Pass
The SALT

PyRASP

Python *Runtime Application Self Protection*

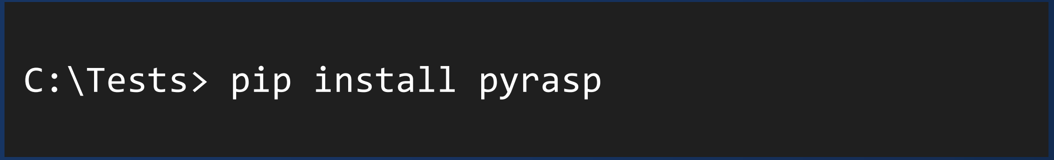
Defending your Python Web Application
From the Inside

Renaud Bidou

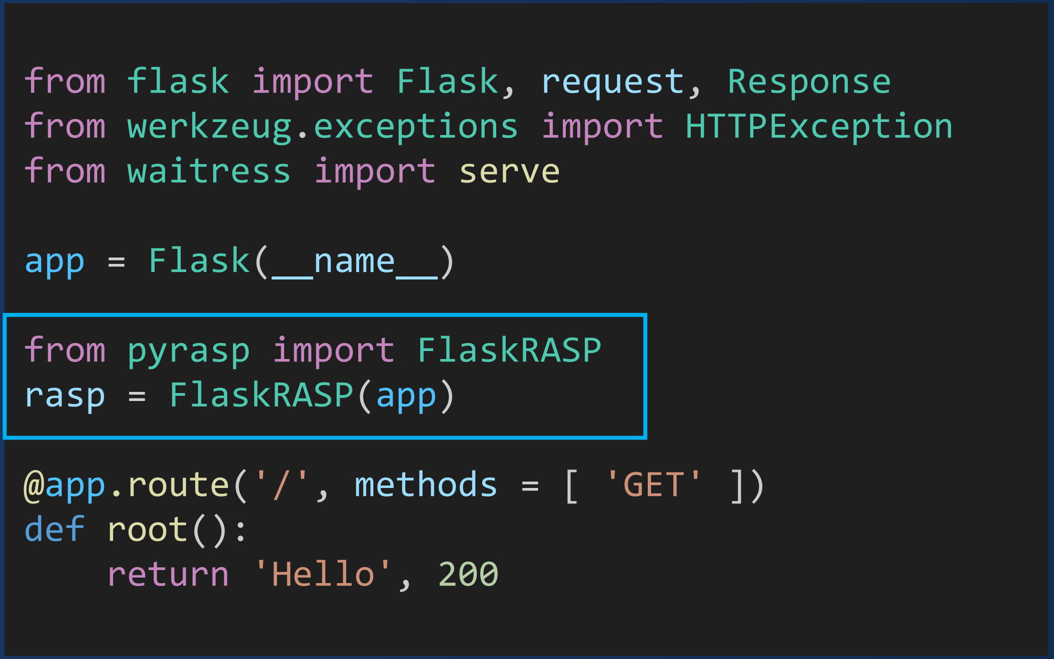
ParaCyberBellum

PyRASP 101

INSTALL & CODE

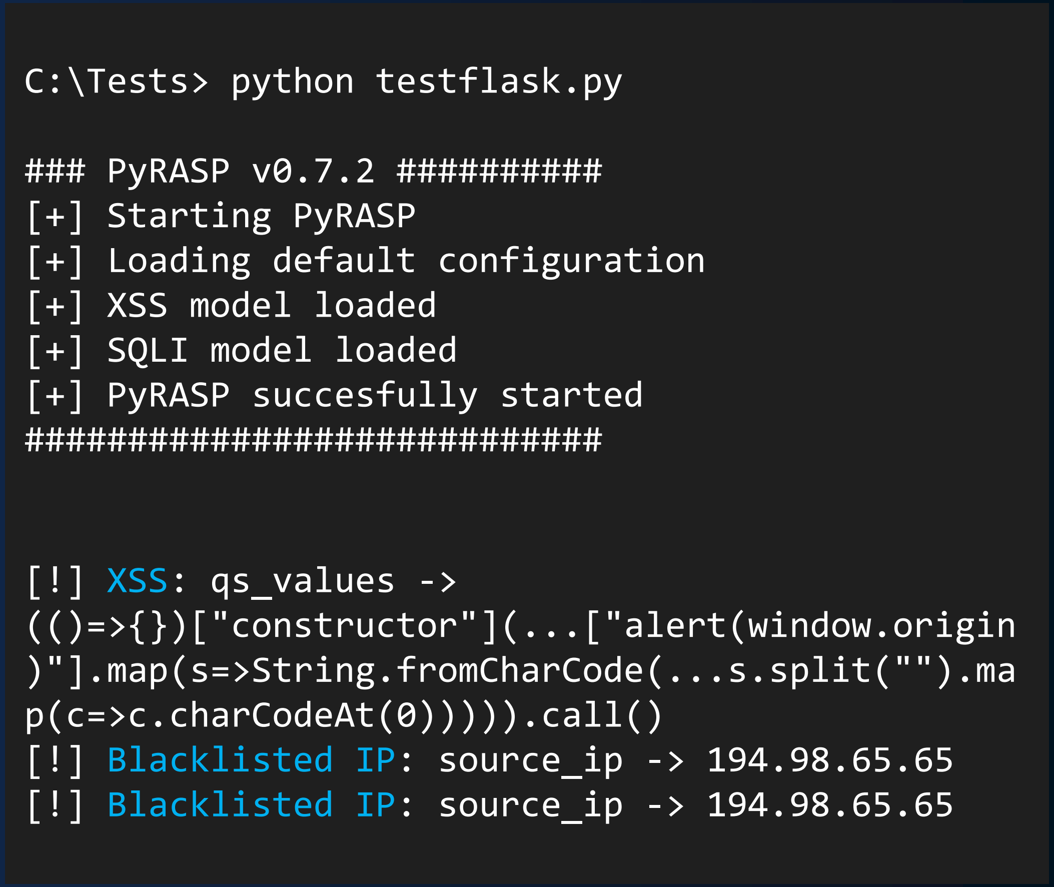


```
C:\Tests> pip install pyrasp
```



```python
from flask import Flask, request, Response
from werkzeug.exceptions import HTTPException
from waitress import serve

app = Flask(__name__)

from pyrasp import FlaskRASP
rasp = FlaskRASP(app)

@app.route('/', methods = [ 'GET' ])
def root():
    return 'Hello', 200
```

RUN & TEST



```
C:\Tests> python testflask.py

### PyRASP v0.7.2 ##########
[+] Starting PyRASP
[+] Loading default configuration
[+] XSS model loaded
[+] SQLI model loaded
[+] PyRASP succesfully started
############################


[!] XSS: qs_values ->
(()=>{})["constructor"](...["alert(window.origin
)"].map(s=>String.fromCharCode(...s.split("").ma
p(c=>c.charCodeAt(0))))).call()
[!] Blacklisted IP: source_ip -> 194.98.65.65
[!] Blacklisted IP: source_ip -> 194.98.65.65
```

DESIGN CRITERIA

1 → Secure & Signature-Free

2 → Lightweight

3 → Oneliner

4 → Distributed & Multi-Platform

5 → Useful Logging & Telemetry

WHY RASP ?

