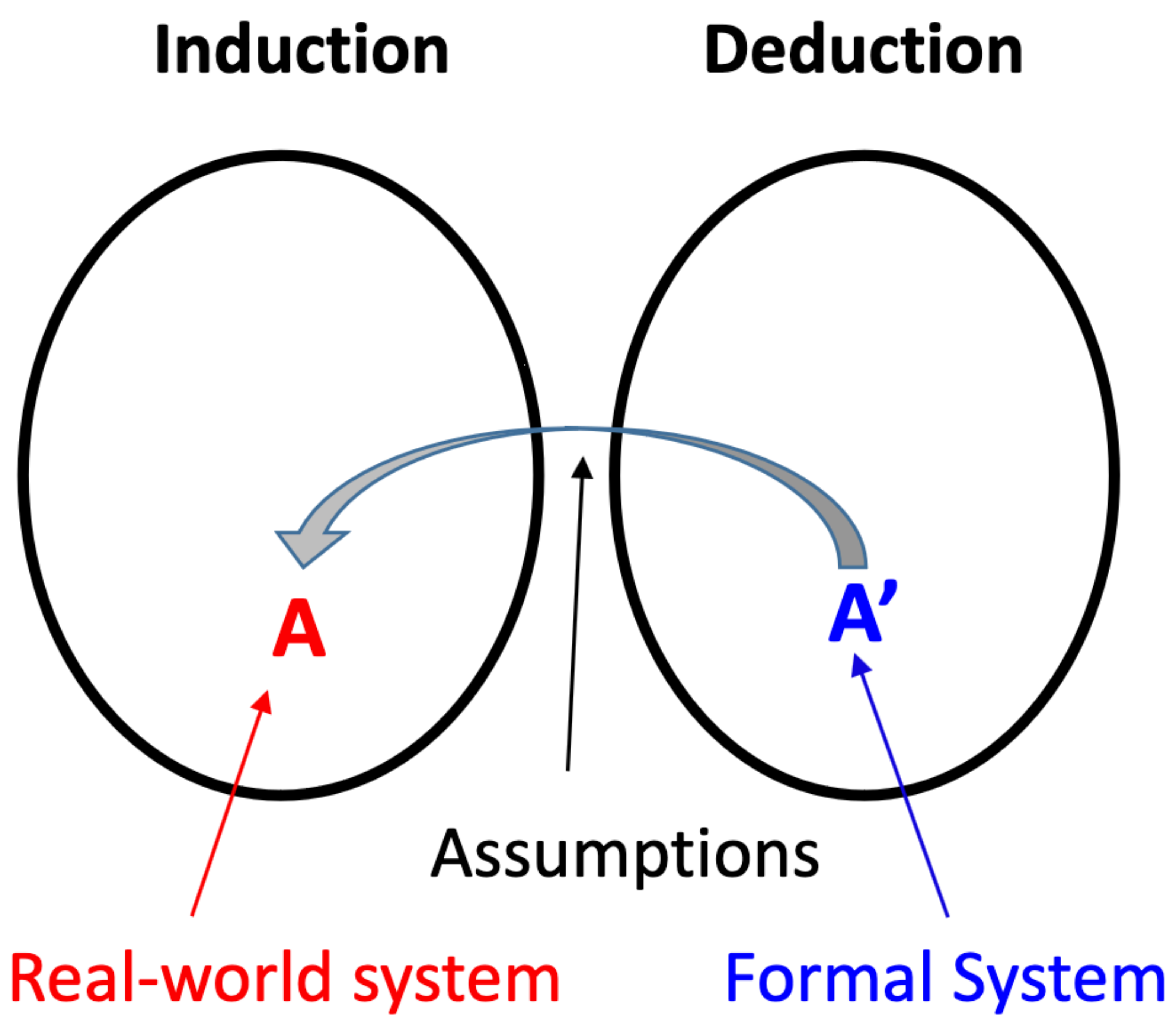
Measurement + Ethics

CS499/579 :: Empirical Computer Security

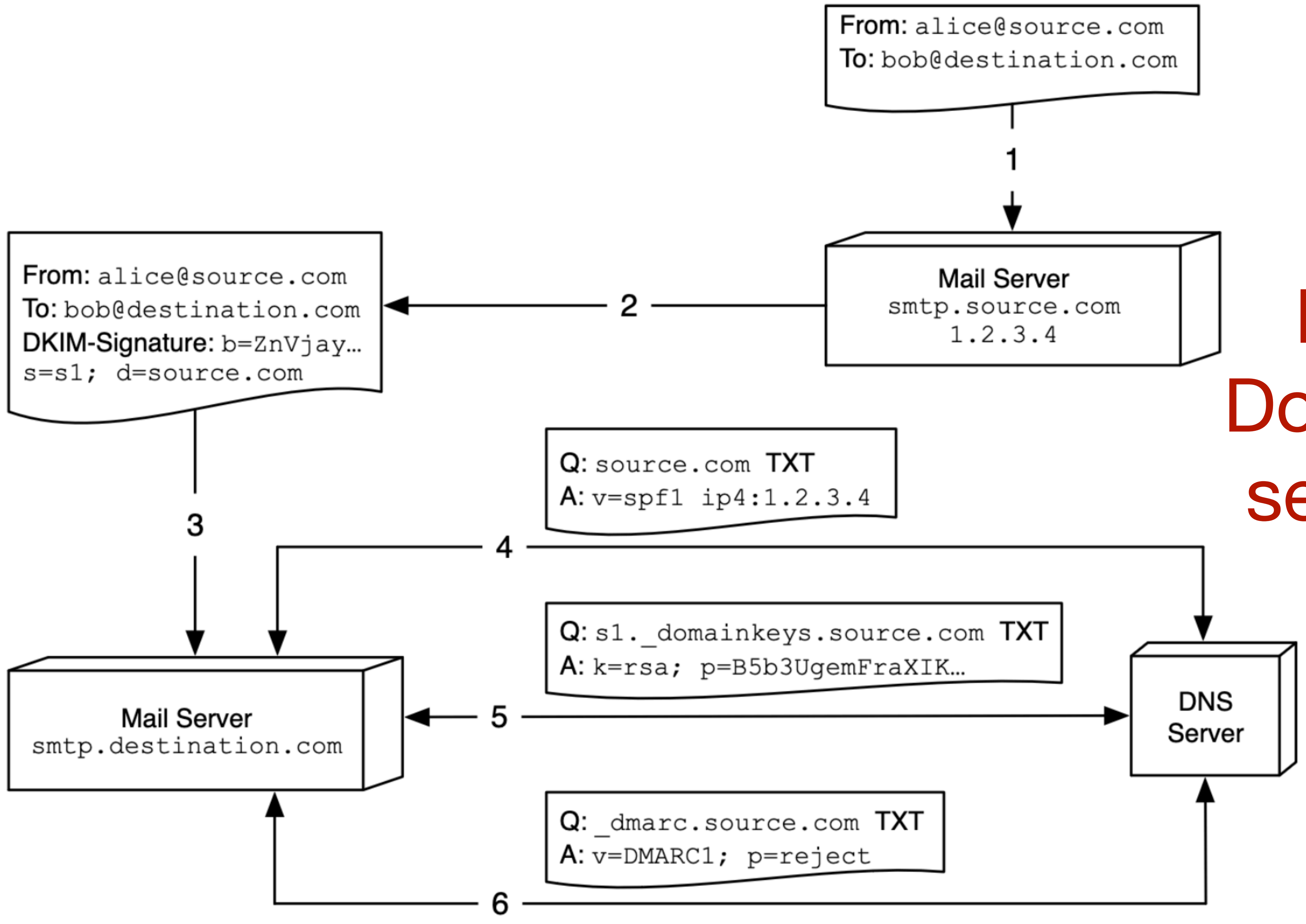
Zane Ma (he/him/his)

