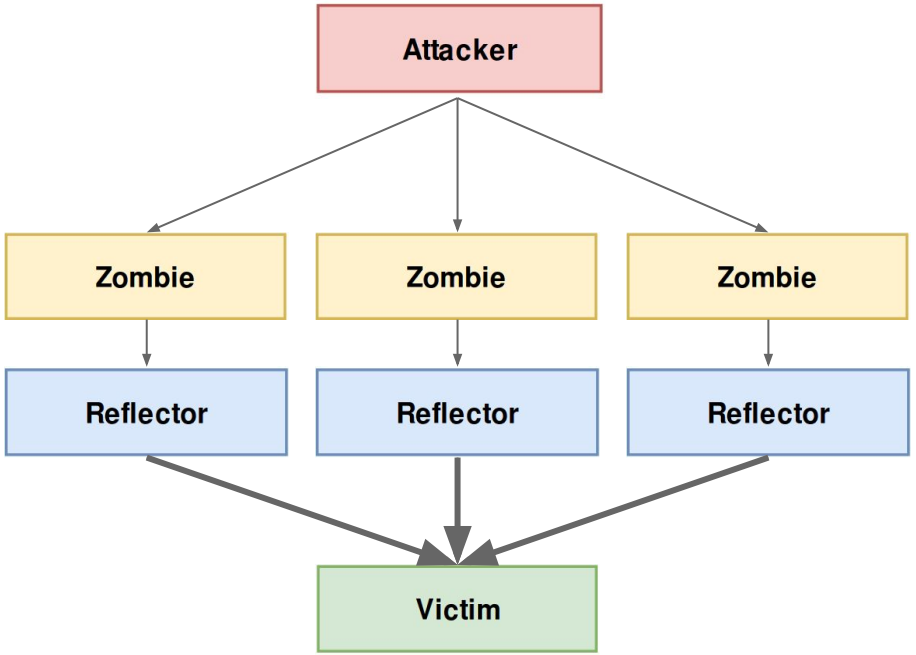
DDoS

- One attacker
- Multiple DoS machines (zombies)
- Often includes a CnC machine



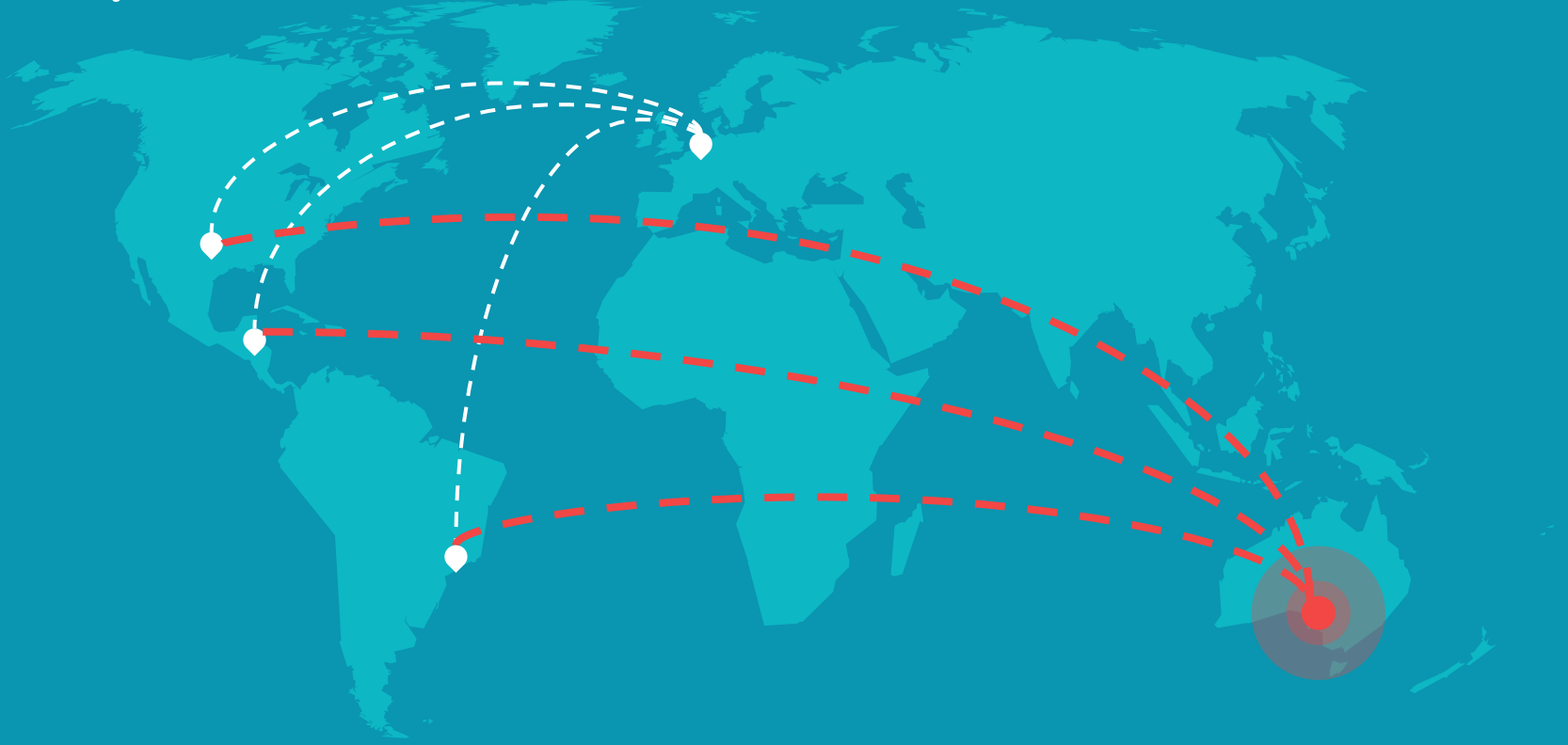
Reflective DDoS

- One attacker
- Multiple DoS machines (zombies)
- Often includes a CnC machine
- One or more reflectors
- Can **amplify** the output



7

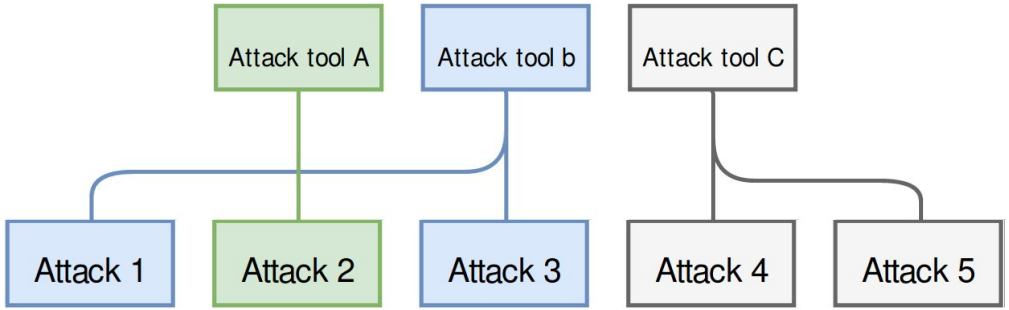
Amplified Reflective DDoS attack



The question

Can we discriminate attack tools used in RDDoS attacks **at the reflector**

- Analyse network traffic
- Extract features
- Perform machine learning



Research question

Can RDDoS tools be identified by looking at the network traffic send to a reflector?