Natively Resistent

Request Smuggling

Encoding Tricks

HTTP Parameter Pollution

DevOps Friendly

Embedded in Application Code

Native CI/CD Pipeline Integration

Environment Aware

Targeted System Protection

Framework Specificities Handling

Access to Application Internals

MAIN SECURITY CHECKS & ENGINES

Flood & Brute Force	Threshold
Request Validation	Application Internals
Spoofing	Header Validation
Decoy	Path
SQL Injection	Grammatical Analysis + Machine Learning
XSS	Machine Learning
Command Injection	System Internals
HPP	Grouping
DLP	RegExp

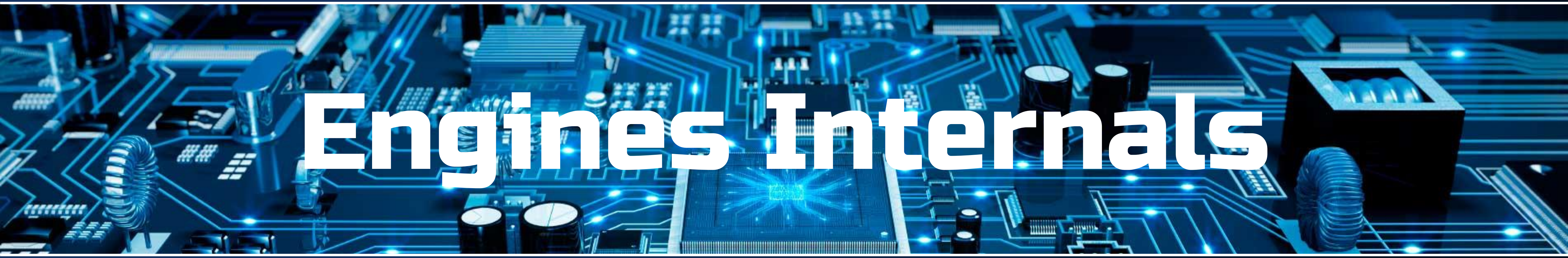
SUPPORTED PLATFORMS

FRAMEWORKS

- FLASK
- DJANGO
- FASTAPI

SERVERLESS

- AWS LAMBDA
- AZURE FUNCTIONS
- GCP FUNCTIONS



Engines Internals

INTERCEPTING REQUESTS & RESPONSES

```
def register_security_checks(self, app)
```

FLASK

```
@app.before_request
@app.after_request
```

FASTAPI

```
@app.middleware('http')
```

DJANGO

```
MIDDLEWARE = [ 'pyrasp.DjangoRASP’, … ]
def __call__(self, request)
```

