OPTIMIZING DNS FILTERING WHEN UNDER ATTACK

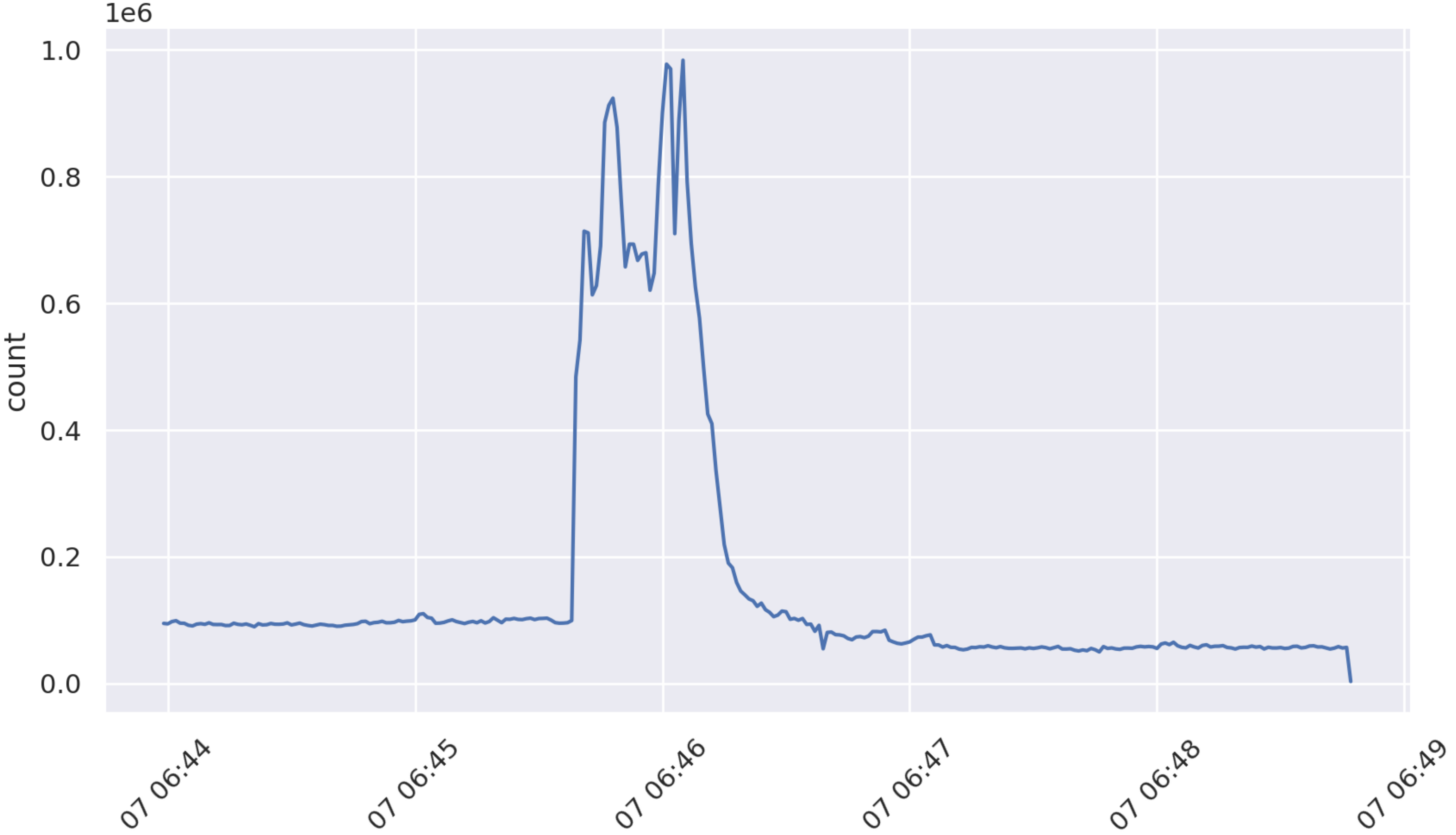
(especially when under pressure)

Wes Hardaker <hardaker@isi.edu>

Rapid response leads to rapid errors

- SECOPS are pressured to react quickly to malicious traffic
  - DDoS
  - Penetration
  - …
- Initial goal: stop as much of it as possible by filtering
  - Source addresses
  - Destination addresses
  - Protocols
  - …

