



MAY 11-12

BRIEFINGS

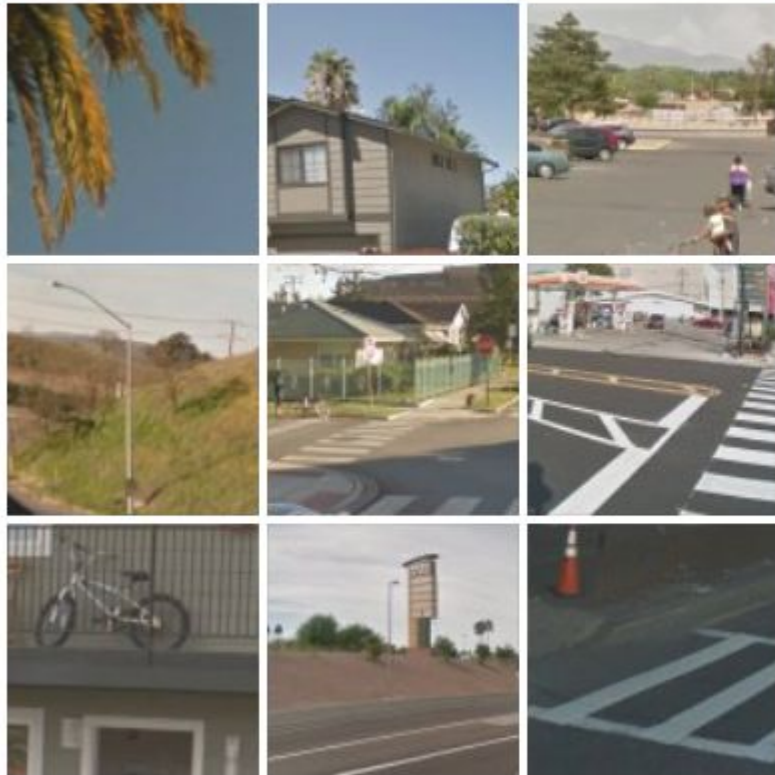
DATA  OME

Leveraging Streaming-Based Outlier Detection and SliceLine to Stop Heavily Distributed Bot Attacks

Antoine Vastel, PhD | Head of Research, DataDome
Konstantina Kontoudi, PhD | Lead Data Scientist, DataDome



Select all images with
crosswalks
Click verify once there are none left.



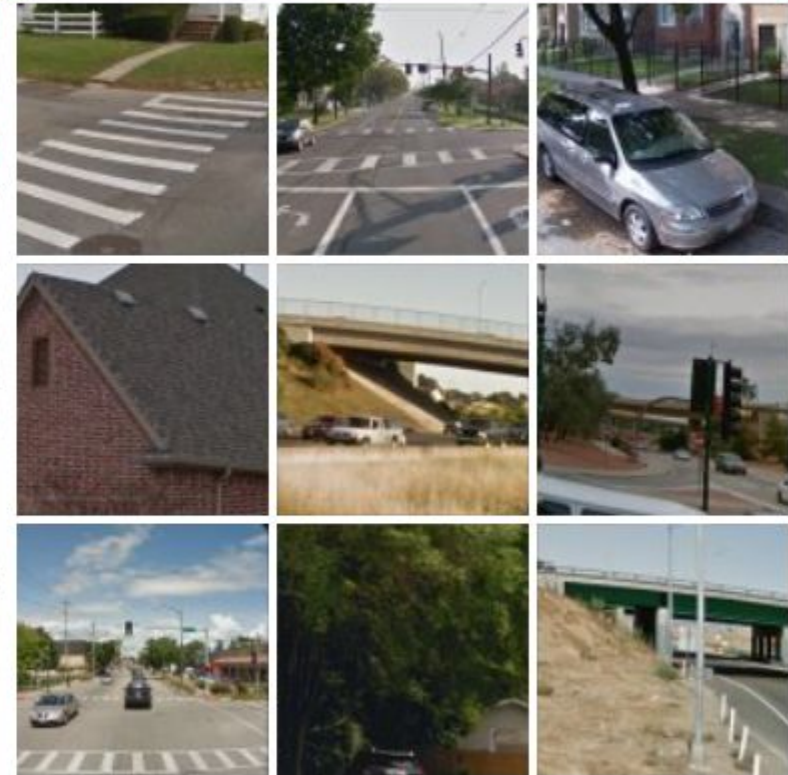
VERIFY

Select all squares with
bicycles
If there are none, click skip



SKIP

Select all images with
crosswalks
Click verify once there are none left.



VERIFY



Select all squares with
fire hydrants
If there are none, click skip

⏪ 🔊 ⓘ [SKIP](#)



Select all squares with
traffic lights
If there are none, click skip

A 3x3 grid of images. The top-left square contains a traffic light with an orange circle drawn around it. The other squares are empty.

A close-up of a traffic light with an orange circle drawn around it. Below this is a close-up of a man's face looking to the right.

A 3x3 grid of images. The top-left square contains a bicycle with an orange circle drawn around it. The other squares are empty. Below this is a close-up of a man's face looking to the right.

**All these CAPTCHAs
because of bad bots.**

**But can we detect bad
bots without
CAPTCHAs?**

Introduction to (Bad) Bots

What's a bot?

Program to **automate actions**.

Can be used for positive purposes:

- Automated **website testing**.

Nefarious purposes:

- Test batch of **stolen credentials**.
- Generate **fake video views/retweets**.

Attacks conducted by bots

Credential stuffing/account takeover → Steal user accounts.

DDoS → Make website/mobile app unavailable.

Carding → Test stolen credit cards.

Vote manipulation → Generate fake views, increase number of likes, retweets, etc.

Bot Technologies: HTTP Clients



Axios
Got



Aiohttp
Requests



net/http

Low CPU/RAM resources needed.

No JavaScript execution.

(Potentially) inconsistent:

- HTTP Headers
- TLS Fingerprints

Bot Technologies: Automated Browsers

Puppeteer +
(headless) Chrome



Selenium +
(headless) Firefox



Playwright +
(headless) Webkit



More CPU/RAM resources needed.

Execute JavaScript (JS) natively.

Consistent **HTTP headers** and **TLS fingerprints**.

Potentially inconsistent browser fingerprint (JS).



How to detect bots?

How to detect bots?

Signatures/(browser/TLS/ HTTP) fingerprints.

Behavioral analysis:

- Volume of requests.
- Browsing patterns.

Reputation: IP/session, proxy detection.

Context: country, time of the day, website targeted.

Detection Example: Selenium

Detection using browser fingerprinting (JS).

Selenium introduces attribute:

`document.$cdc_asdjflasutopfhvcZLmcfl_`

```
chromium / chrome / test / chromedriver / js / call_function.js
459 lines (426 loc) · 14 KB
Blame Raw Copy Edit Search ...
208
209 /**
210  * Returns the global object cache for the page.
211  * @param {Document=} opt_doc The document whose cache to retrieve. Defaults to
212  *   the current document.
213  * @return {!Cache} The page's object cache.
214  */
215 function getPageCache(opt_doc, opt_w3c) {
216   var doc = opt_doc || document;
217   var w3c = opt_w3c || false;
218   // |key| is a long random string, unlikely to conflict with anything else.
219   var key = '$cdc_asdjflasutopfhvcZLmcfl_';
220   if (w3c) {
221     if (!(key in doc))
222       doc[key] = new CacheWithUUID();
223     return doc[key];
224   } else {
225     if (!(key in doc))
226       doc[key] = new Cache();
227     return doc[key];
228   }
229 }
```

Bypass Techniques Used by Bots

Forge TLS fingerprints.



Forge browser fingerprint and HTTP headers.

<https://github.com/intoli/user-agents>

Fake/simulate JavaScript execution:

- Forge JS proof of work payload (reverse engineer).

```
{
  "appName": "Netscape",
  "connection": {
    "downlink": 10,
    "effectiveType": "4g",
    "rtt": 0
  },
  "platform": "Win32",
  "pluginsLength": 3,
  "vendor": "Google Inc.",
  "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML; like Gecko) Chrome/108.0.0.0 Safari/537.36",
  "viewportHeight": 660,
  "viewportWidth": 1260,
  "deviceCategory": "desktop",
  "screenHeight": 800,
  "screenWidth": 1280
}
```

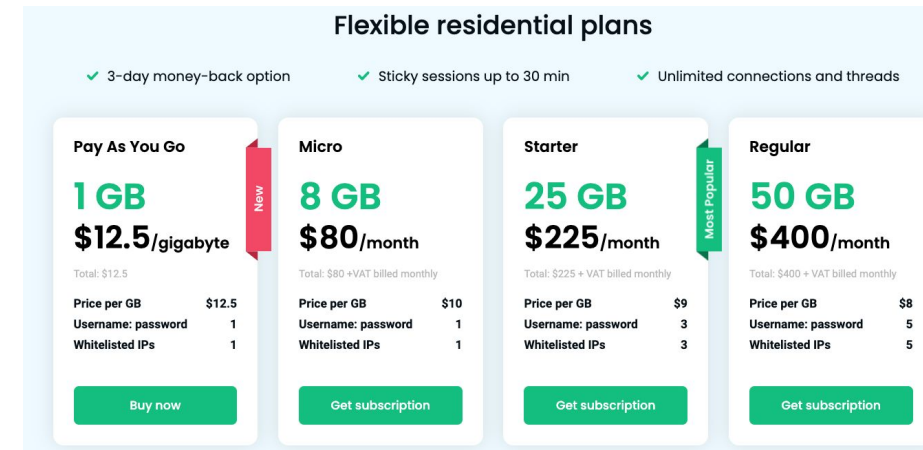
Bypass Techniques Used by Bots

Distribute attack using **proxies**:

→ Avoid IP-based rate limiting.