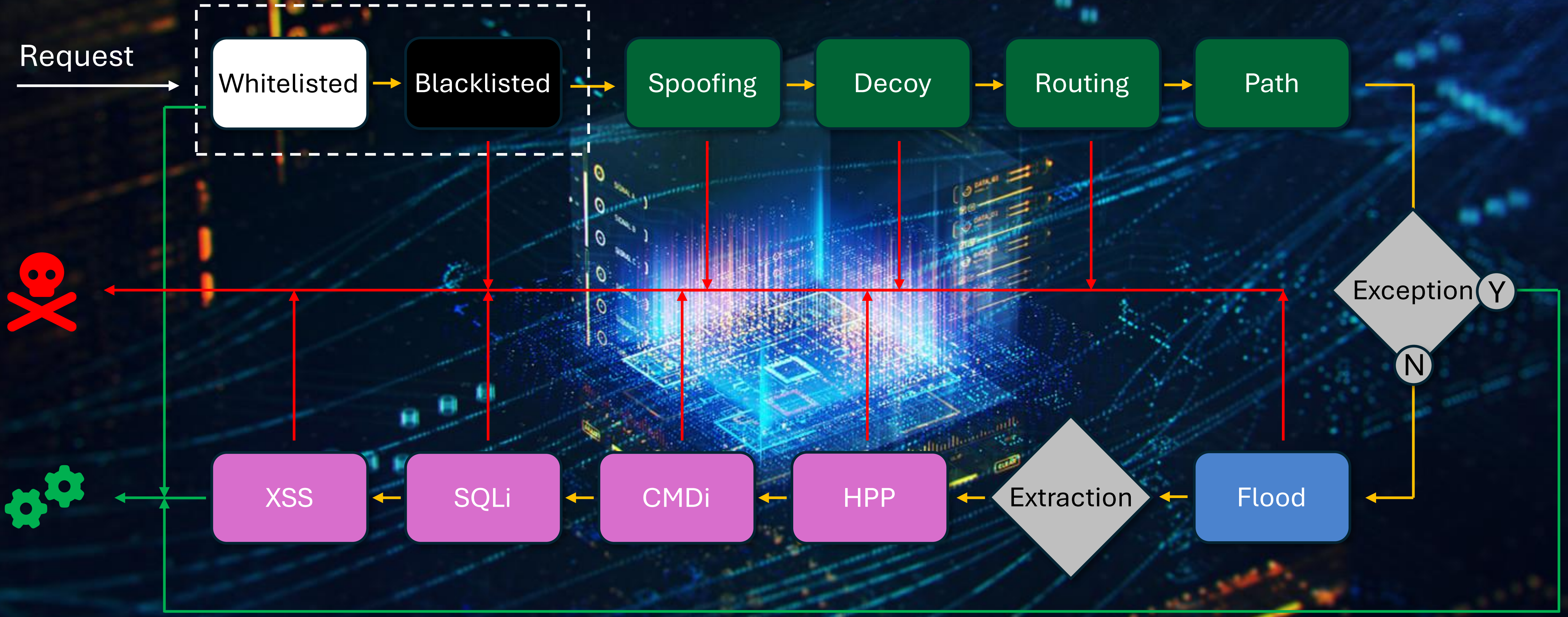
AWS LAMBDA

AZURE FUNCTIONS

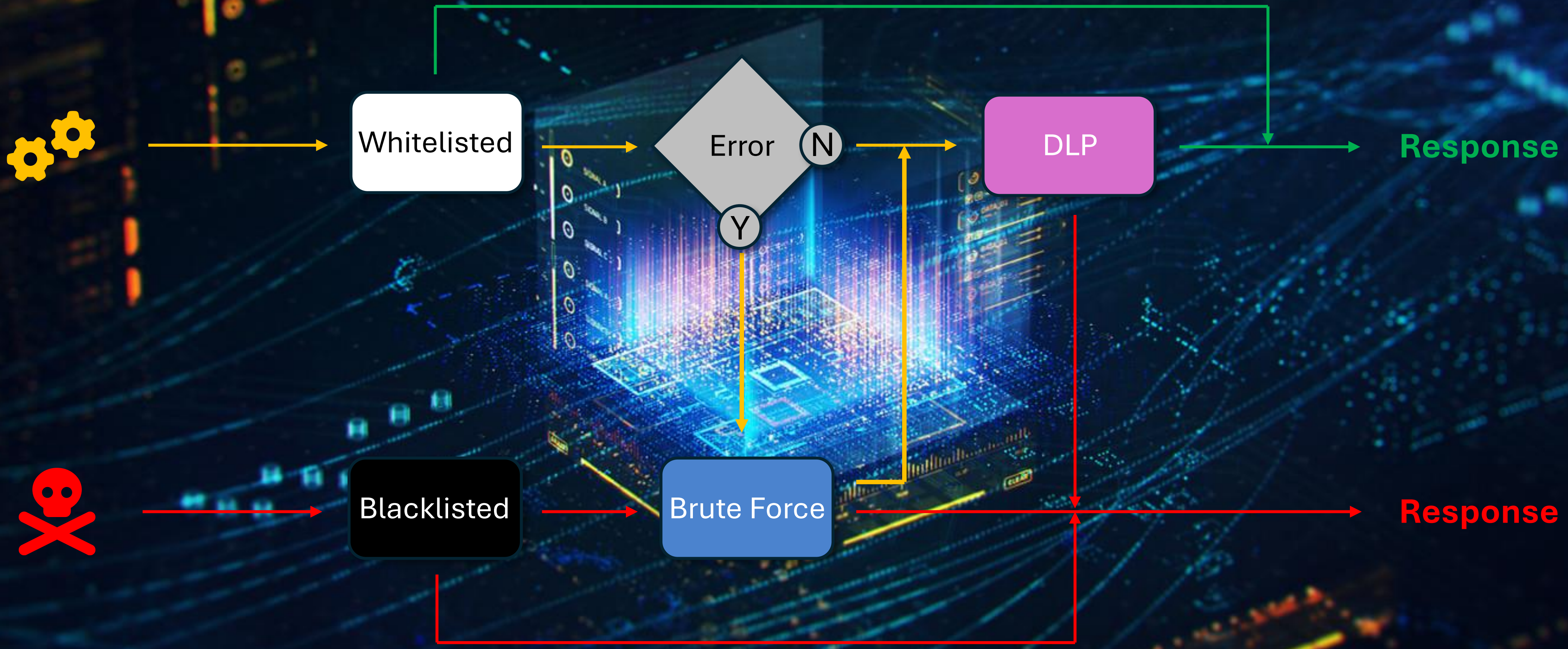
GCP FUNCTIONS

```
def decorator(request, context)
```

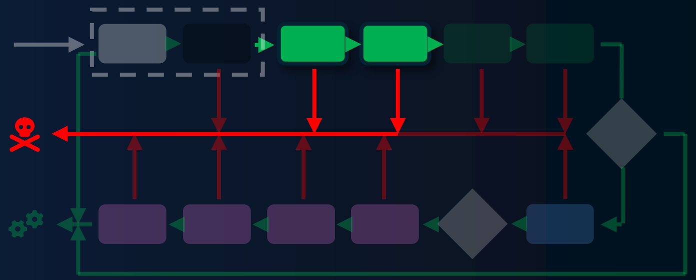
REQUEST PROCESSING OVERVIEW



RESPONSE PROCESSING OVERVIEW



SIMPLER (and most efficient) ENGINES



DECOYS

1. Set of commonly targeted paths

   `^/.env   ^/.git   ^/.aws   /wp-   …`

2. Attempt to connect ⇨ Blacklisted

* 0% False Positive

SPOOFING

1. Check `Host` header

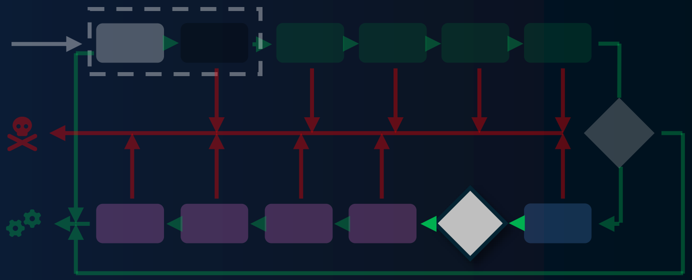
2. Doesn’t match configuration

   ⇨ Blacklisted

* Scanners & Direct access prevention

99.99% Early Attack Detection

EXTRACTION



BASICS

Headers Names & Values

- ✚ Cookies Names & Values
- ✚ Referer
- ✚ User Agent

Query String Variables & Values

Posted Data Variables & Values

JSON Data Variables & Values

TRICKS

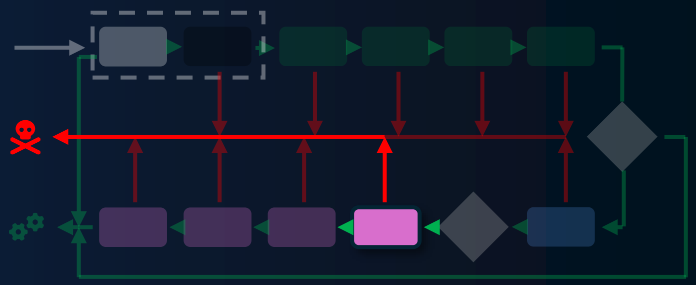
Base64 encoded values

- ➡ Decode
- ➡ Parse JSON
- ➡ Extract Variables & Values
- ➡ Base64 Decode
- ➡ Recurse (rare cases... so mandatory)

Example : JWT

HPP: TRIVIAL BUT...

