Sprocket Security

Watching the Watchers

Analyzing URL Scanning Solutions at Scale

Bypassing modern security controls using “httpot”
AKA the reverse **honeypot.**

WATCHING THE WATCHERS

Meet the **Speaker**

SPEAKER

Nicholas Anastasi

Director of Technical Operations / Hacker

with **Sprocket Security**

nanastasi@sprocketsecurity.com • @sprocket_ed

Enjoys: Python • Candy • Running

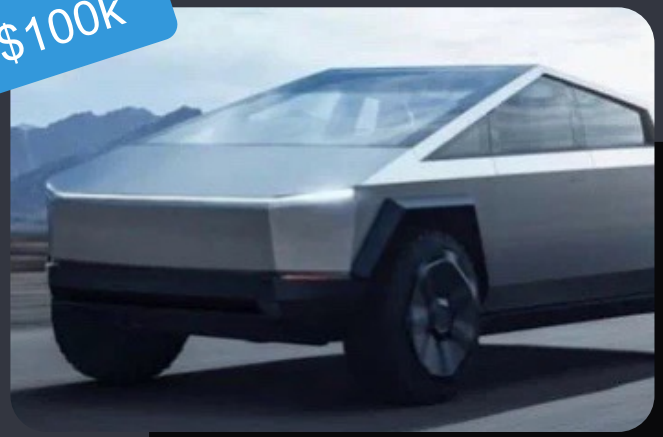
YOU MAY HAVE HEARD…

> Defenders only have to make one mistake for the red team to win.
>
> —Twitter

BLUE TEAM VS RED TEAM

A comparison of **Tools.**

$100k



BLUE TEAM

- Five EDRs
- Advanced Email Gateways & Firewalls
- SIEM Solutions
- Purple Team Engagements

VS

$3.50

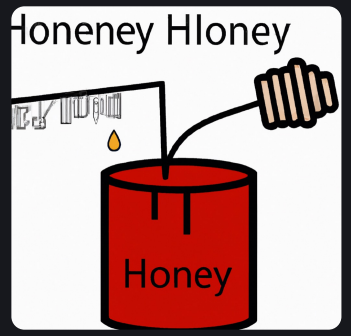
RED TEAM

- Dysfunctional Open-Source Tools
- C2 Solutions by crazies
- Blog articles & Twitter feeds
- Maybe in-house software

GOING UP AGAINST THE BEHEMOTH

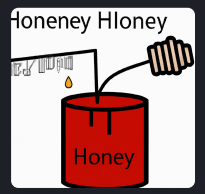
How do we take on **Enterprise Software?**

In-House Software Solutions!



Phishing Sucks

LET ME TELL YOU WHY



PHISHING SUCKS

The Problem

Scanning Solutions

Crawler Internals

Crawler Detection

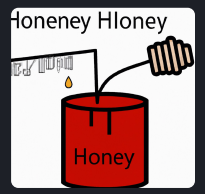
Evasion Methods

> “Why aren’t my phishing email landing in user inboxes?
>
> 
>
> —You

Having to link to your payload or landing page has its setbacks...