Distribute attack using **residential proxies**:

→ Avoid reputation-based blocking.



Plan	Capacity	Price	Price per GB	Username: password	Whitelisted IPs
Pay As You Go	1 GB	\$12.5/gigabyte	\$12.5	1	1
Micro	8 GB	\$80/month	\$10	1	1
Starter	25 GB	\$225/month	\$9	3	3
Regular	50 GB	\$400/month	\$8	5	5

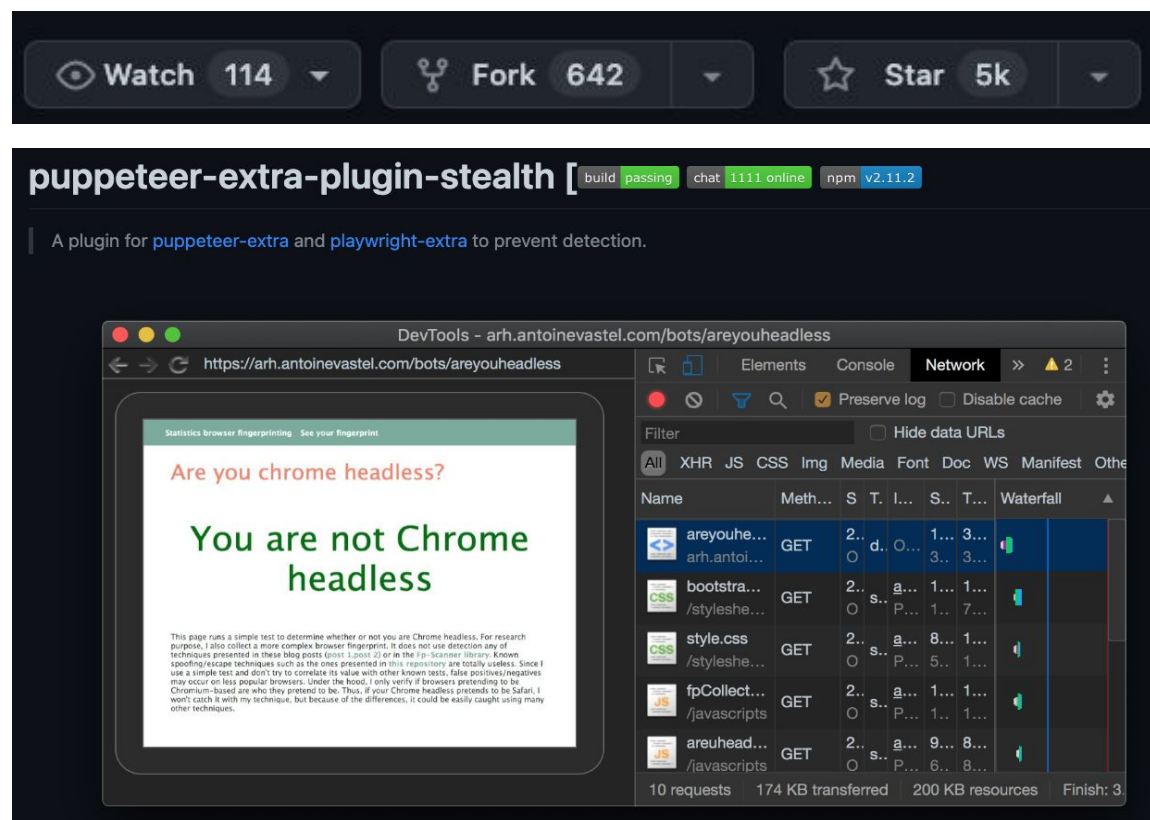
Distribute attack using **residential proxies located in same country as website targeted**:

→ Avoid geo-blocking.



Community Driven Anti-Detection Frameworks

Generic, for (headless) automated browsers:



Watch 114 Fork 642 Star 5k

puppeteer-extra-plugin-stealth [build passing chat 1111 online npm v2.11.2]

A plugin for puppeteer-extra and playwright-extra to prevent detection.

DevTools - arh.antoinevastel.com/bots/areyouheadless

https://arh.antoinevastel.com/bots/areyouheadless

Statistics browser fingerprinting. See your fingerprint

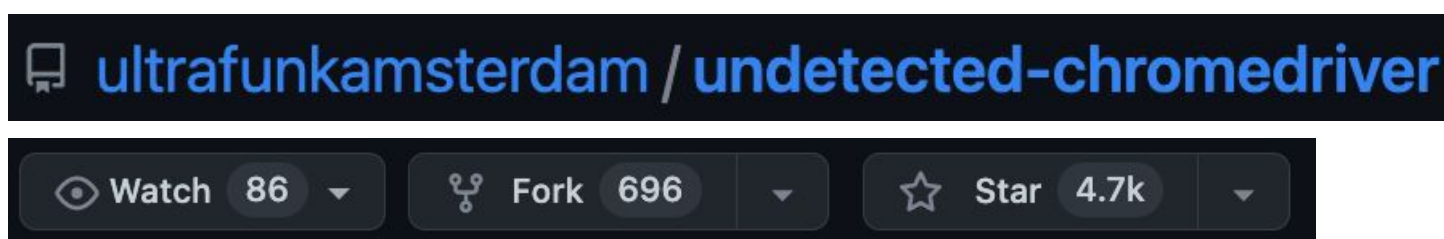
Are you chrome headless?

You are not Chrome headless

This page runs a simple test to determine whether or not you are Chrome headless. For research purposes, I also collect a more complex browser fingerprint. It does not use detection any of techniques presented in these blog posts (post 1, post 2) or in the fp-scanner library. Known spoofing/resize techniques such as the ones presented in this repository are totally useless. Since I use a simple test and don't try to correlate its value with other known tests, false positives/negatives may occur on less popular browsers. Under the hood, I only verify if browser's pretends to be Chromium-based are who they pretend to be. Thus, if your Chrome headless pretends to be Safari, I won't catch it with my technique, but because of the differences, it could be easily caught using many other techniques.

Name	Meth...	S	T	I...	S...	T...	Waterfall
areyouhe... arh.antol...	GET	2..	0	d...	1..	3..	
bootstra... /styleshe...	GET	2..	0	s...	1..	1..	
style.css /styleshe...	GET	2..	0	s...	8..	1..	
fpCollect... /javascripts	GET	2..	0	s...	1..	1..	
areuhead... /javascripts	GET	2..	0	s...	9..	8..	