- Do RDDoS attacks leave distinctive traces?
- Can a fingerprint be build using these traces?
- Can RDDoS attacks be correlated to the same attacker?
- Is it possible to identify the tool used in a RDDoS attack?
- Can machine learning be utilised to automate the identification process?

Methodology

Automating attack and collecting data

Data

Fox-IT data

- Unlabeled
- Collected from honeypots
- Unknown number of attack scripts
- Unsupervised learning

Lab generated data

- Labeled
- Collected from own server
- Known number of attack scripts
- Supervised learning

DNS DDoS scripts

Flooder

Pastebin.com, written in C, multi-threaded, random UDP source port

Saddam

GitHub.com, written in Python, multi-threaded, random UDP source port

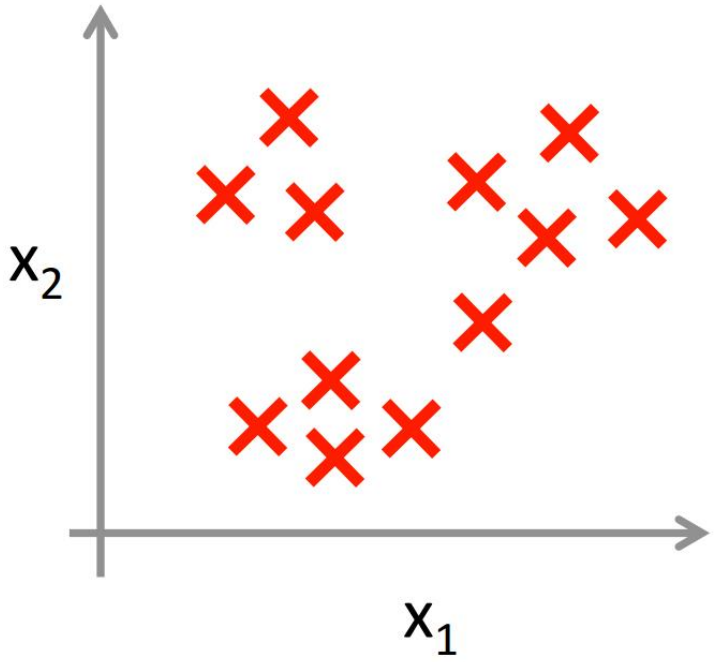
Ethan

GitHub.com, written in C, single-threaded, fixed UDP source port