2023.10.04

From last class…



- In order to understand how computer systems actually work, we need to measure them (e.g., performance / security properties)



Is email secure?
Do people use email security protocols?
Used securely?

Scanning the Internet

- Prior to 2013, scanning the full internet was uncommon
- Why? (Think IPv4)

IPv4 header format

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version				IHL				DSCP						ECN		Total Length															
4	32	Identification																Flags			Fragment Offset												
8	64	Time To Live								Protocol								Header Checksum															
12	96	Source IP Address																															
16	128	Destination IP Address																															
20	160																																
⋮	⋮	Options (if IHL > 5)																															
56	448																																

EFF SSL Observatory: A glimpse at the CA ecosystem (2010)

3 months on 3 Linux desktop machines (6500 CPU-hours)

Census and Survey of the Visible Internet (2008)

3 months to complete ICMP census (2200 CPU-hours)

- 32-bit address! 2^{32} = ~4B destination IPs
- Scanning at 100 IPs / second would take 462 days

ZMap: Fast Internet-Wide Scanning and Its Security Applications

Zakir Durumeric
Michigan (now Stanford)

Eric Wustrow
Michigan (now UC Boulder)

Alex Halderman
Michigan

2013 USENIX

Introducing ZMap

An open-source tool that can port scan the entire IPv4 address space from just one machine in under 45 minutes with 98% coverage

With ZMap, an Internet-wide TCP SYN scan on port 443 is as easy as:

```
$ zmap -p 443 -o results.txt
34,132,693 listening hosts
(took 44m12s)
```

97% of gigabit Ethernet linespeed

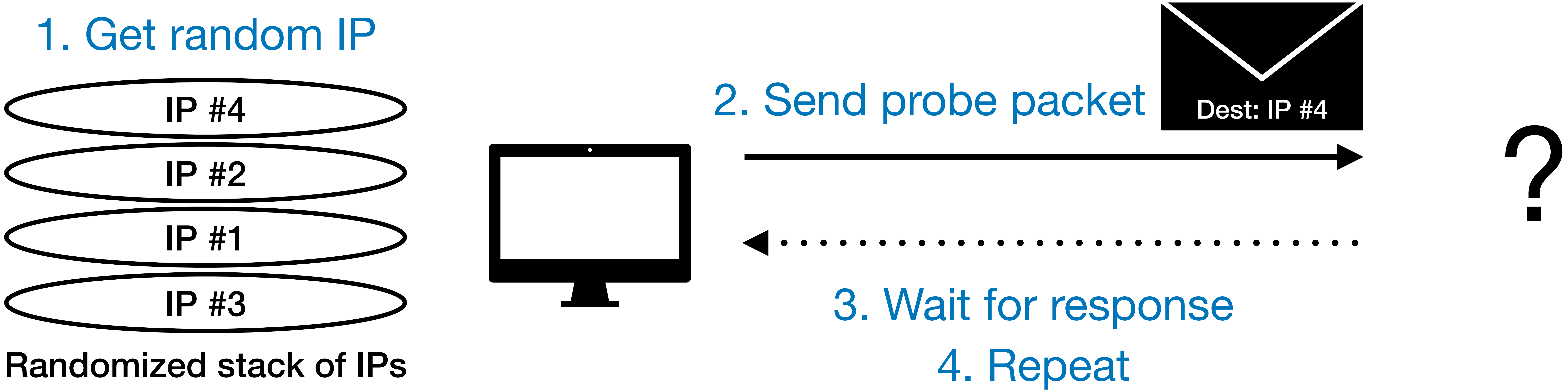
Weeks / months of scanning —> hours

How does it work?

What are the resource / performance costs?
How would you optimize this?

Naive way of scanning an IP address:

1. Make a randomized stack of all IP addresses
2. Send one packet to random destination (pop off the stack)
3. Wait - if response received, log IP + response payload; otherwise, timeout



How does it work?

Short answer: reduce / eliminate state associated with scanning!

In other words, reduce how much the scanner has to remember, so you don't need to wait for responses + you can minimize memory usage

1. Efficient random IP tracking: How can we scan all IPv4 addresses, randomly, without remembering all the ones we have already scanned?
2. Stateless scanning: How can we send out network requests without waiting for a response?