Microsoft Azure Functions join duplicated parameters with comma



The Code

```
def testazure(req: func.HttpRequest) -> func.HttpResponse:
   params = dict(req.params)
```

The Query String

```
?a=select%20login&a=password/*&a=*/%20from%20/*&a=*/%20users#
```

The Outcome

```
params = {
   "a": "select login,password/*,*/ from /*,*/ users#"
}
```

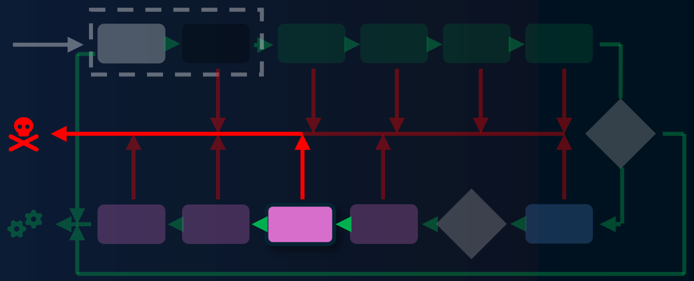
MSRC Case 87582

We examined your report and found that this is not a relevant security threat. The finding describes an assumption present in azure functions.

These are customer owned apps and at the http layer, we don't modify the format of any customer defined parameters. It's the responsibility of the customer to ensure that the parameters they're taking from the internet are not passed onto downstream components in an unsafe manner.

This is not a vulnerability. This case is now closed.

COMMAND INJECTION



1. Split stacked commands

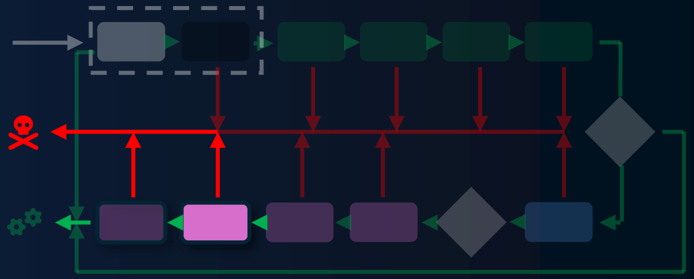
```
command_pattern = '(?:[&;|]|\$IFS)+\s*(\w+)'
commands = re.findall(command_pattern, str(injection)) or []
```

2. Call `shutil.which`

```
for command in commands:
    if shutil.which(command):
        command_injection = True
```

* Stick to the OS

GRAMMATICAL ANALYSIS



1. Define injection points

```
'select * from test where id={{vector}}'
```

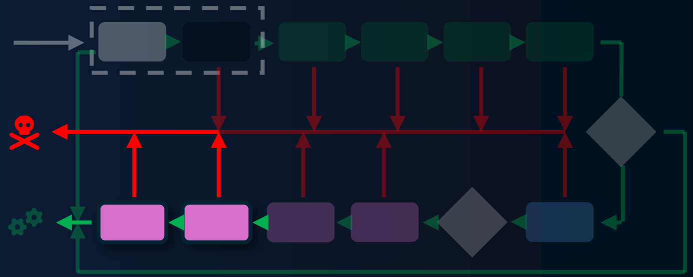
2. Replace {{vector}} with potential injection

3. Test statement against in-memory sqlite DB

```
temp_db = sqlite3.connect(":memory:")
try:
  temp_db.execute(statement)
except Exception as e:
  if 'no such table' in str(e):
    sql_injection = True
```

Was grammatically correct

XSS & SQLI ML ENGINES



Input Data

	XSS	SQLi
Valid	8499	1706
Attacks	8517	1546

Vectorizer

	XSS	SQLi
Features	32645	11974

Classification

Random Forest Classifier

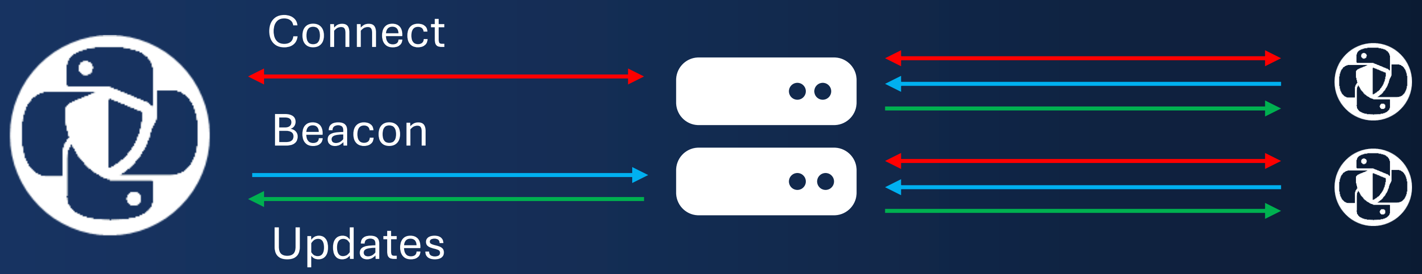
	XSS	SQLi
Accuracy	0.99953	0.99955
Precision	1.00000	0.99901
Recall	0.99906	1.00000
F1 Score	0.99953	0.99950

False-Negative: Recall

False-Positive: Precision

S’More…

DISTRIBUTED INFRASTRUCTURE



Connect

Routes upload
Configuration download
Blacklist download

Beacons

New blacklist entries
Telemetry

Updates

Blacklist updates (new – delete)
Configuration changes

✱ **Blacklisted once** ⇨ **Blacklisted everywhere**

LOGS

Syslog

```
[<event_time>] "<application_name>" - "<event_type>" - "<source_ip>" - "<country>" - "<location>:<payload>",
"<mitre_code> - <pcb_code>", "<action>"
```

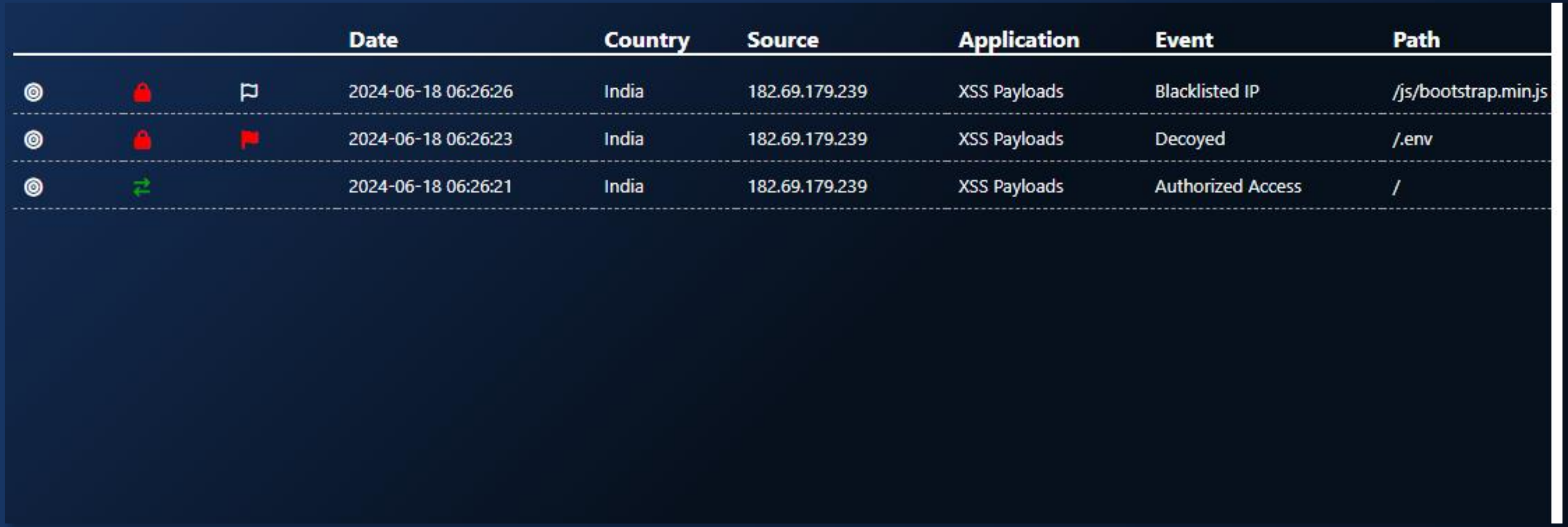
JSON / Webhook

```
{
   "time": "<event_time>",
   "application": "<application_name>",
   "log_data": [
      "<event_type>",
      "<source_ip>",
      "<country>",
      {
         "path": "<path>",
         "location": "<location>",
         "payload": "<payload>",
         "codes": "<codes>",
         "action": "<action>",
         "engine": "<engine>",
         "score": "<machine_learning_score>"
        }
    ]
}
```

