Go with the (net)flow

A tale of fixing ML-based network flow analysis

$ who

ERMES PENNUCCI

- Malware Analyst @ Tinexta Defence
- PhD student @ University of Sannio

ANTONIO REPOLA

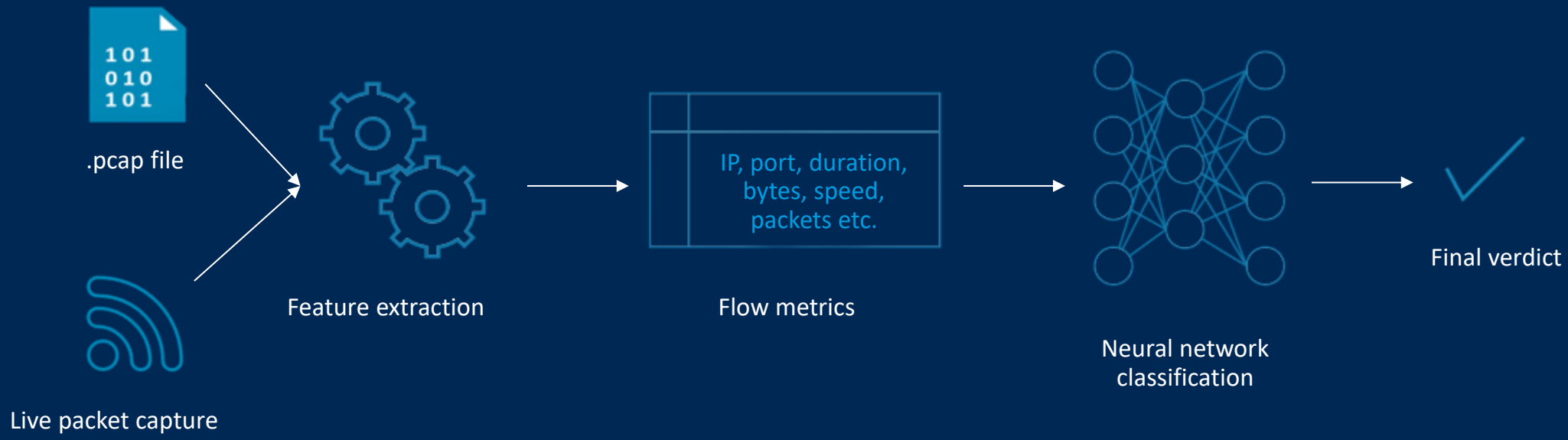
- Data Scientist @ Tinexta Defence

Agenda

- Network IDS with machine learning
- Popular tools and approaches
- Discovering bugs in existing tools
- NetFlowMeter
- Data exploration with decision trees
- Autoencoders for anomaly detection

Why Network IDS with Machine Learning?

- Use pattern recognition capabilities of ML to identify anomalies in traffic and possibly unknown attacks
- Actively researched field



Data collection

- Complete network capture is often infeasible
  - Most data is also encrypted
- Look at the big picture: aggregate packets into flows

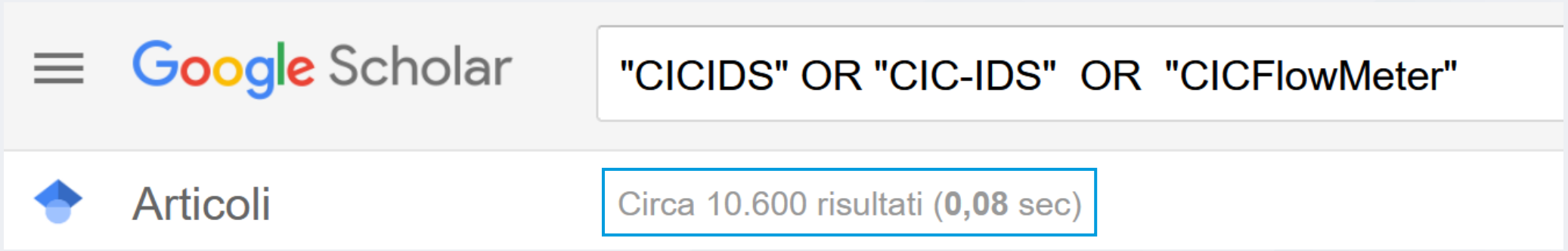
- There are various well-known flow analyzers
  - **CICFlowMeter**: popular in academia, network and time features
  - **OpenArgus**, **Zeek**: popular in industry, application-level features

A brief history of CICFlowMeter

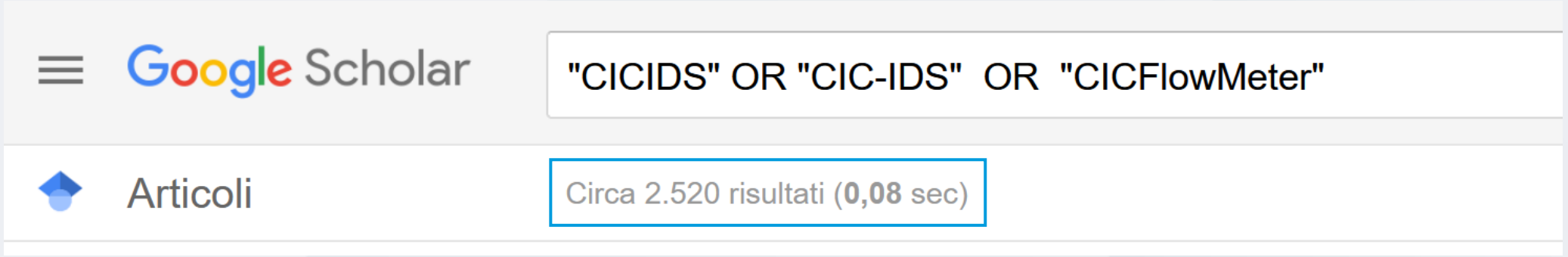
- Most known for the CICIDS2017 dataset
  - **Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization**, Sharafaldin et al., 2018. 10.5220/0006639801080116
  - 5000+ citations (Google Scholar)
- Analyzed and corrected by an independent group of researchers
  - **Error Prevalence in NIDS datasets: A Case Study on CIC-IDS-2017 and CSE-CIC-IDS-2018**, Liu et al., 2022. 10.1109/CNS56114.2022.9947235
  - 100+ citations (Google Scholar)
- Last modified on Apr 27, 2023 to fix residual issues with labeling

A brief history of CICFlowMeter

Any time



Since 2025



CICFlowMeter and us

- AI Team reported performances issues when processing moderately sized PCAPs

```
CICFlowMeter found :1 pcap files
==> 1 / 1
Working on... capture_4.pcap
Exception in thread "main" java.lang.OutOfMemoryError: GC overhead limit exceeded
        at java.util.Arrays.copyOfRange(Arrays.java:3664)
        at java.lang.String.<init>(String.java:207)
        at java.lang.StringBuilder.toString(StringBuilder.java:412)
        at cic.cs.unb.ca.jnetpcap.BasicPacketInfo.bwdFlowId(BasicPacketInfo.java:83)
```

- Code analysis revealed a codebase that's hard to maintain
- We decided to work on a complete rewrite instead

Introducing NetFlowMeter

- Complete rewrite of the project in C#, MIT licensed

  https://github.com/DefenceTechSecurity/NetFlowMeter

- Cleaned up logic and documented most behaviors
- **Critical requirement**: fully compatible on our datasets

Main improvement: high performances

- **Dramatically faster**, especially in worst-case datasets

Reference file size	Processing time CICFlowMeter	Processing time NetFlowMeter	Speedup
400MB	2m 1s	9s	13x (1344%)
500MB	5m 40s	13s	26x (2615%)

- Still more room for improvement
- Unlocks new scenarios such as real-time usage

Scaling memory usage

Reference file size	RAM CICFlowMeter	RAM NetFlowMeter	Efficiency
400MB	3GB	550MB	1/5
500MB	4GB	1GB	1/4

- Memory usage is still high
  - Legacy needed to achieve perfect compatibility
  - Flows never removed from memory due to cumulative metrics
- Optional feature: **break compatibility** to reduce memory usage to just 100MB
  - Scales well for real time processing

Development roadmap

- Version 1.0 - Out now
  - Baseline with CICFlowMeter compatibility
- Version 2.0 - In development
  - Tuned specifically to our use cases
  - Compatibility will be dropped for the sake of fixing bugs
  - Most of the improvements already documented as comments in the code

Flow cutting

- CICFlowMeter splits flows at the 2 minutes mark
  - This affects most metrics since they are continuously reset
  - The reason is unclear, we couldn't find any documentation
- We are experimenting with removing this feature
  - Early results are promising

Improved correctness

- When first packet of a flow has the RST flag, the flow is not terminated
- Flow cutting can in some cases split right before an RST packet
- This new flow stays pending in memory, wrongly carrying cumulative metrics

```
else if (packet.Tcp.HasHeaderFlag(TcpFlags.RST))
{
    // TODO: CICFlowMeter bug, if the first packet happens to be a RST packet the flow state is not terminated
    // This can happen when a flow is split right before a RST packet by the timeout
    // When this happens the next flow will wrongly carry the previous flow's cumulative duration
    if (TotalPackets != 1)
        TcpFlowState = FlowState.TcpTerminated;
```

Improved correctness

- Certain metrics can only be initialized by the first packet
- When this happens, only the direction of this packet is initialized

```
178     public void firstPacket(BasicPacketInfo packet) {
199
200         if (Arrays.equals(this.src, packet.getSrc())) {
201             this.min_seg_size_forward = packet.getHeaderBytes();
202             ...
203         } else {
204             this.min_seg_size_backward = packet.getHeaderBytes();
```

- On new packets they're updated with a `min(current, next)` logic which causes the other direction to always be 0

CICIDS2017 – Rationale

- Public network traffic dataset for evaluating ML-based IDSs
- Widely used as a benchmark
- Both PCAPs and labeled CSV files provided
- Case study for data quality analysis

CICIDS2017 – Overview

- Captured over 5 days in July 2017
- Benign traffic: abstract behavior of 25 users based on common protocols
- Tuesday to Friday: benign + specific attack scenarios

Day	Morning activities	Afternoon activities
Monday	Benign traffic only	
Tuesday	FTP brute force (Patator)	SSH brute force (Patator)
Wednesday	Slow and flood DoS	Heartbleed
Thursday	Web attacks (brute force, XSS, SQL injection)	Infiltration (Dropbox download, Cool disk Mac, port scan)
Friday	Botnet ARES	Port scanning, DDoS LOIC

Labeling (2023)

- Very **imbalanced** dataset

```
BENIGN                       1582566
Portscan                      159066
DoS Hulk                      158468
DDoS                           95144
Infiltration - Portscan        71767
DoS GoldenEye                   7567
Botnet - Attempted              4067
 ...
Infiltration                      36
Web Attack - XSS                  18
Web Attack - SQL Injection        13
Heartbleed                        11
```