1. Efficient random IP tracking

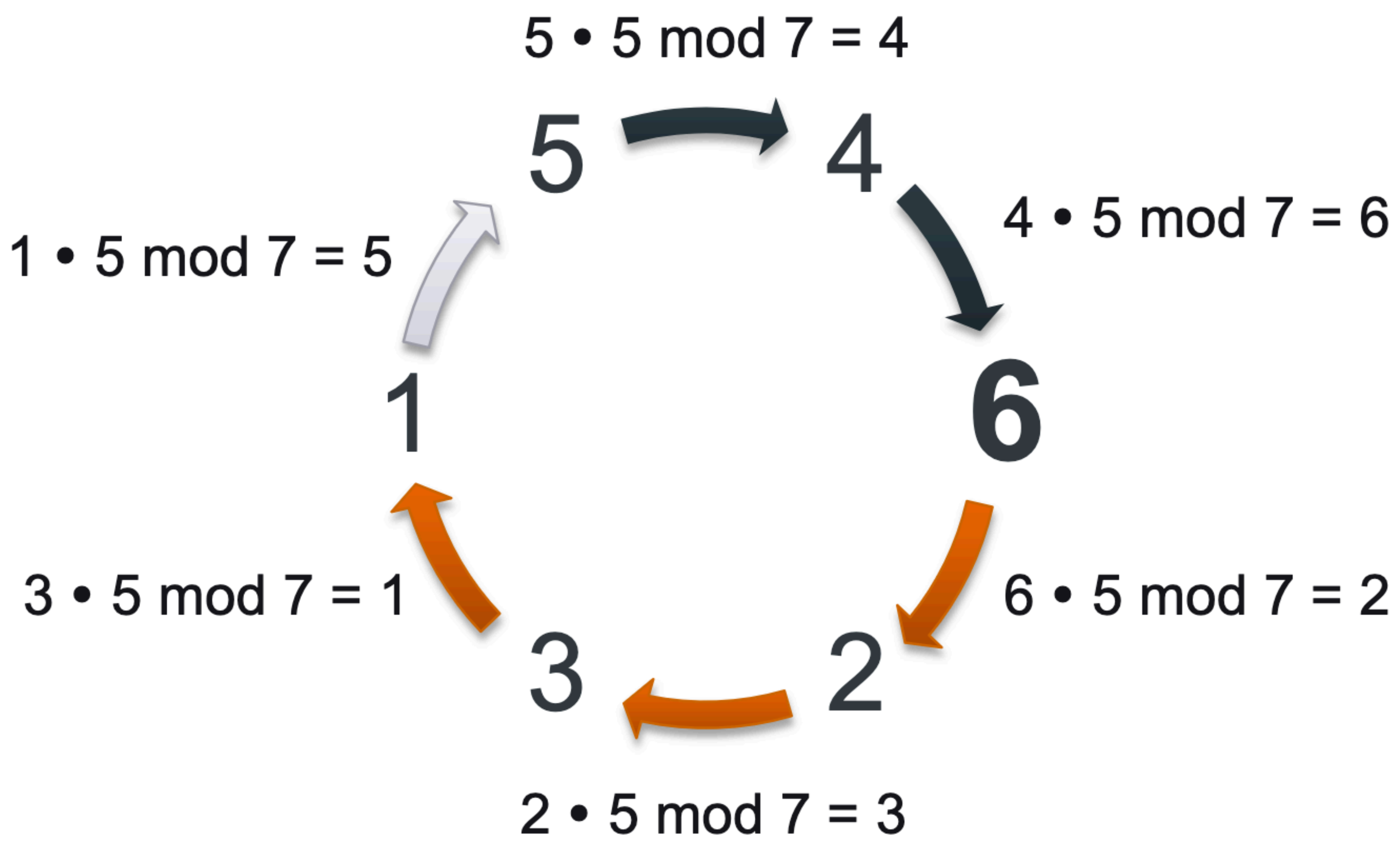
How can we scan all IPv4 addresses (equivalent to 4-byte unsigned integer), ~~randomly~~, without remembering all the ones we have already scanned?

Order them and keep track of:

1. Current IP address (e.g., 128.193.10.29)
2. Increment size (e.g., 1)
3. Starting point (e.g., 0 = 0.0.0.0)

1. Efficient random IP tracking

How can we scan all IPv4 addresses (equivalent to 4-byte unsigned integer), randomly, without remembering all the ones we have already scanned?



Fancy math ordering = multiplicative group of integers modulo *p*, only track:

1. Current location (current IP)
2. Primitive root (increment size)
3. First address (starting/end point)

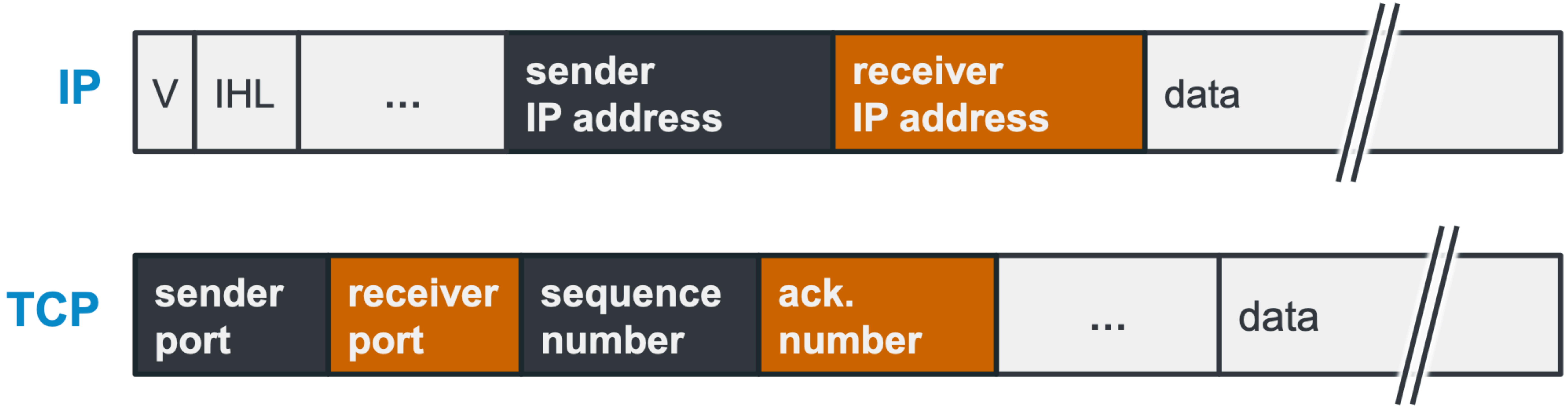
Each primitive root is a different random* ordering