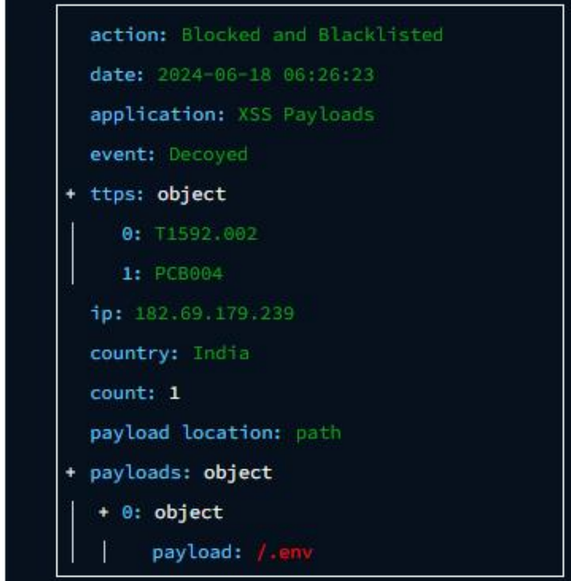
Beacon Telemetry

```
{
   "key": "<agent-key>",
   "version": "<agent-version>",
   "telemetry": {
      "cpu": <cpu_usage_percent>,
      "memory": <memory_usage_percent>,
      "requests": {
         "success": <valid_count>,
         "error": <errors_count>,
         "attacks": <attacks_count>
      }
   }
}
```

LOGS USAGE

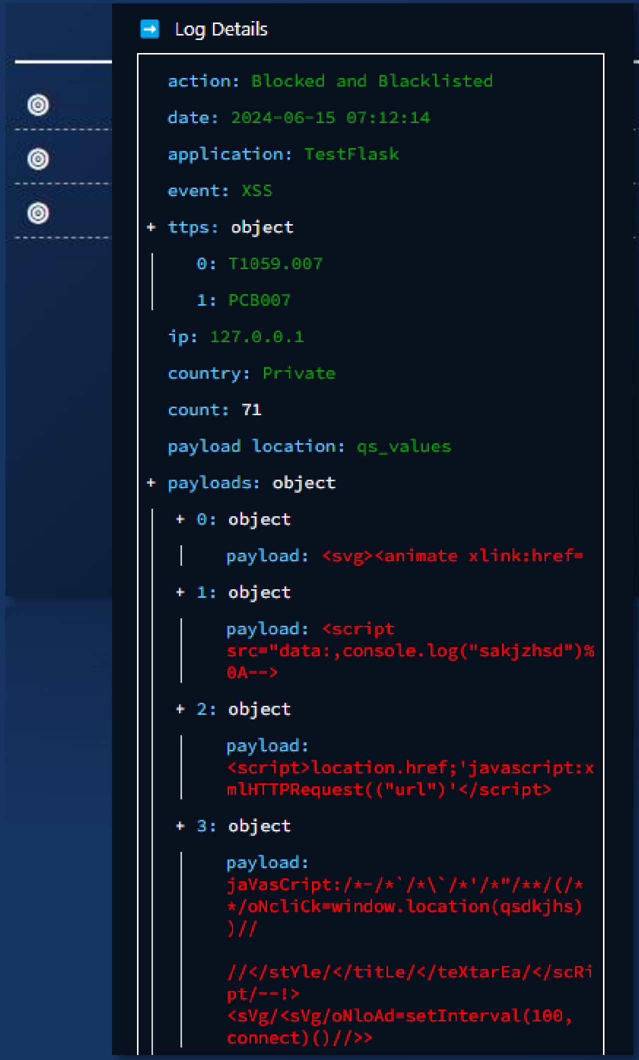


			Date	Country	Source	Application	Event	Path
◎	🔒	⚐	2024-06-18 06:26:26	India	182.69.179.239	XSS Payloads	Blacklisted IP	/js/bootstrap.min.js
◎	🔒	⚑	2024-06-18 06:26:23	India	182.69.179.239	XSS Payloads	Decoyed	/.env
◎	⇄		2024-06-18 06:26:21	India	182.69.179.239	XSS Payloads	Authorized Access	/