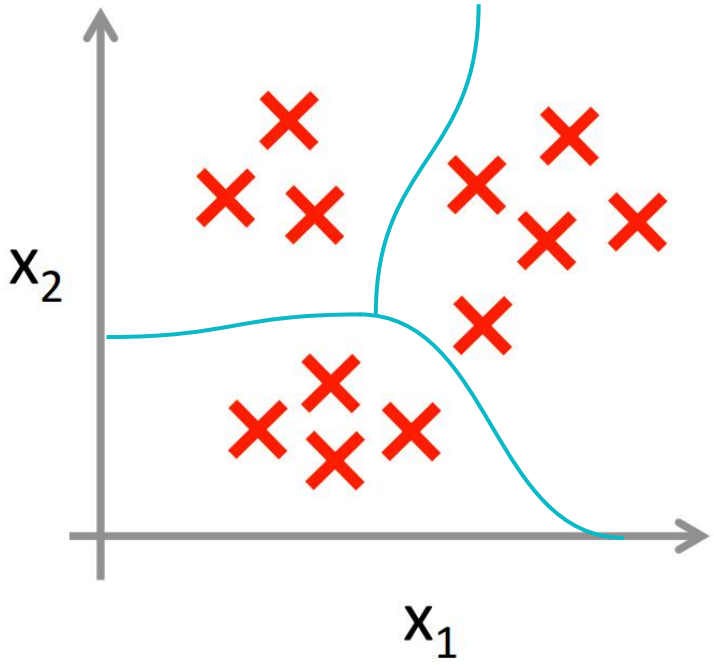
Tsunami

Infosec-Ninjas, written in C, single-threaded, fixed UDP source port

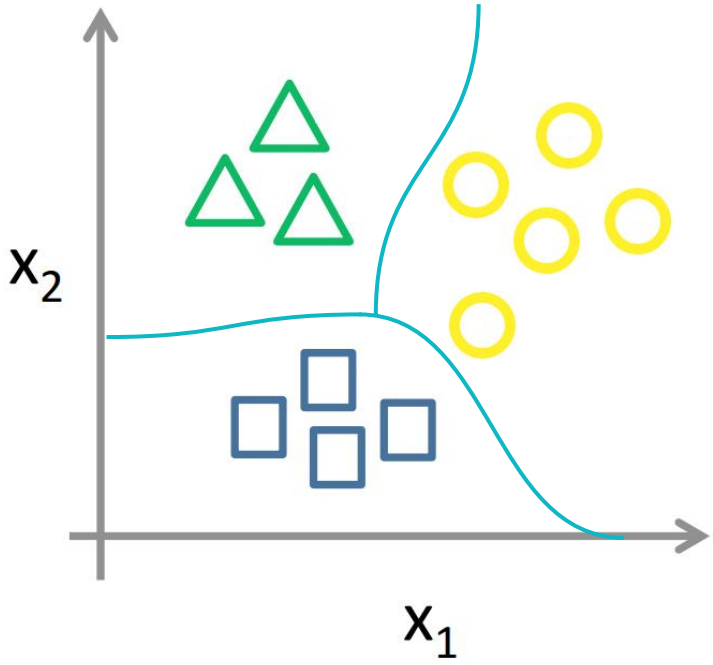
Multiclass classification



Multiclass classification

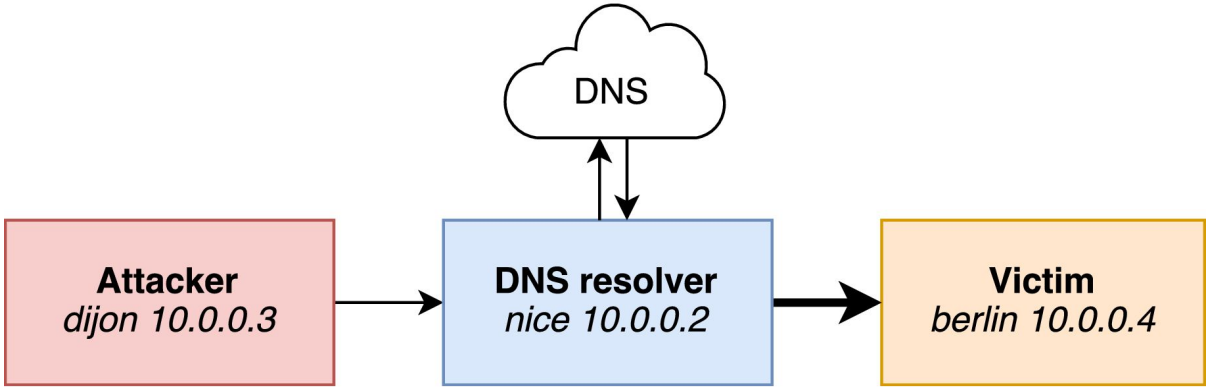


Multiclass classification

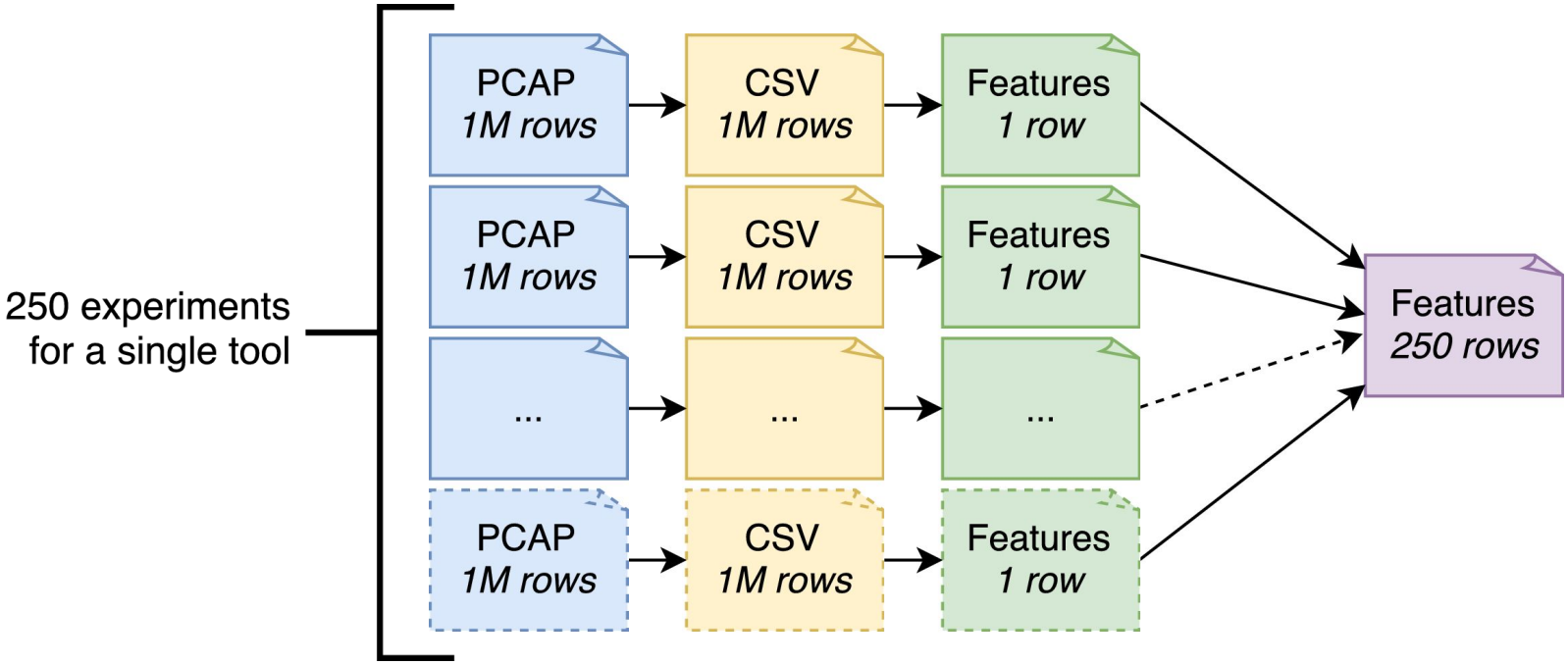


Data collection

- Fully automated attacks
- PCAP's collected at the resolver

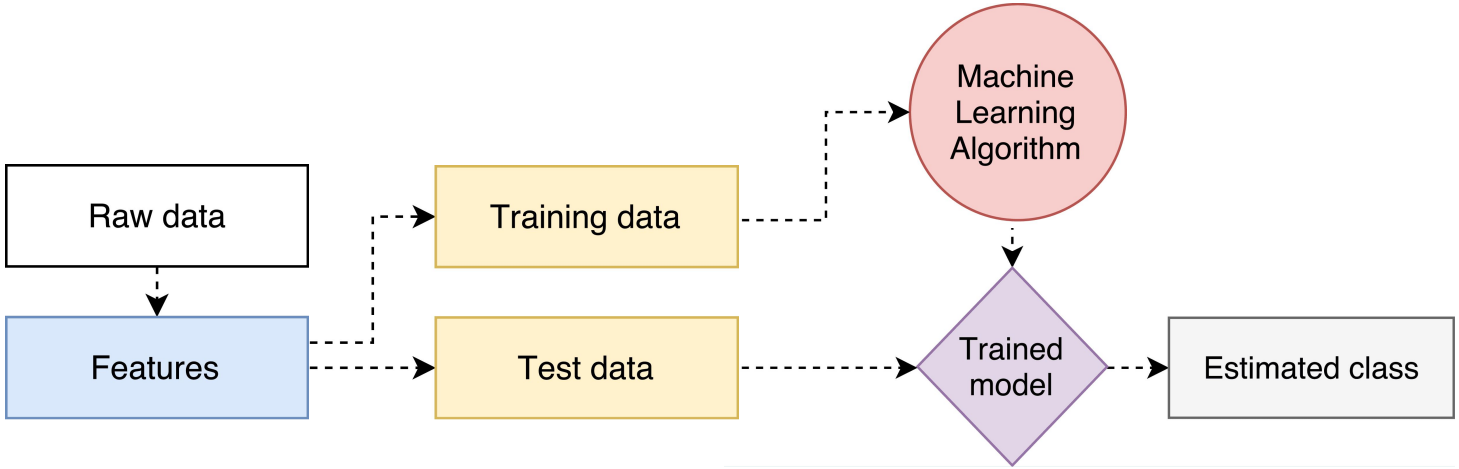


Data collection cont'd



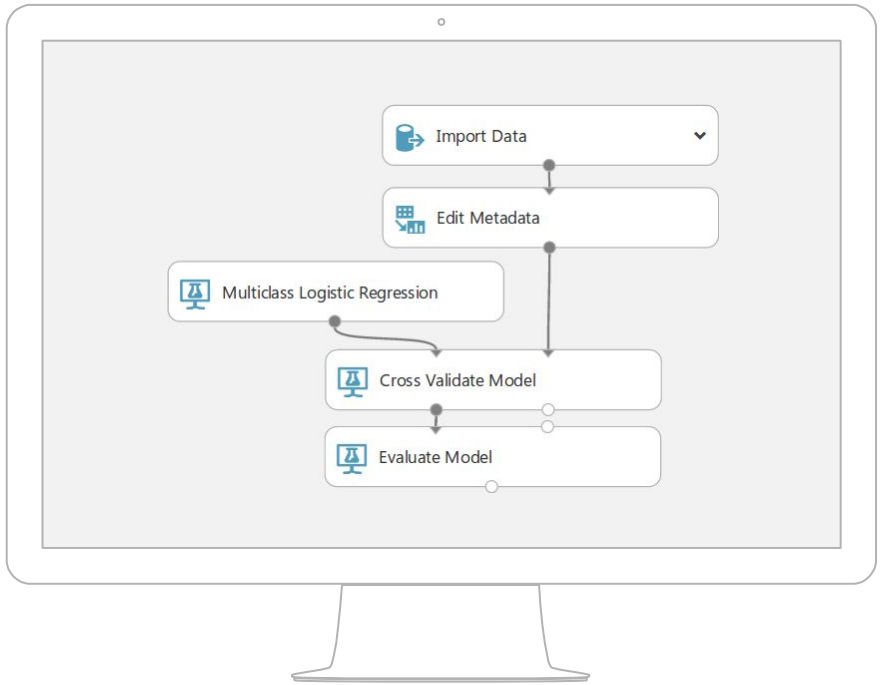
Machine learning

- Randomly split into 90% train- and 10% test data
- 10-fold cross validation



Azure Machine Learning

- SaaS
- Fast prototyping
- Visualisations
- Data import from HTTP server



Results 1/2

Fox-IT data

Fox-IT dataset 1

25 packets per PCAP

Observations:

- All packets almost **identical**
- DNS request in particular **identical** only changing the hostname
- Some field frequently change:
  - DNS ID
  - IP ID
  - UDP Source Port
- Also the IP Total length and header checksum change

Fox-IT dataset 1 (cont'd)

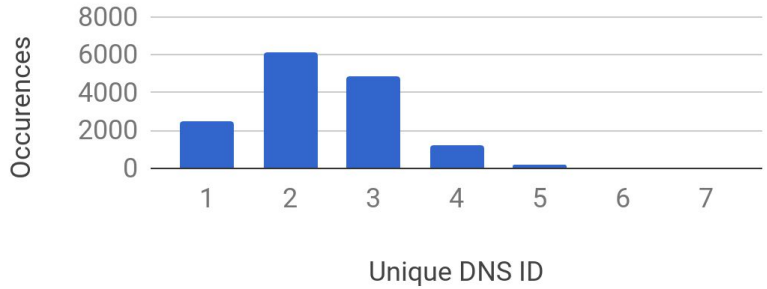
Ignoring the frequently changing data types we find 1 difference:

IP DS Field set to **0x40**

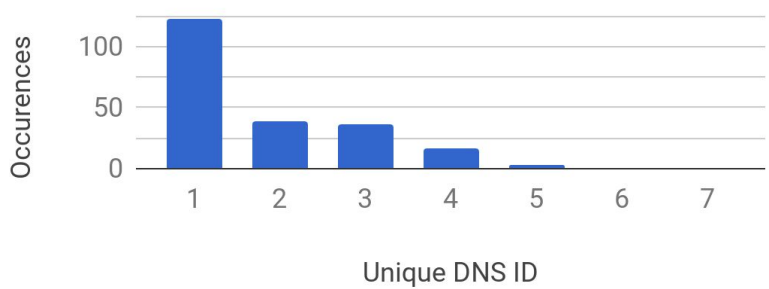
No other differences means we need to recognize patterns

Capatalised domains VS non capatalised

All data

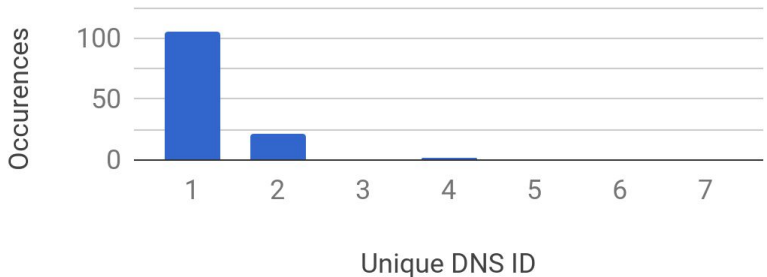


Atleast one packet with a DS Field set to 0x40

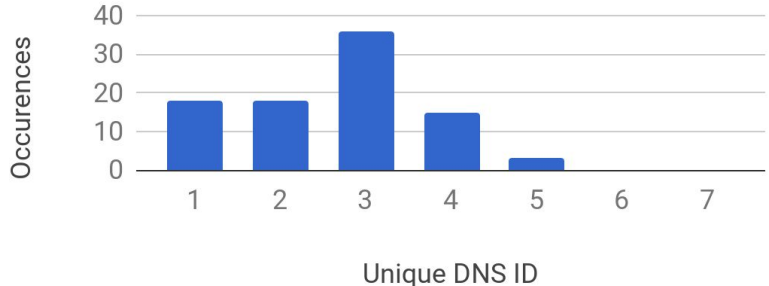


4 domains found: 'ARCTIC.GOV', 'NRC.GOV', 'hoffmeister.be', 'leth.cc'