2. Stateless scanning

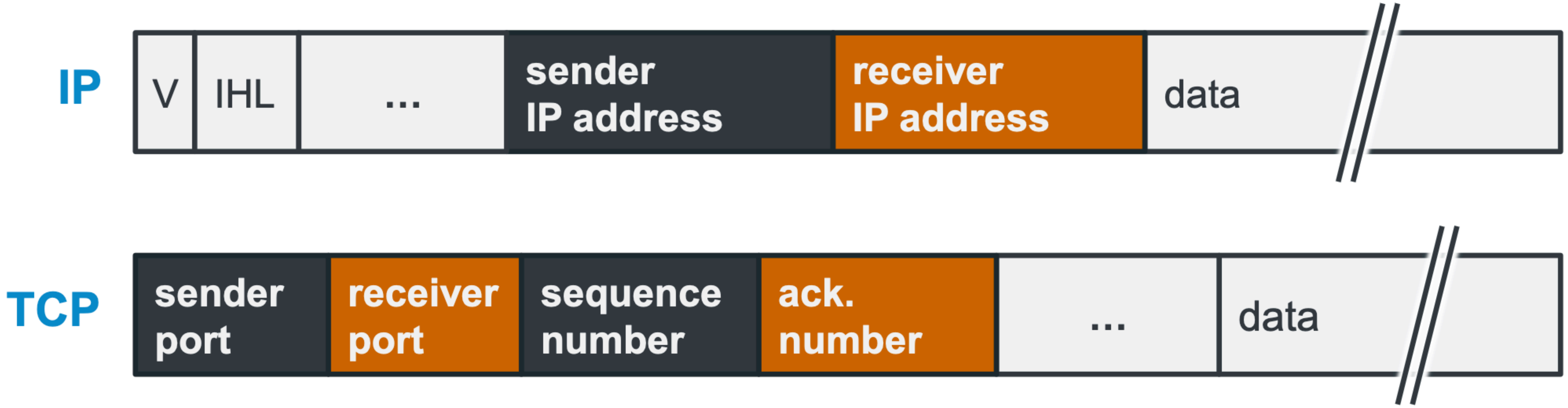
How can we send out network requests without waiting for a response?

But first: why do we need to wait for responses anyways? Random background noise - unsolicited packets are common

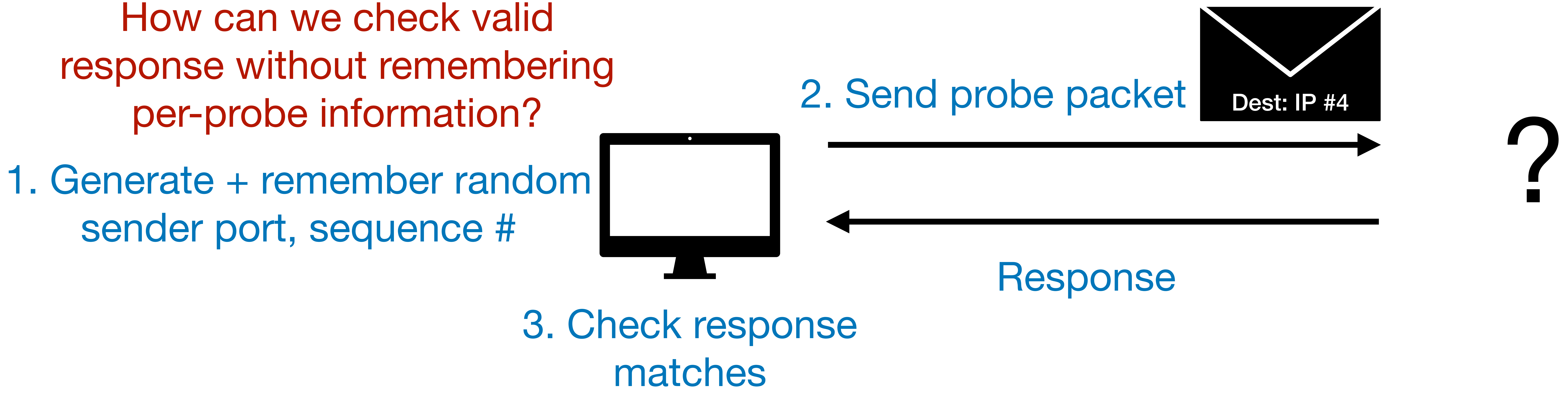
How do we normally distinguish between background noise packets and response packets? Look at response fields predictably related to probe packet



2. Stateless scanning



How can we check valid response without remembering per-probe information?



2. Stateless scanning

1. Use the same sender port and initial sequence number every time

    2^16 (16-bit sender port) * 2^32 (32-bit sequence number) uniqueness

2. Per-probe uniqueness: Set the port + sequence number based on the target IP address

    2^16 * 2^32 * 2^32 (32-bit target IP) uniqueness

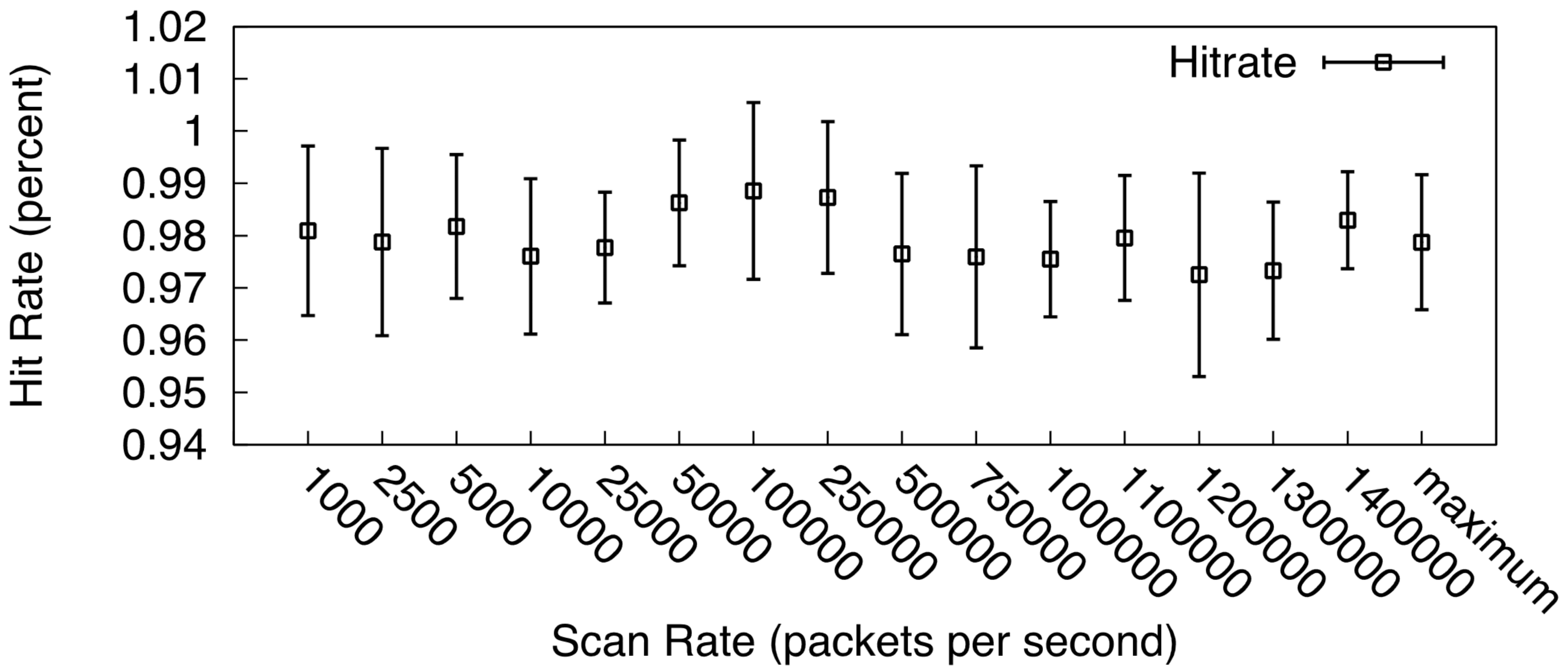
Downside: can't distinguish between responses triggered by previous scans

3. Per-probe + per-scan uniqueness (what ZMap does): set port + sequence number based on Message Authentication Code (MAC) computed over the target IP address, using a per-scan key

Scanning Performance

How fast is too fast?