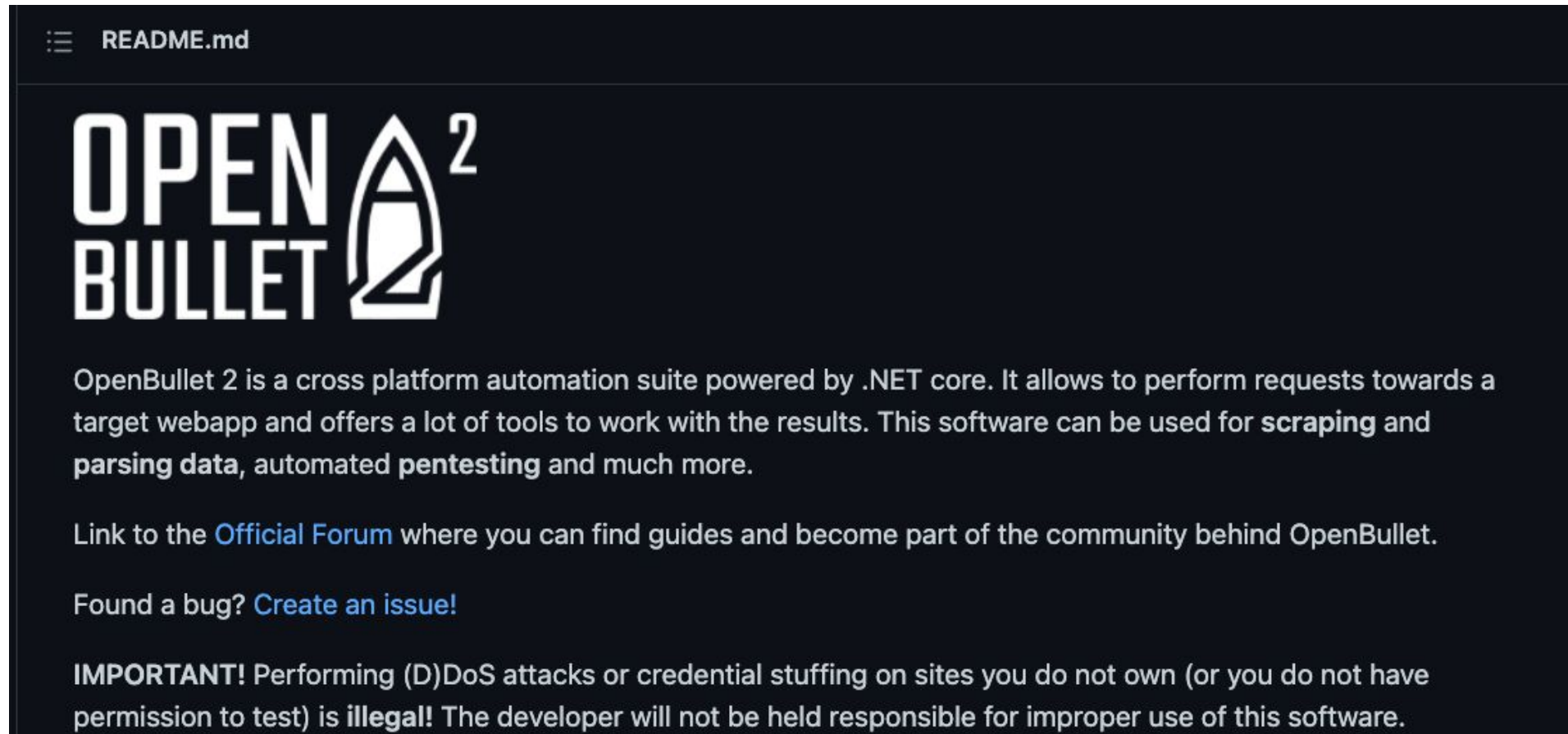
10 requests 174 KB transferred 200 KB resources Finish: 3



ultrafunkamsterdam / undetected-chromedriver

Watch 86 Fork 696 Star 4.7k

To Conduct Credential Stuffing Attacks



☰ README.md

OPEN BULLET ²

OpenBullet 2 is a cross platform automation suite powered by .NET core. It allows to perform requests towards a target webapp and offers a lot of tools to work with the results. This software can be used for **scraping and parsing data**, **automated pentesting** and much more.

Link to the [Official Forum](#) where you can find guides and become part of the community behind OpenBullet.

Found a bug? [Create an issue!](#)

IMPORTANT! Performing (D)DoS attacks or credential stuffing on sites you do not own (or you do not have permission to test) is **illegal!** The developer will not be held responsible for improper use of this software.

Bad Bots in 2023: Summary

Distribute their attacks.

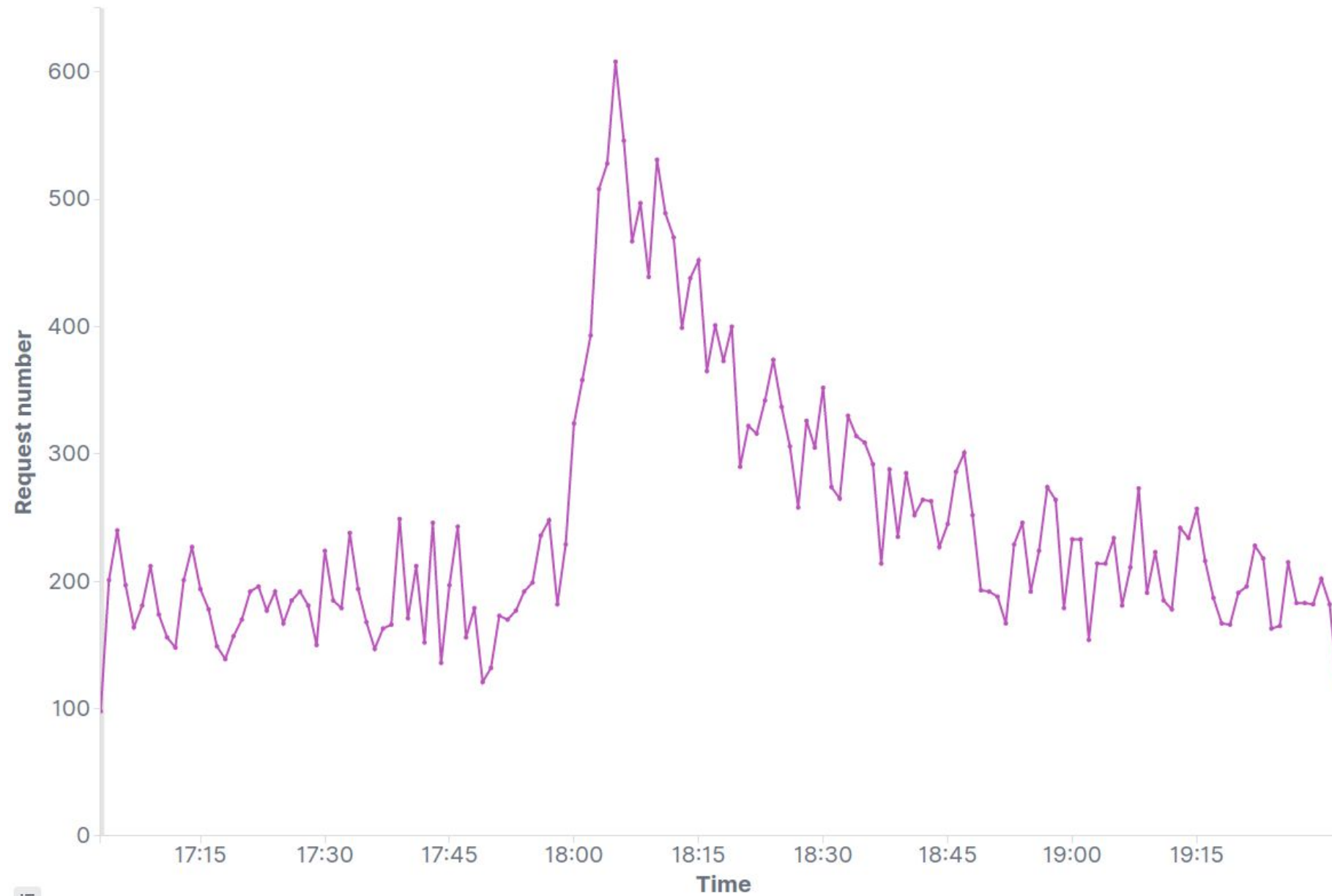
Leverage thousands of residential proxies.

Constantly change and forge their signatures/fingerprints.

Q: How to block these ever-evolving and distributed bots?

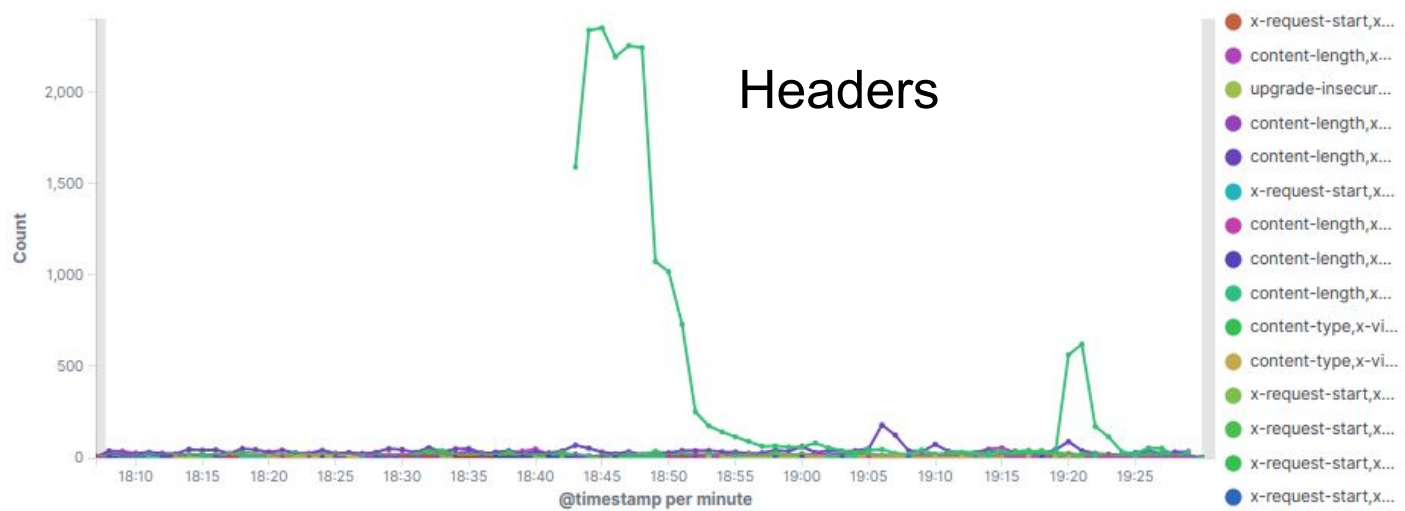
Detecting and Blocking Distributed Attacks Manually

