Who Filters the Filters:

Understanding the Growth, Usefulness and Efficiency of Crowdsourced Ad Blocking

Pete Snyder – Brave Software
Antoine Vastel – University of Lille / INRIA
Ben Livshits – Brave Software / Imperial College London

The talk in a slide…

- **Many web and privacy tools use crowdsource lists**

- **How these lists are maintained is poorly understood**
  Who decides what goes in? What comes out? What exceptions exist? etc…

- **Web measurement of EasyList**
  - Most popular list
  - Mostly “dead weight”, 90.16% of rules unused
  - 10k website measurement over 2+ months → practical optimizations
  - How do advertisers & trackers respond?

Overview

- **Context and Background**
  What, why and how of EasyList

- **Methodology**
  Web scale measurement over two months

- **Measurement Results**
  Whats used and unused, rule lifecycle, how do trackers respond, etc?

- **Applications**
  Mobile and extension optimizations

- **Discussion and Conclusion**

Overview

- **Context and Background**
  What, why and how of EasyList

- **Methodology**
  Web scale measurement over two months

- **Measurement Results**
  Whats used and unused, rule lifecycle, how do trackers respond, etc?

- **Applications**
  Mobile and extension optimizations

- **Discussion**

Context and Background

- EasyList is the most popular list
- Targets ads and tracking from advertisers
- Text format, RegEx-like format
- EasyList is a large project, 15 years of contributions
- Targets English and “global” sites
- Many different rules, acting on different layers