Relabeling and cleaning

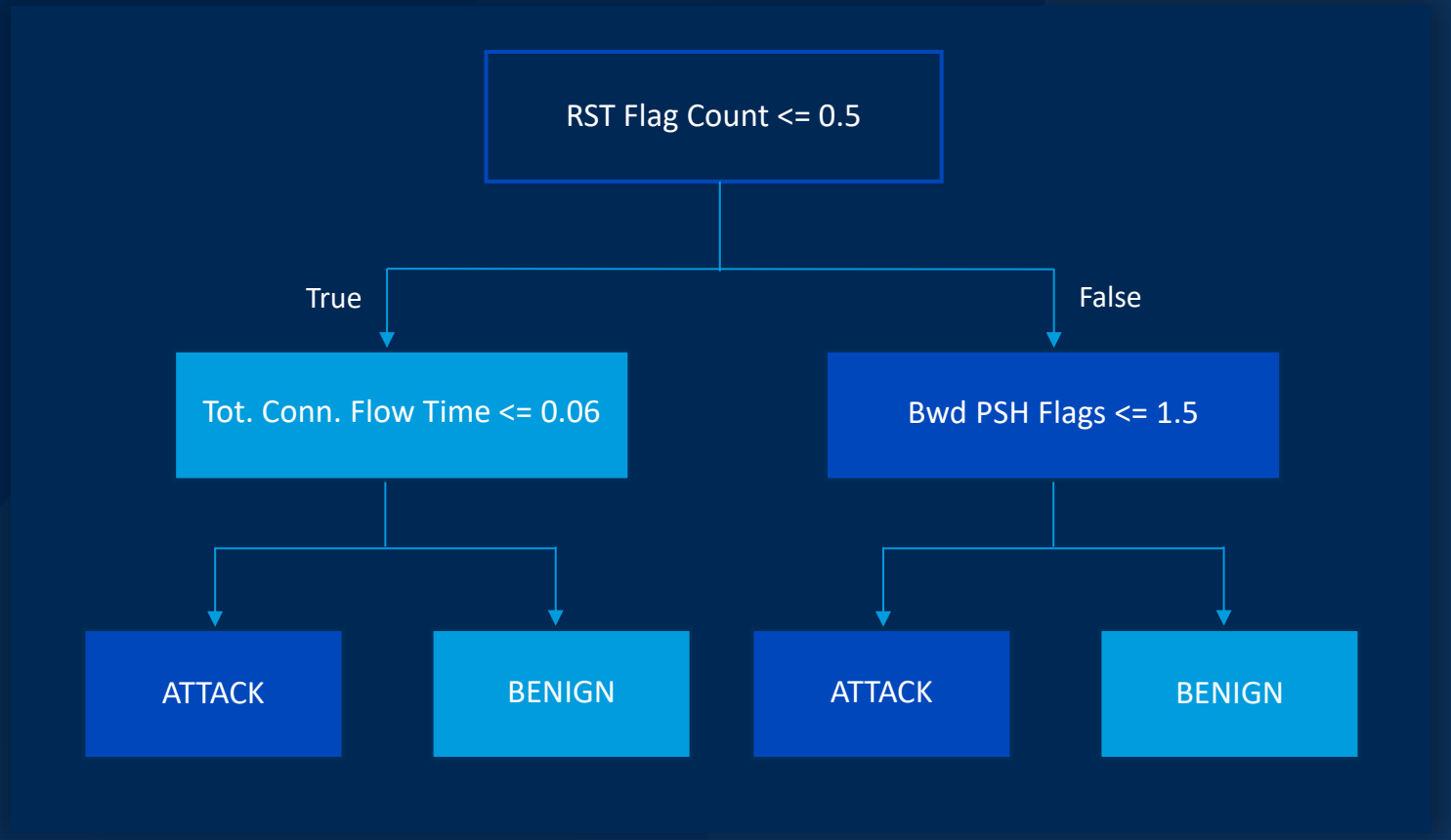
- We referenced the updated labeling procedure (2023)
- Made modifications for NetFlowMeter (**2.0!**)
- Collapsed everything into two classes: BENIGN and ATTACK
- Rows containing infinite values have been removed (**not present in NetFlowMeter!**)

	CICFlowMeter	NetFlowMeter
BENIGN	1594540	1353140
ATTACK	505431	497999

Less flows due to removed flow-cutting

Decision trees for Exploratory Data Analysis

- Good for gaining insights into data quality issues and identifying important features
- Classification and regression
- Produce explainable, rule-based trees



Binary classification

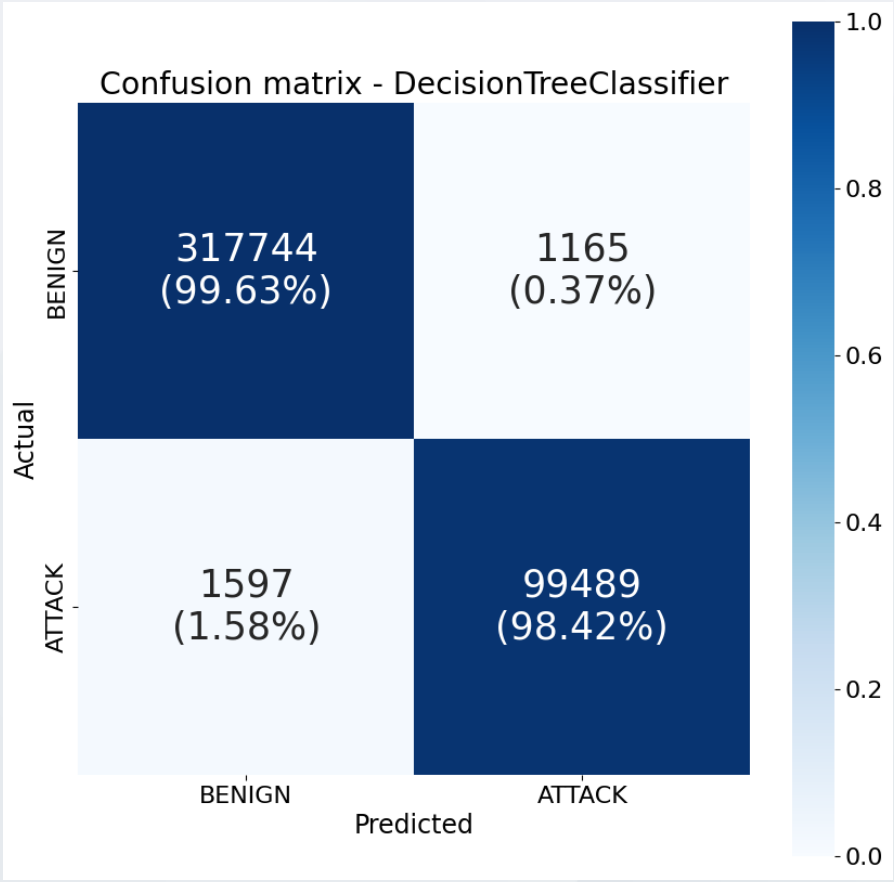
- Decision tree classifier using scikit-learn
- Some columns are excluded from training
- 80/20 training/test split, no data scaling needed

```
excluded_cols= ['id','Flow ID','Src IP','Src Port','Dst IP','Dst Port','Protocol',
                'Timestamp','Fwd URG Flags','Bwd URG Flags','URG Flag Count',
                'Attempted Category','Label']

X = df.drop(columns=excluded_cols, axis=1)
y = df['Label']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=RANDOM_SEED, stratify=y
)

dtc = DecisionTreeClassifier(max_depth=MAX_DEPTH, random_state=RANDOM_SEED)
dtc.fit(X_train, y_train)
y_pred = dtc.predict(X_test)
```

Binary classification

max_depth=5

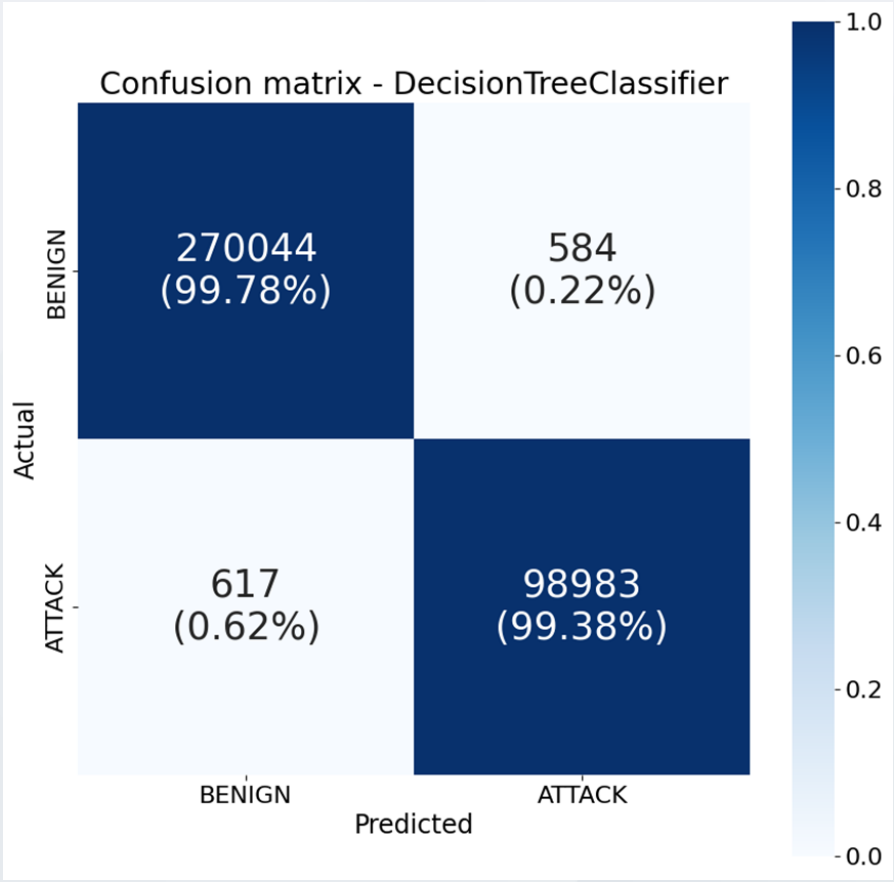
CICFlowMeter (2023)