Deeply scrutinized by security controls

Traditional evasion methods are working less and less



PHISHING SUCKS

The Problem

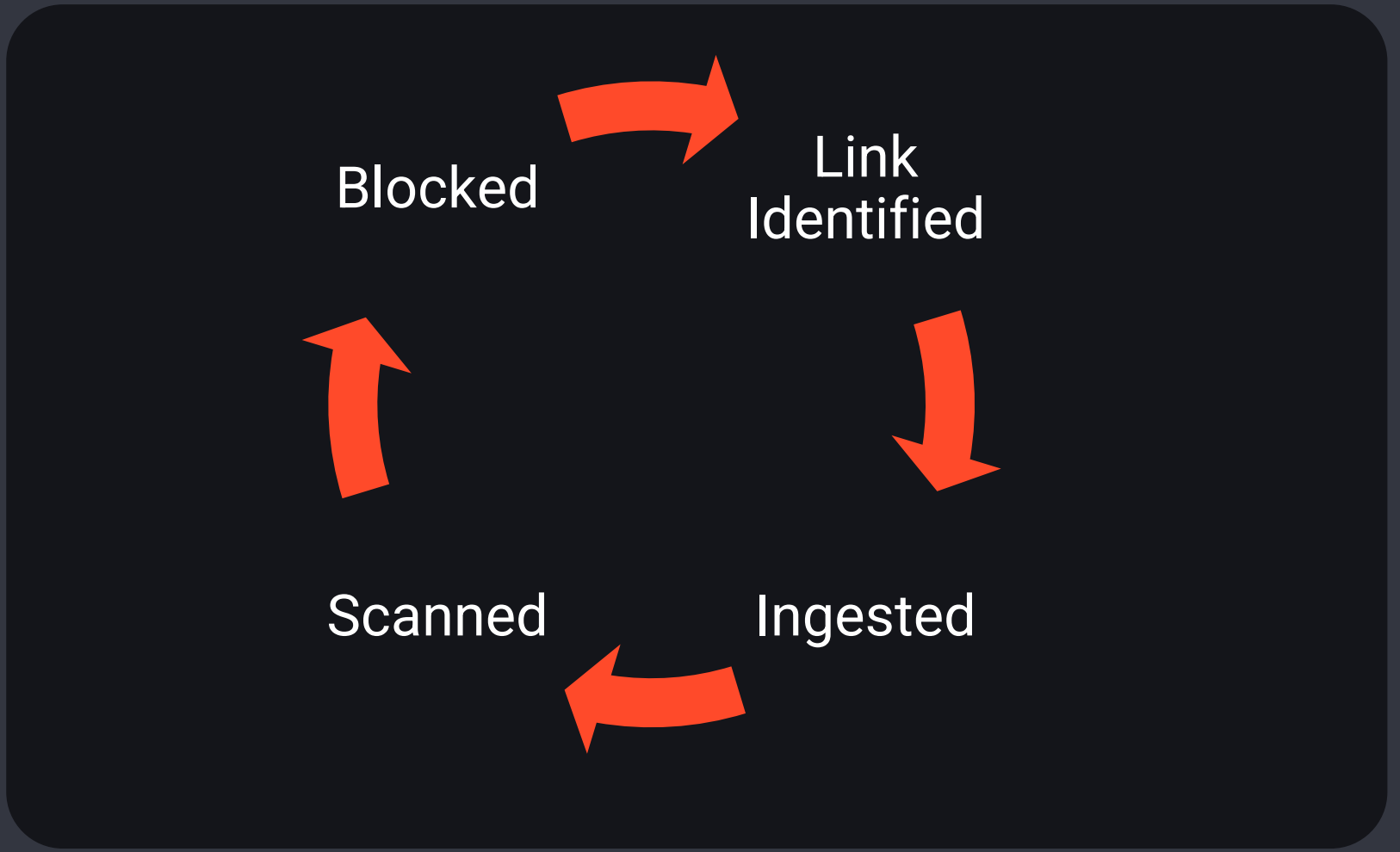
Scanning Solutions

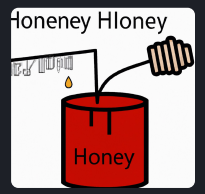
Crawler Internals

Crawler Detection

Evasion Methods

URL Scanning





PHISHING SUCKS

The Problem

Scanning Solutions

Crawler Internals

Crawler Detection

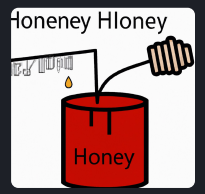
Evasion Methods

Crawler Internals

Using headless Chrome

Emulate user clicks and interactions

Scale and can crawl millions of pages at once



PHISHING SUCKS

The Problem

Scanning Solutions

Crawler Internals

Crawler Detection

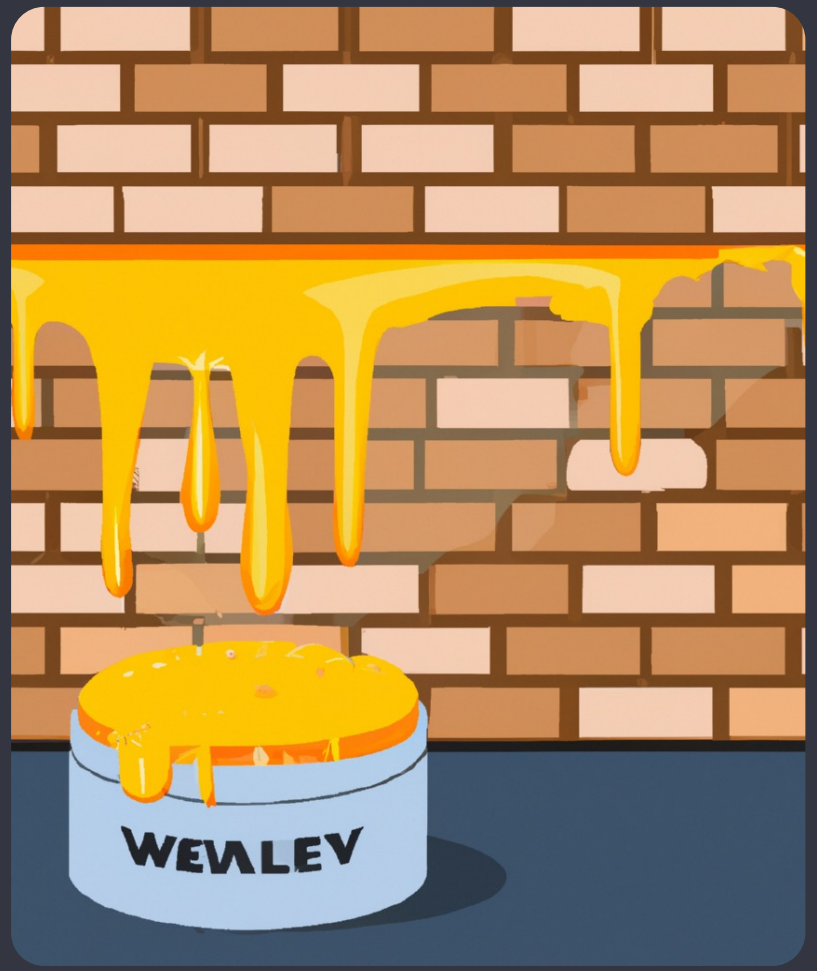
Evasion Methods

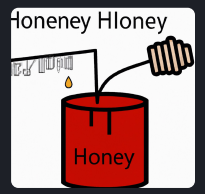
Crawler Detection

Cloudflare

Cloudfront

BotD





PHISHING SUCKS

The Problem

Scanning Solutions

Crawler Internals

Crawler Detection

Evasion Methods

Current Evasion Methods

HTML Smuggling

Apache Redirect Rules (sneaky_proxy)

Redirect bouncing and meta refreshes

Introduction to httpot

A HONEY ~~HOME~~ BAKED SOLUTION

INTRODUCTION

Honeypots

Reverse Honeypot

httpot

> A honeypot is a decoy system or network set up to simulate a vulnerable target and attract cyber attackers.