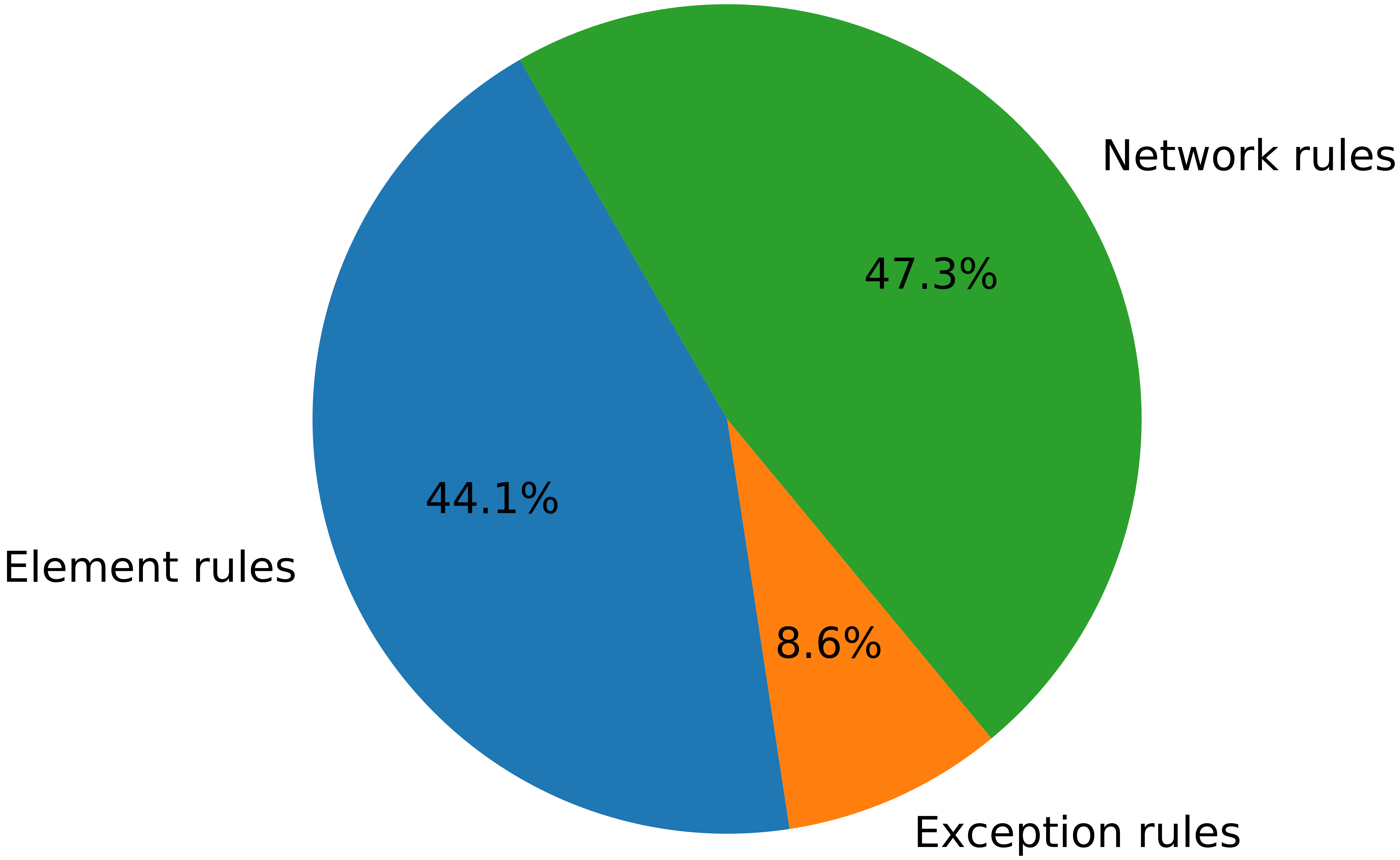
EasyList Types of Rules

- **Network rules**
  `||example.org/ad`

- **Element rules**
  `site.com###iframe`

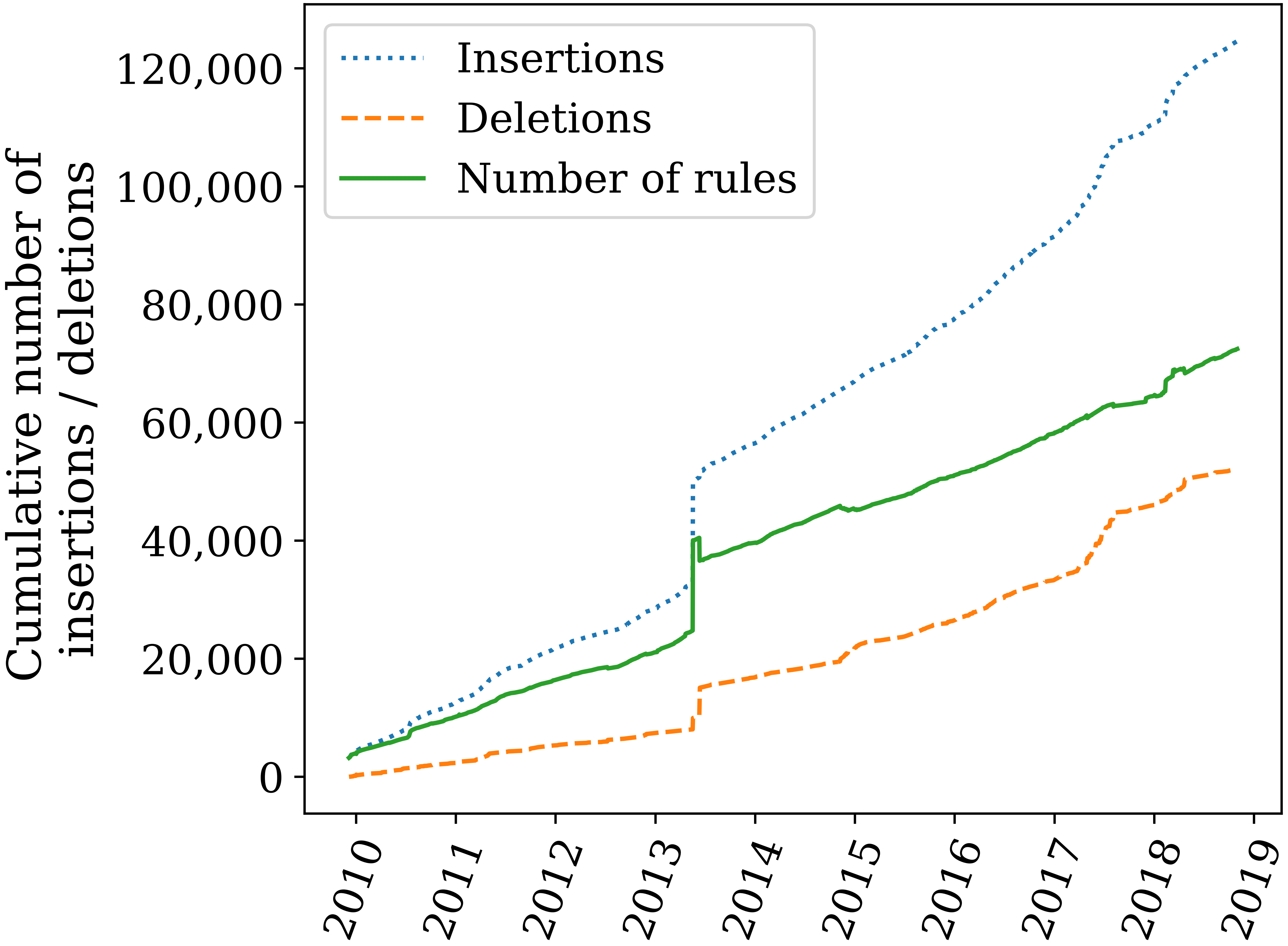
- **Exception rules**
  @@||example.org/advice

- **Filters**
  `||example.org^script`



EasyList Over Time

- 2005: Started by Rick Petnel
- 2009: Moves to GitHub
- 2013: Merges with "Fanboy's list"
- 2019: Reaches 72,469 rules
- 2020 (May): Shrinks to ~69k

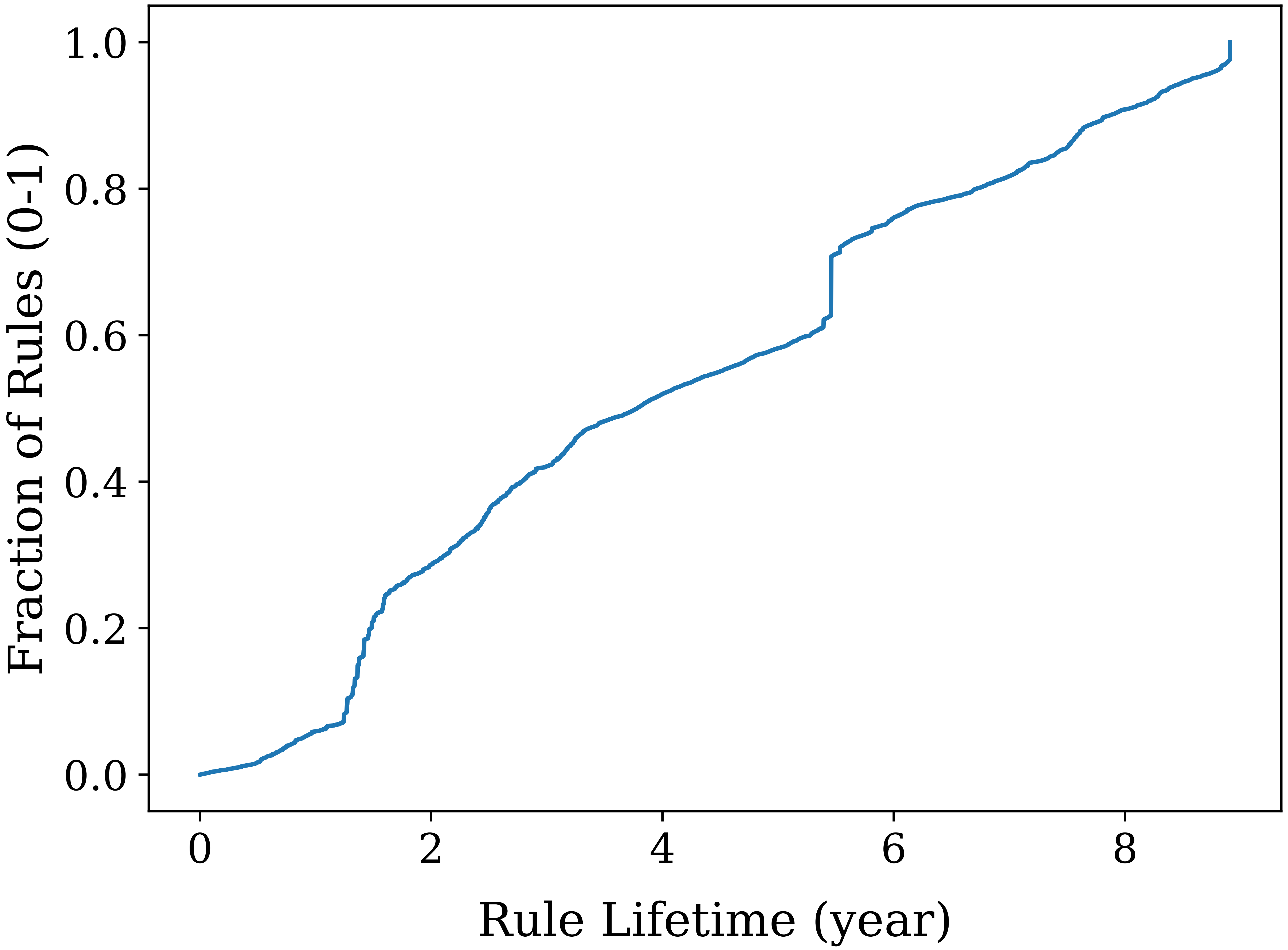


Rule “Life Cycle”