Binary classification

max_depth=5

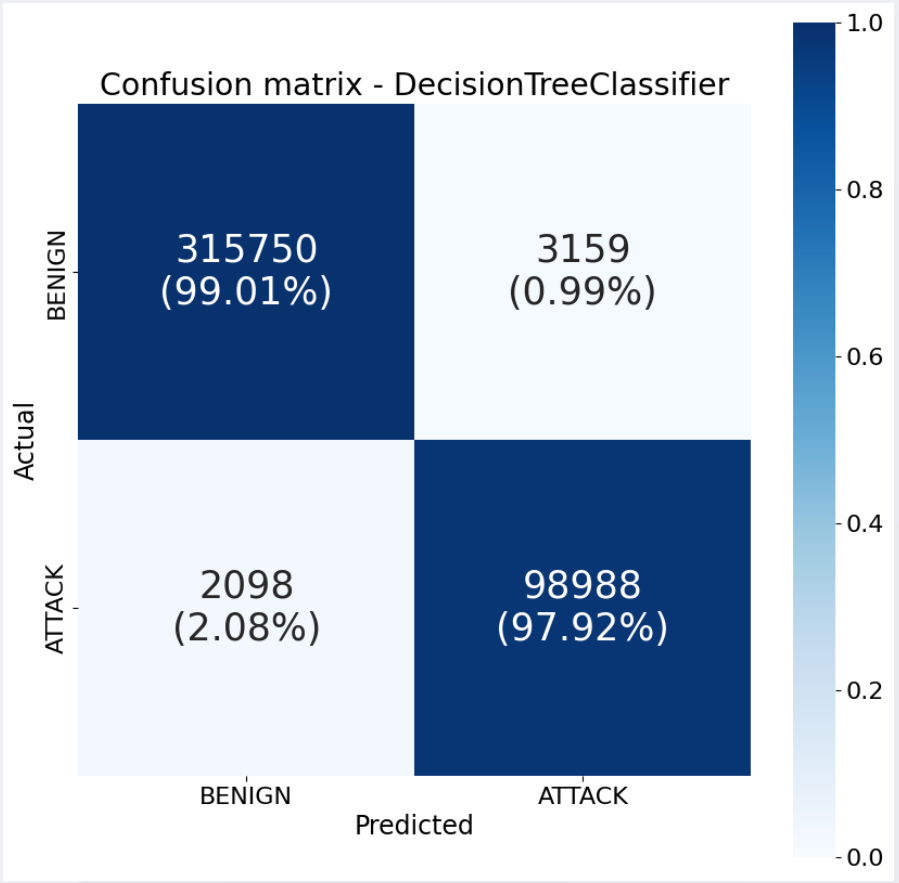
NetFlowMeter



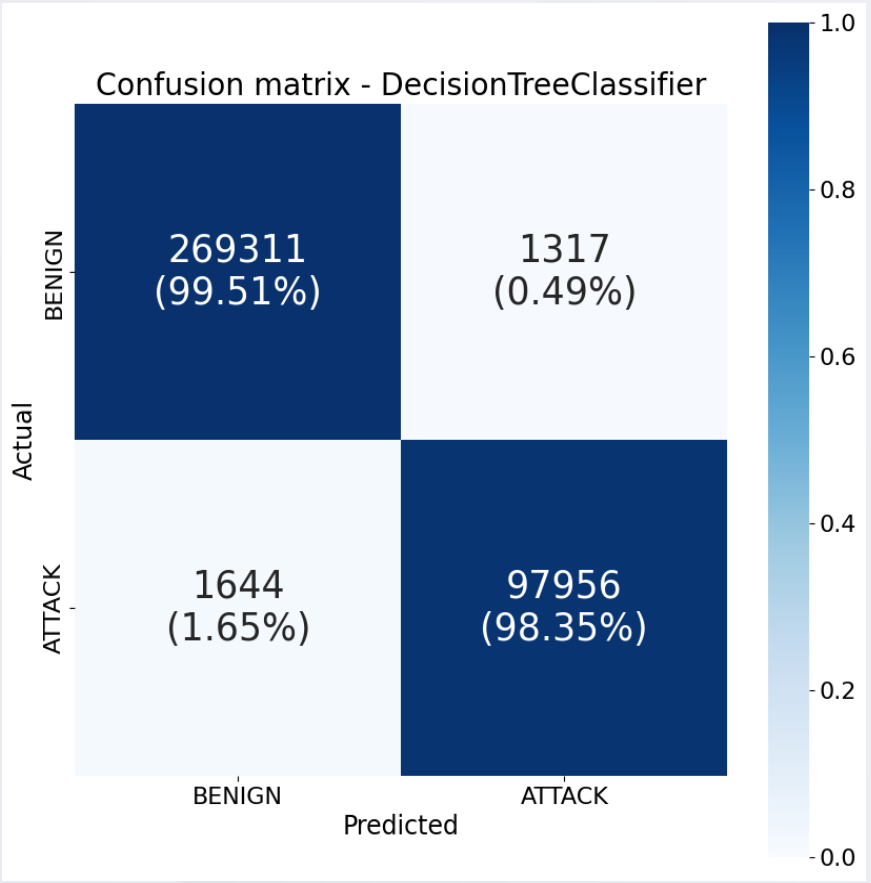
Binary classification

max_depth=3

CICFlowMeter (2023)

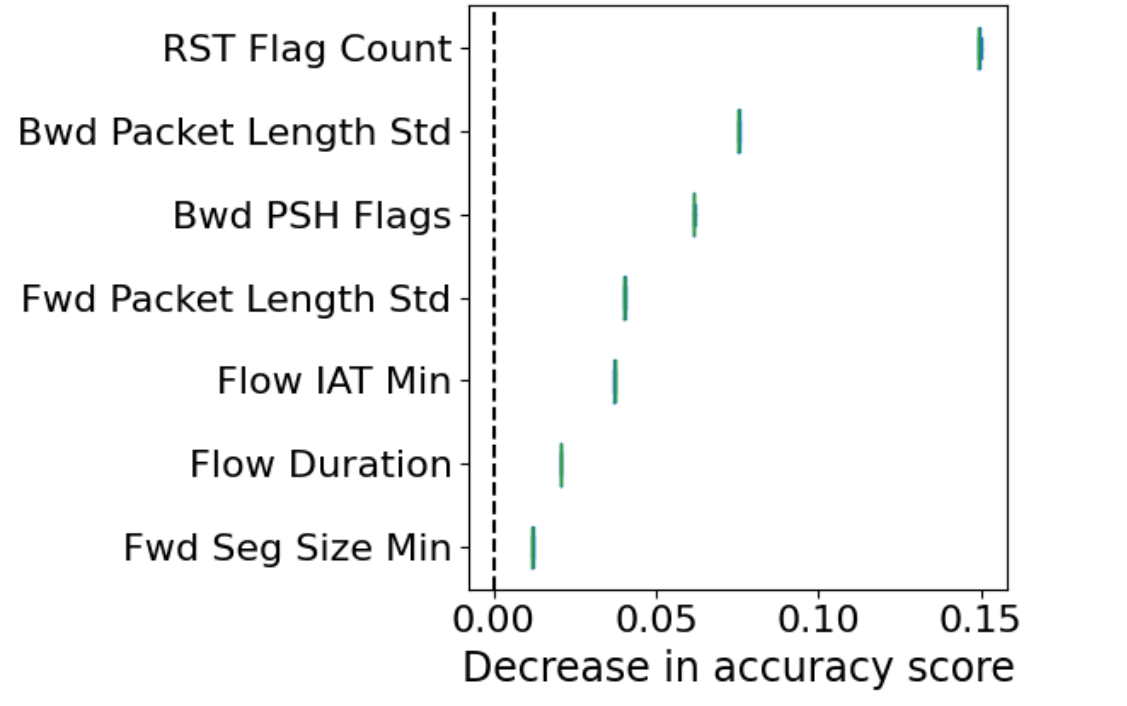


NetFlowMeter

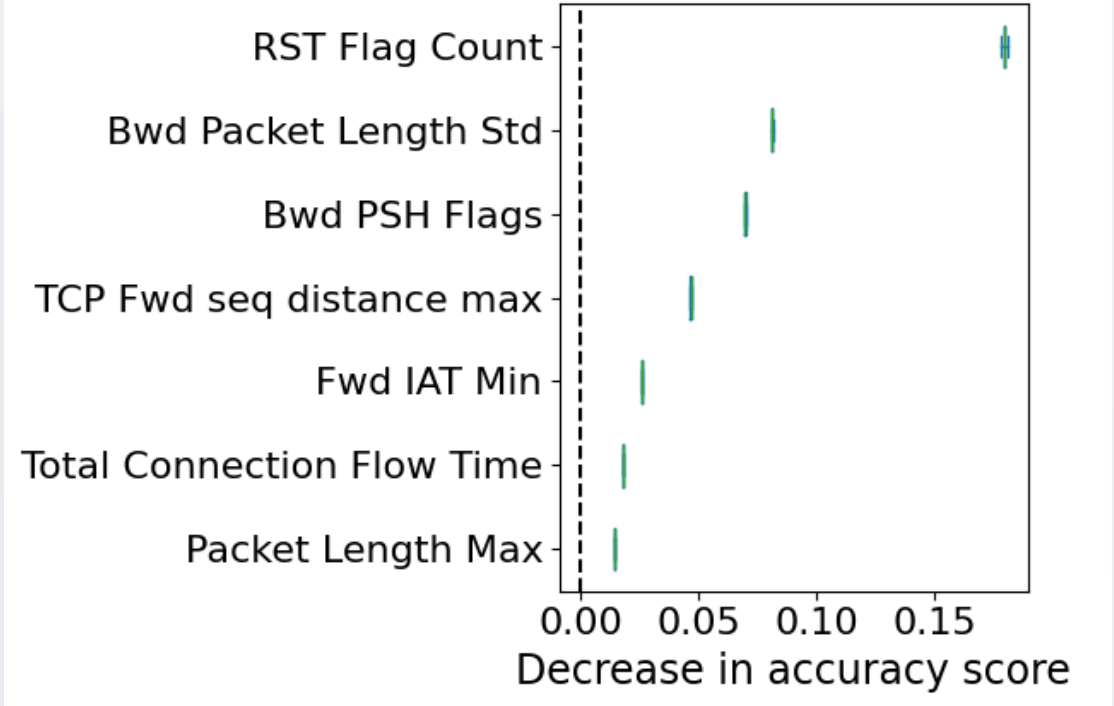


Permutation feature importance (test set)

CICFlowMeter (2023)