—ChatGPT

INTRODUCTION

Honeypot

Reverse Honeypot

httpot

COLLECTING INFORMATION ABOUT DEFENDERS

Links within phishing emails are scanned for malicious content.

Simulate anti-bot solutions to identify data points detecting whether the visitor is human.

Most Security Solutions clicking links don't have the tech to evade their own products.

INTRODUCTION

Honeypot

Reverse Honeypot

httpot

Core functions of httpot

Hosts a webpage including 50+ checks to collect info about user/systems.

Stores info and submits it to an Elasticsearch instance for analysis.

How we entice scanners to crawl our links

Generate unique URLs with a UUID and provider name.

Send emails to several domains using email gateways/firewalls.

Submits the URL to VirusTotal and URLScan.

“Providers” crawl links and info about the engine and stored in a JSON blob.

…Then the JSON blob is submitted to Elasticsearch for analysis.

httpot

MEAT & POTATOES

ENDPOINTS

- Generate
- Submit
- Landing Page
- Load & Collect

```
# Provider  # UUID                                  # Timestamp
/virustotal/8397f9d5-7ae0-410f-922c-c8a3651b17e2/2023032514401
```


```
{
   "virustotal":"https://api.dev.proxy.events/virustotal/8397f9d5-7ae0-410f-922c-c8a3651b17e2/20230325144015",
   "urlscan":"https://api.dev.proxy.events/urlscan/b03bfebd-2317-4c6a-8619-783093f3c725/20230325144015",
   "sophos":"https://api.dev.proxy.events/sophos/8be6f2a4-7679-46d6-b840-f5d731ff093d/20230325144015",
   "barracuda":"https://api.dev.proxy.events/barracuda/ea28c340-8a1f-4c02-b247-3cb3f9cf3f06/20230325144015",
   "ironport":"https://api.dev.proxy.events/ironport/c59e2bd5-e9f8-44c0-8a4d-2d9c7c1e7b70/20230325144015",
   "trustwave":"https://api.dev.proxy.events/trustwave/1be69b63-3a99-4612-89d0-ba35ed240d3b/20230325144015",
   "symantec":"https://api.dev.proxy.events/symantec/be3c607e-4664-4aba-8d59-86917e0b0e36/20230325144015",
   "clearswift":"https://api.dev.proxy.events/clearswift/527bd1f8-d29e-4efa-a0ad-8f2b2ee5e182/20230325144015",
   "macafee":"https://api.dev.proxy.events/macafee/afd95855-dc6a-454d-970f-d254441bda09/20230325144015",
   "fireeye":"https://api.dev.proxy.events/fireeye/c2618c7e-0957-44bf-a8e0-8ad9d83b04fe/20230325144015",
   "forcepoint":"https://api.dev.proxy.events/forcepoint/9a1ad52a-2adb-44f9-9df0-f2735ce5108c/20230325144015",
   "trendmicro":"https://api.dev.proxy.events/trendmicro/249c4614-9abe-4077-a2de-d55dd23e4898/20230325144015",
   "google":"https://api.dev.proxy.events/google/f9d7766e-9ab0-4363-879c-529615b24583/20230325144015",
   "microsoft":"https://api.dev.proxy.events/microsoft/72a083c6-3006-45bd-b097-b8b6eb04e5e2/20230325144015",
   "proofpoint":"https://api.dev.proxy.events/proofpoint/f6eb9e33-e51c-422a-a610-e7bc942549b1/20230325144015",
   "mimecast":"https://api.dev.proxy.events/mimecast/36b45741-944d-43cc-a787-5e98b9254744/20230325144015"
}
```

ENDPOINTS

- Generate
- Submit
- Landing Page
- Load & Collect

```
def _api_virustotal(self):
    """Submit VirusTotal API request"""

    # Getting a timestamp for the submission itself
    timestamp = arrow.utcnow().format("YYYYMMDDHHmmss")

    # Setting up the data for the request
    data = {
        "apikey": f"{self.config['targets']['api']['virustotal']['apikey']}",
        "url": f"{self.urls['virustotal']}",
    }

    # Issuing the API request
    response = requests.post(
        "https://www.virustotal.com/vtapi/v2/url/scan",
        data=data,
    )

    # Storing metadata about the submission
    self.submission_data["api"]["virustotal"] = {
        "timestamp": timestamp,
        "url": self.urls["virustotal"],
        "response_code": response.status_code,
        "response": response.json(),
    }
```