- Measurement of how long a rule stays in the List

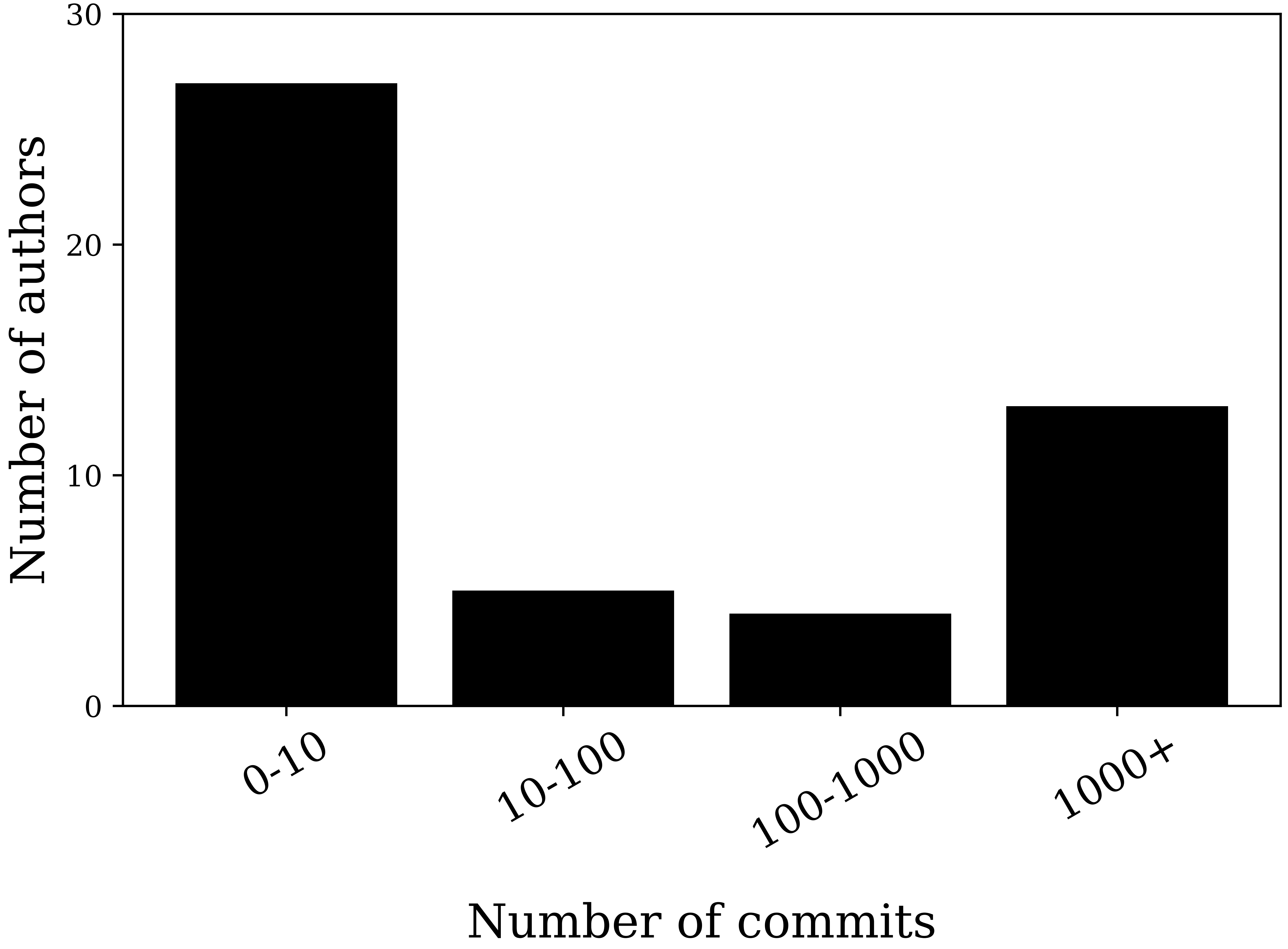
- Measured using git commit history

- ~50% rules remain for > ~4 years



Who Contributes To EasyList

- **From forum and GitHub**
- **Five main contributors**
  76.87% of commits
- **Many small contributors**
  65.3% of contributors made <= 100 commits



Also in the Paper…

- How commit history was tracked across project structure changes
- How often commits are made
- How other tools use EasyList
- Tooling details

Overview

- **Context and Background**
  What, why and how of EasyList

- **Methodology**
  Web scale measurement over two months

- **Measurement Results**
  Whats used and unused, rule lifecycle, how do trackers respond, etc?

- **Practical Applications**
  Mobile and extension optimizations

- **Discussion and Conclusion**

Measurement Goals

- **Broad Goal:** Understand how EasyList and the web interact
- **Sub Goals:**
  - How is “rule usefulness” distributed?
  - Relationship between rule age and rule usefulness?
  - How to advertisers respond to being listed?

Methodology

- **Instrument a browser:**
  Record all network requests when visiting a page

- **Representative automated crawl**
  Both popular and unpopular websites

- **Apply EasyList to crawl data:**
  Determine what would be blocked if that day’s EasyList was applied

Browser Instrumentation

- **Stock Chromium:**
  Current stable version of Chromium at time of measurement

- **Puppeteer automation:**
  Record all URLs fetched, along with response type, hash and body size

- **Passive instrumentation:**
  No changes to page loading or resource requesting

- **No measurement of page contents:**
  Omitted measurements of element hiding rules

Representative Automated Crawl

- **Web domain selection:**
  “Popular": Alexa 5k
  “Unpopular”: Random selection from Alexa 5,000-1m

- **Page selection:**
  Measured landing page, and three same-eTLD+1 links

- **Measurement times:**
  - Every day for 74 days
  - Measured each page for 30 seconds

- **Passive measurement:**
  No changes to page loading or resource requesting

https://cnn.com

30 sec

https://cnn.com

30 sec

```
<a href=“https://advertiser.com">
<a href=“https://cnn.com/page1”>
<a href=“https://othersite.org”>
<a href=“https://cnn.com/page3">
<a href=“https://neat.advertiser.com”>
<a href=“https://cnn.com/page2">
<a href=“https://youtube.com">
<a href=“https://cnn.com/page5">
…
```

https://cnn.com

30 sec

```
~~<a href=“https://advertiser.com">~~
<a href=“https://cnn.com/page1”>
~~<a href=“https://othersite.org”>~~
<a href=“https://cnn.com/page3">
~~<a href=“https://neat.advertiser.com”>~~
<a href=“https://cnn.com/page2">
~~<a href=“https://youtube.com">~~
<a href=“https://cnn.com/page5">
…
```