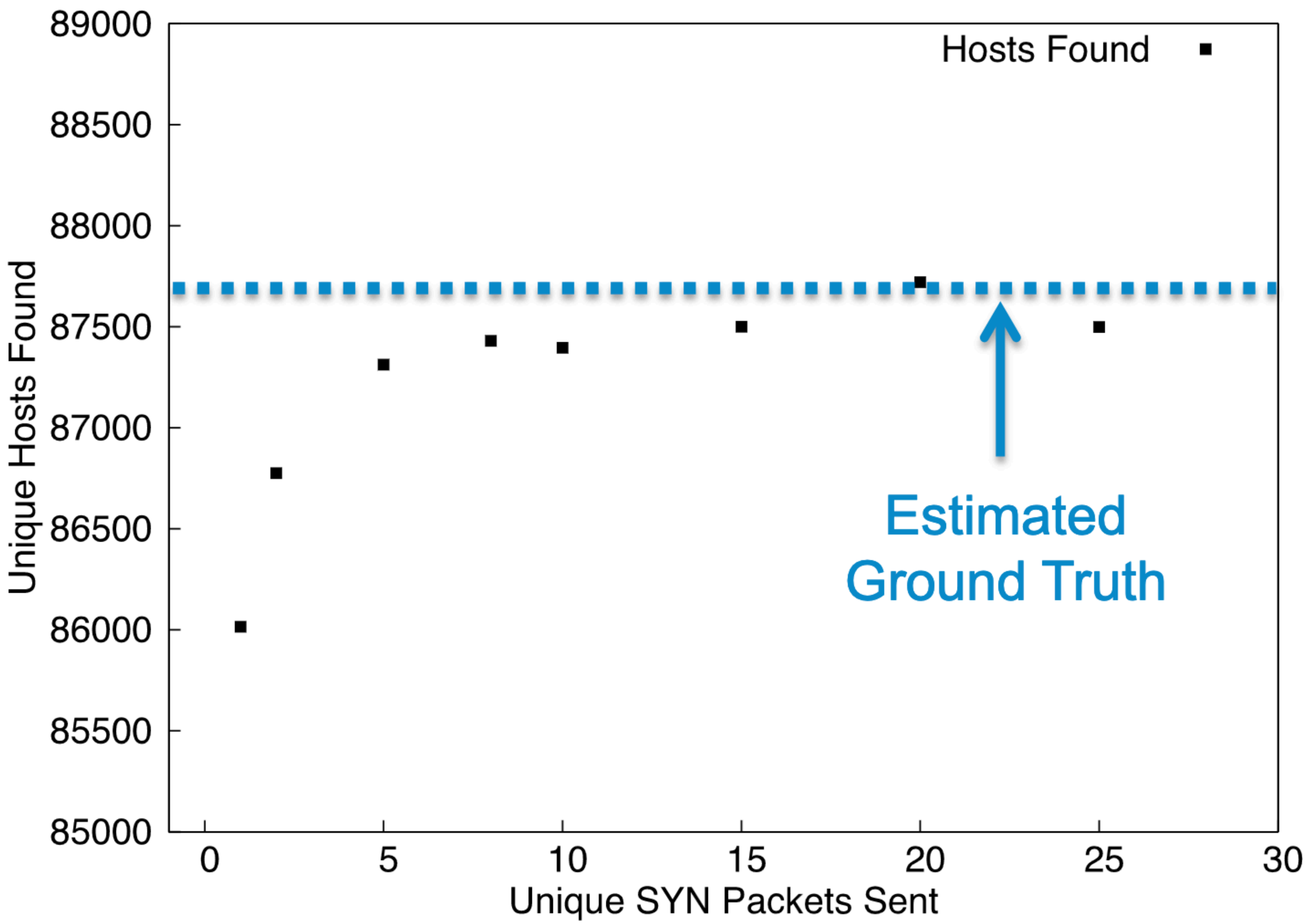
No correlation between hit-rate and scan-rate. Slower scanning does not reveal additional hosts



Oregon State University

Scanning Coverage

Is one probe packet per destination IP sufficient?



We expect an eventual plateau in responsive hosts, regardless of additional probes.