ENDPOINTS

- Generate
- Submit
- Landing Page
- Load & Collect

```yaml
# Mail gateway targets
mail:
  sophos:
    domain: ocwen.com
  barracuda:
    domain: terra-gen.com
  ironport:
    domain: cisco.com
  trustwave:
    domain: lek.com
  symantec:
    domain: lenovo.com
  clearswift:
    domain: globalbankers.com
  macafee:
    domain: pfgc.com
  fireeye:
    domain: alcoa.com
  forcepoint:
    domain: musl.com
  trendmicro:
    domain: amarfinance.com
  google:
    domain: match.com
  microsoft:
    domain: cdw.com
  proofpoint:
    domain: wellsfargo.com
  mimecast:
    domain: iwgr.com
```

```python
def _mailgun(self, domain, junk_user, subject, html, target):
        """Send emails via mailgun"""

        # Getting a timestamp for the submission itself
        timestamp = arrow.utcnow().format("YYYYMMDDHHmmss")

        # Define junk users for sending
        junk_users = [
            "contact",
            "sales",
            junk_user,
        ]

        # Issue API request to mailgun
        try:
            for user in junk_users:
                response = requests.post(

                    # Mailgun API endpoint
                    "https://api.mailgun.net/v3/entitymail.com/messages",
                    auth=("api", f"{self.config['mailgun']['api_key']}"),

                    # Email data to send
                    data={
                        "from": f'John Smith <{self.config["mailgun"]["username"]}>',
                        "to": f"{user}@{domain}",
                        "subject": subject,
                        "html": html,
                    },
                )
                log.info(f"Successfully sent email to: {domain}")
        except Exception as a:
            log.error(f"Failed to send email to: {domain}")
            pass
```

ENDPOINTS

- Generate
- Submit
- Landing Page
- Load & Collect

Hello!

Click this link please:
{{ url }}

Click Here

Company Inc, 3 Abbey Road, San Francisco CA 94102
Don't like these emails? Unsubscribe.

ENDPOINTS

- Generate
- Submit
- Landing Page
- Load & Collect

```
Failed to load resource: net::ERR_NAME_NOT_RESOLVED
Failed to load resource: net::ERR_BLOCKED_BY_CLIENT
Uncaught (in promise) TypeError: Failed to fetch dynamically imported module: h
Failed to load resource: the server responded with a status of 404 (Not Found)
Failed to load resource: net::ERR_BLOCKED_BY_CLIENT
```

```
<!DOCTYPE html>
<html>
<head>
    <title>Hi mom</title>
</head>
<body>
    <script src="https://api.dev.proxy.events/static/ua-parser.min.js"></script>
    <script src="https://api.dev.proxy.events/static/fpCollect.min.js"></script>
    <script src="https://api.dev.proxy.events/static/fpScanner.js"></script>
    <script src="https://detector.incolumitas.com/pd-lib.js"></script>
    <script src="https://api.dev.proxy.events/static/newTests.js"></script>
</body>
</html>
```

ENDPOINTS

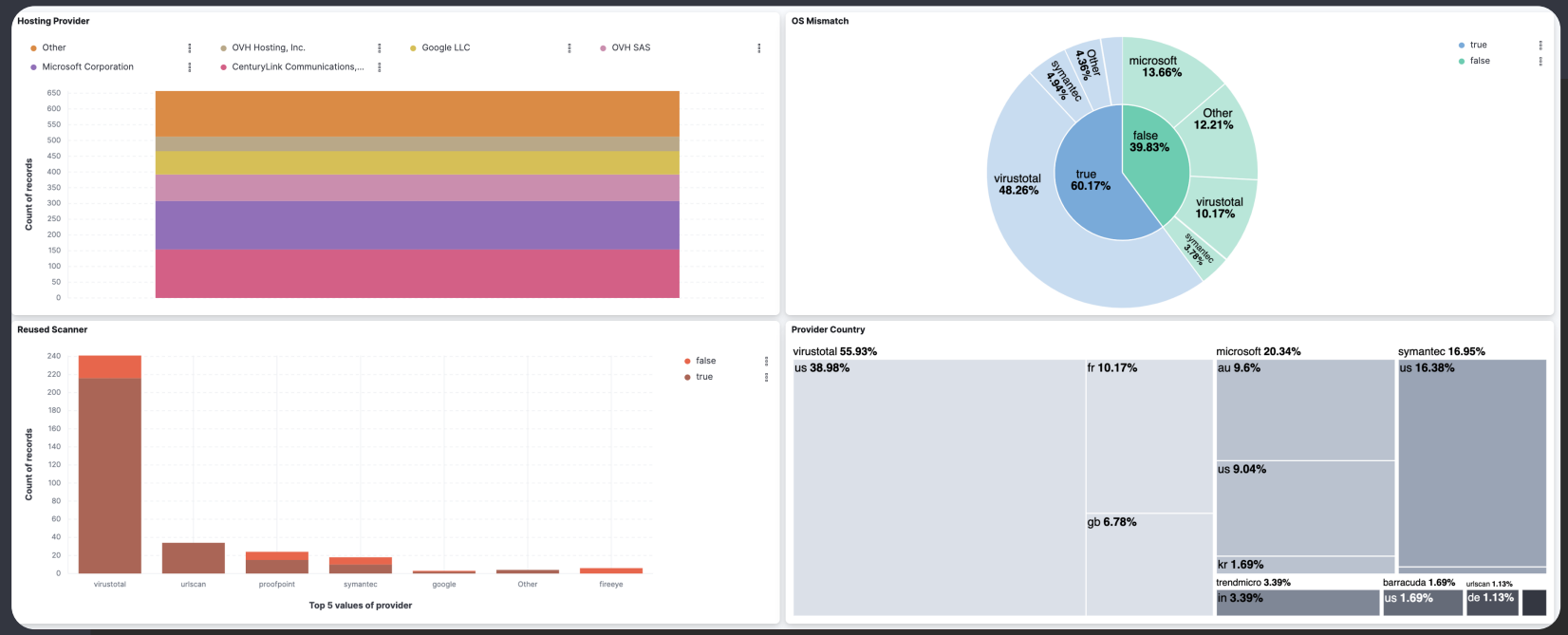
- Generate
- Submit
- Landing Page
- Load & Collect

```python
@router.post("/collect/{path:path}")
async def collect(request: Request, path: str):
    """Endpoint for ingesting metadata collected from scanners"""

    # awaiting the data
    results = await request.json()

    # Storing request information
    try:
        new_results = {}
        new_results["timestamp"] = int(arrow.utcnow().format("YYYYMMDDHHmmss"))
        new_results["provider"] = urlparse(path).path.split("/")[0]
        new_results["uuid"] = urlparse(path).path.split("/")[1]
        new_results.update(results)
    except Exception as e:
        log.error(e)
        pass

    try:
        # Loading the data and parsing it
        load = Load(new_results, path)
        load.preflight()

        # Transforming the data
        parsed = load.transform()
        load.submit(parsed)
        load.write(parsed)
    except Exception as e:
        log.error(e)
        pass

    # Returning a 200 response
    return await JSONResponse({"status": "ok"})
```