https://cnn.com

30 sec

~~<a href=“https://advertiser.com">~~

✅ <a href=“https://cnn.com/page1”>

~~<a href=“https://othersite.org”>~~

<a href=“https://cnn.com/page3">

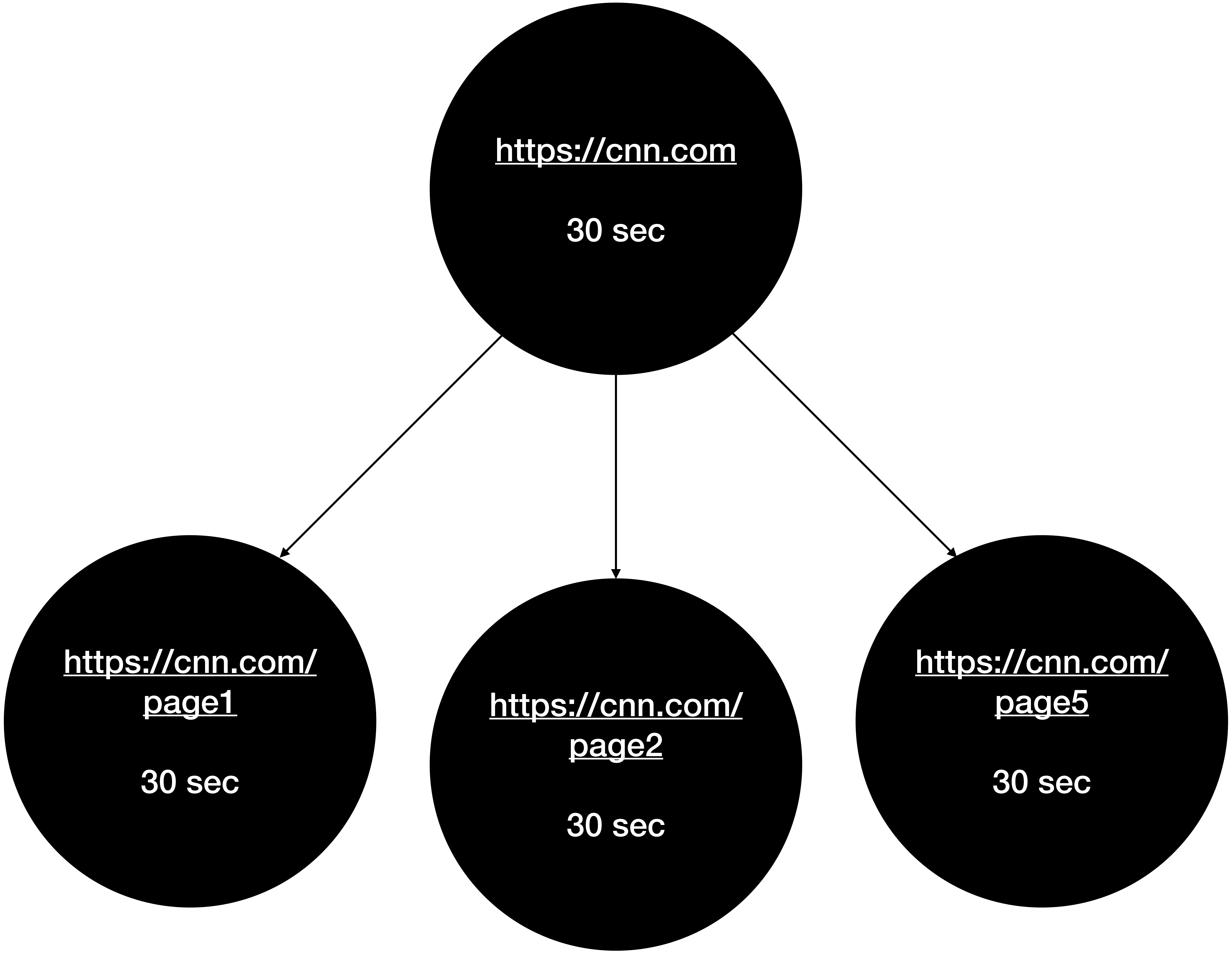
~~<a href=“https://neat.advertiser.com”>~~

✅ <a href=“https://cnn.com/page2">

~~<a href=“https://youtube.com">~~

✅ <a href=“https://cnn.com/page5">

…


https://cnn.com
30 sec
https://cnn.com/page1
30 sec
https://cnn.com/page2
30 sec
https://cnn.com/page5
30 sec

Applying EasyList to Crawl Data

Fetched Page	**Requested Sub-Resources**	**EasyList Rule**

Applying EasyList to Crawl Data

Fetched Page	Requested Sub-Resources	EasyList Rule
`https://cnn.com`		

Applying EasyList to Crawl Data

Fetched Page

https://cnn.com

Requested Sub-Resources

- https://cnn.com/header.png
- https://cnn.com/ad/img.png
- https://doubleclick.com/iframe
- https://cnn.com/js/script.js
- …

EasyList Rule

Applying EasyList to Crawl Data

Fetched Page	Requested Sub-Resources	EasyList Rule
`https://cnn.com`	`https://cnn.com/header.png`	
	`https://cnn.com/ad/img.png`	`*/ad/*`
	`https://doubleclick.com/iframe`	`||doubleclick.com^`
	`https://cnn.com/js/script.js`	
	…	…

On Omitting Element Rules

- **Noted network and exception rules**
  Did not include element (i.e., cosmetic) rules

- **Reasoning**
  - Measurement focus is on privacy and performance
  - Highly variable and dependent on user interaction
  - Many EasyList consuming tools also omit them (e.g., Privoxy, PiHole)

Summary

- **Instrumented automated Chromium**
- **Visited 10k sites (5k popular, 5k unpopular)**
- **Recorded:**
  - Domains visited
  - Subpages visited
  - Resource requests and responses
  - Matching EasyList network rules

Overview

- **Context and Background**
  What, why and how of EasyList

- **Methodology**
  Web scale measurement over two months

- **Measurement Results** ←
  Whats used and unused, rule lifecycle, how do trackers respond, etc?