Scan Coverage

1 Packet: 97.9%

2 Packets: 98.8%

3 Packets: 99.4%

Comparison with Nmap

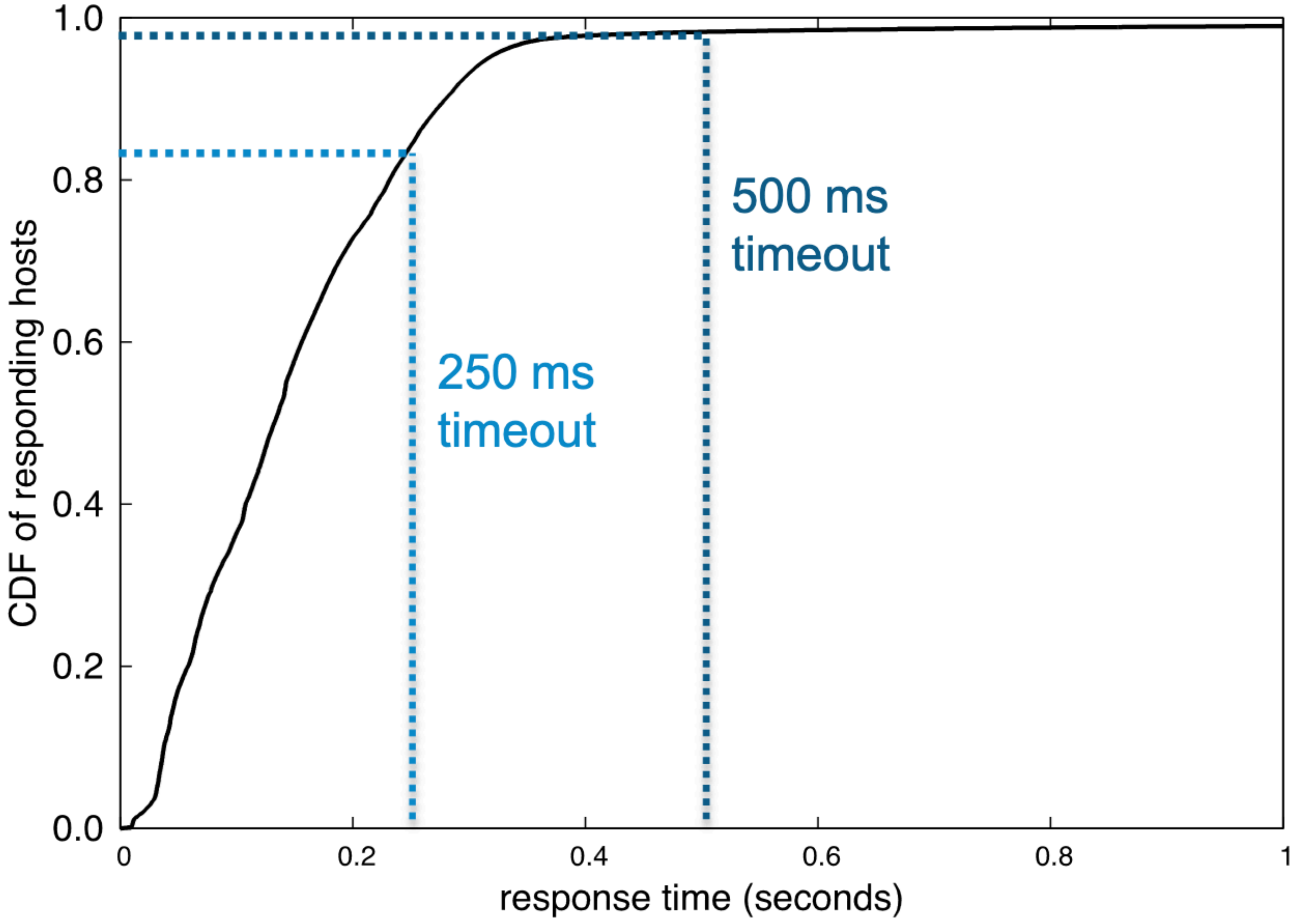
	Normalized Coverage	Duration (mm:ss)	Est. Internet Wide Scan
Nmap (1 probe)	81.4%	24:12	62.5 days
Nmap (2 probes)	97.8%	45:03	116.3 days
ZMap (1 probe)	98.7%	00:10	1:09:35
ZMap (2 probes)	100.0%	00:11	2:12:35

ZMap is capable of scanning more than 1300 times faster than the most aggressive Nmap default configuration ("insane")

Surprisingly, ZMap also finds more results than Nmap

Probe Response Times

Why does ZMap find more hosts than Nmap?



Response Times

250 ms:	**< 85%**
500 ms:	**98.2%**
1.0 s:	**99.0%**
8.2 s:	**99.9%**

Statelessness leads to both higher performance ***and*** increased coverage.

Ethics of Active Scanning

Ethics requires the balancing of harms with benefits

What are potential negative consequences of scanning? Potential mitigations?

- Overwhelming traffic that slows down / takes down network
  - Randomize / spread out probes to a given network
- Sysadmins believe they are under attack + waste resources responding
  - Signal benign nature over HTTP, reverse DNS entries
- Access or modify sensitive or private user data
  - Test locally beforehand; only collect what is needed; remove sensitive data
- Other unforeseen / unknown issues
  - Provide contact info and honor requests to be excluded from future scans

Meta: Do we need to scan the full internet?

- Depends what we are trying to find

<u>When we **don't** need to scan everything</u>

Determining what percent of websites use HTTPS

Collecting different types of phishing websites to categorize strategies

Make sure to get a random or representative sample!

<u>When we **do** need to scan everything</u>

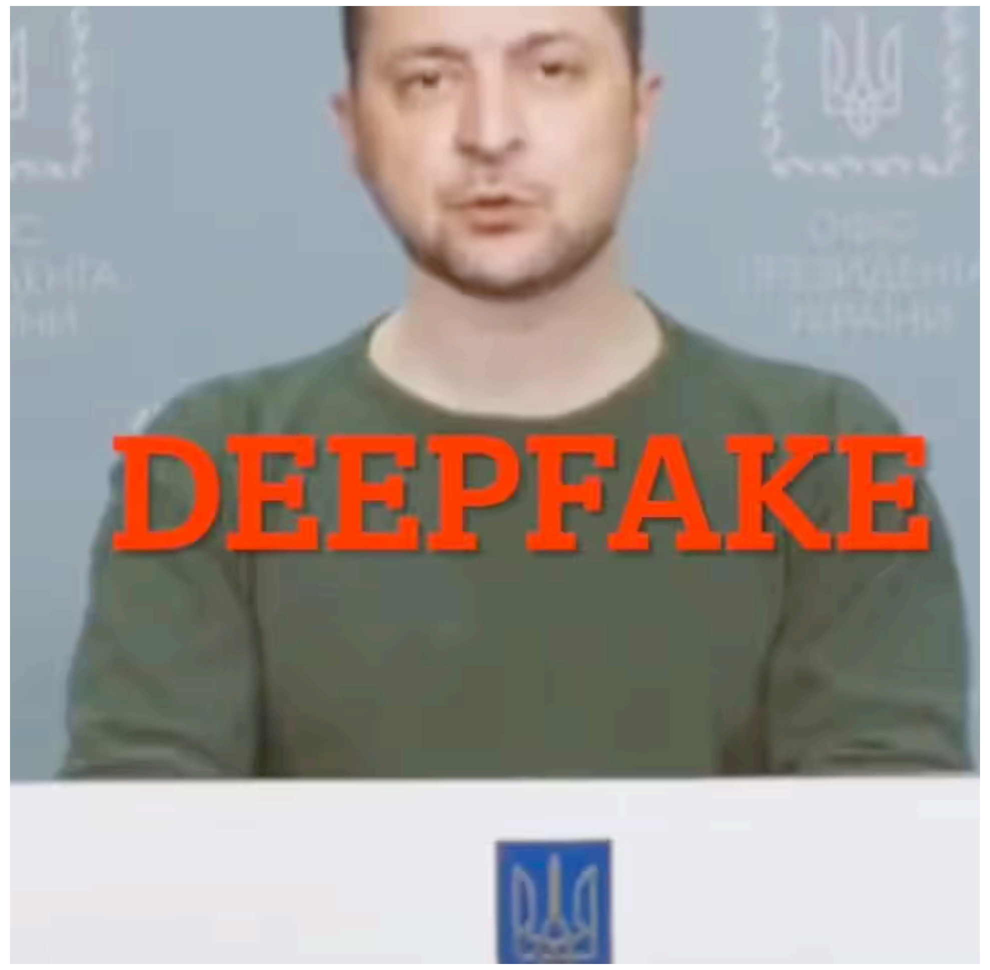
Finding really rare (but possibly very impactful) phenomenon