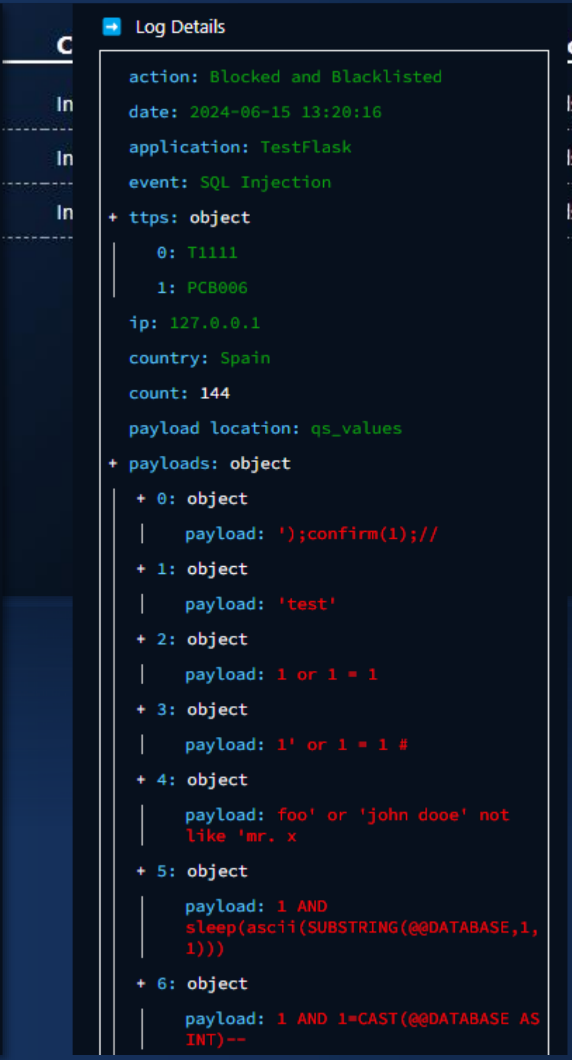


```
  action: Blocked and Blacklisted
  date: 2024-06-18 06:26:23
  application: XSS Payloads
  event: Decoyed
+ ttps: object
|   0: T1592.002
|   1: PCB004
  ip: 182.69.179.239
  country: India
  count: 1
  payload location: path
+ payloads: object
|  + 0: object
|  |   payload: /.env
```

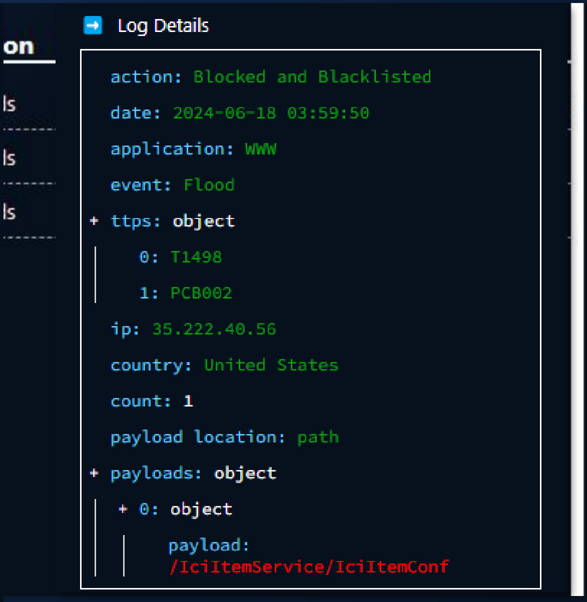
LOGS USAGE


Log Details

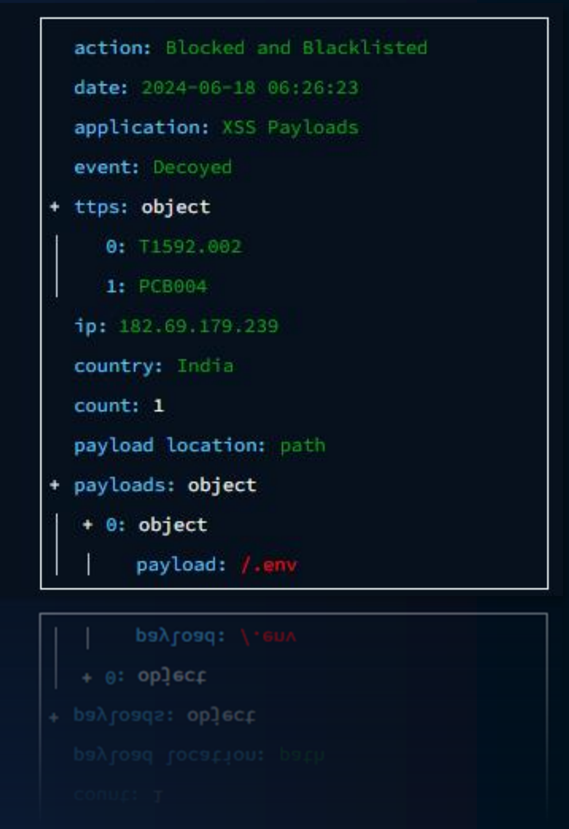
```
action: Blocked and Blacklisted
date: 2024-06-15 07:12:14
application: TestFlask
event: XSS
+ ttps: object
    0: T1059.007
    1: PCB007
ip: 127.0.0.1
country: Private
count: 71
payload location: qs_values
+ payloads: object
  + 0: object
        payload: <svg><animate xlink:href=
  + 1: object
        payload: <script
        src="data:,console.log("sakjzhsd")%
        0A-->
  + 2: object
        payload:
        <script>location.href;'javascript:x
        mlHTTPRequest(("url")'</script>
  + 3: object
        payload:
        jaVasCript:/*-/*`/*\`/*'/*"/**/(/*
        */oNcliCk=window.location(qsdkjhs)
        )//

        //</stYle/</titLe/</teXtarEa/</scRi
        pt/--!>
        <sVg/<sVg/oNloAd=setInterval(100,
        connect)()//>>
```


Log Details

```
action: Blocked and Blacklisted
date: 2024-06-15 13:20:16
application: TestFlask
event: SQL Injection
+ ttps: object
    0: T1111
    1: PCB006
ip: 127.0.0.1
country: Spain
count: 144
payload location: qs_values
+ payloads: object
  + 0: object
        payload: ');confirm(1);//
  + 1: object
        payload: 'test'
  + 2: object
        payload: 1 or 1 = 1
  + 3: object
        payload: 1' or 1 = 1 #
  + 4: object
        payload: foo' or 'john dooe' not
        like 'mr. x
  + 5: object
        payload: 1 AND
        sleep(ascii(SUBSTRING(@@DATABASE,1,
        1)))
  + 6: object
        payload: 1 AND 1=CAST(@@DATABASE AS
        INT)--
```


Log Details

```
action: Blocked and Blacklisted
date: 2024-06-18 03:59:50
application: WWW
event: Flood
+ ttps: object
    0: T1498
    1: PCB002
ip: 35.222.40.56
country: United States
count: 1
payload location: path
+ payloads: object
  + 0: object
        payload:
        /IciItemService/IciItemConf
```