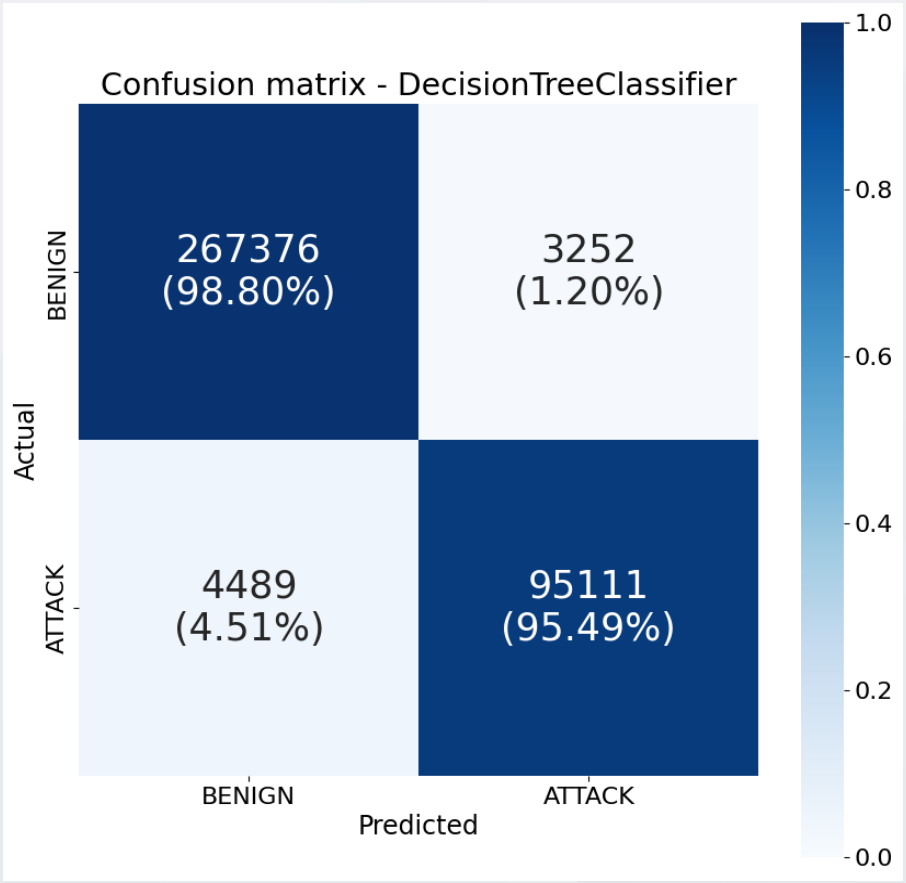
NetFlowMeter



Binary classification

max_depth=2

NetFlowMeter

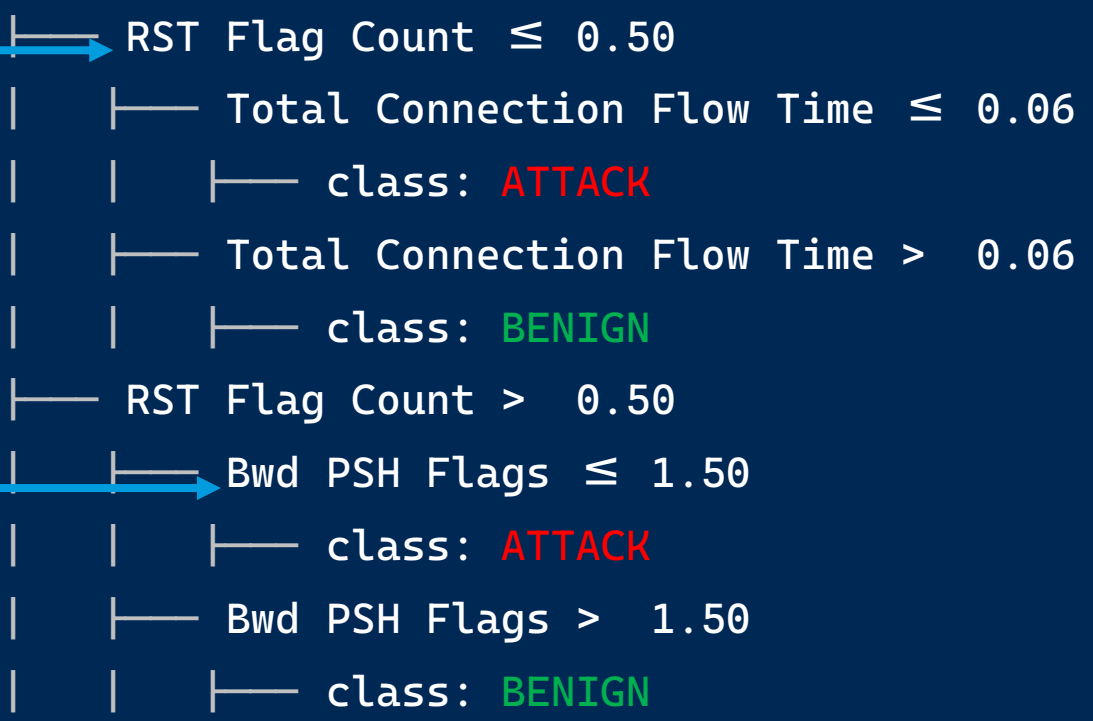


Decision tree

The class imbalance is likely affecting the results

Number of packets with the RST flag set

Bwd indicates only packets sent by the server

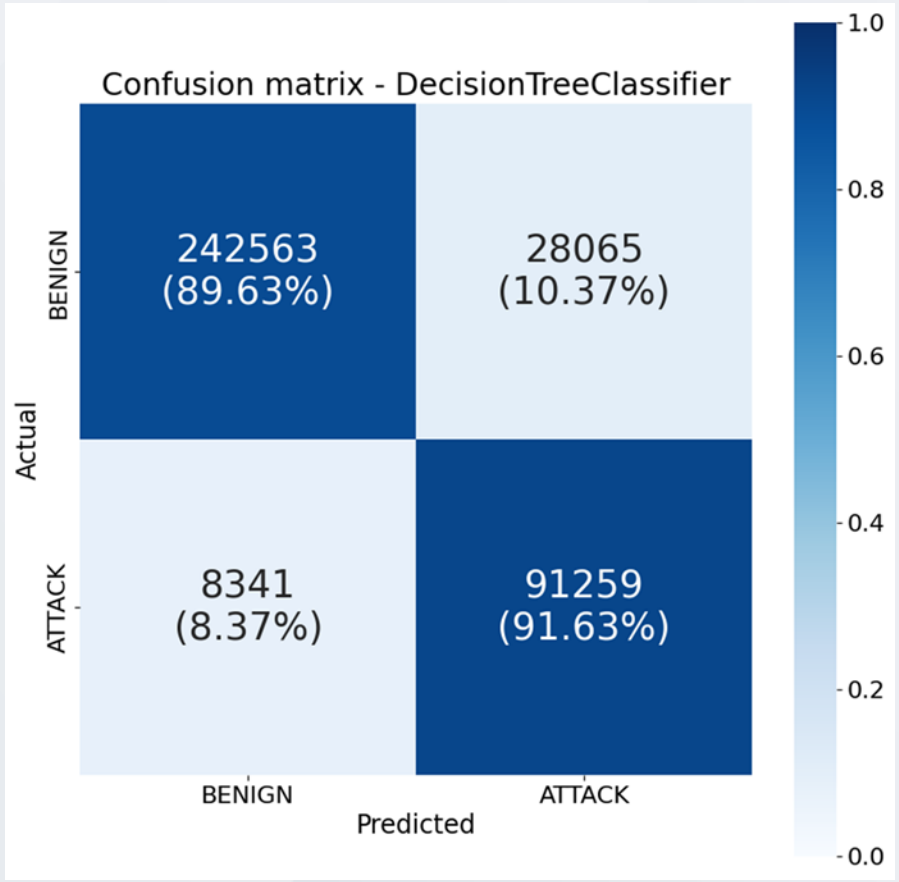


Binary classification

max_depth=1

```
├── RST Flag Count ≤ 0.50
│   ├── class: BENIGN
├── RST Flag Count >  0.50
│   ├── class: ATTACK
```

NetFlowMeter



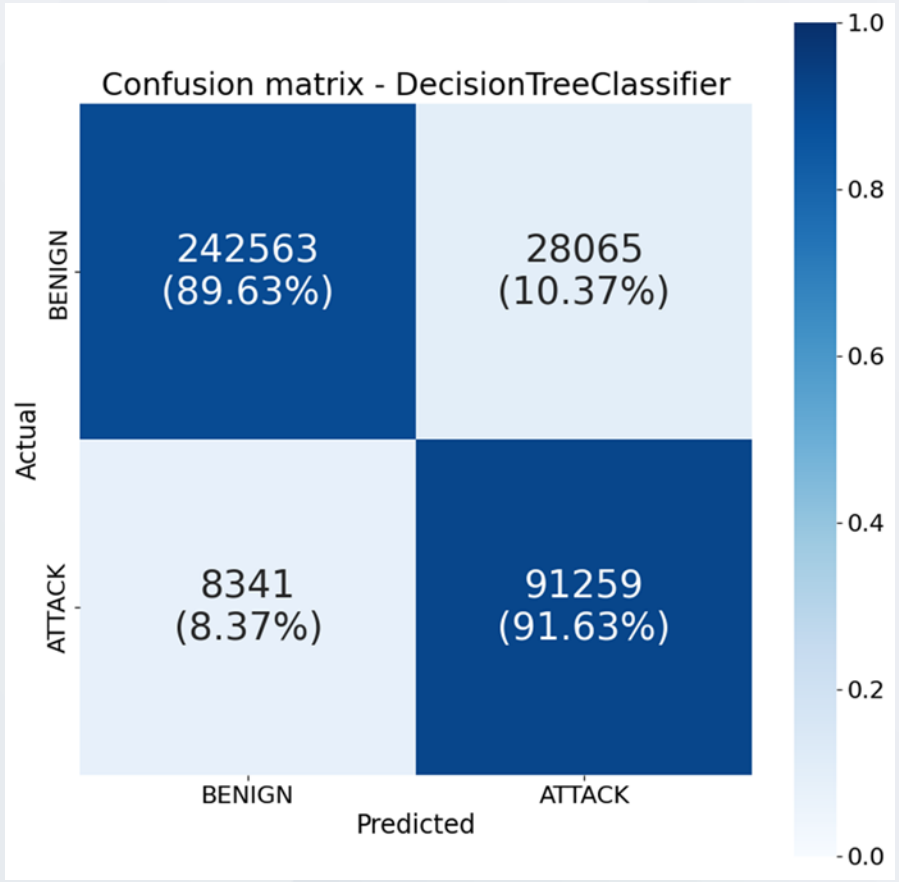
Binary classification

max_depth=1

```
├── RST Flag Count ≤ 0.50
│   ├── class: BENIGN
├── RST Flag Count >  0.50
│   ├── class: ATTACK
```

Is every reset connection an attack?

NetFlowMeter



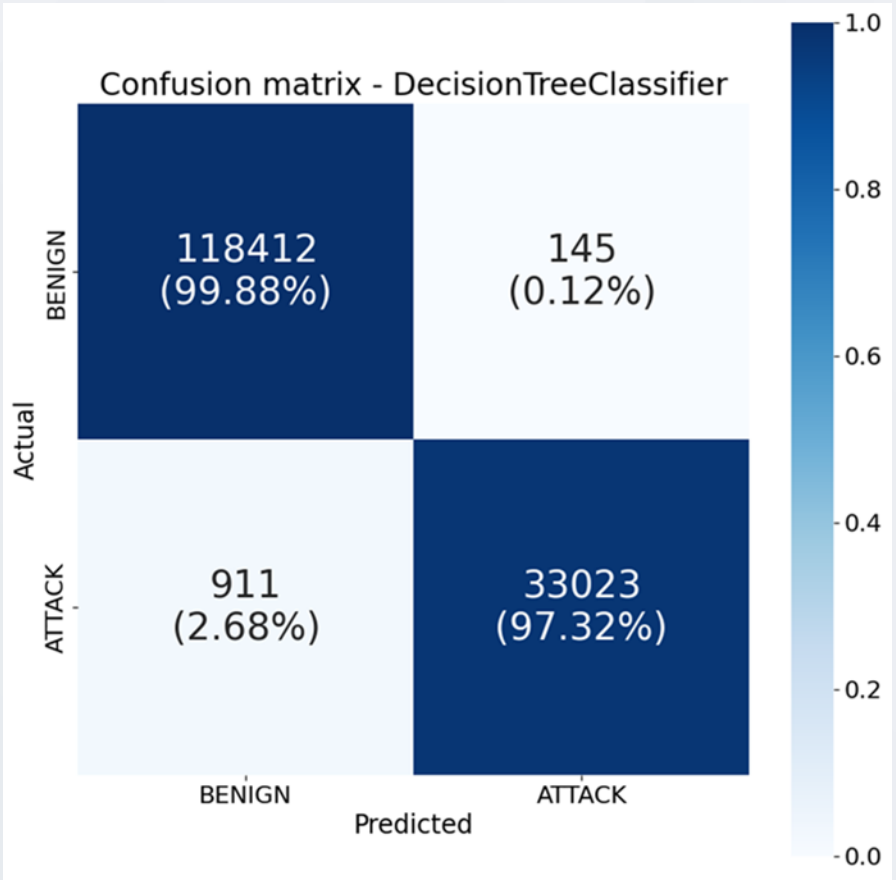
Binary classification

max_depth=1
Only Monday and Wednesday
(mostly HTTP traffic)

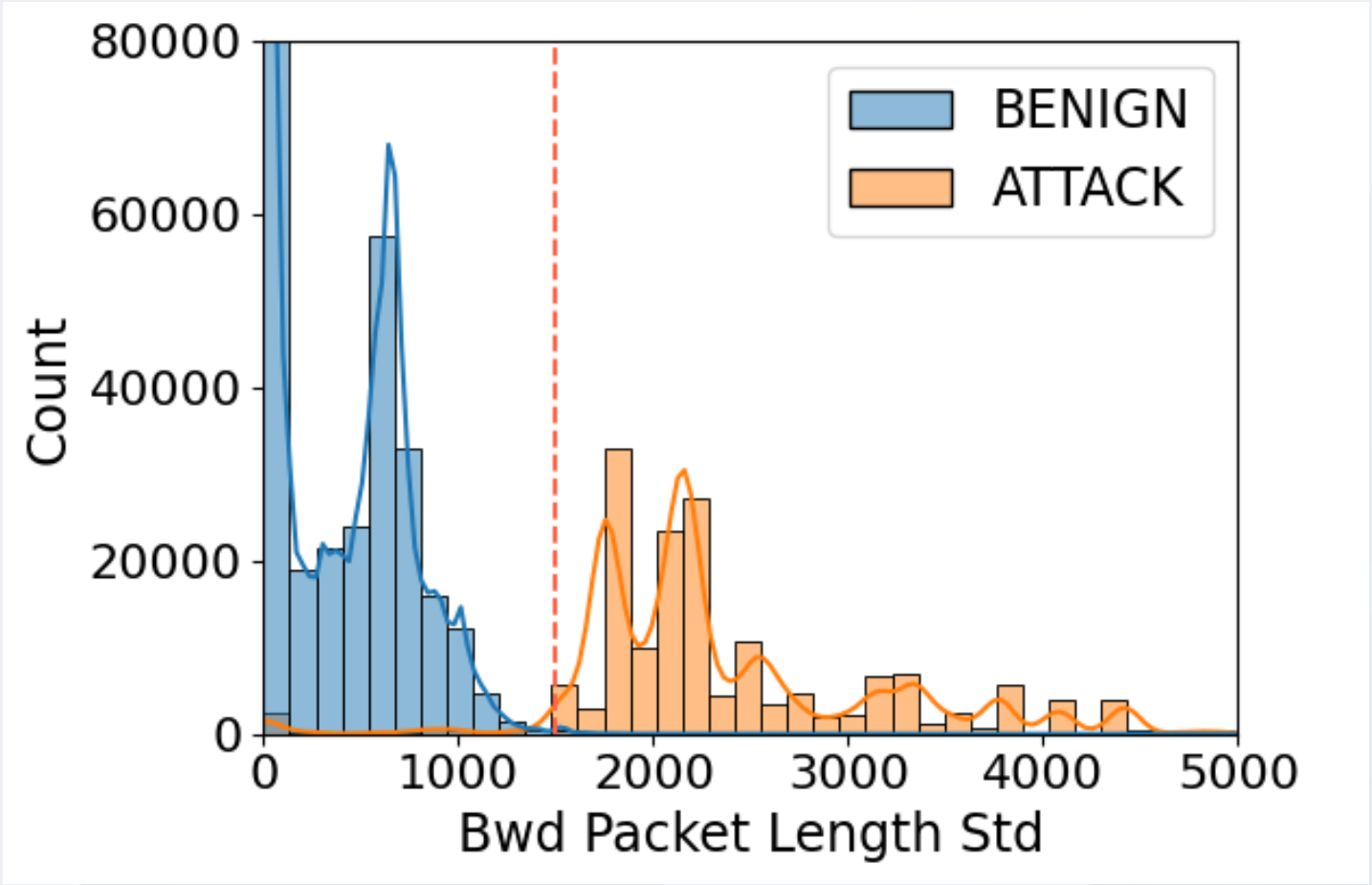
```
├── Bwd Packet Len Std ≤ 1496.86
│   ├── class: BENIGN
├── Bwd Packet Len Std >  1496.86
│   ├── class: ATTACK
```