- **Applications**
  Mobile and extension optimizations

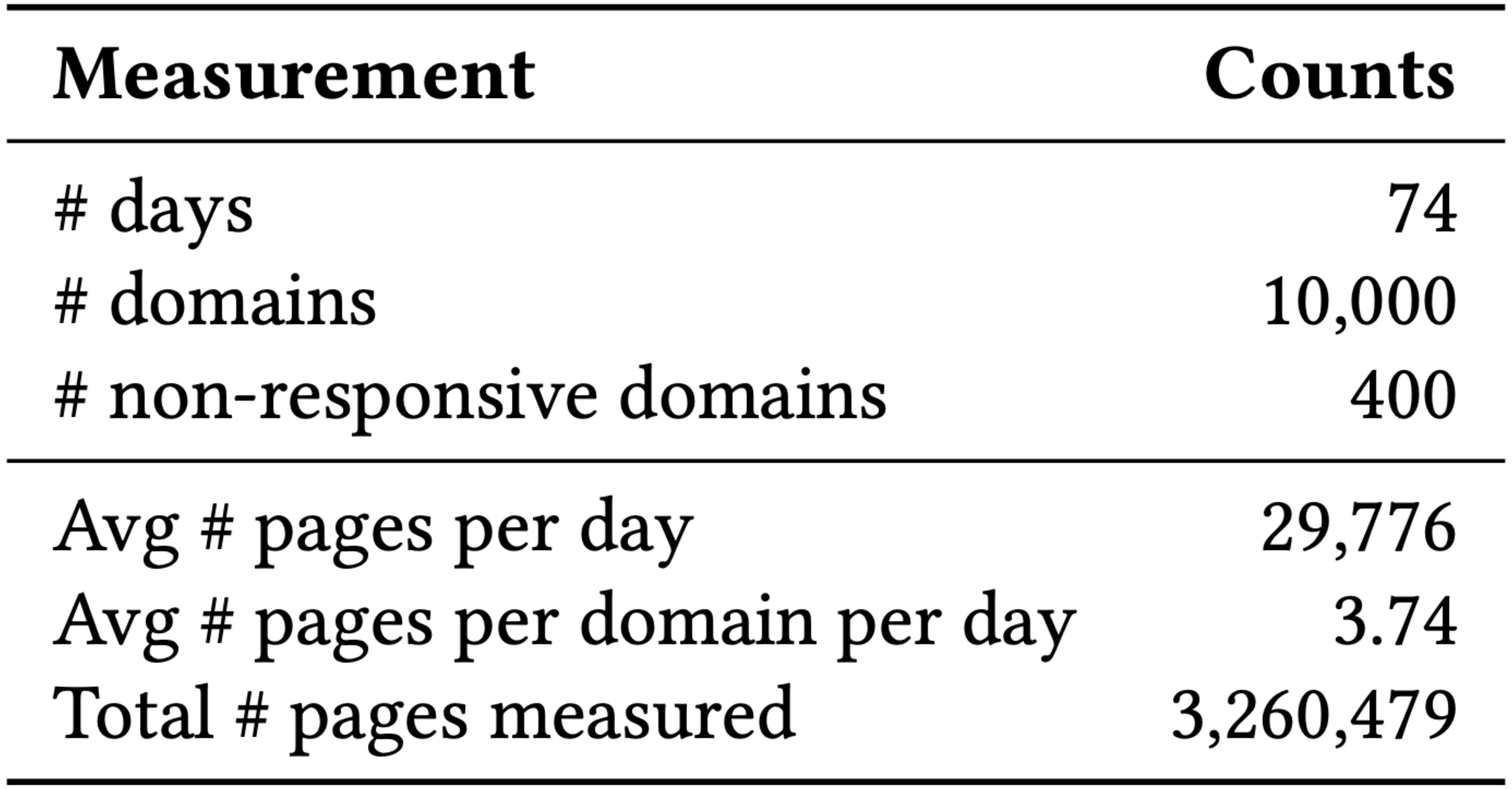
- **Discussion and Conclusion**

Measurement Results

- **Study period:**
  July 24th → October 5th, 2018

- **Unresponsive domains:**
  400 domains never replied

- **3.74 pages per domain:**
  Difference b/c single
  page apps, CF CAPTCHA, etc.



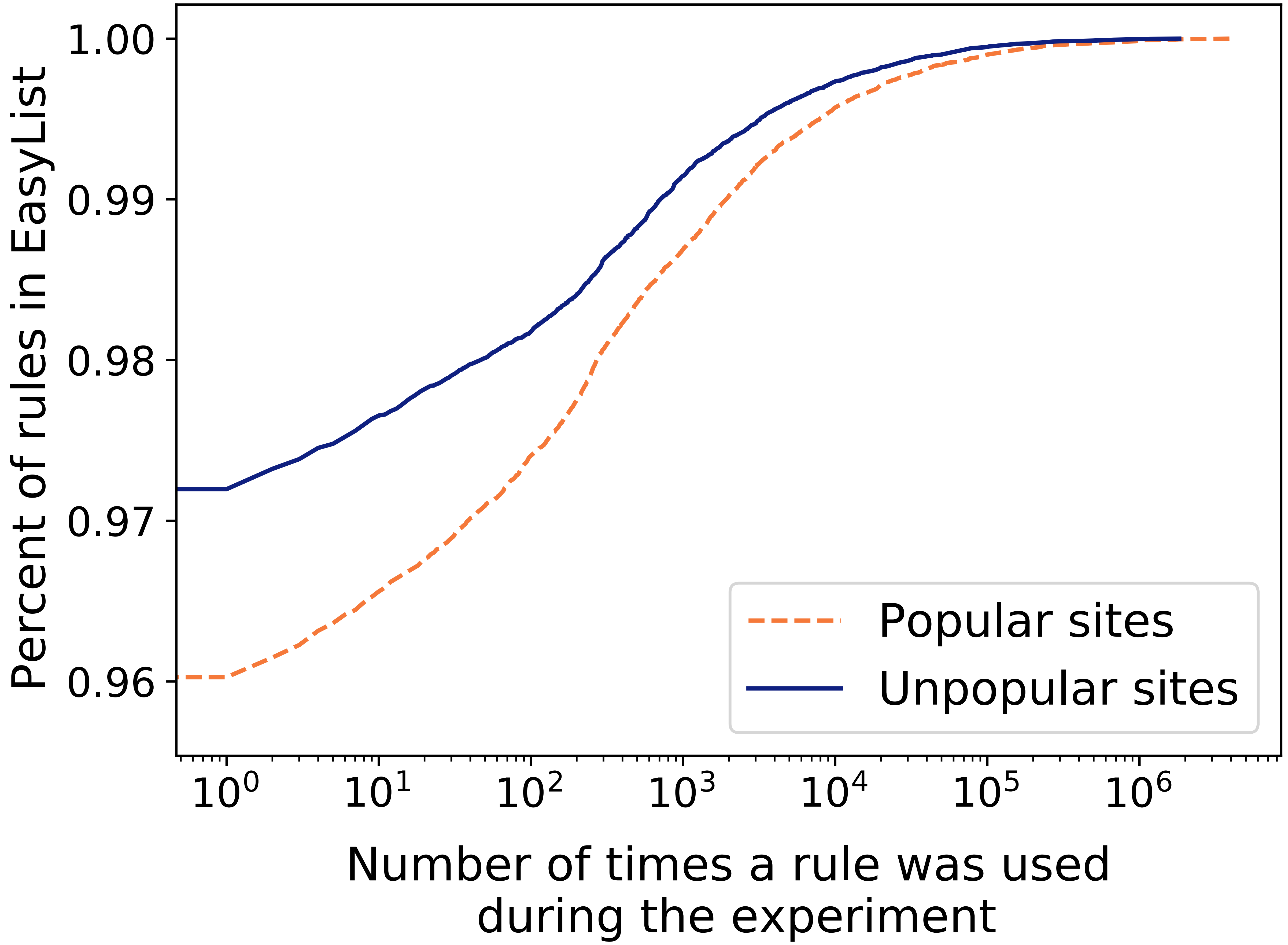
Measurement	**Counts**
# days	74
# domains	10,000
# non-responsive domains	400
Avg # pages per day	29,776
Avg # pages per domain per day	3.74
Total # pages measured	3,260,479

Proportion of EasyList Rules Used

- **Measurement**
  % of rules used at least once during the entire experiment

- **Most rules were not used**
  90.16% never applied
  5.39% used >= 100 times

- **Domain popularity not sig**



Relationship of Rule Age and Usefulness

- **Measurement:**
  Are newer rules more useful?