Findings

THE FUN STUFF

Kibana



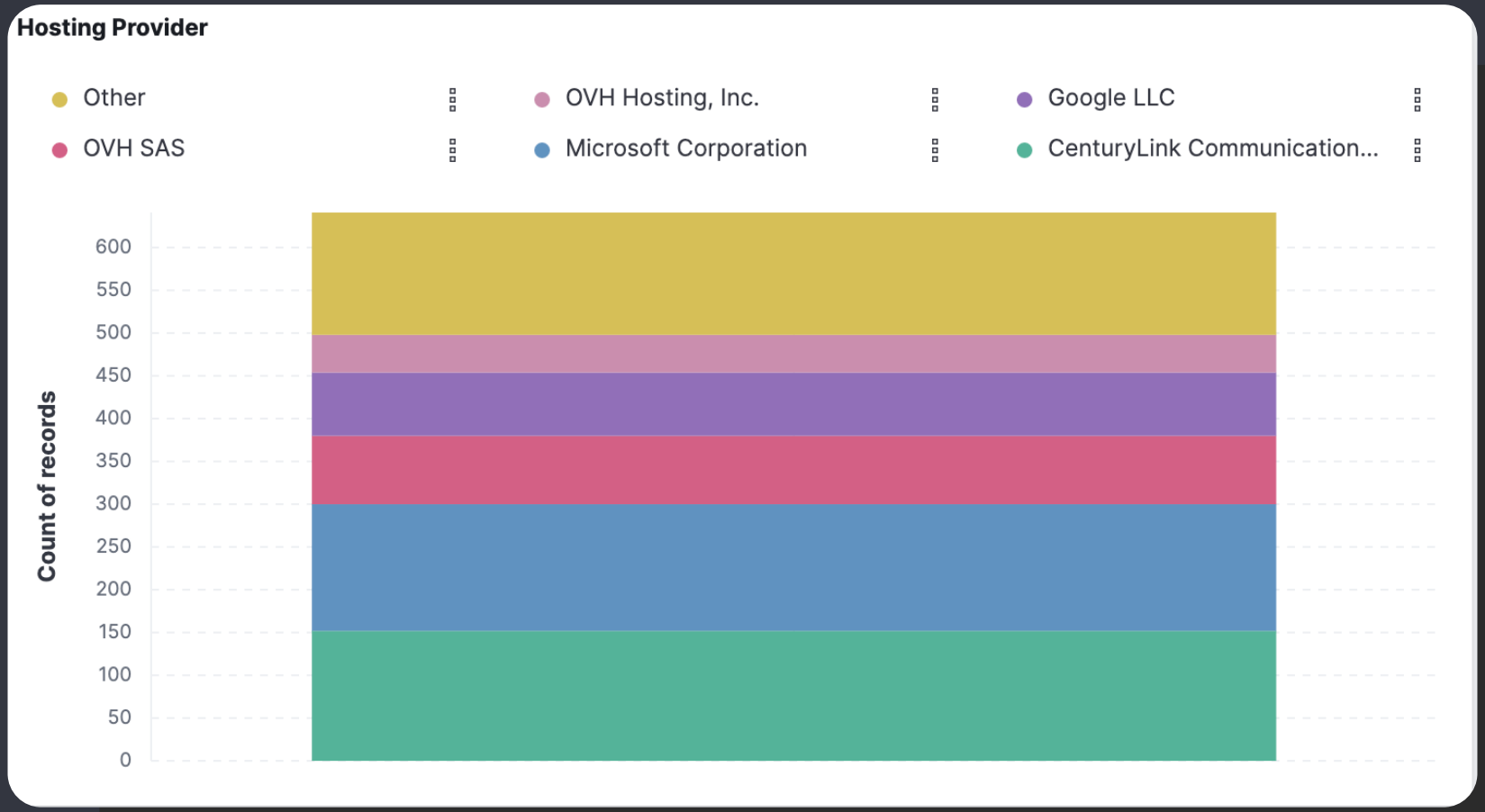
FINDINGS

- Kibana Overview
- Findings
- Proofpoint
- In Action

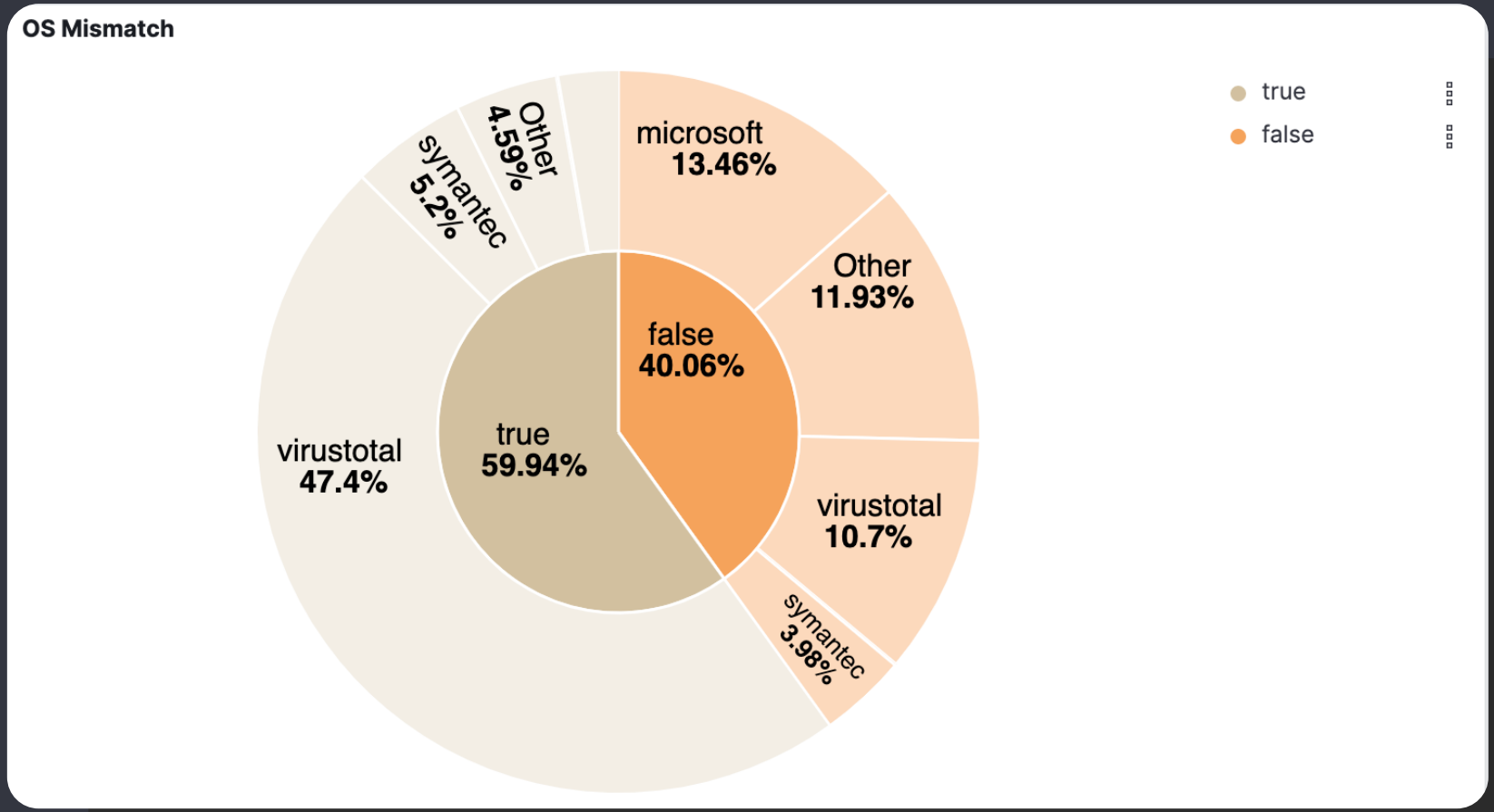
FINDINGS

- Kibana Overview
- Findings
- Proofpoint
- In Action

Source IP



TCP Mismatch



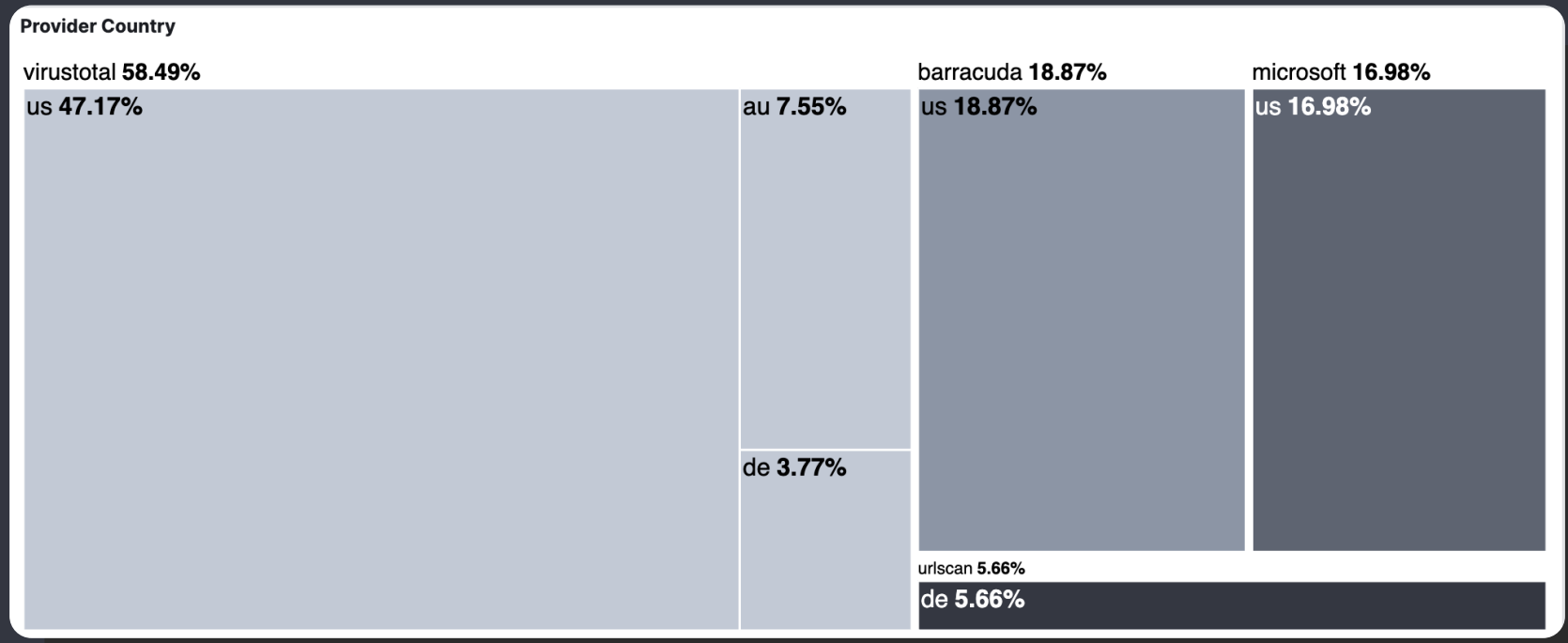
FINDINGS

- Kibana Overview