Case Study: A DDoS attack on b.root-servers.net



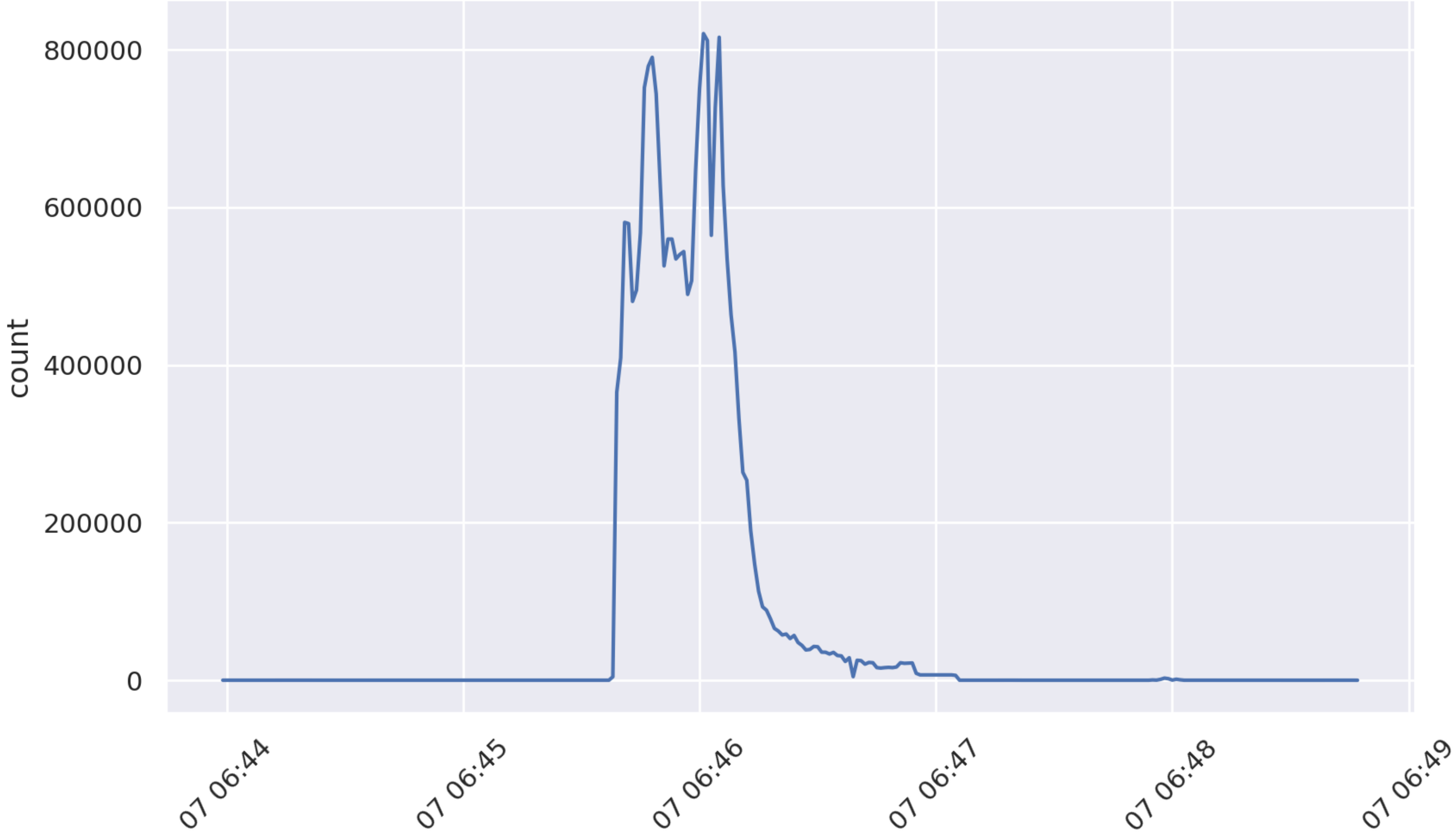
Dataset is available on comunda.isi.edu

The dataset’s published analysis

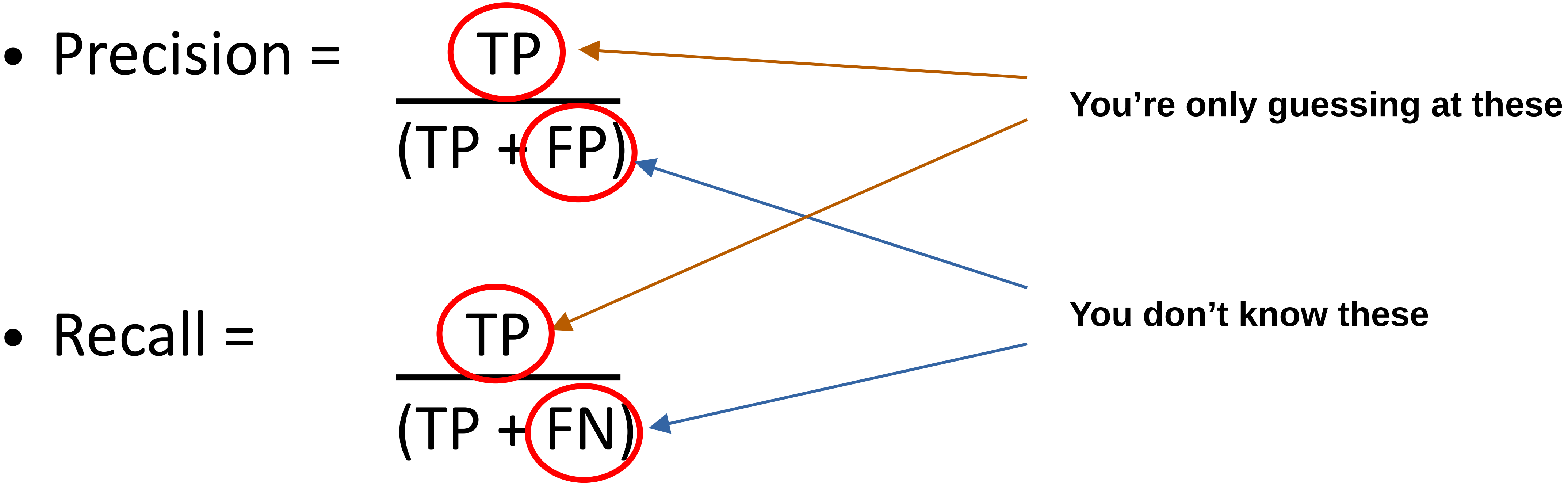
- Attack characteristic: Randomized sources
- Query name: Random
- Response codes: Random
- *… more randomness not shown …*
- Packet size: **540 bytes IP packets**

Clearly we should filter on this

Packet Size == 540 for the win!