- **Answer:**
  Mixed, but mostly no

- New and old rules are used at least once equally

- Most blocking is done by old rules

	Added during experiment	**Added before experiment**
Absolute #	2,002	37,826
% used at least once	9.45%	9.84%
Use frequency (of those used at least once)	0.65 per day	6.14 per day

Advertiser Reactions

- **Methodology:**
  - Same resource, multiple URLs, only some blocked
  - Non-blocked URLs occurred after relevant rule
  - Compare URLs to observe why not blocked

Advertiser Reactions

- **Methodology:**
  - Same resource, multiple URLs, only some blocked
  - Non-blocked URLs occurred after relevant rule
  - Compare URLs to observe why not blocked

T_0	T_1	T_2
a.com/ad-script.js b.com/ad-script.js c.com/ad-script.js		

Advertiser Reactions

- **Methodology:**
  - Same resource, multiple URLs, only some blocked
  - Non-blocked URLs occurred after relevant rule
  - Compare URLs to observe why not blocked

T_0	T_1	T_2
a.com/ad-script.js b.com/ad-script.js c.com/ad-script.js	New Rule: /ad-script.js	

Advertiser Reactions

- **Methodology:**
  - Same resource, multiple URLs, only some blocked
  - Non-blocked URLs occurred after relevant rule
  - Compare URLs to observe why not blocked

T_0	T_1	T_2

`a.com/ad-script.js`
`b.com/ad-script.js`
`c.com/ad-script.js`
	New Rule:
	`/ad-script.js`
		~~a.com/ad-script.js~~
		~~b.com/ad-script.js~~
		`c.com/sneaky.js`

Advertiser Reactions

- **Methodology:**
  - Same resource, multiple URLs, only some blocked
  - Non-blocked URLs occurred after relevant rule
  - Compare URLs to observe why not blocked

$\underline{T_0}$

a.com/ad-script.js
b.com/ad-script.js
c.com/ad-script.js

$\underline{T_1}$

New Rule:

/ad-script.js

$\underline{T_2}$

~~a.com/ad-script.js~~
~~b.com/ad-script.js~~
c.com/sneaky.js

a.com/ad-script.js
b.com/ad-script.js
c.com/ad-script.js

V.S.

c.com/sneaky.js

Advertiser Reactions

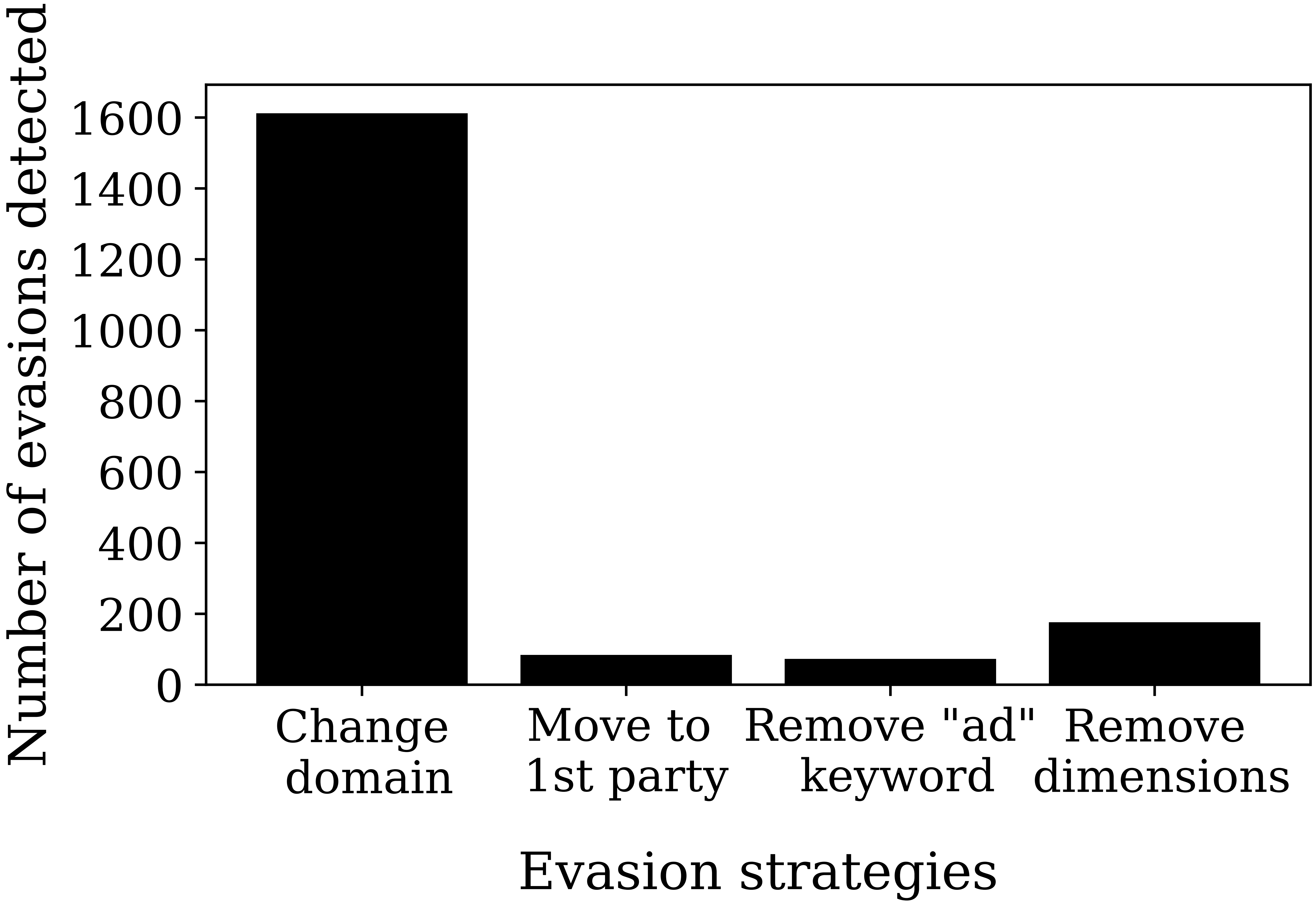
- **Changing domain:**
  tracker.com/script.js → benign.com/script.js

- **Move to 1st party:**
  google-analytics.com/ga.js → cnn.com/ga.js

- **Remove “ad” keyword:**
  example.org/ads/shoes.png → example.org/images/shoes.png

- **Remove dimensions:**
  example.org/shoes-320x240.png → example.org/shoes-standard.png

Advertiser Reactions



Also in the Paper…

- **How quickly advertisers respond to new rules?**
  Most don't…

- **Statistical correlation between rule age and use frequency**
  Significant positive correlation

- **Specific examples of filter list evasions**
  We name names…

Overview

- **Context and Background**
  What, why and how of EasyList

- **Methodology**
  Web scale measurement over two months

- **Measurement Results**
  Whats used and unused, rule lifecycle, how do trackers respond, etc?