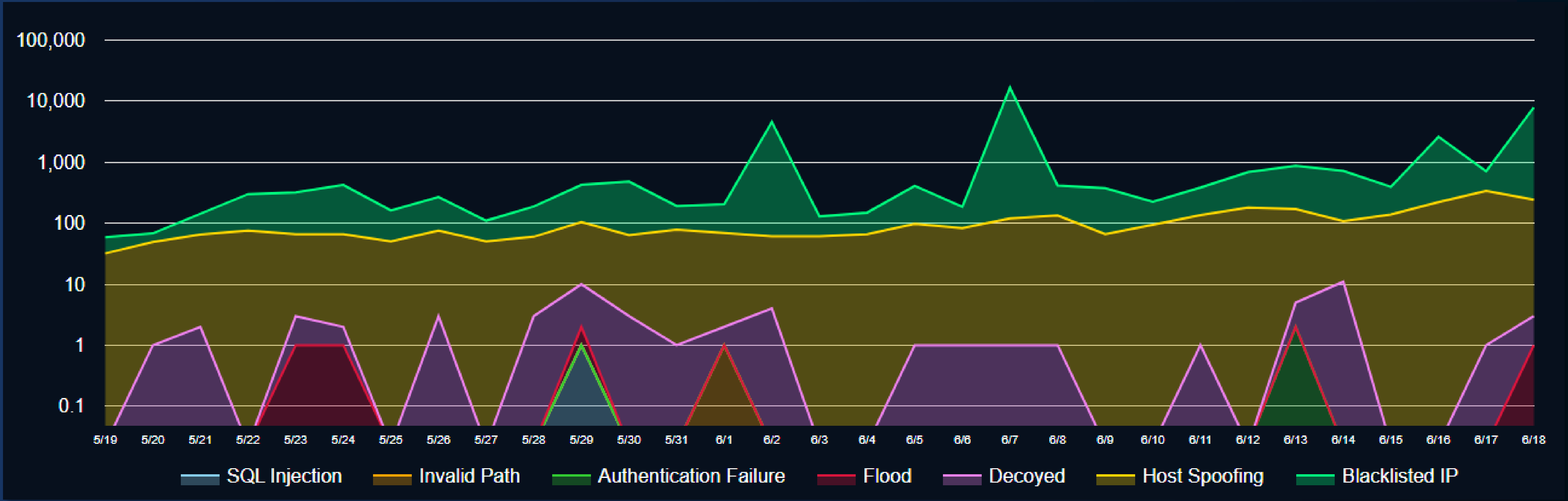


```
action: Blocked and Blacklisted
date: 2024-06-18 06:26:23
application: XSS Payloads
event: Decoyed
+ ttps: object
    0: T1592.002
    1: PCB004
ip: 182.69.179.239
country: India
count: 1
payload location: path
+ payloads: object
  + 0: object
        payload: /.env
```