Detect a Traffic Spike

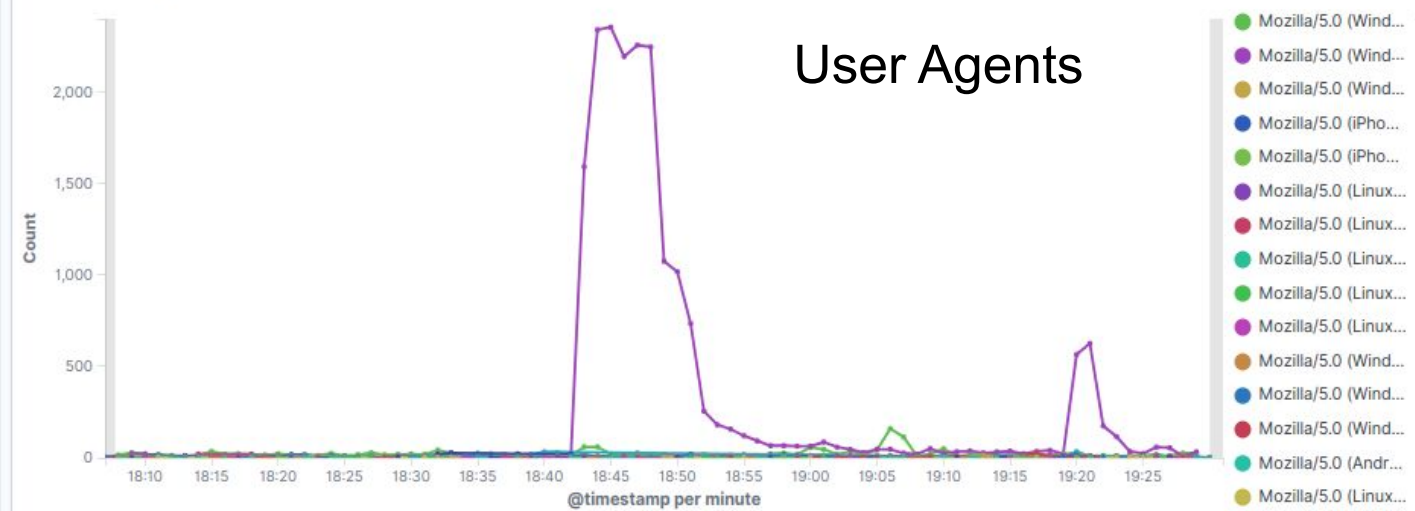


Drill-Down on Different Features

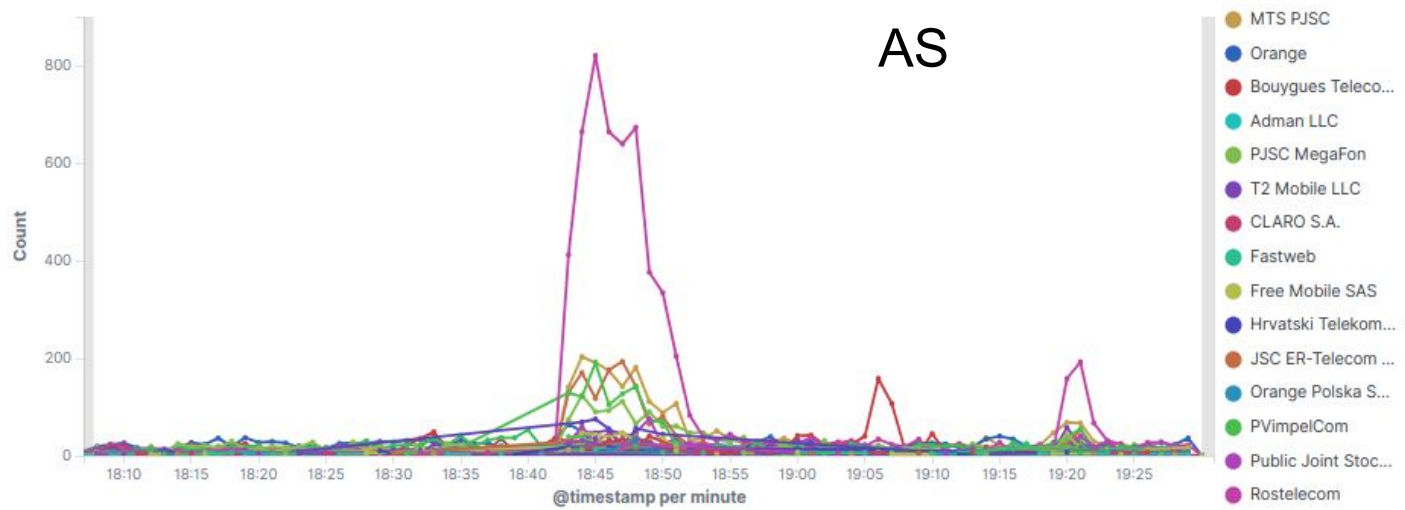
login-attack-fingerprint-HeadersList



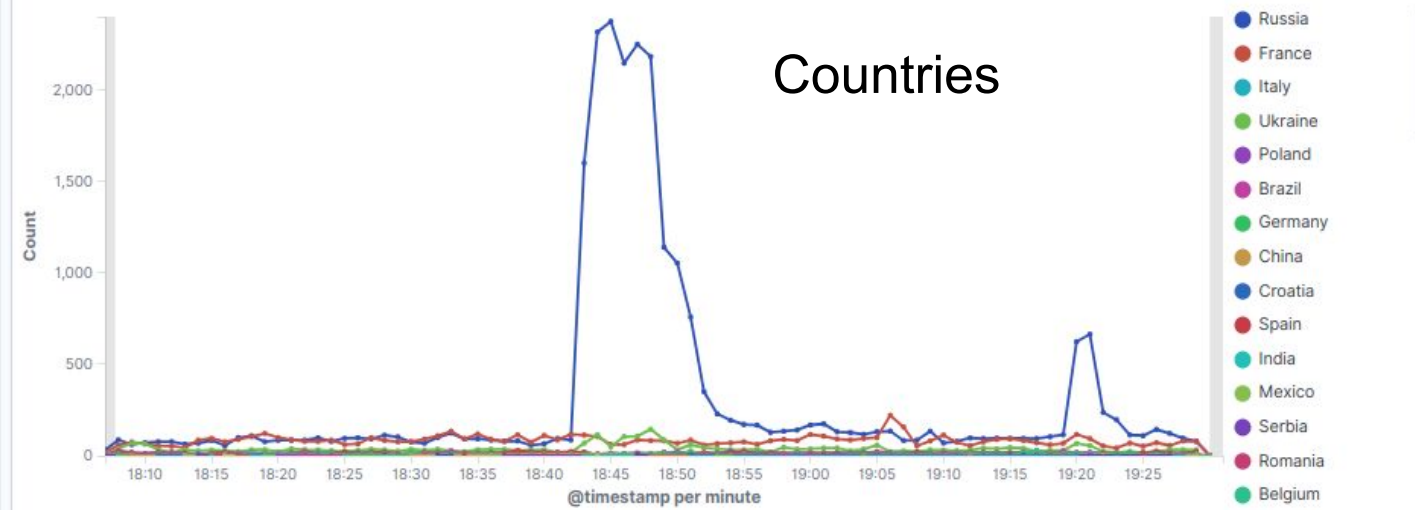
login-attack-fingerprint-UserAgent



login-attack-fingerprint-AS



login-attack-fingerprint-Countries



Derive Rules

Find a rule.

```
Country=Russia && User Agent=Mozilla/5.0 (Macintosh;  
Intel Mac OS X 10_15_3) AppleWebKit/537.36 (KHTML, like  
Gecko) Chrome/99.0.4844.84 Safari/537.36
```