Ensuring we’re right: calculating our filter’s precision and recall



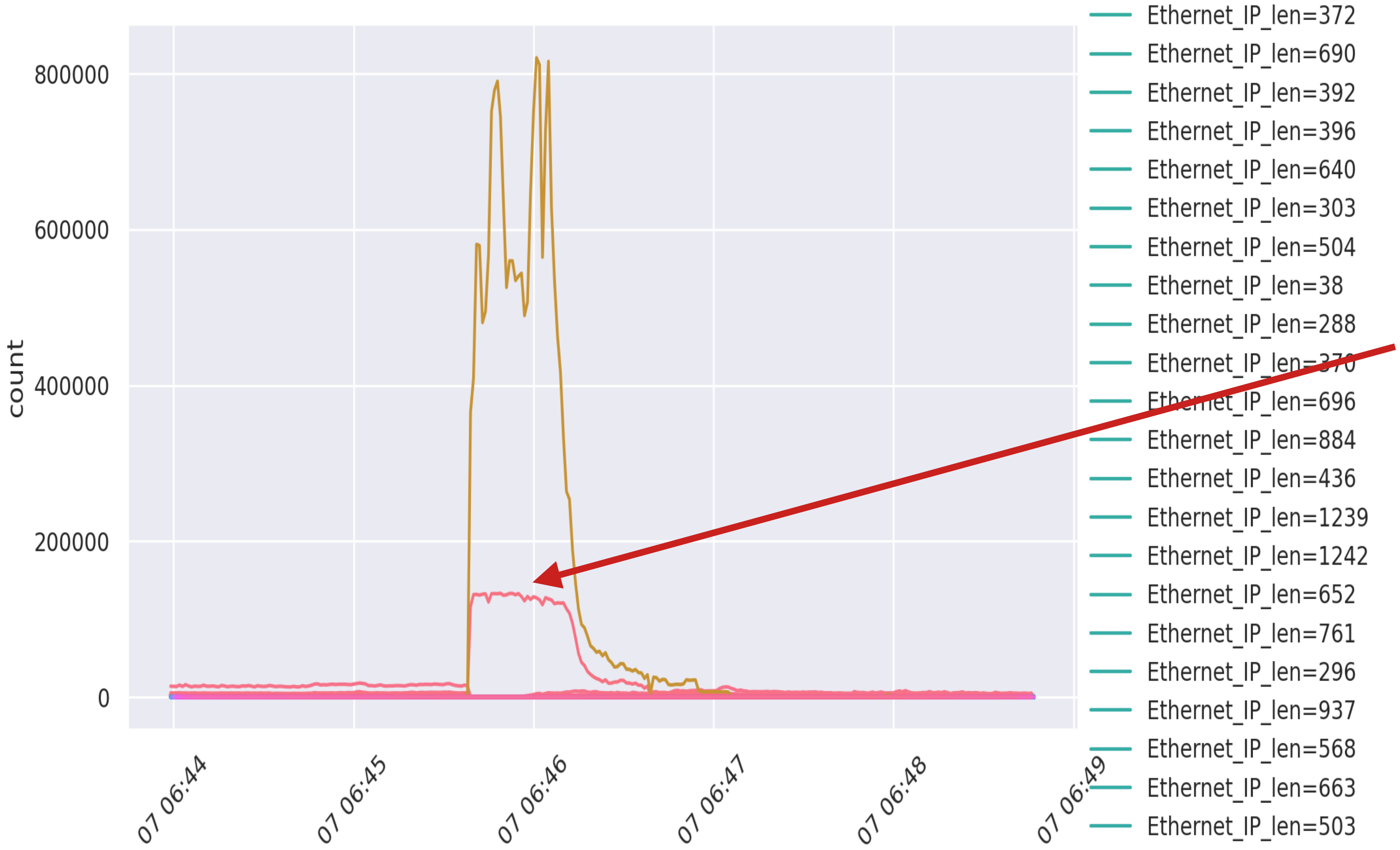
- Precision = $\frac{TP}{(TP + FP)}$
- Recall = $\frac{TP}{(TP + FN)}$

You’re only guessing at these

You don’t know these

You can’t evaluate how well you did without Ground Truth!

Let’s analyze further: graph the other lengths too



Oh no!!!
what’s that???
?????

(Hint: It’s a FN)

New analysis shows:
www.example.com

Point 1: You’re not done yet!

- If you created the first filter and stopped:
  - You would be missing second order attacks FNs
  - You might be filtering things you shouldn’t FPs