Std dev of the length of packets from the server

NetFlowMeter



Distribution of ‘Bwd Packet Length Std’



Repeated attacks

- All Wednesday DoS attacks look the same in Wireshark
- GET / with a random query
- The server always returns the same page

- The launchpad URL provides a good marker

Wireshark · Segui flusso HTTP (tcp.stream eq 136320) · Wednesday-workingHours.pcap

```
GET /?NZIAQHXMO=MXAZYZUJR HTTP/1.1
Accept-Encoding: identity
Host: 205.174.165.68
Keep-Alive: 118
User-Agent: Mozilla/5.0 (Windows; U; MSIE 7.0; Windows NT 6.0; en-US)
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Connection: close
Referer: http://engadget.search.aol.com/search?q=OIAFMAMNON
Cache-Control: no-cache


HTTP/1.1 200 OK
Date: Wed, 05 Jul 2017 13:54:17 GMT
Server: Apache/2.4.18 (Ubuntu)
Last-Modified: Mon, 26 Jun 2017 14:32:04 GMT
ETag: "2c39-552ddd09283d0"
Accept-Ranges: bytes
Content-Length: 11321
Vary: Accept-Encoding
Connection: close
Content-Type: text/html


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
<html xmlns="http://www.w3.org/1999/xhtml">
  <!--
    Modified from the Debian original for Ubuntu
    Last updated: 2014-03-19
    See: https://launchpad.net/bugs/1288690
  -->
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" /
    <title>Apache2 Ubuntu Default Page: It works</title>
    <style type="text/css" media="screen">
  * {
    margin: 0px 0px 0px 0px;
    padding: 0px 0px 0px 0px;
  }
```

Repeated attacks

- Searching the URL in the PCAP reveals 163k hits
- All attacks are essentially the same
- The model is simply learning the shape of a response

```
1E850C560   6E 3A 20 30 70 78 20 30 70 78 20 30 70 78 20 30   n: 0px 0px 0px 0
```

Checksum | Cerca (163791 corrispondenze)

Scostamento	Estratto (hex)	Estratto (testo)
1E84E89...	65 3A 20 68 74 74 70 73 3A 2F 2F 6C 61 75 6E 63 **68 70 61 64 2E 6E 65 74 2F 62**	e: https://launc**hpad.net/bugs/12**
1E84ECE...	65 3A 20 68 74 74 70 73 3A 2F 2F 6C 61 75 6E 63 **68 70 61 64 2E 6E 65 74 2F 62**	e: https://launc**hpad.net/bugs/12**
1E84F04...	65 3A 20 68 74 74 70 73 3A 2F 2F 6C 61 75 6E 63 **68 70 61 64 2E 6E 65 74 2F 62**	e: https://launc**hpad.net/bugs/12**
1E84F68...	65 3A 20 68 74 74 70 73 3A 2F 2F 6C 61 75 6E 63 **68 70 61 64 2E 6E 65 74 2F 62**	e: https://launc**hpad.net/bugs/12**
1E84F8C...	65 3A 20 68 74 74 70 73 3A 2F 2F 6C 61 75 6E 63 **68 70 61 64 2E 6E 65 74 2F 62**	e: https://launc**hpad.net/bugs/12**
1E84FBB...	65 3A 20 68 74 74 70 73 3A 2F 2F 6C 61 75 6E 63 **68 70 61 64 2E 6E 65 74 2F 62**	e: https://launc**hpad.net/bugs/12**
1E84FF5...	65 3A 20 68 74 74 70 73 3A 2F 2F 6C 61 75 6E 63 **68 70 61 64 2E 6E 65 74 2F 62**	e: https://launc**hpad.net/bugs/12**

Protocol confusion

```
1 id,Flow ID,Src IP,Src Port,Dst IP,Dst Port,Protocol,Timest
2 1,192.168.10.5-192.168.10.3-49159-445-6,192.168.10.5,49159
3 2,8.6.0.1-8.0.6.4-0-0-0,8.6.0.1,0,8.0.6.4,0,0,2017-07-04 1
4 3,192.168.10.5-192.168.10.3-123-123-17,192.168.10.5,123,19
```

- One of the very first rows in CICIDS2017 has a weird IP address
- Port numbers are also set to 0
- This can't be right

Hostname:	8.6.0.1
ASN:	3356
ISP:	Giglinx Inc
Services:	Data Center/Transit

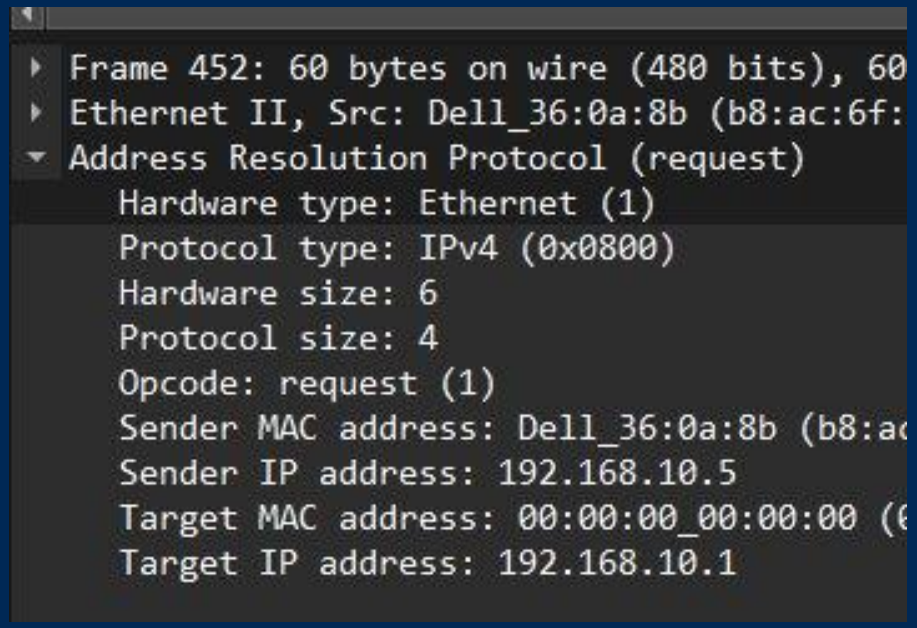
Protocol confusion

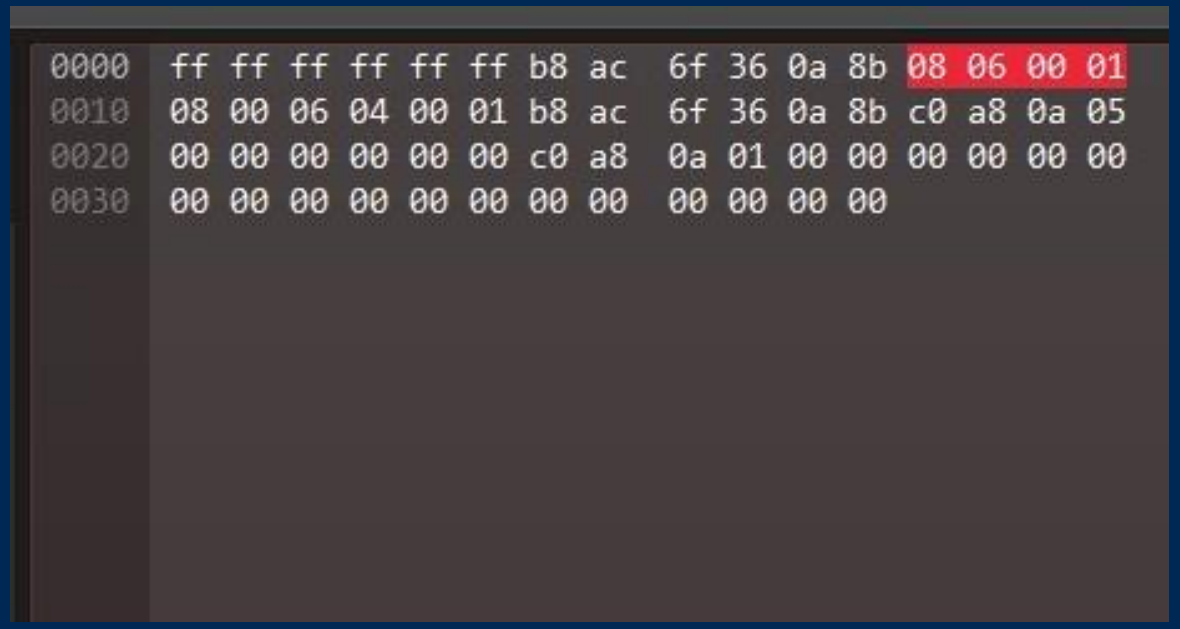
- Filtering the relevant PCAP for this IP addresses yields no results

```
$ tshark -r Tuesday-WorkingHours.pcap -Y "ip.addr == 8.0.6.4" -w output.pcap
$ xxd output.pcap
00000000: 0a0d 0d0a 4800 0000 4d3c 2b1a 0100 0000  ....H...M<+.....
00000010: ffff ffff ffff ffff 0300 1600 4c69 6e75  ............Linu
00000020: 7820 342e 382e 302d 3232 2d67 656e 6572  x 4.8.0-22-gener
00000030: 6963 0000 0400 0800 6d65 7267 6563 6170  ic......mergecap
00000040: 0000 0000 4800 0000 0100 0000 1400 0000  ....H...........
00000050: 0100 0000 0000 0400 1400 0000            ............
$
```

Protocol confusion

- Searching for the binary representation of the IP address however returns one hit
- Wireshark reports this as an ARP packet





Protocol confusion

- The byte pattern appears at offset 12 which matches the offset of the source address in an IP packet
- Certain packets have been mistakenly parsed as the wrong protocol leading to no-sense entries in the dataset

```
0000  ff ff ff ff ff ff b8 ac  6f 36 0a 8b 08 06 00 01
0010  08 00 06 04 00 01 b8 ac  6f 36 0a 8b c0 a8 0a 05
0020  00 00 00 00 00 00 c0 a8  0a 01 00 00 00 00 00 00
0030  00 00 00 00 00 00 00 00  00 00 00 00
```