Only capatalized domains



Non capatalized domains



Fox-IT dataset 1

Conclusion: Confident we found at least 2 different tools

Need more packets / PCAP to perform pattern analysis

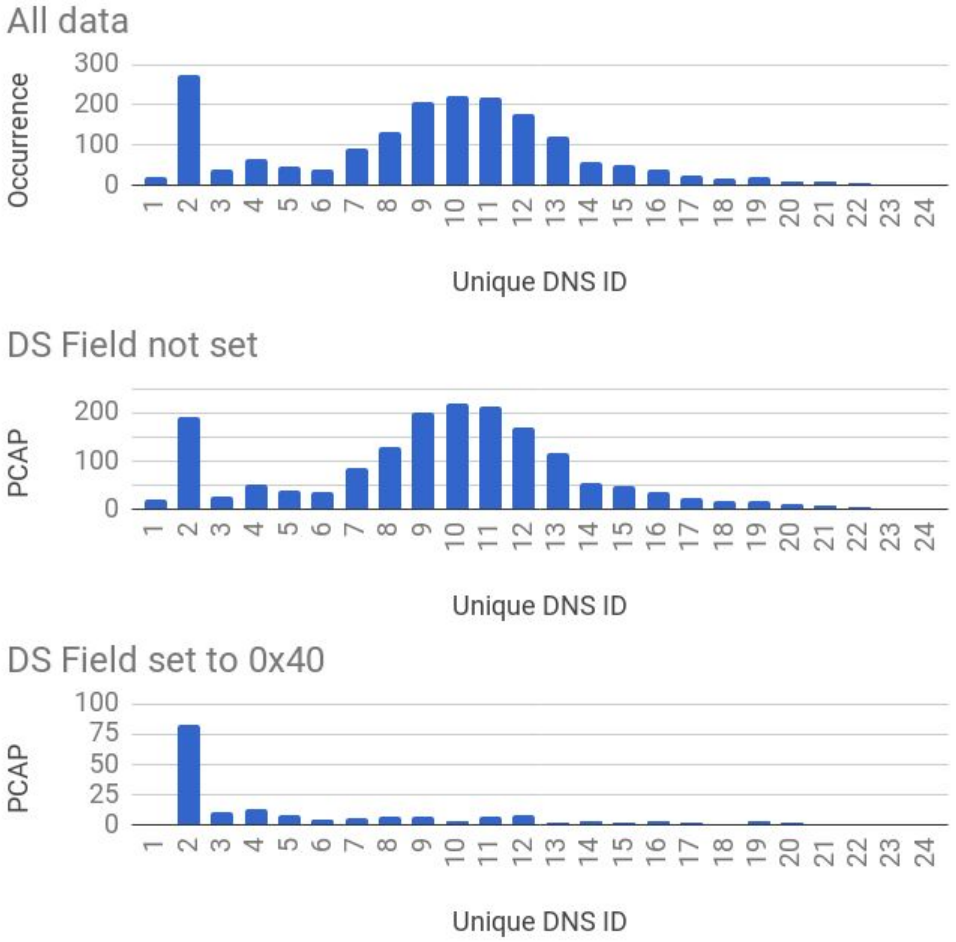
Fox-IT dataset 2

Contains 250 packets per PCAP

1868 PCAPs

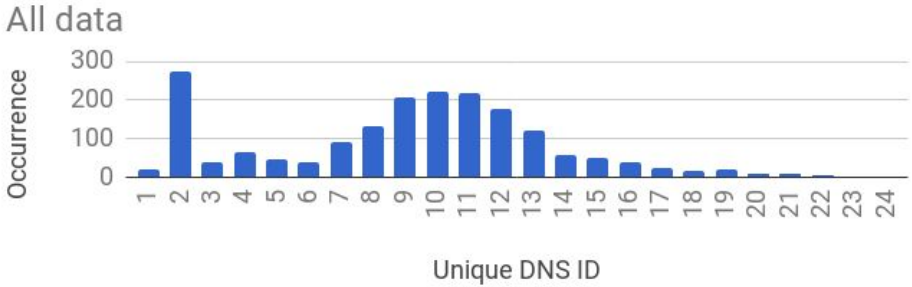
Dataset 2: DS-Field

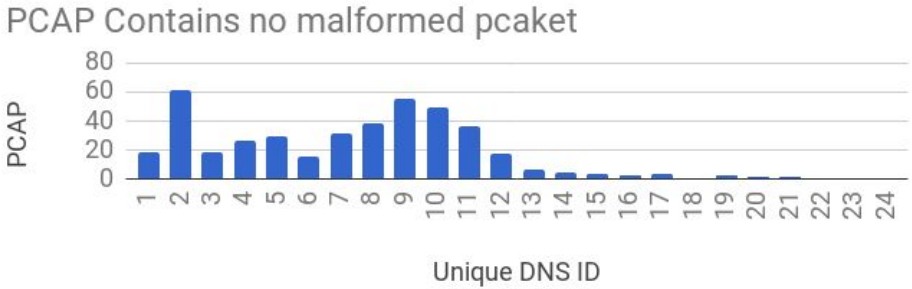
PCAPs with at least one packet with a DS field set to 0x40 change DNS ID very little on average

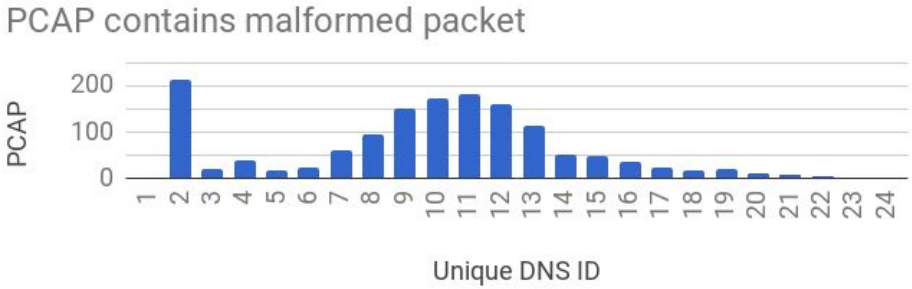


Dataset 2: Malformed packets

PCAPs containing 1 DNS ID never have malformed packets or have their DS field set







There is more

- Large group of PCAPs have not had their DS field set but have a significantly different DNS ID counts
- Some packets change the DNS ID, IP ID, and UDP sourceport together, some do not
- 3 PCAPs found with static DNS ID, IP ID and UDP sourceport

How many tools did we find?

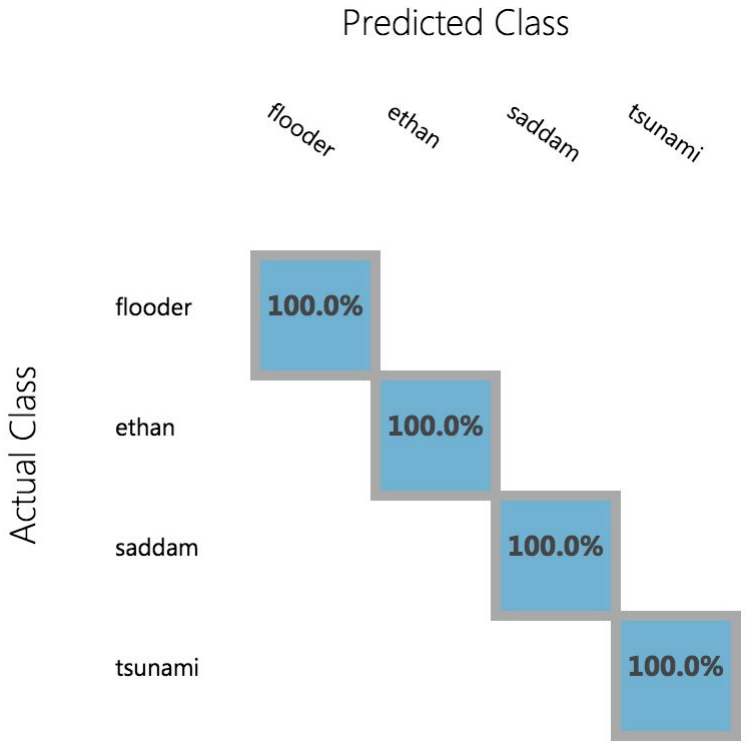
- Tool A: ~2 Unique DNS id's / 250 packets and DS Field set to 0x40
- Tool B: Static DNS ID, UDP source port and IP ID
- Tool C: ~1 Unique DNS ID with changing UDP source port and IP ID, no DS Field / malformed packets
- Tool D: ~10-13 unique DNS ID's / 250 packets and no DS field set

Results 2/2

Lab generated data

Accuracy results

# captures	Multiclass Neural Network accuracy	Multiclass Logistic Regression accuracy
1.000.000	**100%**	**100%**
10.000	**100%**	**100%**
1.000	**100%**	**100%**



Training with fewer features

- Trained with 71 features
- Can we work with less?

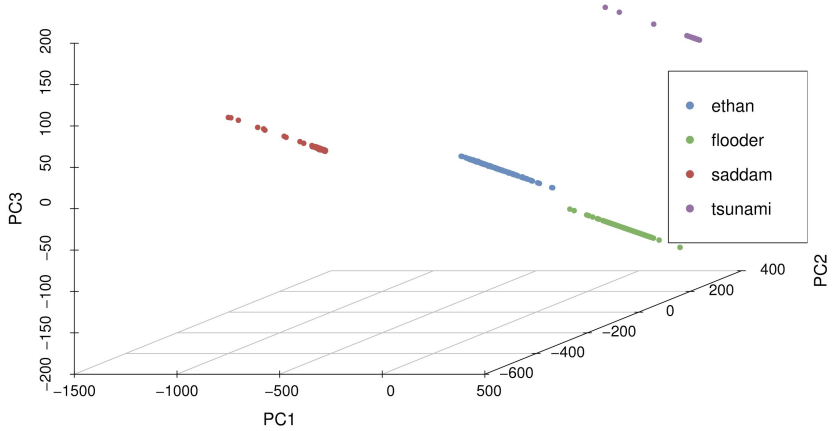
MLR: Feature weighting

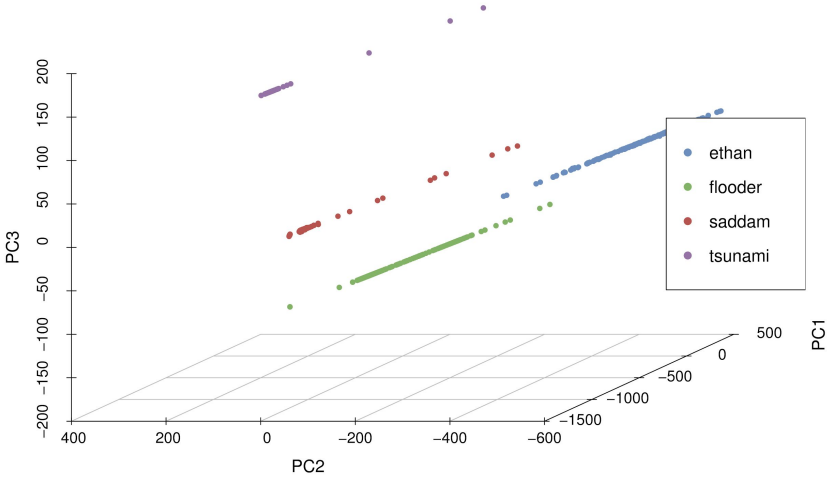
	flooder	ethan	saddam	tsunami
dns.qry.class_unique	**0.622728**	**2.57913**	**-1.90491**	**-1.29728**
dns.id_unique_len	**-0.79392**	**0**	**1.90643**	**0**
dns.qry.type_unique	**-0.761273**	**0**	**1.87811**	**0**
ip.dsfield.dscp_unique	**-0.122946**	**0**	**0**	**1.79175**
udp.srcport_unique_len	**-0.117052**	**0**	**1.53162**	**0**
ip.id_longest_cons	**-1.4457**	**0**	**0.421945**	**0.0336367**
udp.checksum_used	**0**	**1.07789**	**0**	**-0.249253**
...	**...**	**...**	**...**	**...**
dns.flags.z_unique	**0**	**0**	**0**	**0**