Two problems: false positives and false negatives

	Before	During
IP Size == 540	2844	2960052

FP

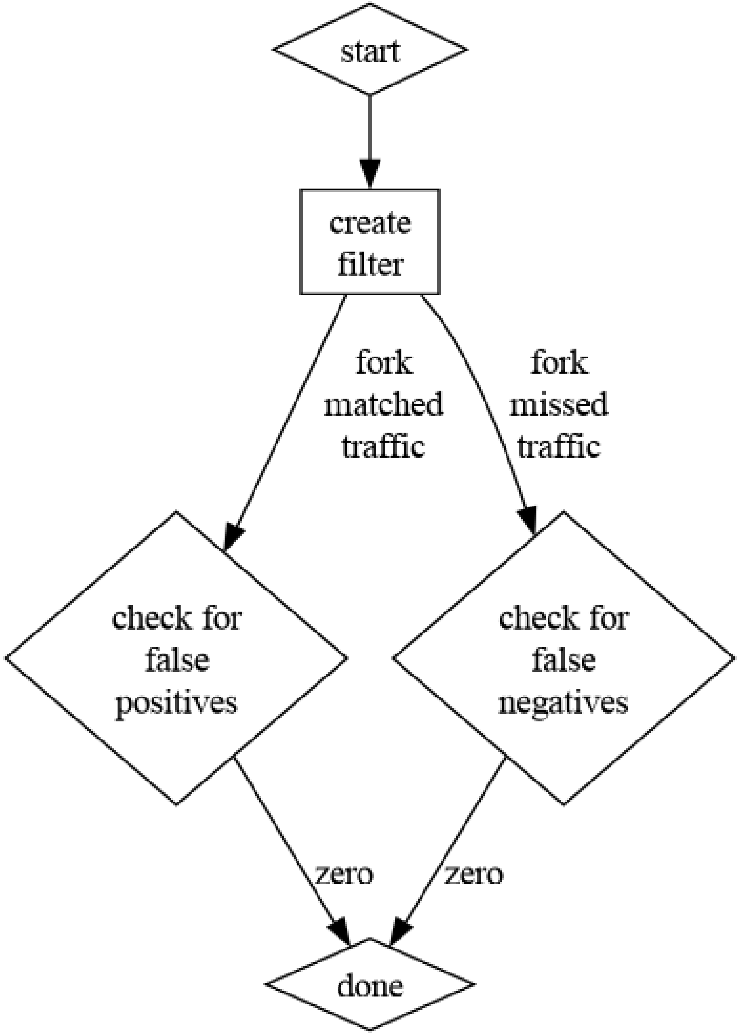
Some FPs???

Searching for missed attack traffic revealed:

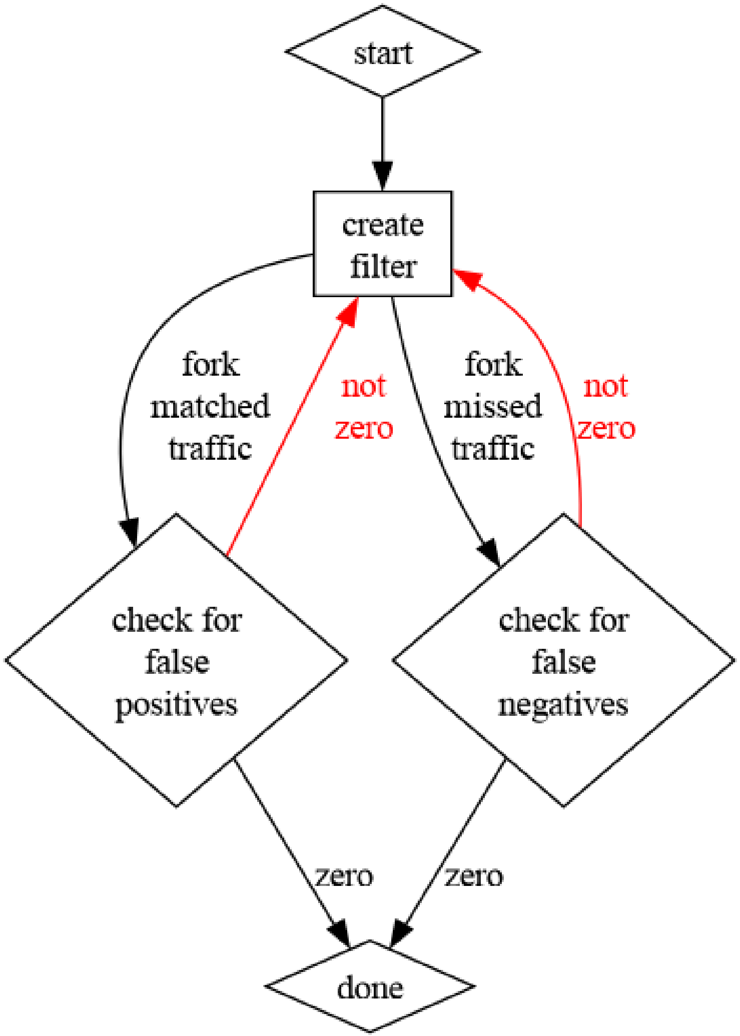
	Before	During
QName = www.example.com	259 New FP	787526

Some FPs???