- **Applications**
  Mobile and extension optimizations

- **Discussion and Conclusion**

Applications

- **Mobile content blocking**
  Fitting filter lists in mobile devices, performantly

- **Improving performance of extensions**
  Left for the paper

Applications

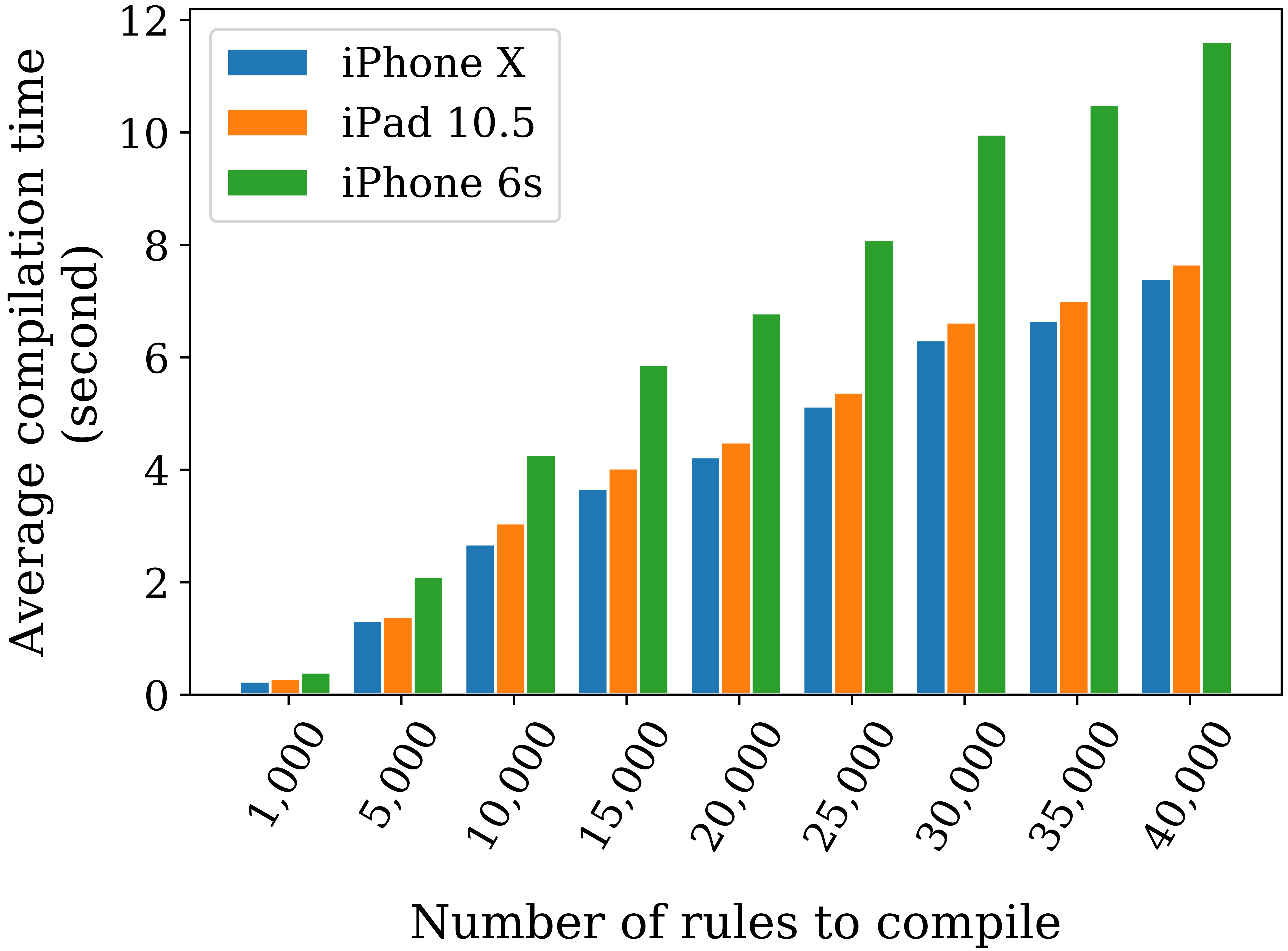
- **Mobile content blocking**
  Fitting filter lists in mobile devices, performantly