Training with fewer features

- Leaves **21 features**
- Still 100% accuracy

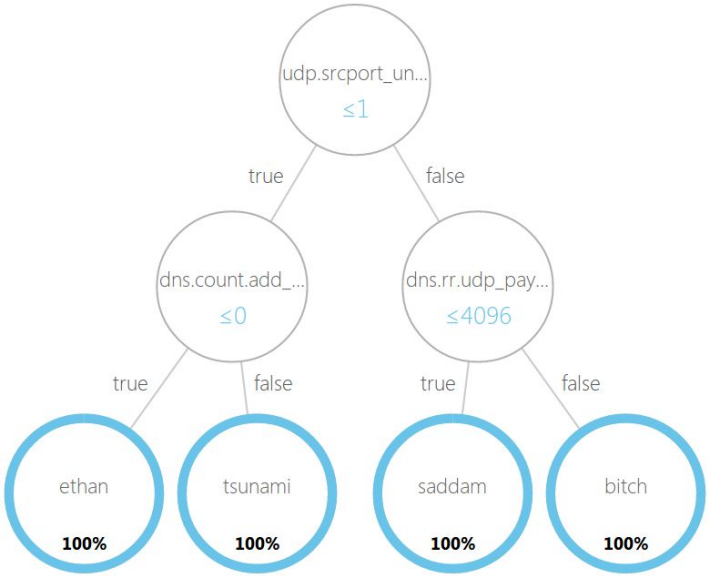
Principal Component Analysis





Multiclass Decision Jungle

- Builds multiple trees
- Downside: probability score always 100%



One tree is enough for 100% accuracy

Decision tree code

```
import os, csv

def classify_tree(o):
    a = int(o['dns.count.add_rr_min']) <= 0
    b = int(o['dns.rr.udp_payload_size_min']) <= 4096

    if int(o['udp.srcport_unique_len']) <= 1:
        return 'ethan' if a else 'tsunami'
    return 'saddam' if b else 'flooder'

all_files = filter(lambda x: x.endswith('csv_feature'), os.listdir('.'))
for filename in all_files:
    data = list(csv.DictReader(open(filename, 'r')))[0]
    print(data['label'] == classify_tree(data))
```

Conclusion

Conclusion

Do RDDoS attacks leave distinctive traces?

Likely, though not necessarily true

- In practice, tools appear to be very similar
  - Individual packets are practically identical
  - Groups of packets show distinctive patterns
- Doable to create a 100% similar behaving tool

- Real possibility that **one** attacker uses **multiple** tools

Conclusion (cont'd)

Can machine learning be utilised to automate the identification process?

- In **practice**, clustering algorithms successfully used to identify different clusters of attacks
  - Recognitions may be incomplete
  - May be used to detect presence of new attacks
- In a **lab environment**, supervised learning looks promising
  - May be tools out there that show identical behaviour
  - Needs trained dataset in order to work

Future work

Training more tools

Add more attack scripts to the dataset

Other protocols

Test if it's possible to discriminate attacks on other protocols:

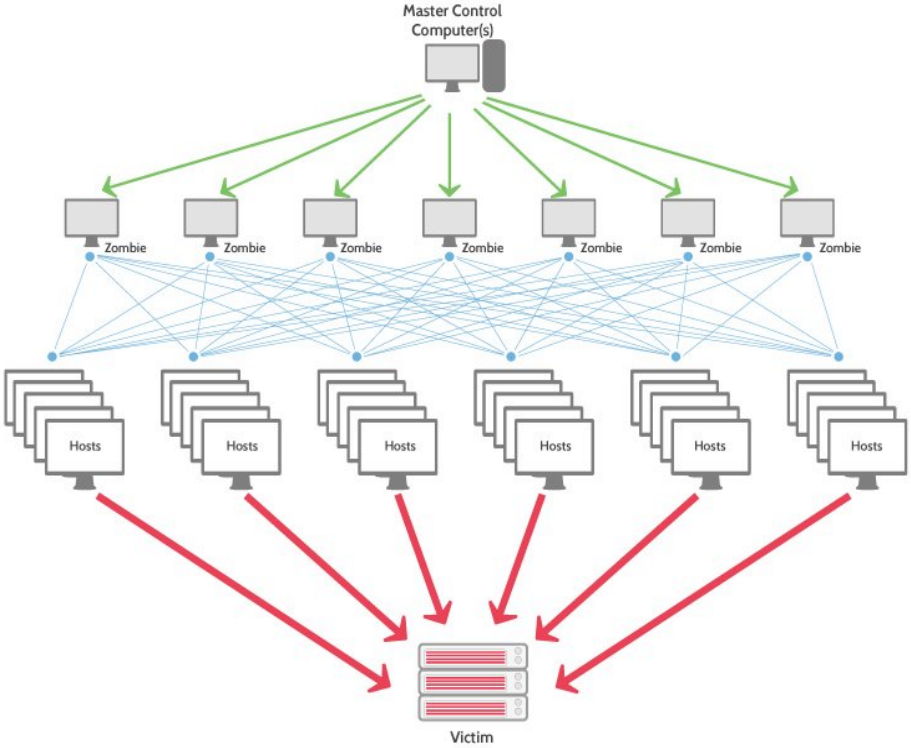
- NTP
- SNMP
- SSDP
- CharGen
- etc.

Combining victim side data

Can captures at the victim side help to identify more attacks?