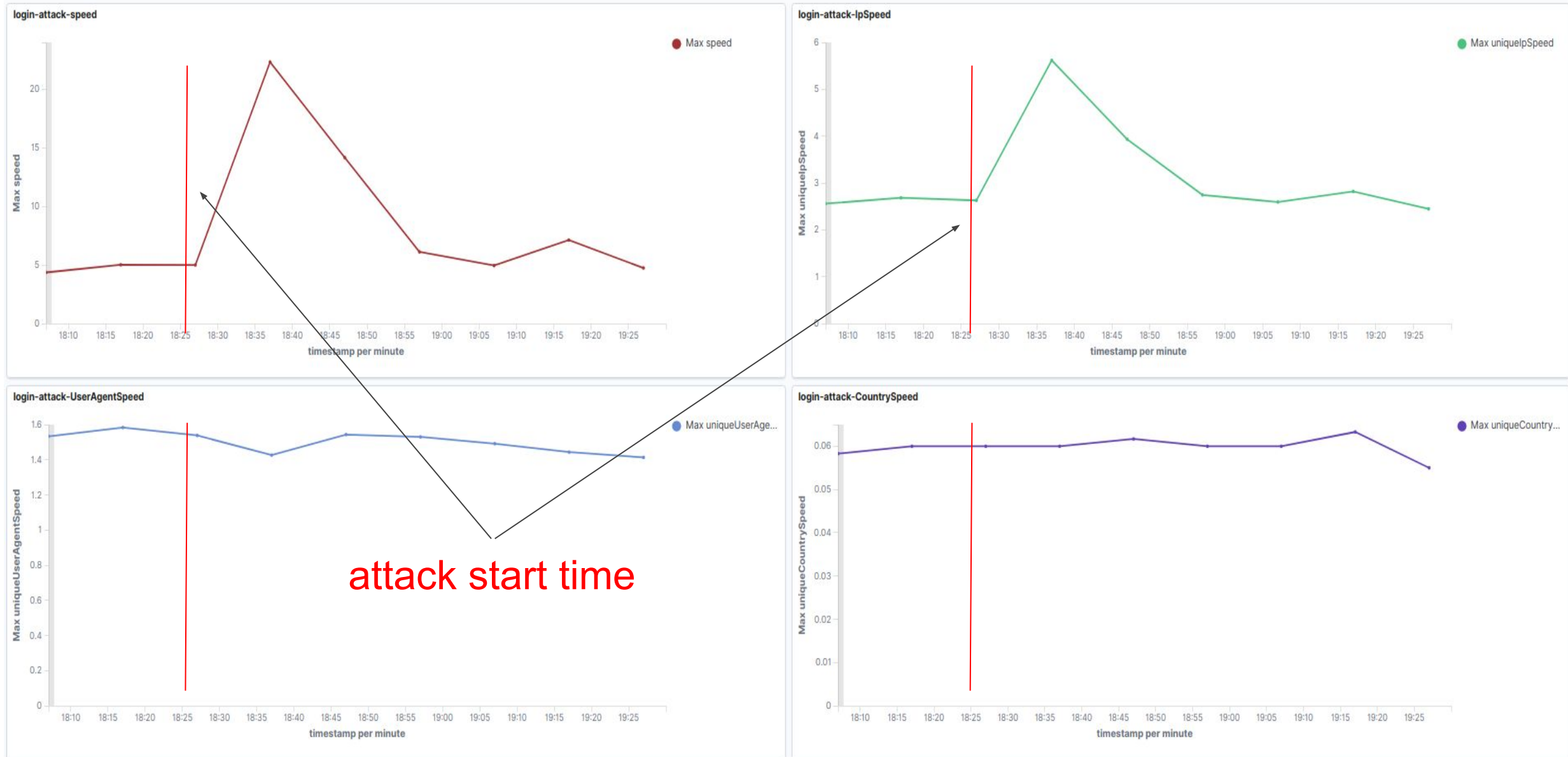
How to automate the analysis?

Detecting Distributed Attacks

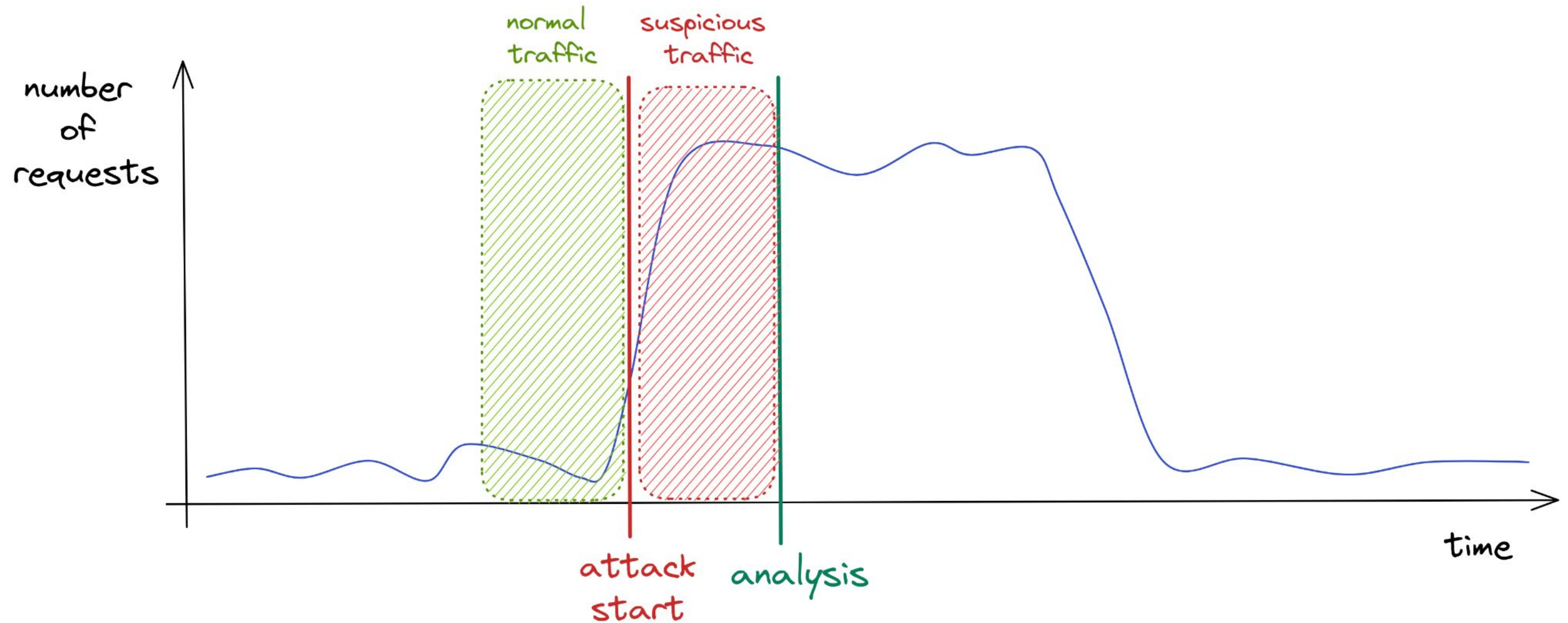
Detecting Distributed Attacks

- Compute **aggregate statistics on login**:
 - Number of unique User Agents.
 - Number of unique IPs.
 - Number of sessions.
- Detect **anomalies** on the resulting **time-series** using a **z-score based anomaly detection algorithm**.
- **Push an event** describing the attack (customer, start time).
- Implemented in streaming (Apache Flink).