- Findings
- Proofpoint
- In Action

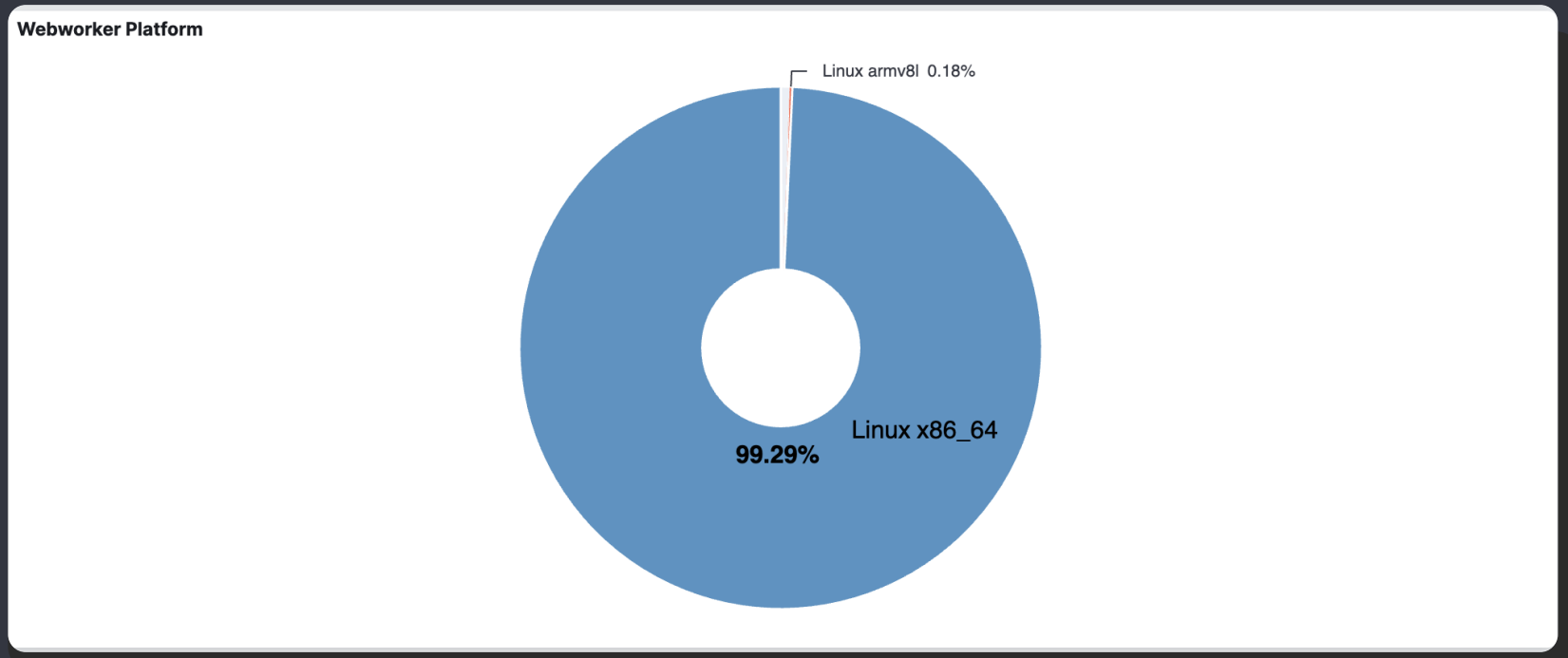
Source Country

FINDINGS

- Kibana Overview
- Findings
- Proofpoint
- In Action



Webworker Platform



FINDINGS

- Kibana Overview
- Findings
- Proofpoint
- In Action

FINDINGS

- Kibana Overview
- Source IP
- OS Mismatch
- Proofpoint
- In Action


```
    "webworker": {
      "userAgent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/
Safari/537.36",
      "appVersion": "5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/104.0.5
      "platform": "Linux x86_64",
      "deviceMemory": 8,
      "hardwareConcurrency": 56,
      "language": "en-US",
      "languages": [
    },
```