Special thanks

Lennart Haagsma from Fox-IT



Thank you

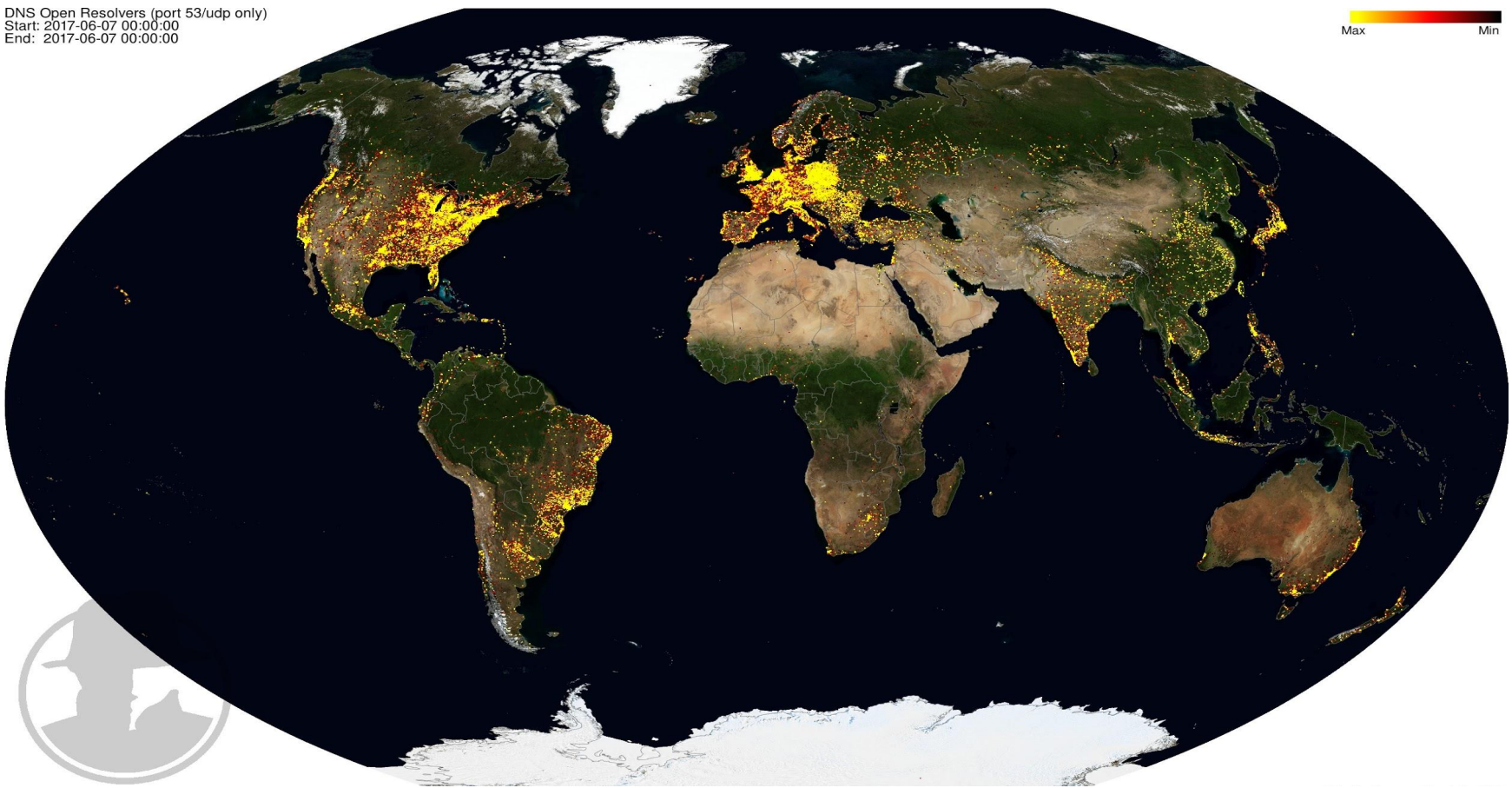
Any questions?

For more details, drop by or:

- fons.mijnen@os3.nl
- max.grim@os3.nl

This template is free to use under Creative Commons Attribution license and provided by SlidesCarnival.