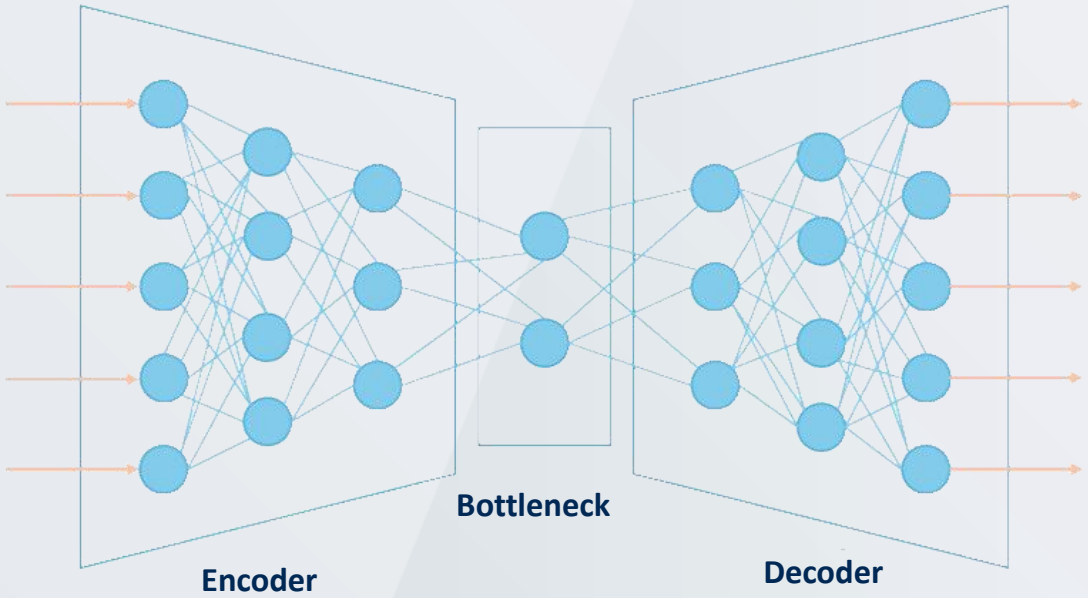
Offset	Octet	0										
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10
12	96	*Source address*										
16	128	*Destination address*										

Intrusion detection – Autoencoder (AE)

- Semi-supervised anomaly detection
  - Train exclusively on normal traffic

- AE architecture:
  - **Encoder**: compress the input into a lower-dim. representation in the **bottleneck**
  - **Decoder**: reconstruct it

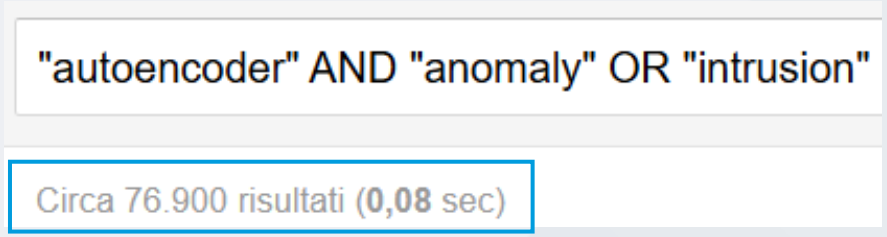
- Exploit the **reconstruction error** (RE)
  - Low RE = NORMAL
  - High RE = ANOMALY
- Need a **threshold**



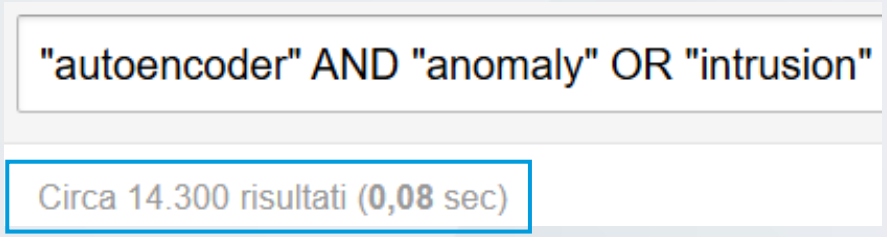
Key advantages

- Doesn’t require attacks for training
  - Anomalies are often rare, expensive or impossible to collect and label
- Can identify unseen and novel threats

Any time



Since 2025



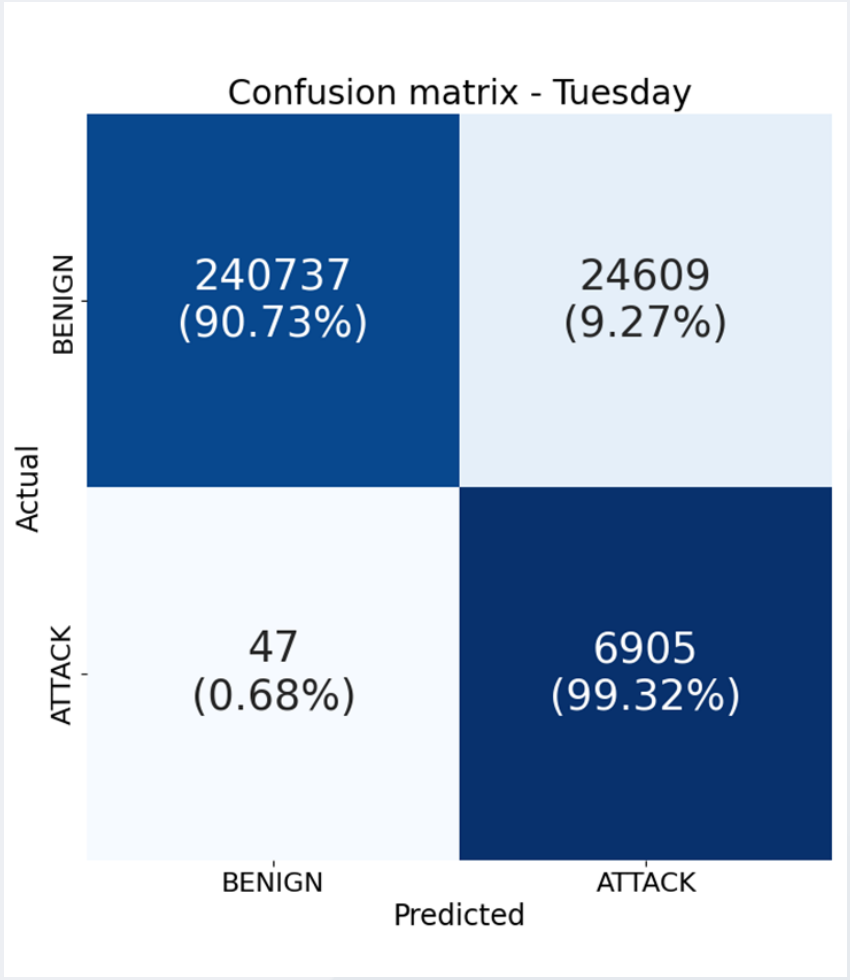
Data splitting and preprocessing – CICIDS2017

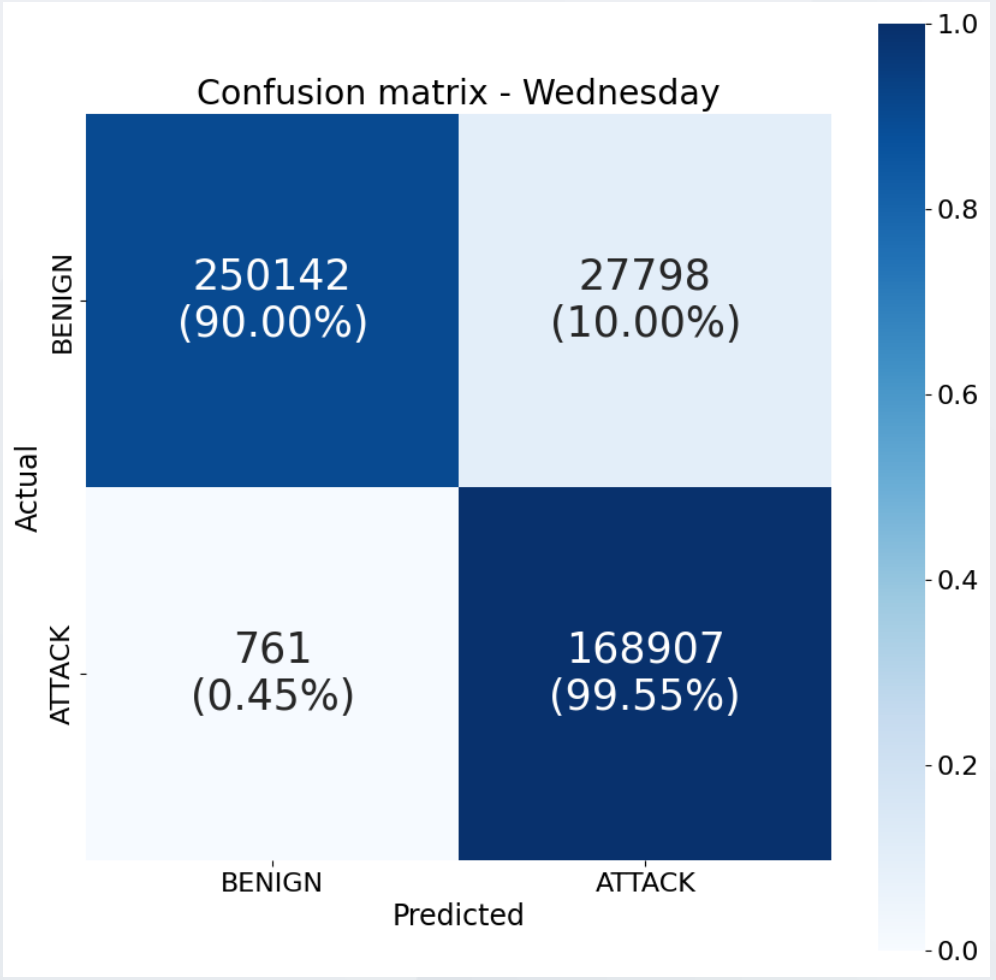
- NetFlowMeter 2.0
- Training and validation on Monday
  - 80/20 split
  - Compute the threshold on validation split
- Testing on every other day
  - Keeping the days separate
- Excluding the same columns as in the decision tree
- Scaling with MinMaxScaler

Architecture

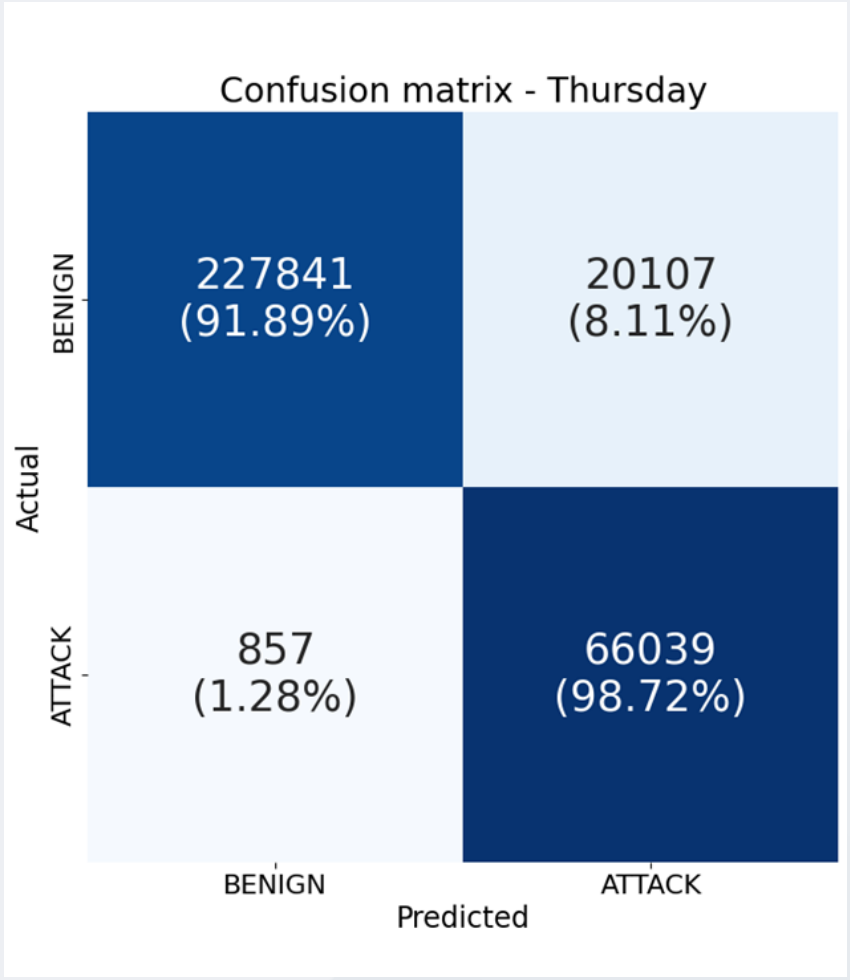
```
input_dim = X_train_scaled.shape[1]
leaky_slope = 0.1

# Encoder
input_layer = Input(shape=(input_dim, ))
layer = Dense(56, kernel_initializer=initializers.he_normal())(input_layer)
layer = BatchNormalization()(layer)
layer = LeakyReLU(negative_slope=leaky_slope)(layer)

# Bottleneck
layer = Dense(8, kernel_initializer=initializers.he_normal(),
              activity_regularizer=regularizers.l1(1e-5))(layer)
layer = BatchNormalization()(layer)
layer = LeakyReLU(negative_slope=leaky_slope)(layer)

# Decoder
layer = Dense(56, kernel_initializer=initializers.he_normal())(layer)
layer = BatchNormalization()(layer)
layer = LeakyReLU(negative_slope=leaky_slope)(layer)

# Output layer
output_layer = Dense(input_dim, activation='sigmoid',
                     kernel_initializer=initializers.he_normal())(layer)

model = Model(inputs=input_layer, outputs=output_layer)
model.compile(optimizer='adam', loss='mean_squared_error')
```