Notifying insecure websites about how to patch vulnerabilities

When we don't feel like doing statistics

Machine Learning

- Step 1: Collect lots of data
- Step 2: Analyze data to see current state of security
- Step 3: Use ML for prediction: perform attacks, automate defenses, etc.
- Step 4: …

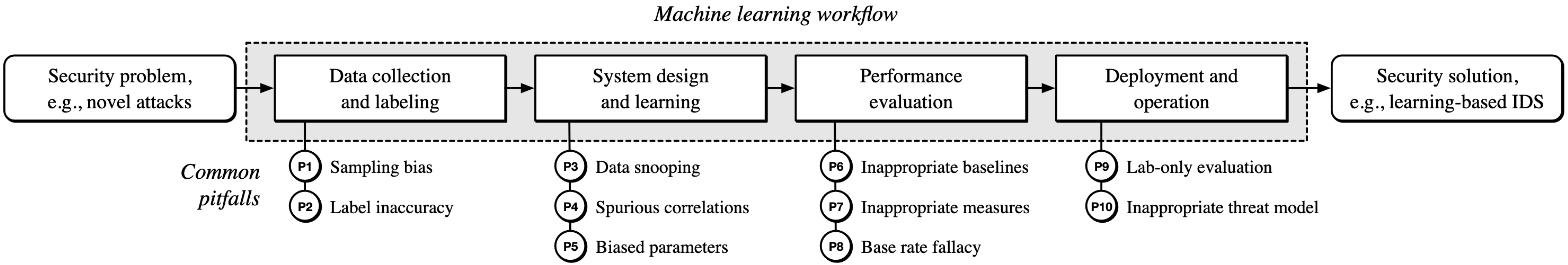


Dos and Don’ts of Machine Learning in Computer Security

Daniel Arp (Technische Universität Berlin) et al.