Extra



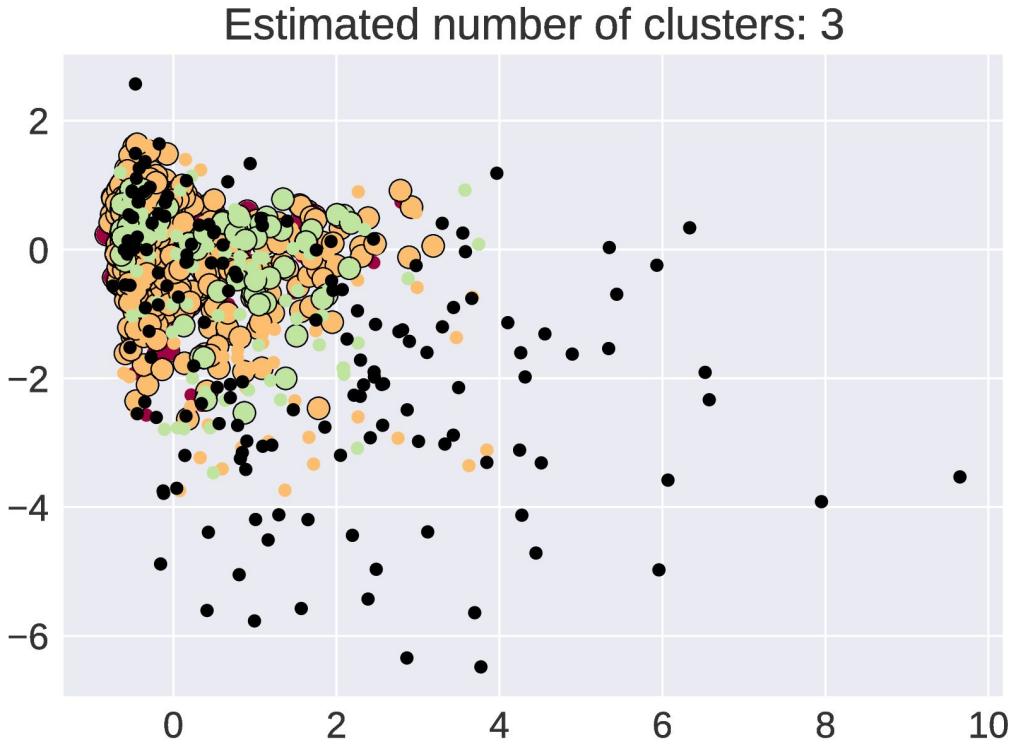
4,131,684

Distinct IP addresses appear to be openly recursive

- The Shadowserver Foundation

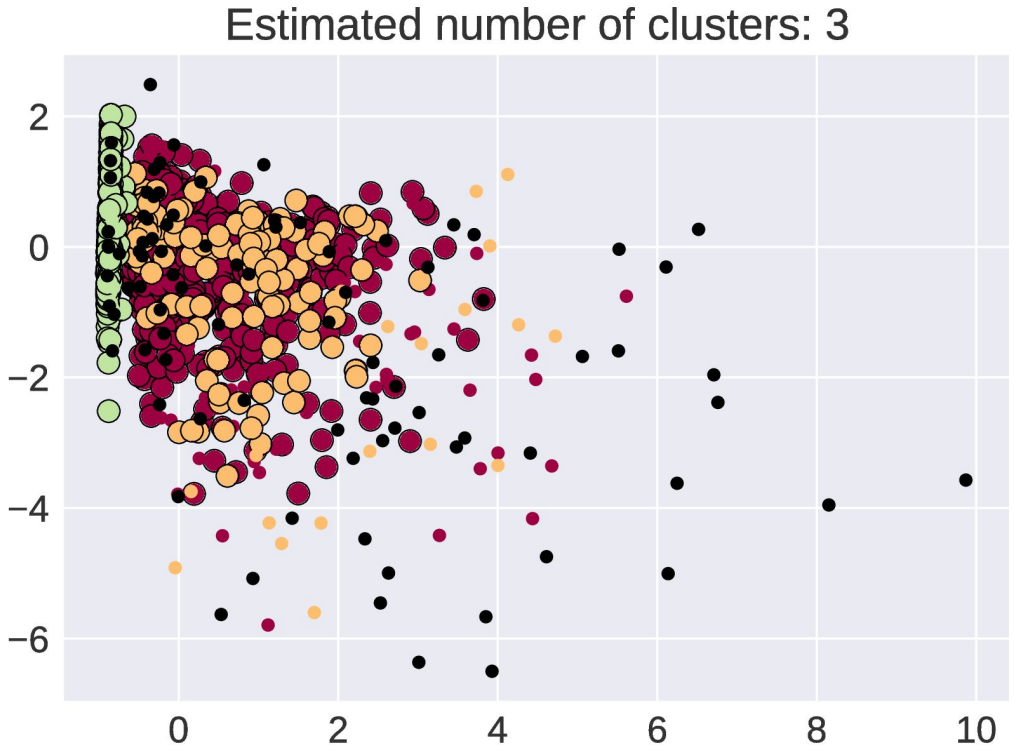
DBSCAN cluster of Fox-IT dataset 2

- By setting a high **ε** we can create clusters



DBSCAN cluster of Fox-IT dataset 2

- By setting a high **ε** we can create clusters

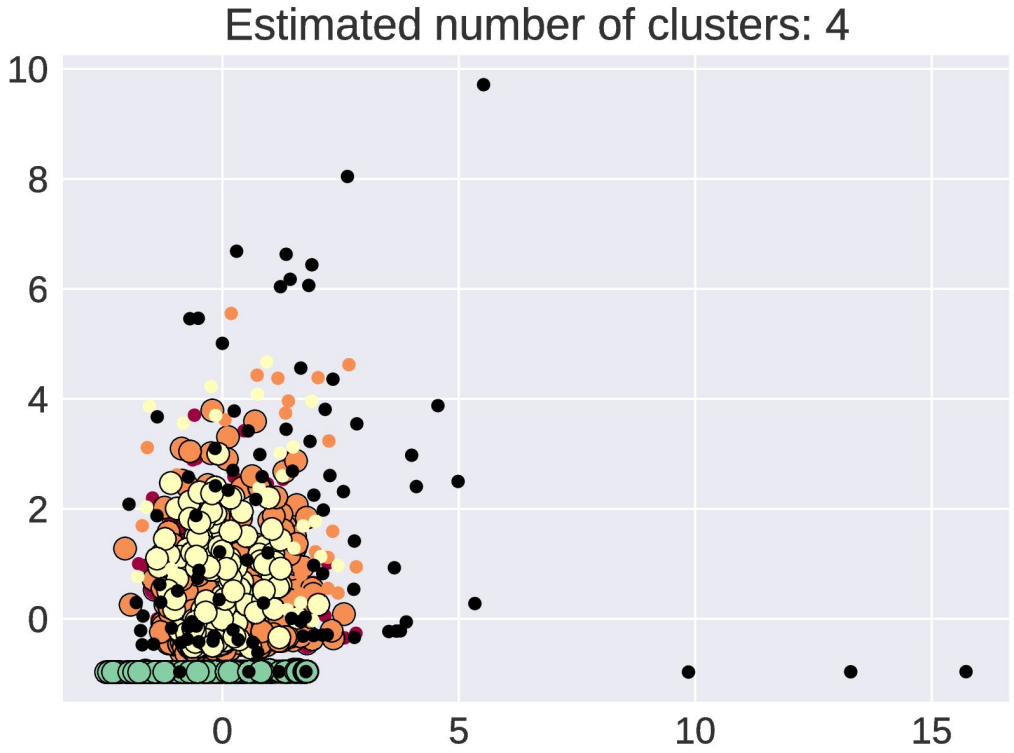


Adding **flooder**

DBSCAN cluster of Fox-IT dataset 2

- By setting a high **ε** we can create clusters

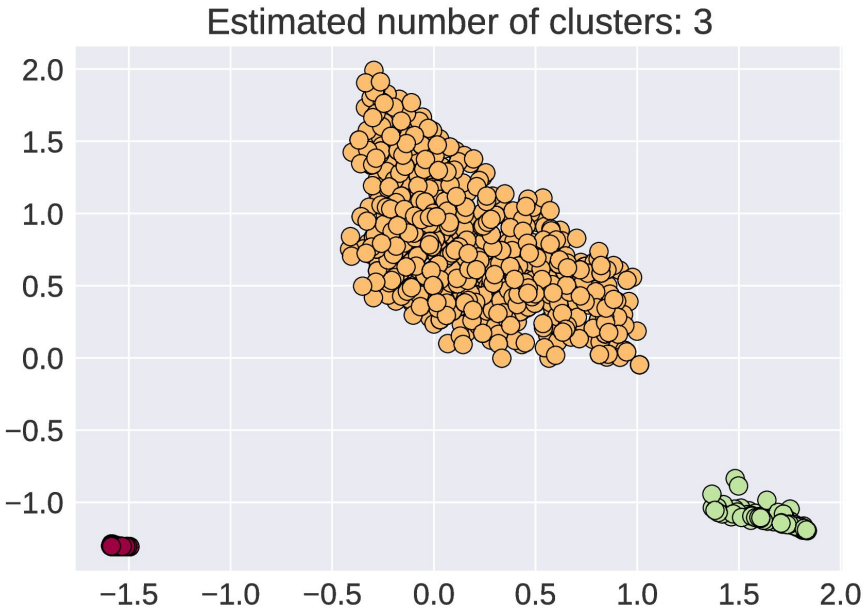
Adding **sadam**



DBSCAN cluster of Fox-IT dataset 2

Clustered based on:

- dns.id_longest_repeat
- dns.id_unique_len
- dns.rr.udp_payload_size_min
- ip.id_longest_repeat
- ip.id_unique_len
- ip.dsfield_unique_len
- udp.srcport_longest_repeat
- udp.srcport_unique_len

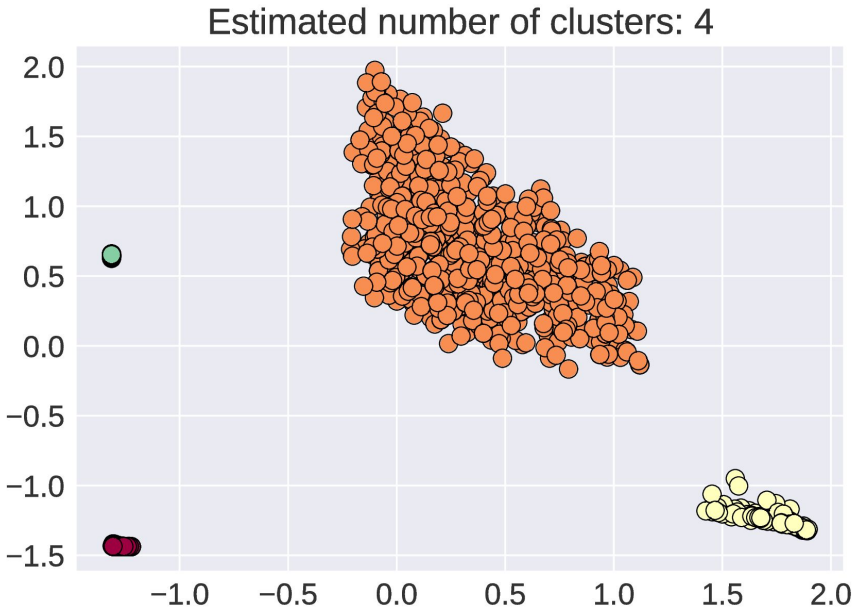


DBSCAN cluster of self generated dataset

- 4 clusters for 4 tools

DBSCAN cluster of merged dataset with sadam

- Shows new cluster for new attack tool



DBSCAN cluster of merged dataset with dns flooder

- Does not show new cluster