- **Improving performance of extensions**
  Left for the paper

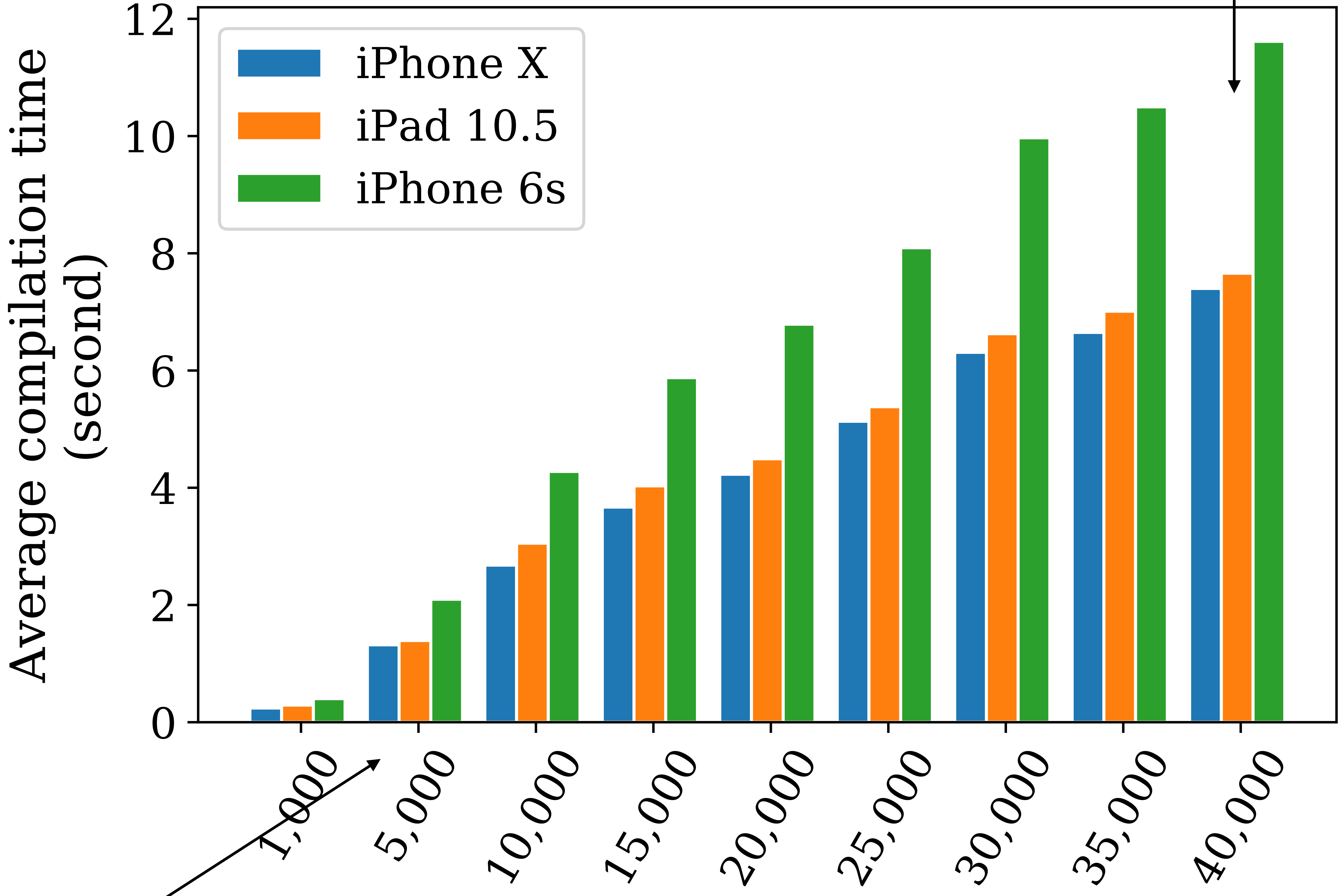
Mobile Content Blocking

- **Two related problems**
  - iOS limits to 50k rules
  - Compiling rules is slow on first load
- **Its not only EasyList…**
  - EasyPrivacy
  - Regional lists
- **Solution**
  - Use crawl data to identify likely useful rules
  - Only load those rules on iOS
  - “Slim List”

Mobile Content Blocking



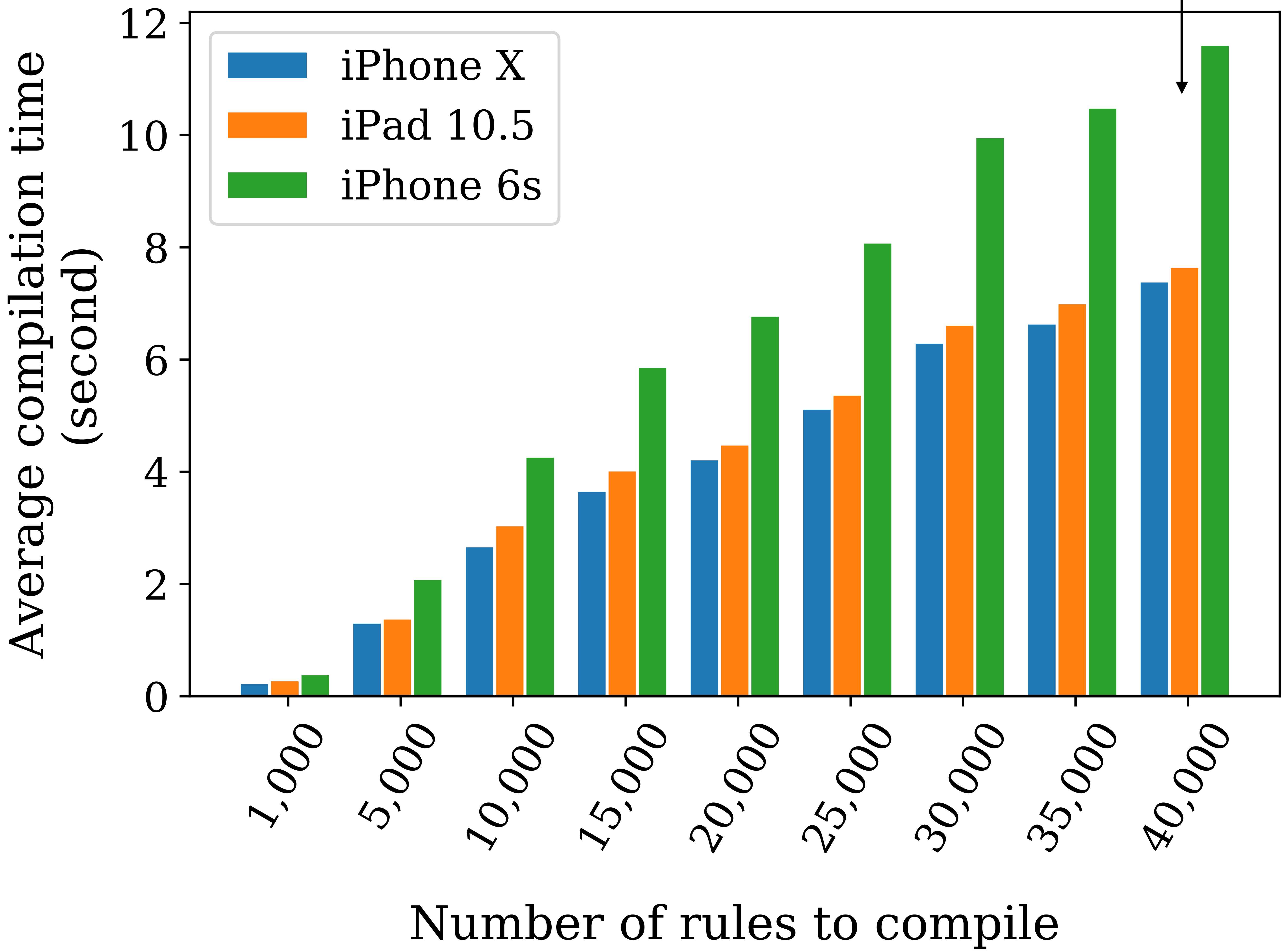
Mobile Content Blocking



Full EasyList

“Useful” subset

Mobile Content Blocking



Full EasyList

Applications

- **Mobile content blocking**
  Fitting filter lists in mobile devices, performantly

- **Improving performance of extensions**
  Left for the paper

Overview

- **Context and Background**
  What, why and how of EasyList

- **Methodology**
  Web scale measurement over two months

- **Measurement Results**
  Whats used and unused, rule lifecycle, how do trackers respond, etc?

- **Applications**
  Mobile and extension optimizations

- **Discussion and Conclusion**

Limitations And Future Work

- **Web site selection generalizability**
  We assume interactivity isn't vital
  We assume "shallow" pages are similar to "deep" pages

- **Web region and language generalizability**
  We assume measuring from US IP generalizes
  We assume good division between English / global EasyList and regional lists

- **Varying resource blocking importance**
  We assume all blocking is equally useful
  We assume vital, security level protections are dealt with through other means

Summary

- First measurement of how EasyList affects the web
- Broadly used, maintained by five people
- >90% of EasyList provides little benefit
- Quantified taxonomy of filter list evasion
- Measurement allows for use on mobile

Summary and Thank You!

- First measurement of how EasyList affects the web
- Broadly used, maintained by five people
- >90% of EasyList provides little benefit
- Quantified taxonomy of filter list evasion
- Measurement allows for use on mobile

Pete Snyder (@pes10k) – Brave Software
Antoine Vastel – University of Lille / INRIA
Ben Livshits – Brave Software / Imperial College London