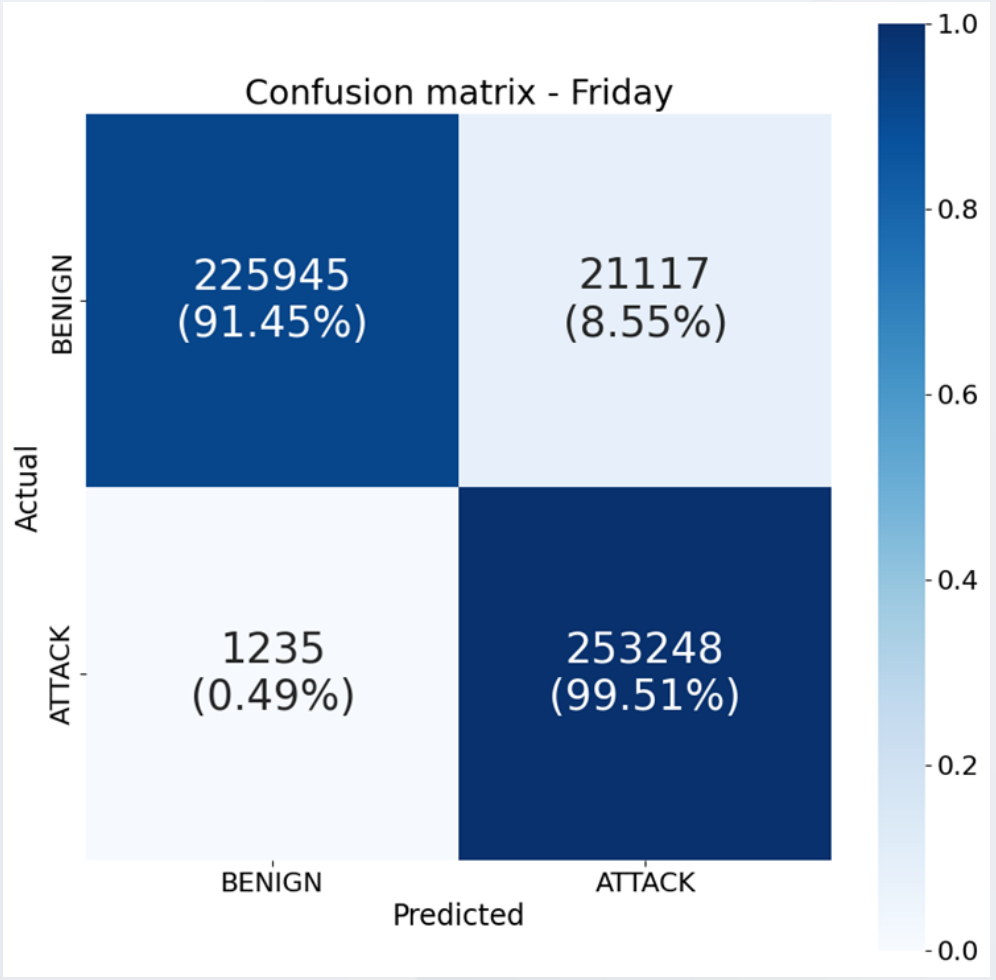
Results – NetFlowMeter



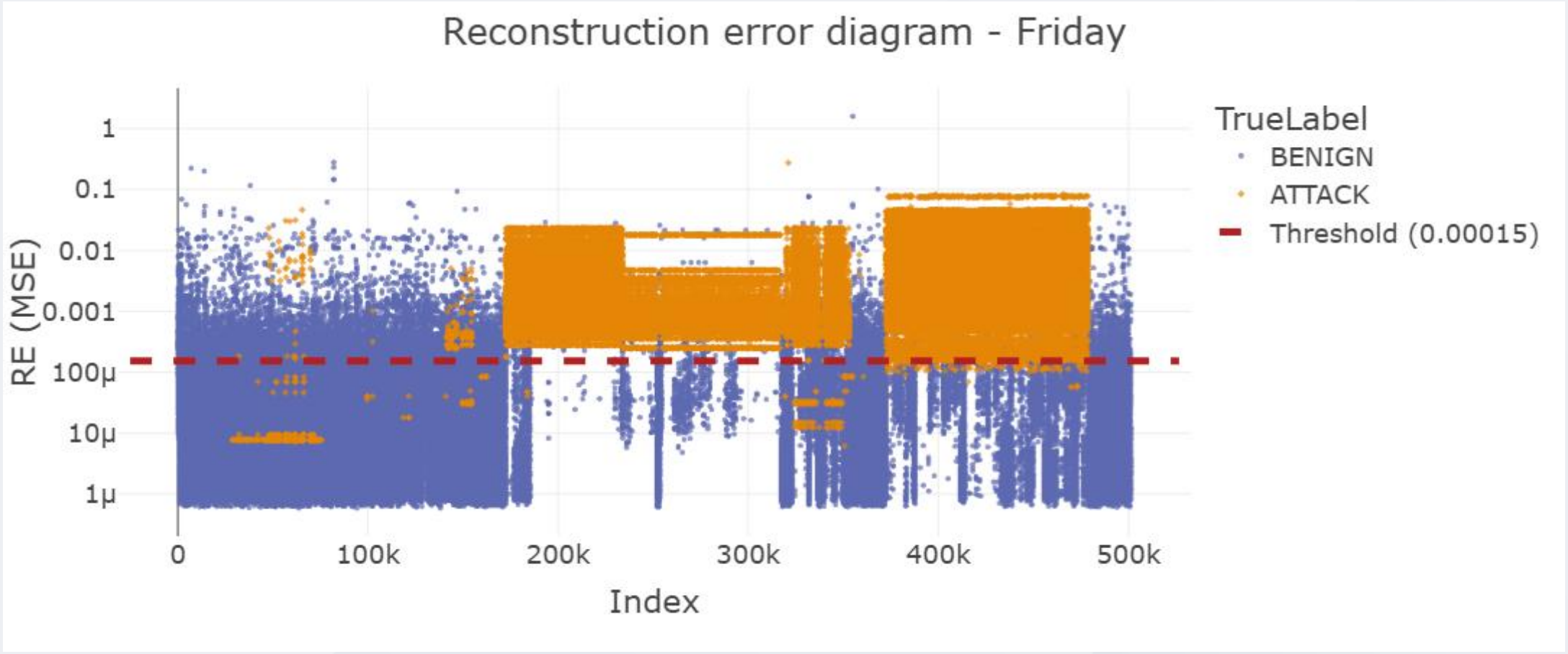


Results – NetFlowMeter



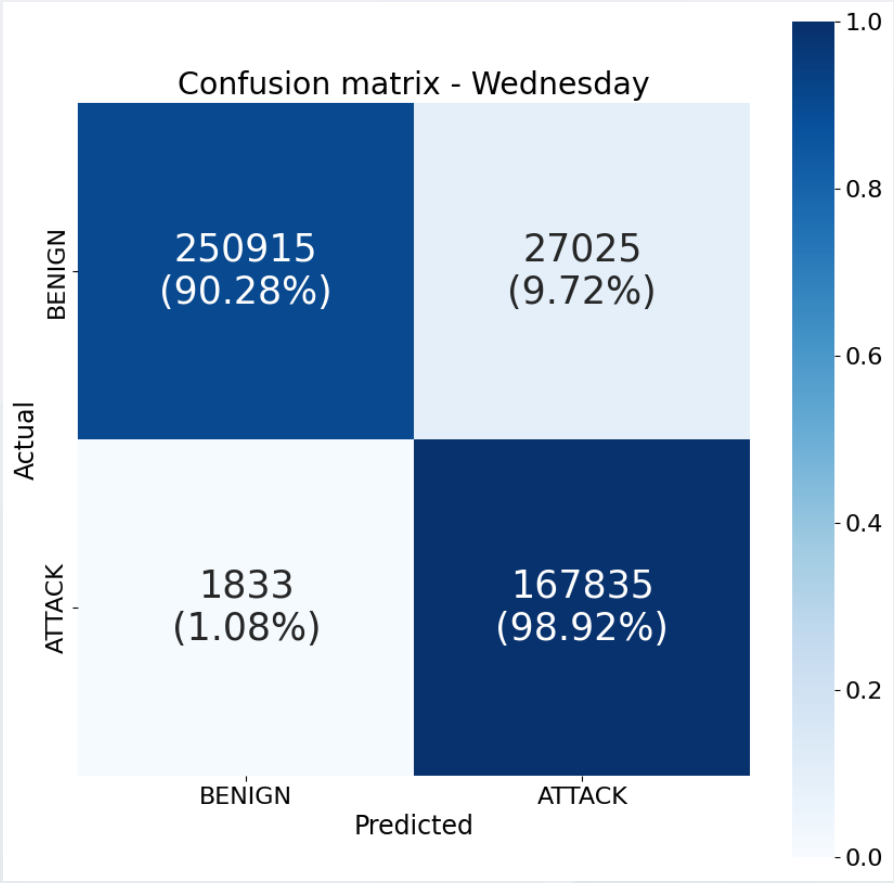


Results – NetFlowMeter



What happens if…

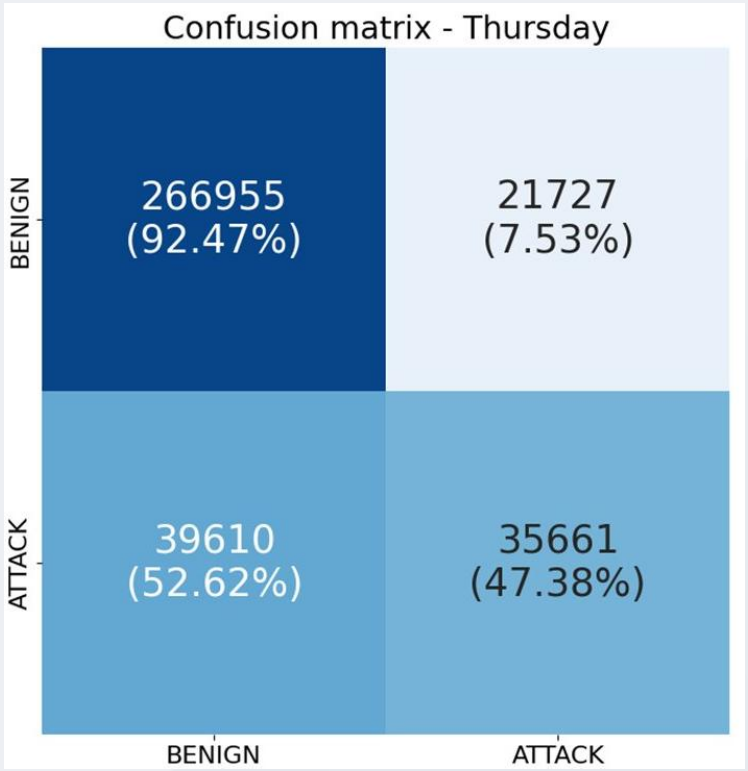
- … we only put 1 unit in the bottleneck layer?
- Results are bad, except on Wednesday
  - This is due to data redundancy