Detecting Distributed Attacks

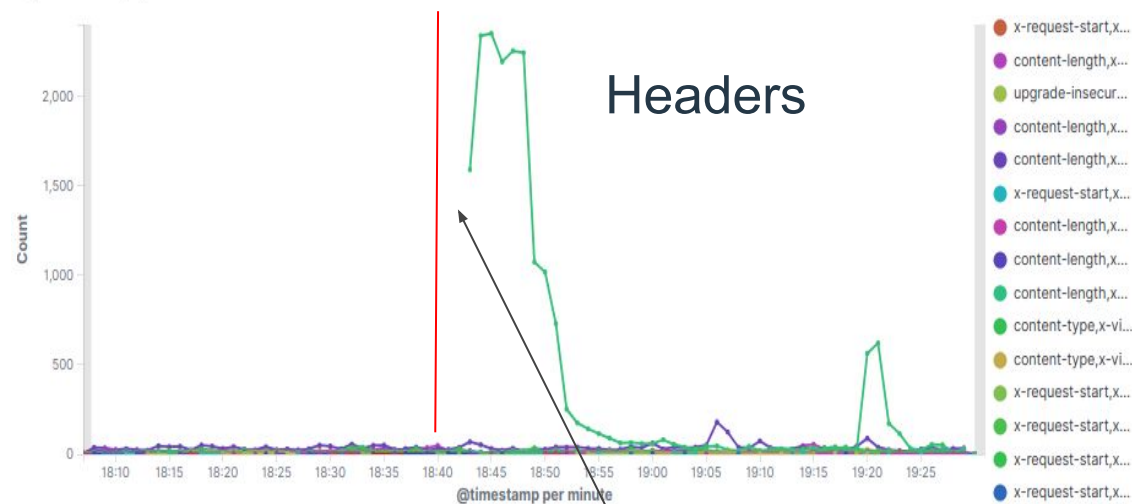


General Idea

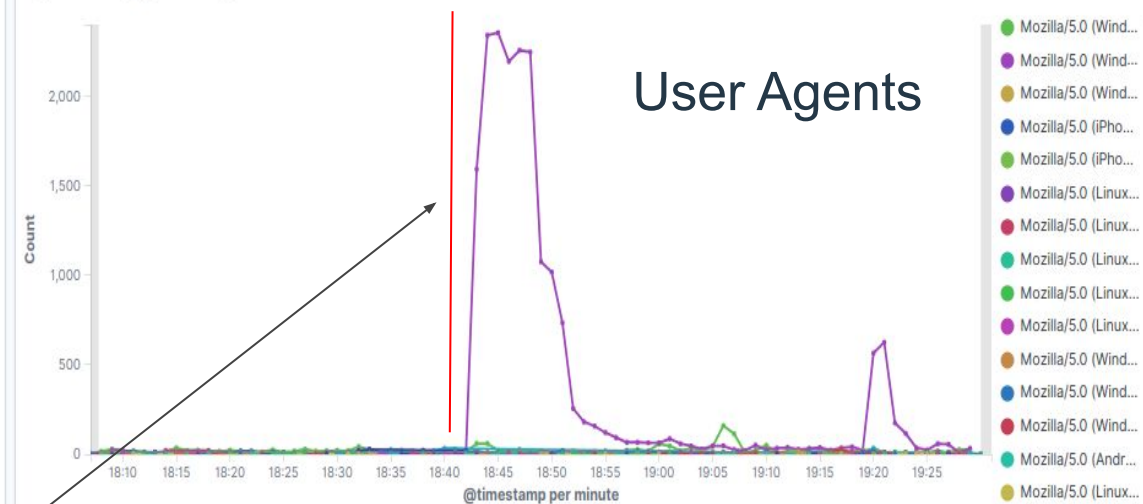


Next Step: Automate Rule Generation

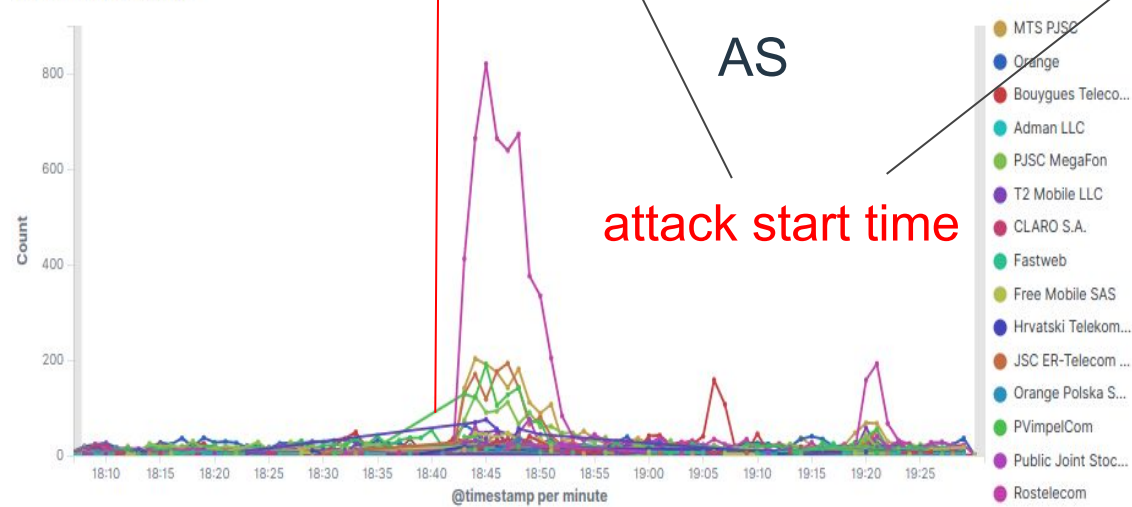
login-attack-fingerprint-HeadersList



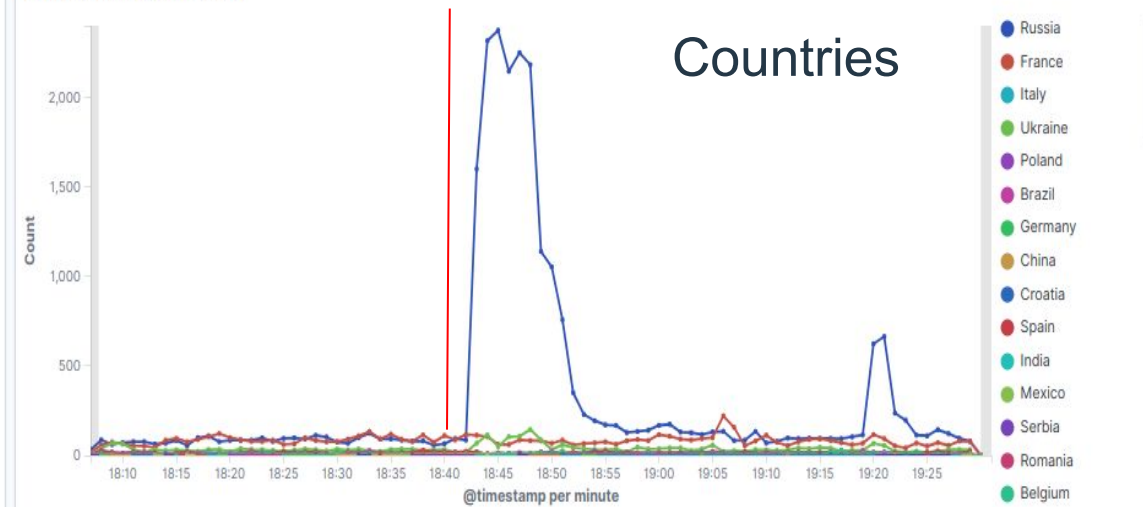
login-attack-fingerprint-UserAgent



login-attack-fingerprint-AS



login-attack-fingerprint-Countries





Introducing Sliceline

What is Sliceline?

Algorithm to find **slices** of data where an ML model performs badly.

country	UA	IP type	error
FR	Chrome	residential	0.1
FR	Firefox	data_center	0.2
DE	Chrome	data_center	0.9
FR	Chrome	data_center	0.8
CN	Firefox	residential	0.2
CN	Chrome	residential	0.1

Slice: a conjunction of conditions.

country = DE

UA=Firefox && IP
type=data_center

Generating Rules Using Sliceline



Use Sliceline to target groups of data instead of bad-performing slices.

country	UA	IP type	group
FR	Chrome	residential	0
FR	Firefox	data_center	0
DE	Chrome	data_center	1
FR	Chrome	data_center	1
CN	Firefox	residential	0
CN	Chrome	residential	0

Group definitions:

- **Human traffic** => group 0
- **Suspicious traffic** => group 1

How does it really work?

- Searches for slices with large errors.
 - **Need a way to quickly compute slice errors.**
- Uses matrix algebra to evaluate slices.
 - **Can profit from optimized implementations of matrix multiplication.**
- Uses pruning to reduce the search space without accessing the data.
 - **Reduces memory access for large datasets.**

Sliceline Internals: Encoding

country	UA	IP type	error
FR	Chrome	residential	0.1
FR	Firefox	data_center	0.2
DE	Chrome	data_center	0.9
FR	Chrome	data_center	0.8
CN	Firefox	residential	0.2
CN	Chrome	residential	0.1

1. One-hot encoding.

2. Express rules as binary vectors.

country = FR	country = DE	country = CN	UA = Chrome	UA = Firefox	IP type = residential	IP type = data_center	error
1	0	0	1	0	1	0	0.1
1	0	0	0	1	0	1	0.2
0	1	0	1	0	0	1	0.9
1	0	0	1	0	1	1	0.8
0	0	1	0	1	1	0	0.2
0	0	1	1	0	1	0	0.1