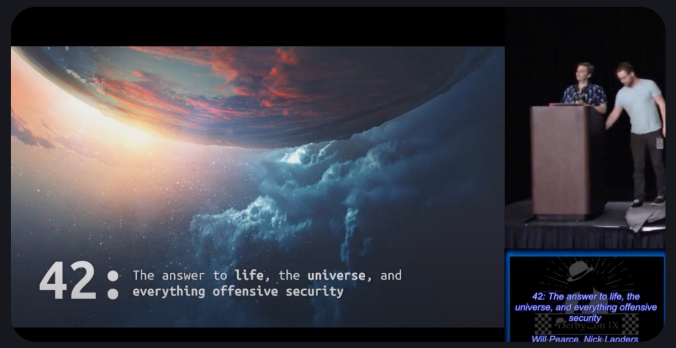
```
    "datacenter": {
      "ip": "205.169.39.63",
      "rir": "ARIN",
      "is_bogon": false,
      "is_datacenter": false,
      "is_tor": false,
      "is_proxy": false,
      "is_abuser": true,
      "company": {
        "name": "CenturyLink Communications, LLC",
        "domain": "aup.lumen.com",
        "network": "205.168.0.0 - 205.171.255.255",
        "whois": "https://api.incolumitas.com/?whois=205.168.0.0"
      },
```

FINDINGS

- Kibana Overview
- Source IP
- OS Mismatch
- Proofpoint
- In Action

```
// const definitions
const IP_FIELDS = [
    'ip',
    'is_bogon',
    'is_datacenter',
    'is_tor',
    'is_proxy',
    'is_vpn',
    'is_abuser',
];

const TCP_FIELDS = ['os_mismatch'];

// Primary function calls
Promise.all([
    fetchData('https://api.incolumitas.com/', IP_FIELDS),
    fetchData('https://tcpip.incolumitas.com/classify', TCP_FIELDS),
    workerPlatformString(),
]).then(([ipResult, tcpResult, isLinuxOrNoWebWorkers]) => {
    if (!ipResult || !tcpResult) {

        .map((key) => ipResult[key]);

    if (ipFlags.some((flag) => flag) || isLinuxOrNoWebWorkers ) {
        // Helping out creepjs ;)
        window.location.href = 'https://creepjs-
api.web.app/analysis';
    }
});
```

MAJOR CREDITS

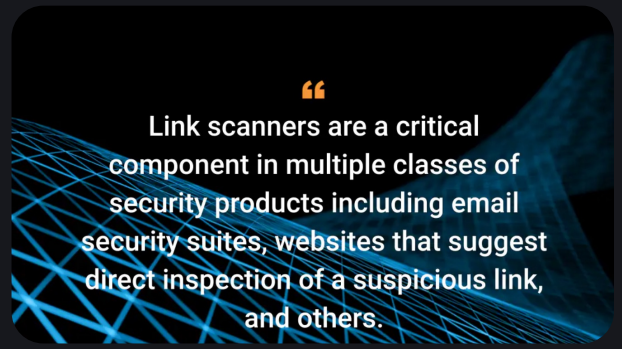


Proofpudding

Talk @ Derbycon

Detecting scraping

Blog Article



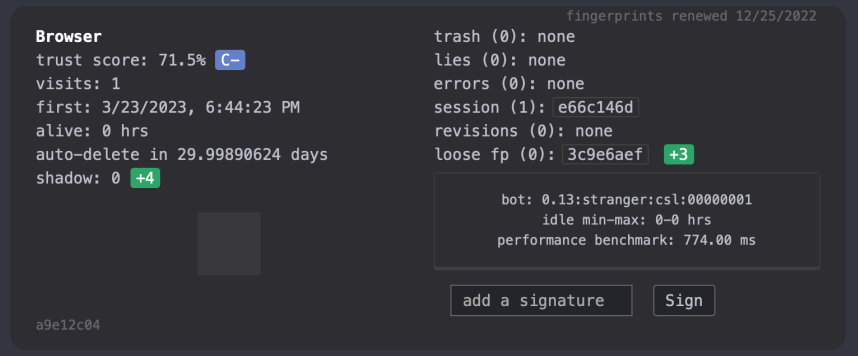
Evading Link Scanning

Blog Article

MY INSPIRATION

incolumitas.com

WEB APP SECURITY BLOG



abrahamjuliot.github.io/creepjs

BROWSER TESTING TOOL

LOOKING FORWARD

Email Gateways

Publish Data

Evade everything using data collection and finally stop guessing!

Thanks for joining!

Watching the Watchers

Analyzing URL Scanning Solutions at Scale

Brought to you by

Sprocket Security

With:

Nicholas Anastasi

nanastasi@sprocketsecurity.com

@sprocket_ed

DOWNLOAD THE PRESENTATION

https://sprkt.io/r00tz

www.sprocketsecurity.com/grassr00tz