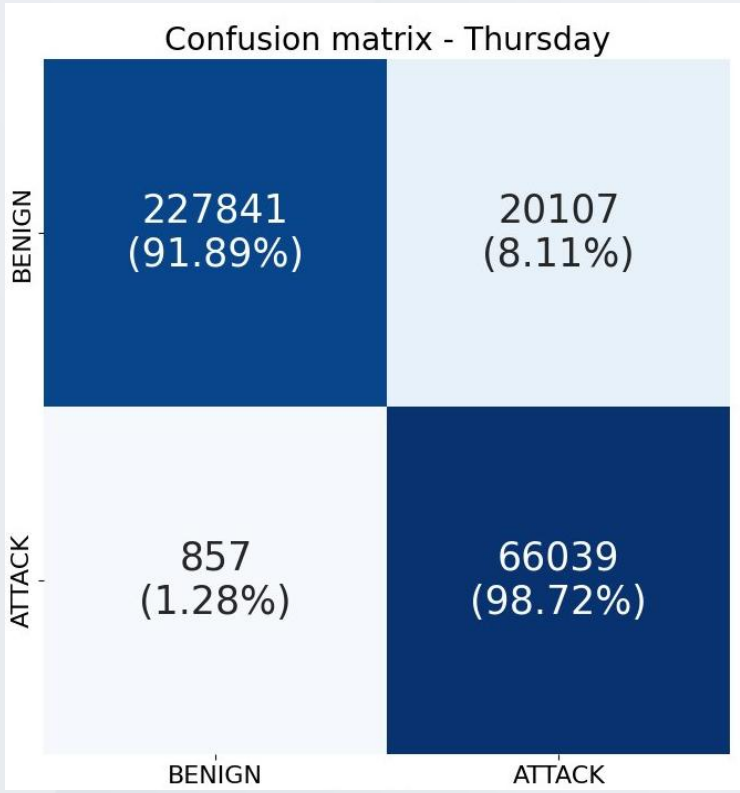
What happens if…

- … we restore the flow cutting behavior of CICFlowMeter?
- Performance drops significantly on Thursday and Friday

Flow cutting

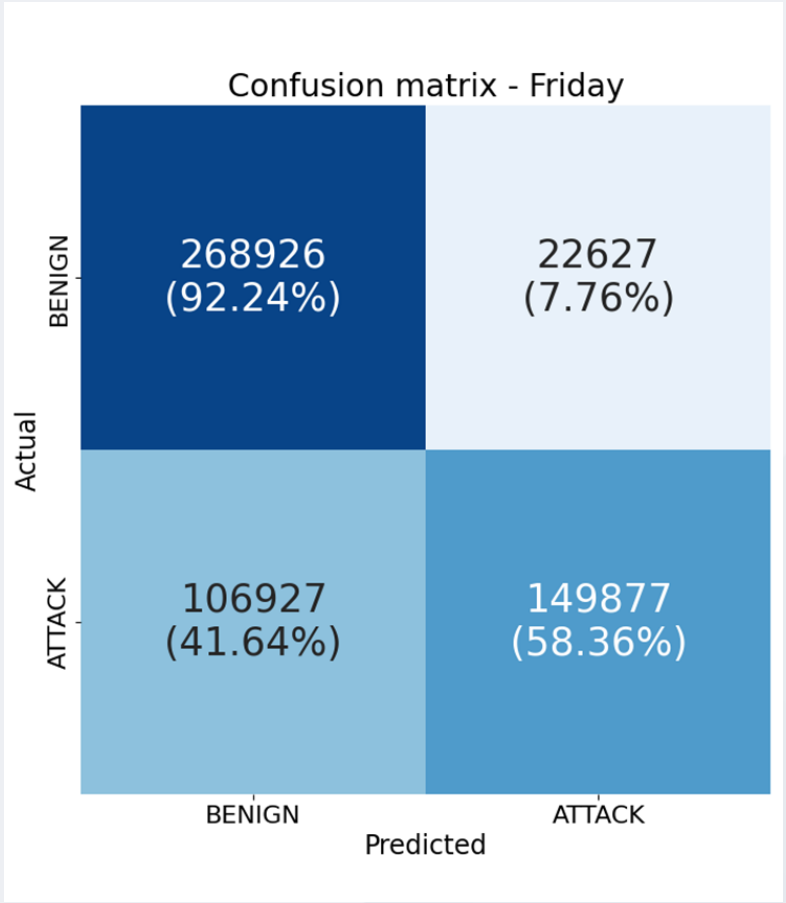


No flow cutting

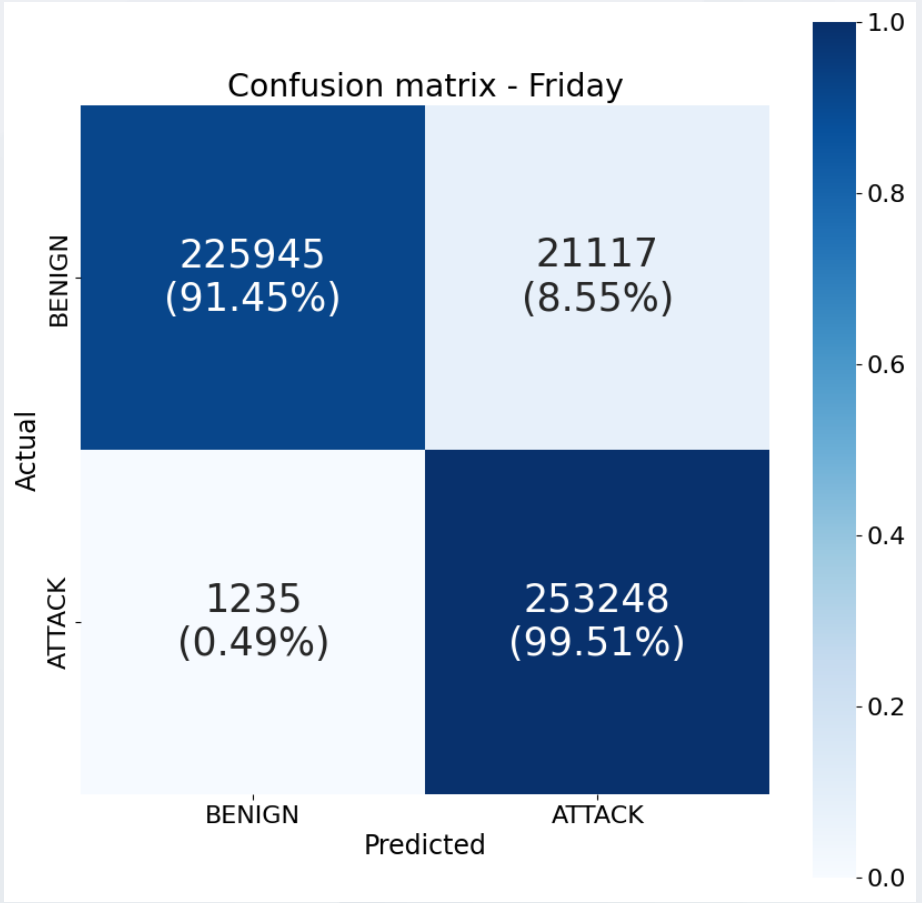


What happens if…

Flow cutting

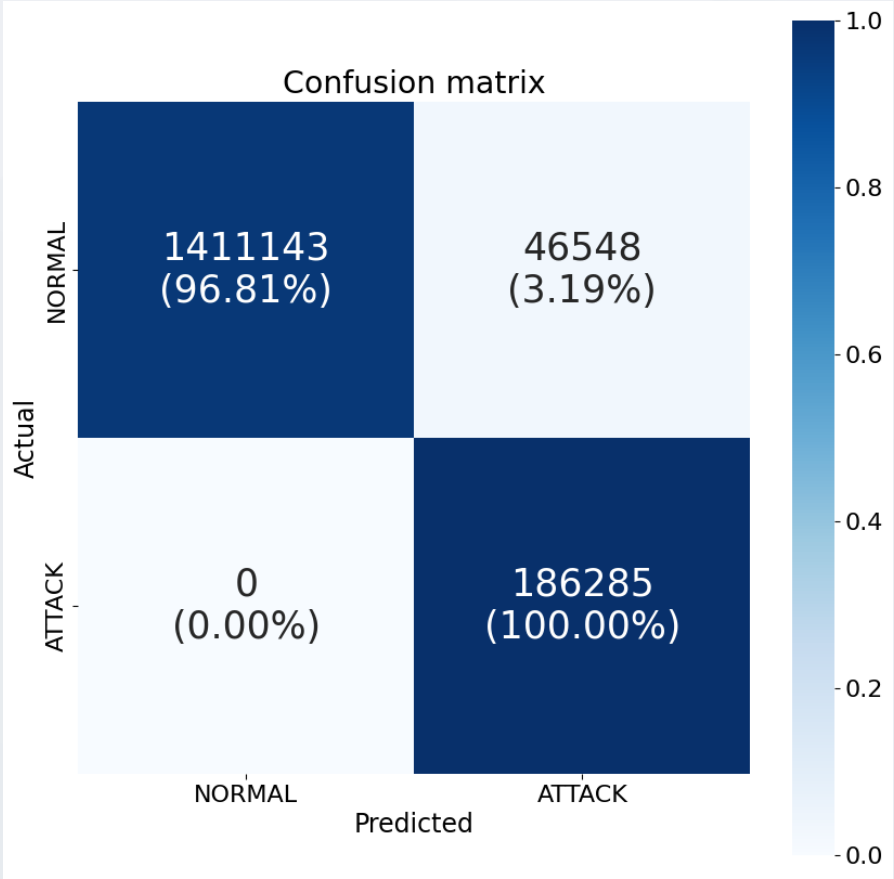


No flow cutting



A real-world example

- Curated dataset collected by us
- Honeypot environment with multiple protocols
- More work needed for false positives
  - Different models (e.g. VAE)
  - Ensemble learning (e.g. voting)
  - Rule-based IDS filtering



Adversarial threats to ML-based IDS

Tools to defend and evaluate ML models against evasion, poisoning, extraction and inference

- Feature space vs problem space evasion attacks
  - **Intriguing Properties of Adversarial ML Attacks in the Problem Space**, Pierazzi et al., 2020. 10.1109/SP40000.2020.00073
  - *"Modifying real objects that correspond to an adversarial feature vector"*

Takeaways

- Do not trust third-party tools and datasets blindly
- Always explore your PCAP and extracted features before applying ML
- For real-world use, additional processing may be needed before/after the ML classification
- Generating attack data with a single tool and configuration may lead to profiling the tool or the server instead of the attack

References

- https://www.unb.ca/cic/datasets/ids-2017.html
- https://github.com/GintsEngelen/CICFlowMeter
- https://intrusion-detection.distrinet-research.be/CNS2022/Datasets
- https://intrusion-detection.distrinet-research.be/CNS2022/CICIDS2017.html
- https://github.com/Trusted-AI/adversarial-robustness-toolbox
- **Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction**, Sakurada and Yairi, 2014. 10.1145/2689746.2689747

Questions?