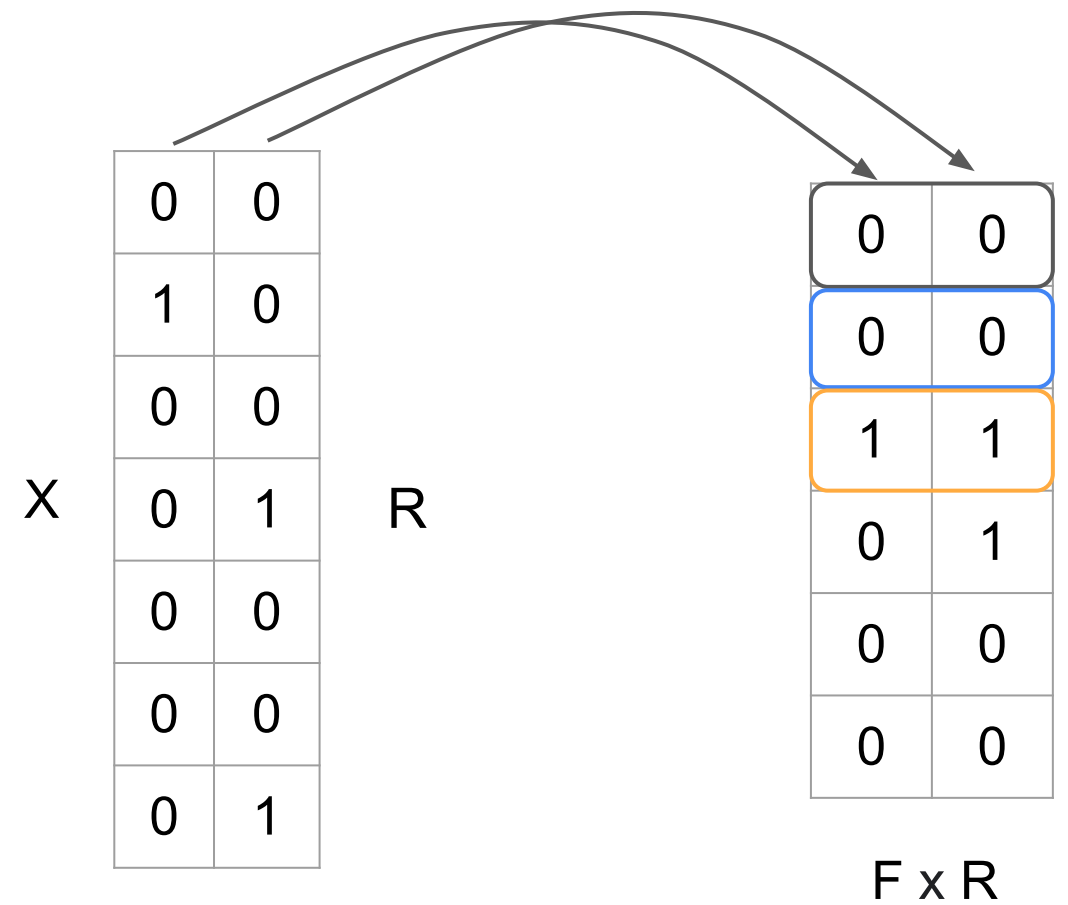
0	1	0	0	0	0	0
---	---	---	---	---	---	---

0	0	0	1	0	0	1
---	---	---	---	---	---	---

Sliceline Internals: Matching Rules With Matrix Multiplication

country = FR	country = DE	country = CN	UA = Chrome	UA = Firefox	IP type = residential	IP type = data_center	error
1	0	0	1	0	1	0	0.1
1	0	0	0	1	0	1	0.2
0	1	0	1	0	0	1	0.9
1	0	0	1	0	1	1	0.8
0	0	1	0	1	1	0	0.2
0	0	1	1	0	1	0	0.1

F: feature matrix



E **country = DE** UA=Firefox && IP type=data_center

Sliceline Internals: Slice Errors

error
0.1
0.2
0.9
0.8
0.2
0.1

E

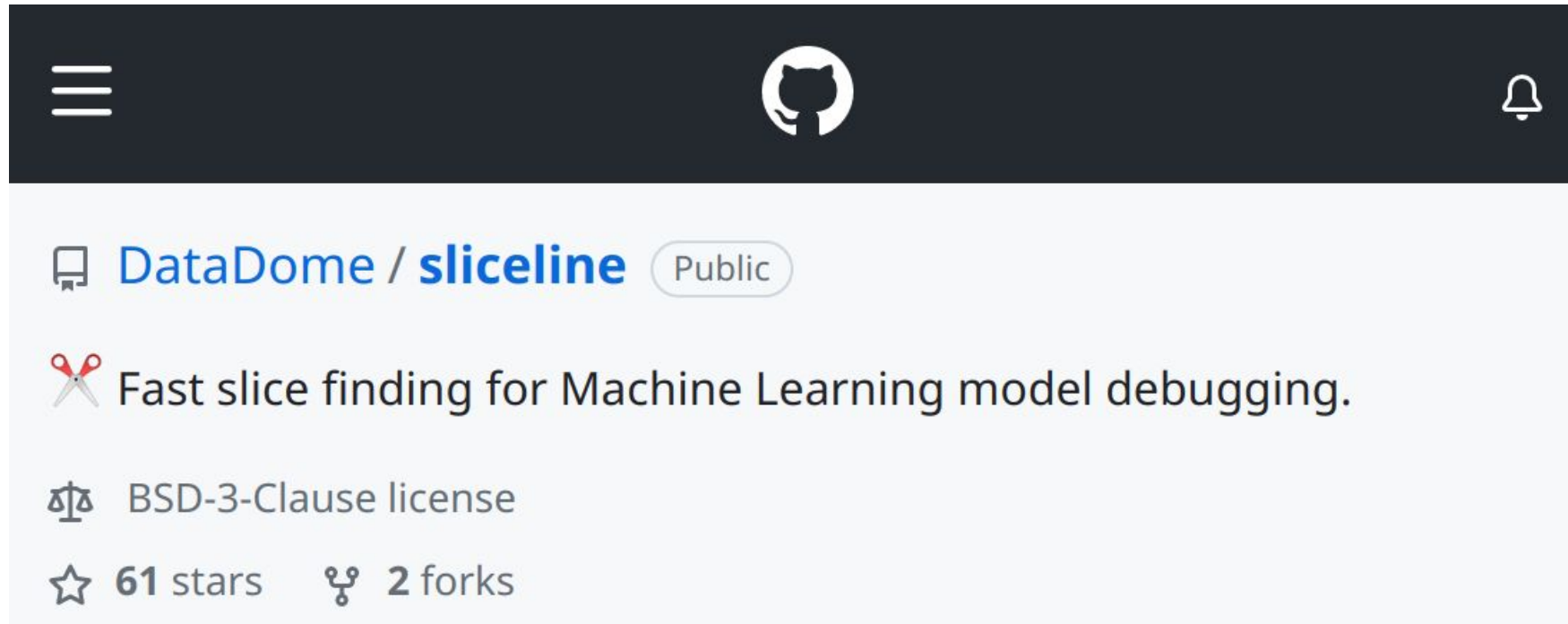
L = FxR

0	0
0	0
1	1
0	1
0	0
0	0

country = DE UA=Firefox &&
IP type=data_center

	$E^T \times L$		
total slice error	<table border="1"><tr><td>0.9</td><td>1.7</td></tr></table>	0.9	1.7
0.9	1.7		
	$I^T \times L$		
slice size	<table border="1"><tr><td>1</td><td>2</td></tr></table>	1	2
1	2		
mean slice error	<table border="1"><tr><td>0.9</td><td>0.85</td></tr></table>	0.9	0.85
0.9	0.85		

Open Source Package



Original algorithm implemented in R.

Implemented in Python:

- Rewrote some part of R implementation using **matrix multiplications**.
- Leverage **numpy optimizations**.
- Compatible with pandas.

Speed up > x 1000

Syntax-agnostic: can generate **rules for any rule-engine**.

Code Example

User Agent	User Agent Version	Country	Accept Language	Headers List	Autonomous System	Accept Header	group
Chrome	111.0	Germany	en-US,en;q=0.9	host,user-agent,sec-ch-ua,sec-ch-ua-mobile,sec...	Vodafone GmbH	text/html,application/xhtml+xml,application/xml...	1
Chrome	111.0	Germany	en-US,en;q=0.9	host,user-agent,sec-ch-ua,sec-ch-ua-mobile,sec...	First Root UG (haftungsbeschraenkt)	text/html,application/xhtml+xml,application/xml...	1
Chrome	111.0	Germany	en-US,en;q=0.9	host,user-agent,sec-ch-ua,sec-ch-ua-mobile,sec...	First Root UG (haftungsbeschraenkt)	text/html,application/xhtml+xml,application/xml...	1
Chrome	111.0	Germany	en-US,en;q=0.9	host,user-agent,sec-ch-ua,sec-ch-ua-mobile,sec...	Hivelocity Inc	text/html,application/xhtml+xml,application/xml...	1
Chrome	110.0	Germany	en-US,en;q=0.9	host,user-agent,sec-ch-ua,sec-ch-ua-mobile,sec...	M247 Europe SRL	text/html,application/xhtml+xml,application/xml...	1
...
Chrome	112.0	France	fr,fr-FR;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6	content-length,sec-ch-ua,sec-ch-ua-platform,se...	Orange	*/*	0
Chrome Mobile	112.0	France	fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7	host,user-agent,content-length,cookie,sec-ch-u...	Free SAS	application/json	0
Chrome	111.0	France	en-US,en;q=0.9	host,user-agent,content-length,sec-ch-ua,conte...	Free SAS	*/*	0
Chrome	109.0	France	fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7	host,user-agent,cookie,sec-ch-ua,sec-ch-ua-mob...	Orange	text/html,application/xhtml+xml,application/xml...	0
Chrome	112.0	France	fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7	host,user-agent,cookie,sec-ch-ua,x-datadome-cl...	Orange	*/*	0

Example Dataset:

- Gathered from a French e-commerce website.
- Human traffic (group 0): requests with old session.
- Suspicious traffic (group 1): requests from non french speaking countries and datacenter IPs.