Point 2: check both filtered and unfiltered traffic



Point 2: check both filtered and unfiltered traffic



Downside:

Every fork is 2x more work

Accuracy is an iterative process

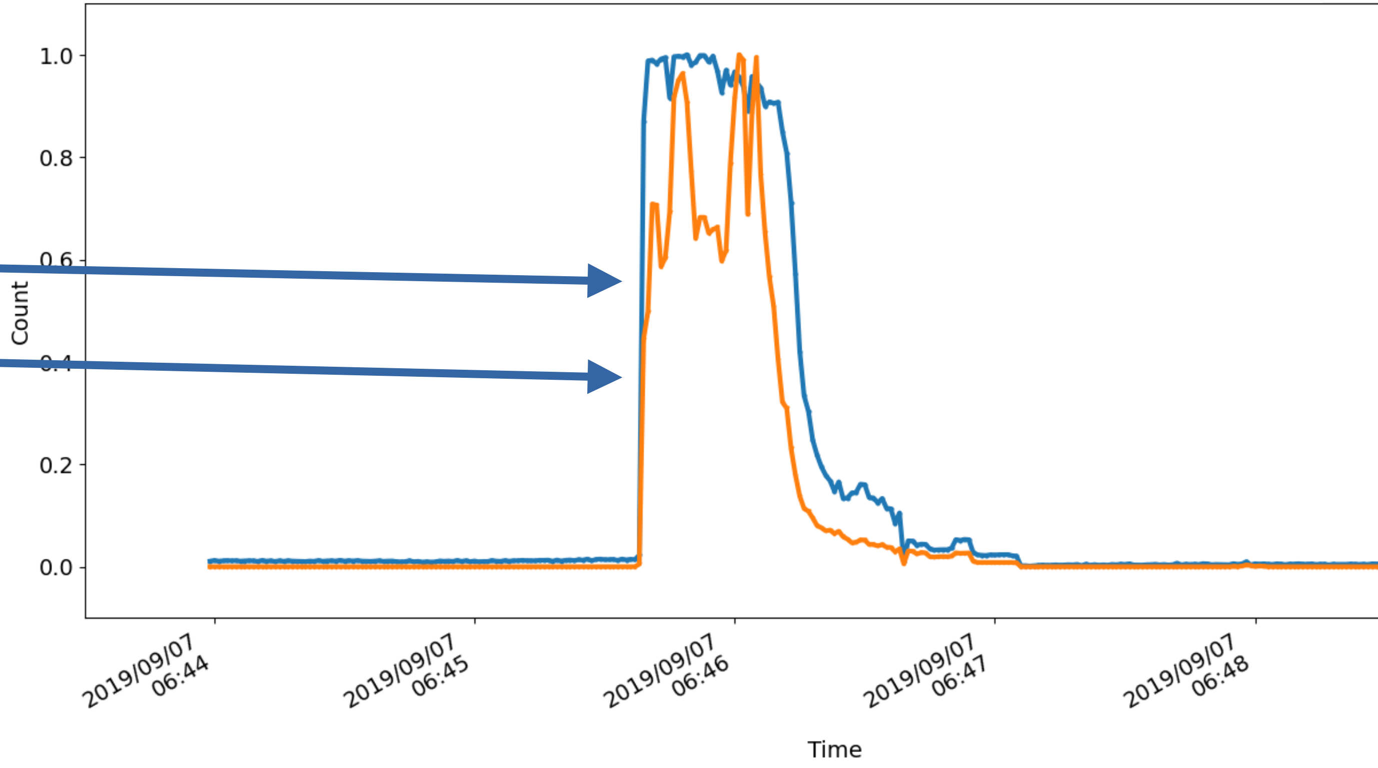
- Find FP
- Find FN
- Find FP
- Find FN
- Find FP
- …

Success requires smart, automated tooling

- Compare by eyesight is great, but….

- Look for other packet similarities
- Look for similar waveforms
- Look for similar edge-detection
- Compare against normal traffic loads
  - (you are recording these right?)