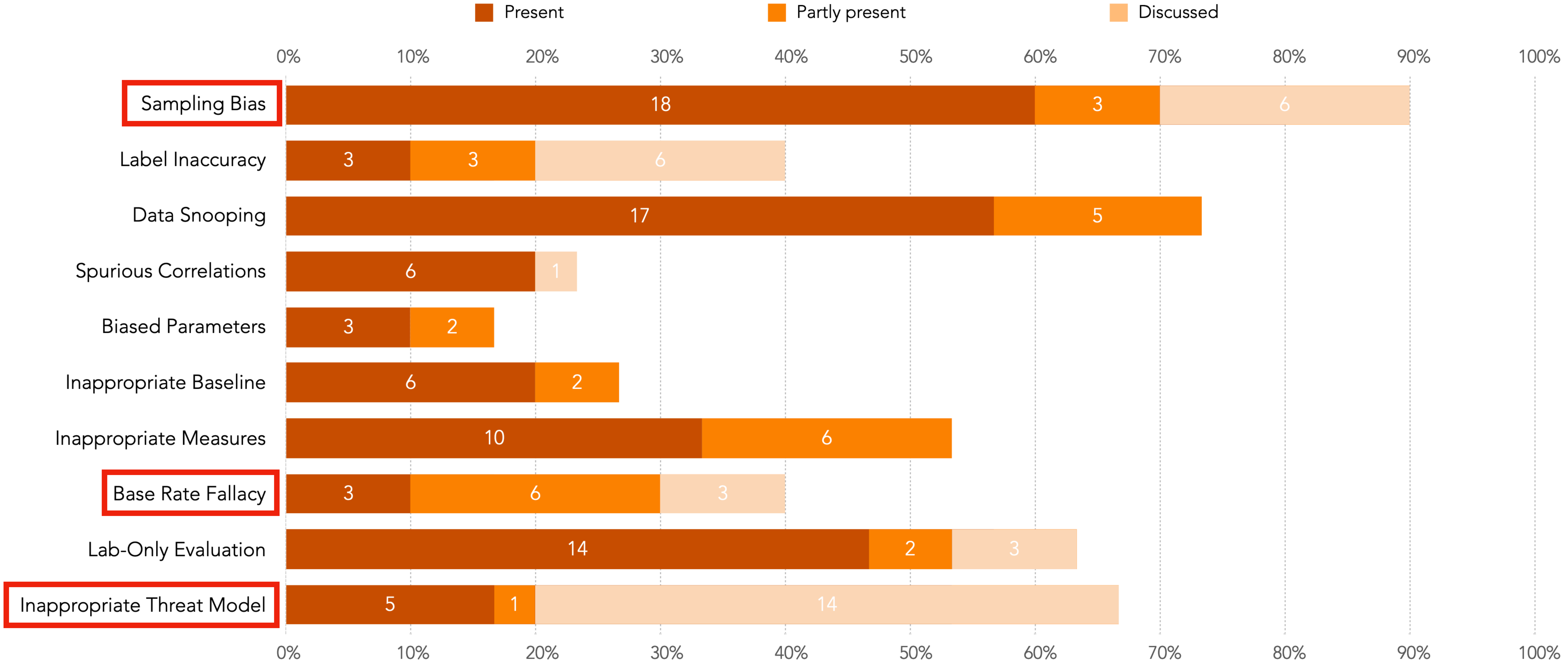
2022 USENIX

Machine Learning Workflow



Machine Learning Flaws

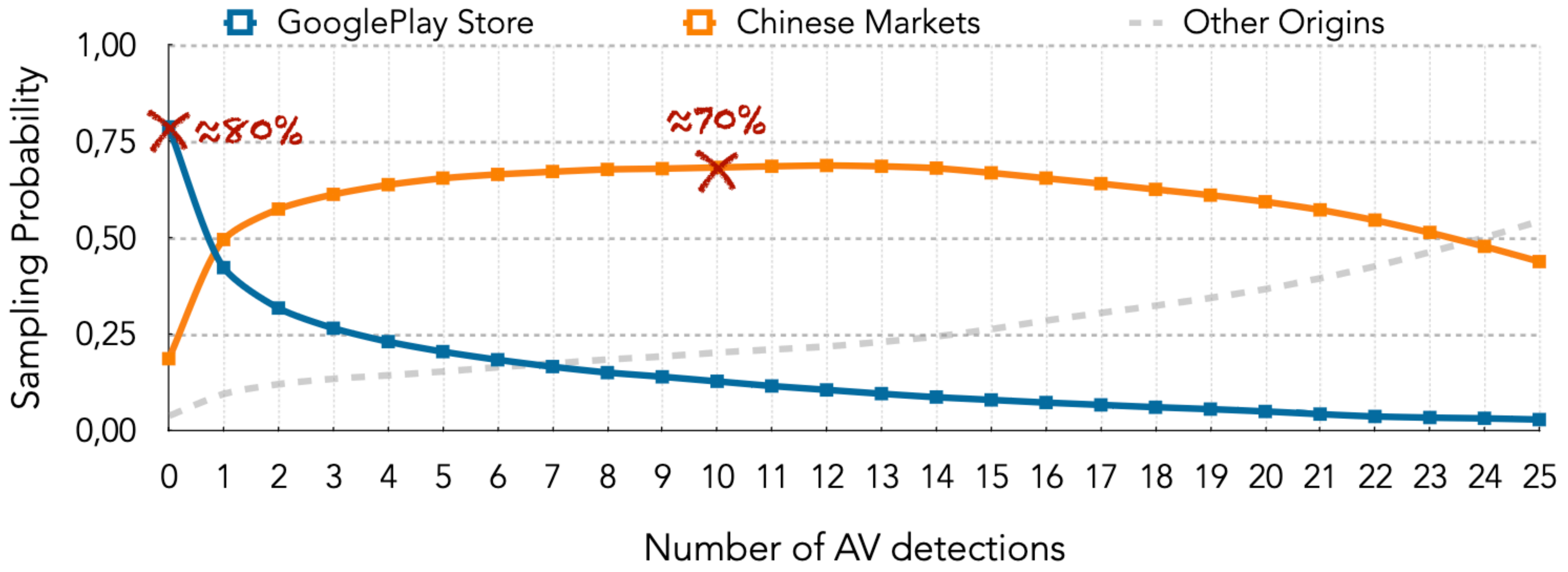
Measured 30 top security papers



Sampling Bias

"The collected data does not sufficiently represent the true data distribution of the underlying security problem"

When the training data for a model does not represent the intended use case

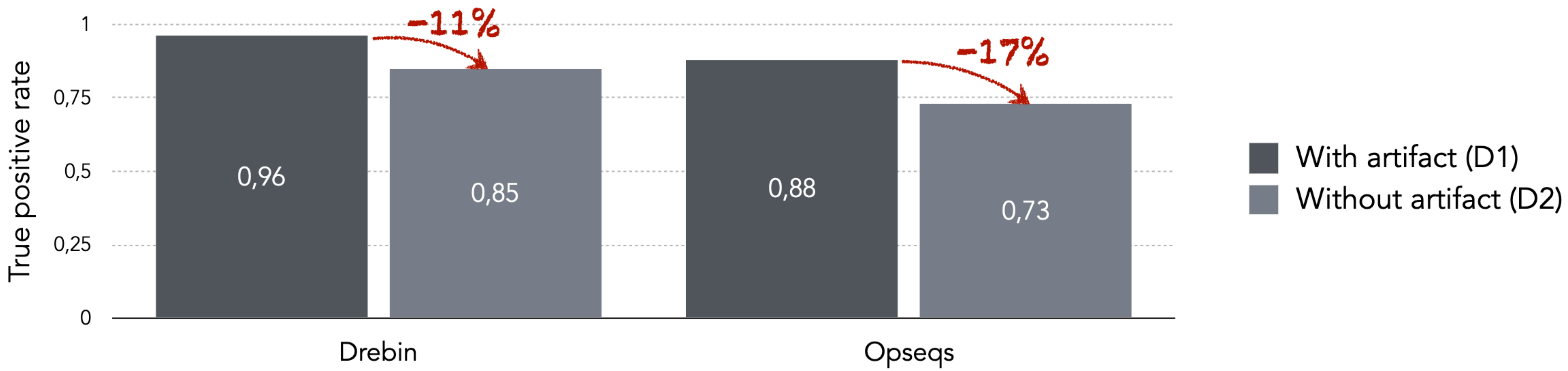


How should we collect benign (0 AV detection) and malicious (10+ AV detections) datasets?

Sampling Bias

What prior study did: randomly sample from all benign apps and all malicious apps to generate training / test data

Outcome: the URL “play.google.com" is one of the top distinguishing features for malware detection (Problem #4: Spurious correlations)



Base rate fallacy

Assume: medical test with 5% false positive rate and no false negative rate

How good is this test when the base rate of infection in the population is 40%?

400 infected / 430 positive = 93% confident

Number of people	Infected	Uninfected	Total
Test positive	*400* (true positive)	*30* (false positive)	430
Test negative	0 (false negative)	570 (true negative)	570
Total	400	600	**1000**

How good is this test when the base rate of infection in the population is 2%?

20 infected / 69 positive = 29% confident

Number of people	Infected	Uninfected	Total
Test positive	*20* (true positive)	*49* (false positive)	69
Test negative	0 (false negative)	931 (true negative)	931
Total	20	980	**1000**

https://en.wikipedia.org/wiki/Base_rate_fallacy

Base rate fallacy

A tendency to ignore the base rate (across a full population) in favor of the accuracy of an individual test

Takeaway: Low positive rate (FPR) is super critical for security systems that handle large amounts of data, and base rate is relatively low (e.g., email spam, malicious network packets)

Especially when cost of false positive is high! For example, blocking a legitimate email, or requiring manual analysis of a (not-actually) malicious network signal

https://en.wikipedia.org/wiki/Base_rate_fallacy

Improper threat model

Building a ML model is not enough to counter a threat - it's possible, often trivial, to break machine learning models.

Example: model for code authorship, 95% accuracy - can reveal relationships between malware, potential cheating / copying for assignments

Attack: removing unused code decreased code attribution accuracy by 48%

How to mitigate? Think like an attacker! Take Prof. Sanghyun Hong's class, CS499/579, AI539 :: Trustworthy Machine Learning

TODOs for you

Specify presentation preferences by **9PM tonight**. Sign-up link on the syllabus at https://empirical-security.net/syllabus

> I will send out presentation + reading (which 1 of the 2 papers to read for each class) assignments tomorrow morning on Canvas

First paper reading + questions will be due by 6PM **Tuesday, October 10th**.

Create a project team by **Friday, October 6th**. Reach out if you need help