Code Example

```
sf = Slicefinder(  
    alpha = 0.80,  
    k = 4,  
    max_l = df.shape[1],  
    min_sup = 1,  
    verbose = True  
)  
  
sf.fit(df.drop("group", axis=1), df["group"])
```

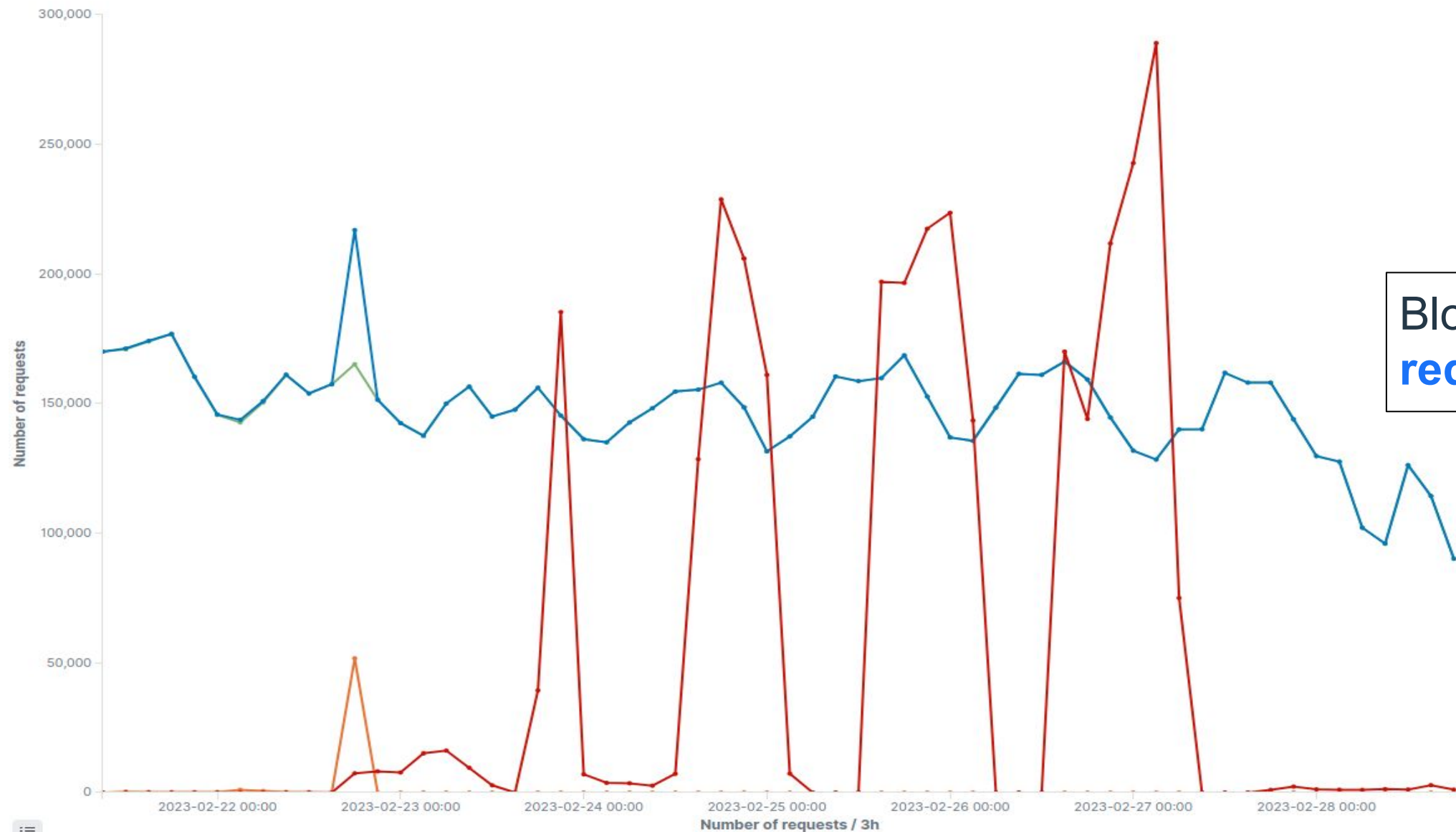
Get rules in the format
you want!

Apply the algorithm with 2 lines of code!

```
for slice, stats in zip(sf.top_slices_, sf.top_slices_statistics_):  
    rule = None  
    for feat, value in zip(df.columns, slice):  
        if value is not None:  
            if rule is None:  
                rule = f"`{feat}`=`{value}` "  
            else:  
                rule += f" && `{feat}`=`{value}` "  
    print(f"{rule} | slice size: {stats['slice_size']}")
```

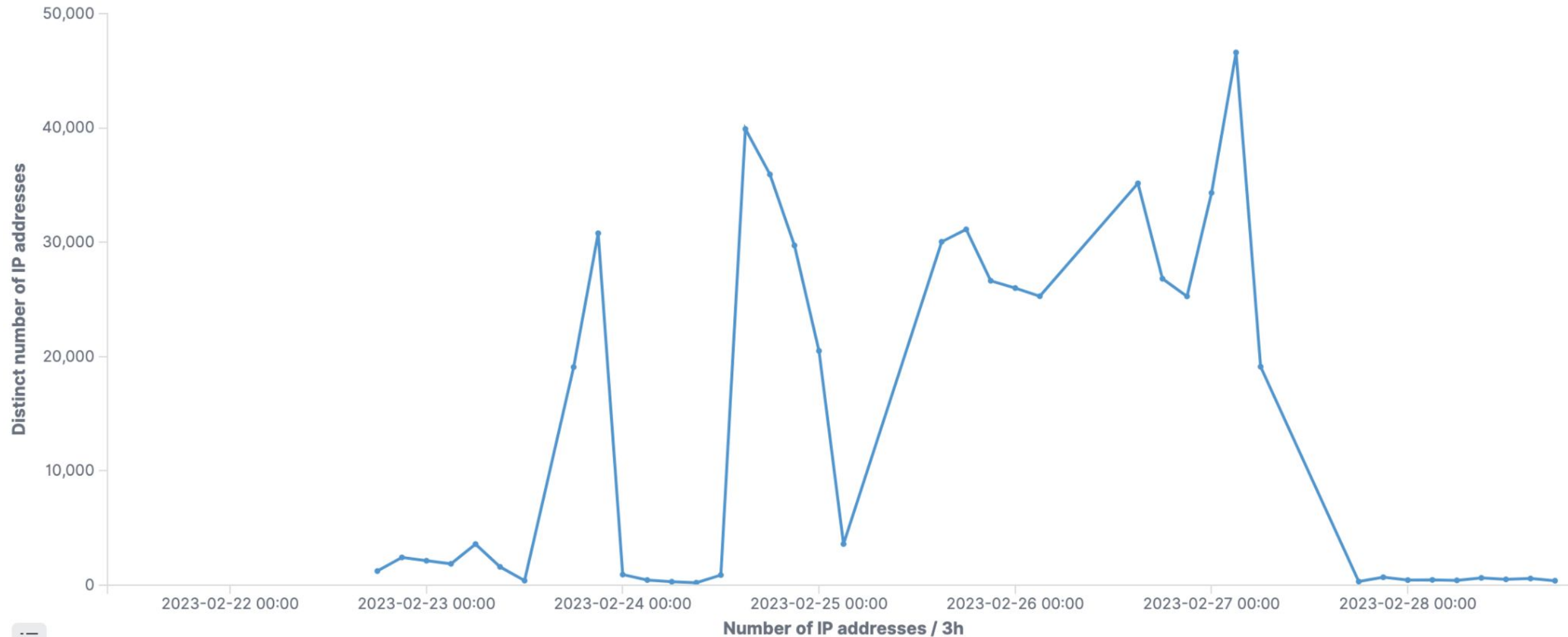
```
`Country`=`Germany` | slice size: 4149.0  
`User Agent`=`Chrome` && `Country`=`Germany` | slice size: 4133.0  
`Country`=`Germany` && `Accept Language`=`en-US,en;q=0.9` | slice size: 4097.0  
`User Agent`=`Chrome` && `Country`=`Germany` && `Accept Language`=`en-US,en;q=0.9` | slice size: 4097.0
```

Credential Stuffing Attack on Gaming Platform



Blocked more than **3M requests** in a week.

Heavily Distributed Attack: > 187k IP Addresses



Takeaways

Approach to **detect distributed attacks** using traffic aggregations and anomaly detection.

We leverage Sliceline to **infer malicious signatures and to generate rules**.

Efficient against **bots that frequently adapt** and modify/forge their fingerprints.

Sliceline **can be applied to other security use cases** that rely on a rule engine.

Contact

DATA  DOME

Antoine Vastel PhD,
Head of Research @DataDome,
antoine.vastel@datadome.co.

Konstantina Kontoudi PhD,
Lead Data Scientist @DataDome,
konstantina.kontoudi@datadome.co.