Normalized size=540 and www.example.com waveforms

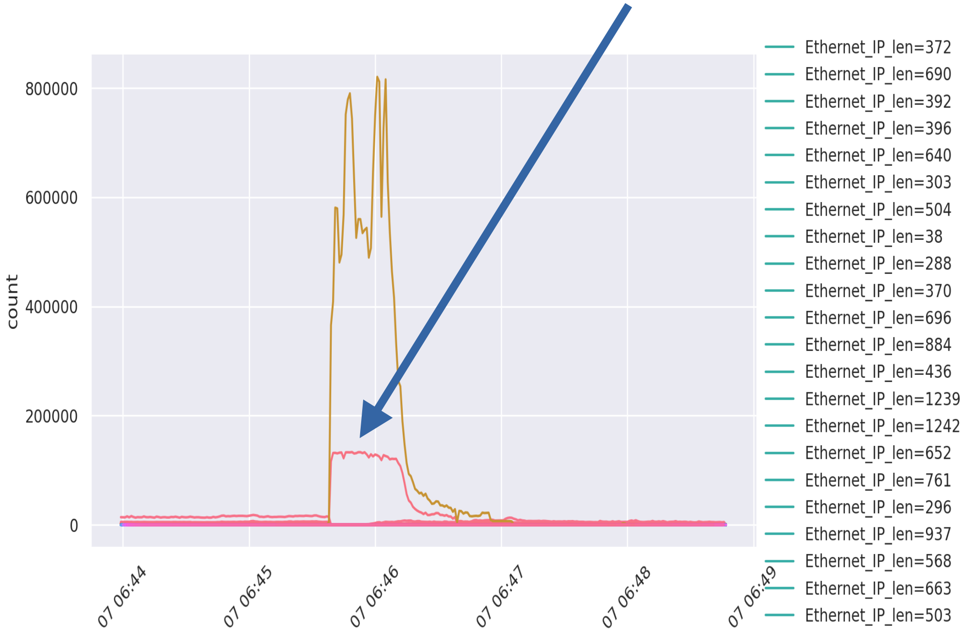


Point 3: trust but verify

The truth is: I’ve been lying to you

Because my tools lied to me

This was not www.example.com queries

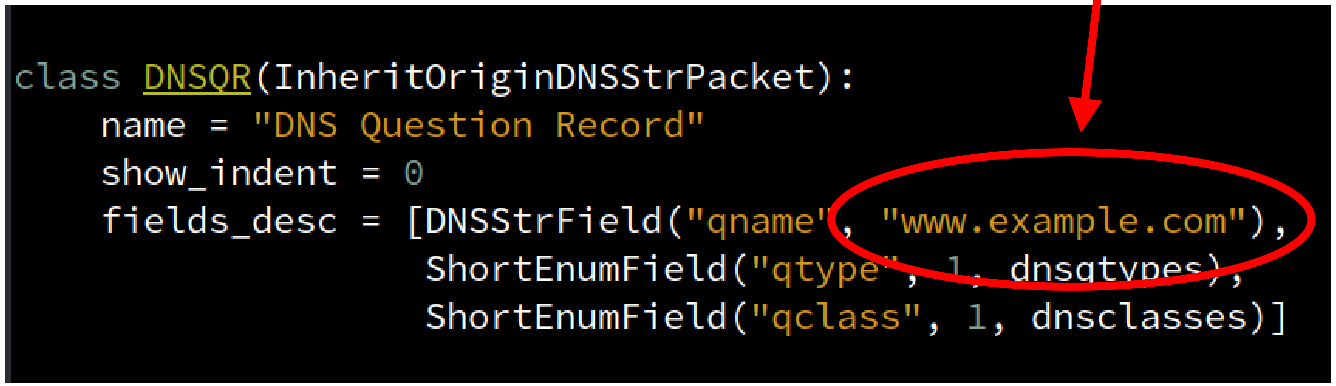


My bad

It was actually ICMP responses containing partial DNS packets

Their bad

From scapy.py:



```
class DNSQR(InheritOriginDNSStrPacket):
    name = "DNS Question Record"
    show_indent = 0
    fields_desc = [DNSStrField("qname", "www.example.com"),
                   ShortEnumField("qtype", 1, dnsqtypes),
                   ShortEnumField("qclass", 1, dnsclasses)]
```

Take-Aways

- 1. You’re not done *You’re never done*
- 2. Check your results *Both filtered and unfiltered*
- 3. Trust no one *Double check everything*

- Prioritize your findings: Hurting you vs hurt your clients
- Use multiple search methodologies, automation, ...
  - Volume, shape, time, edge cases, similarity analysis, etc

This is where I’m actively working