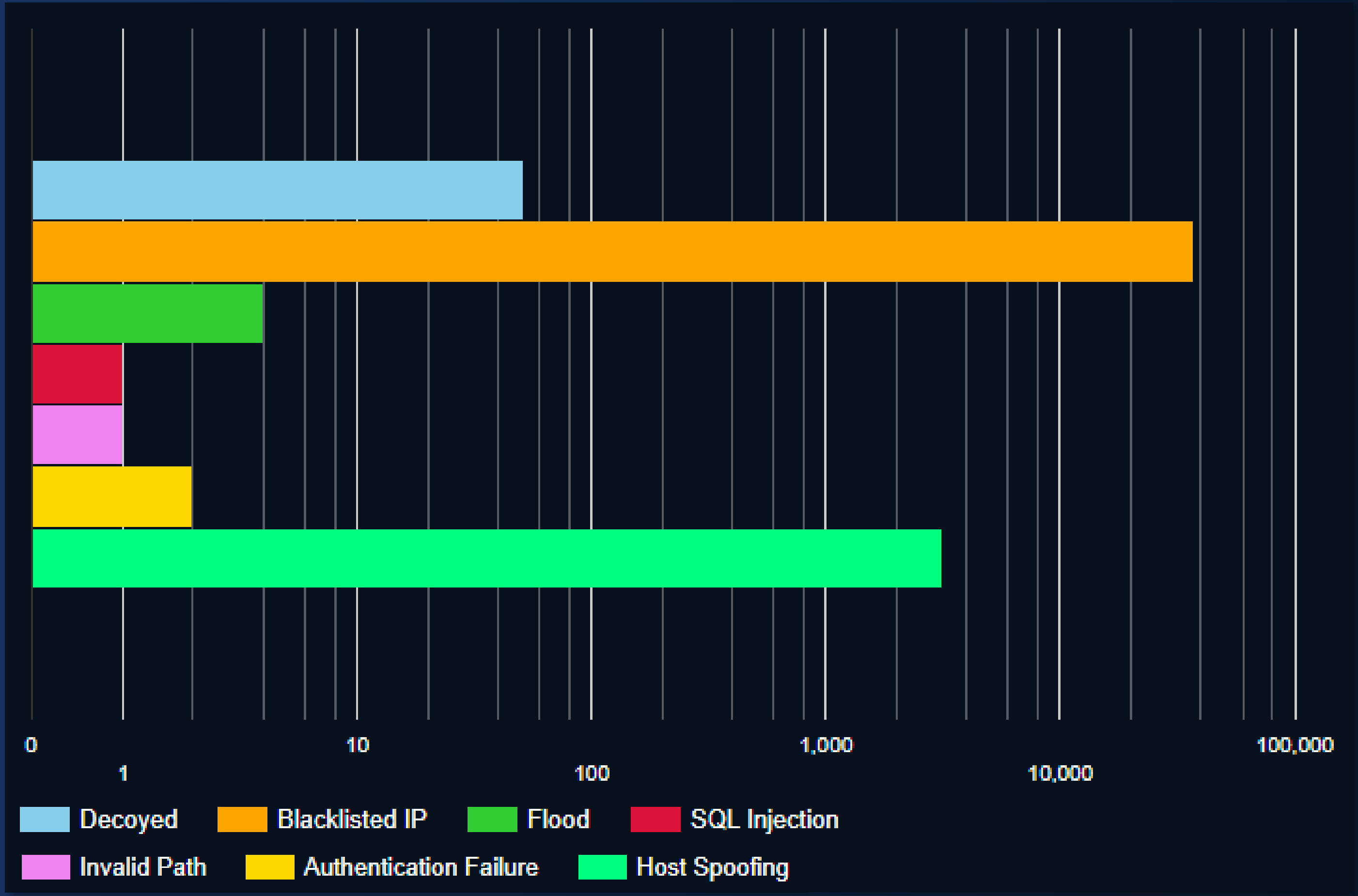
ParaCyberBellum

Pass
The SALT

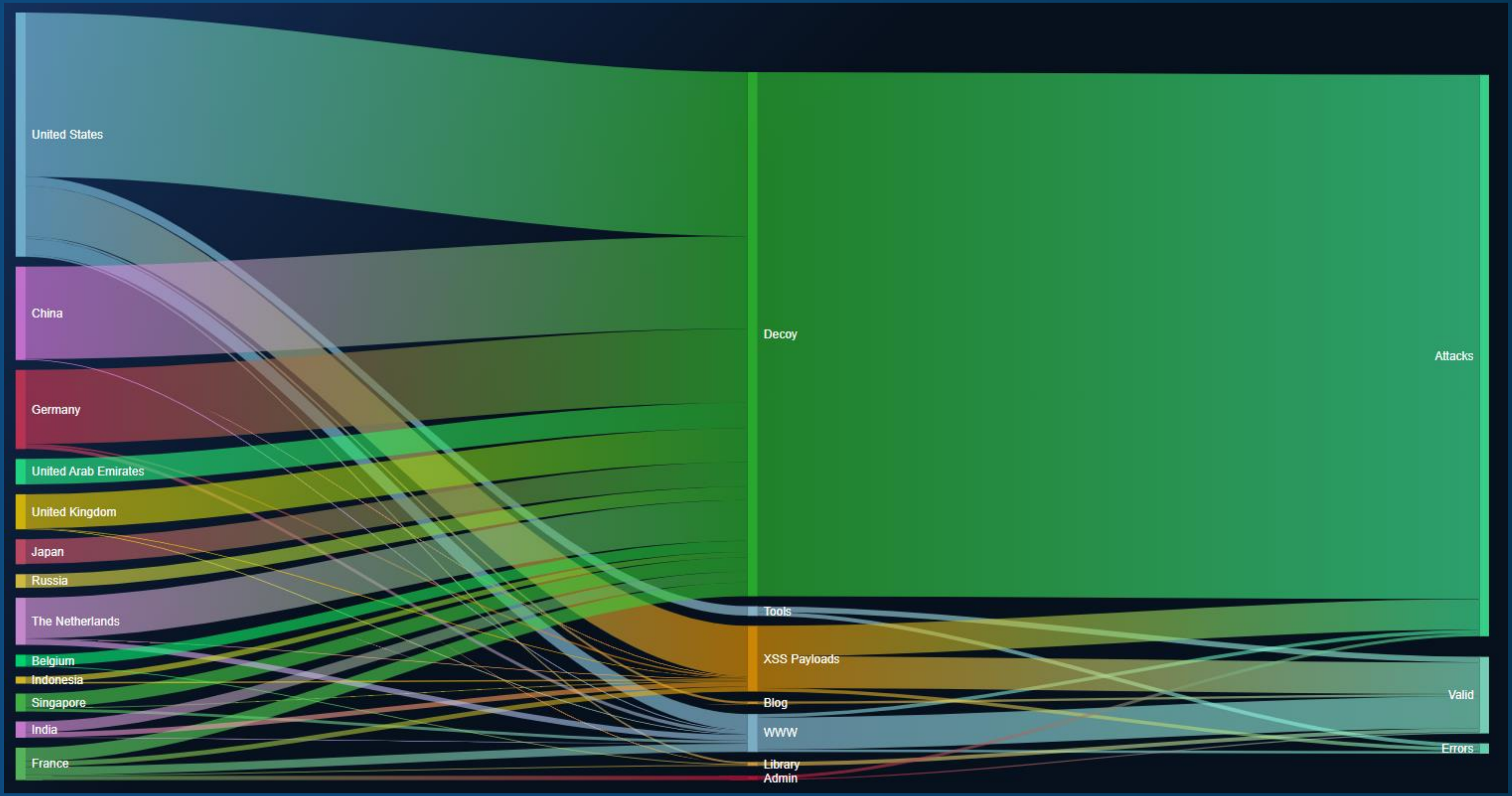
EARLY DETECTION EFFICIENCY



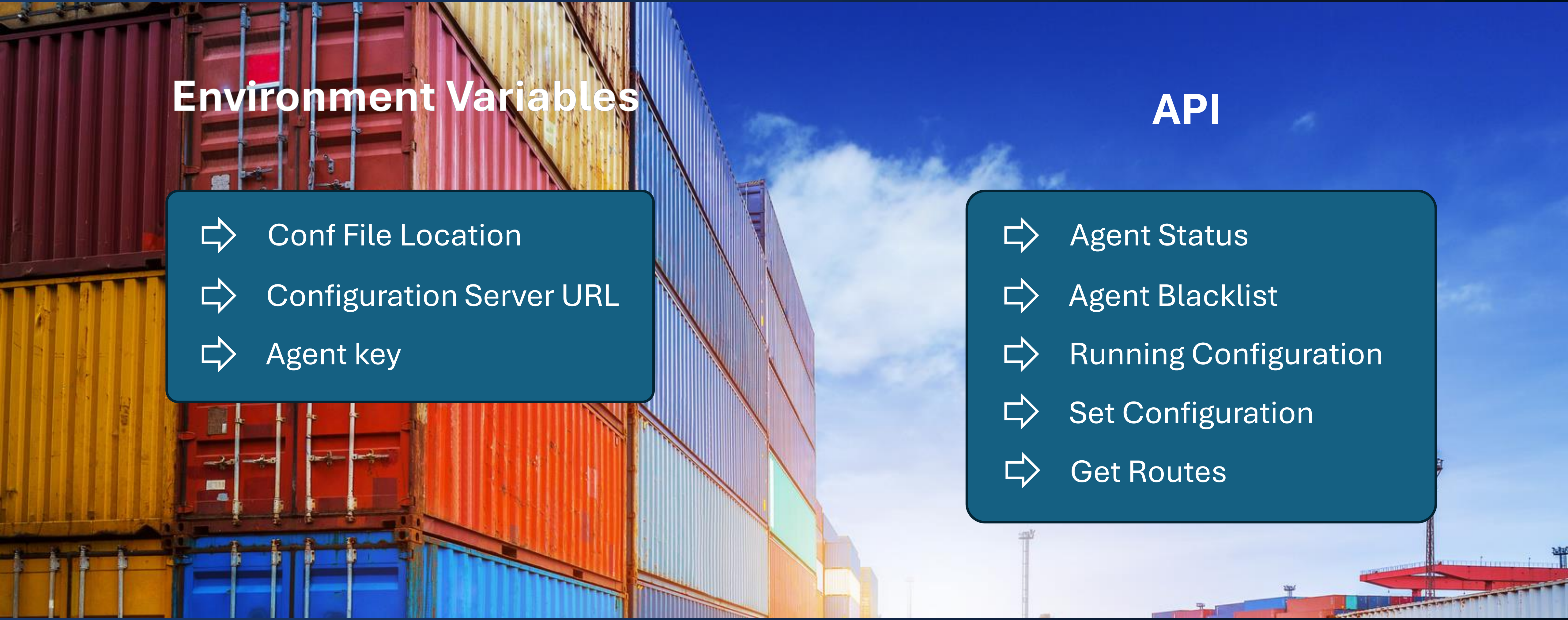
EARLY DETECTION EFFICIENCY



TELEMETRY: OUTPUTS



DEVOPS FRIENDLY



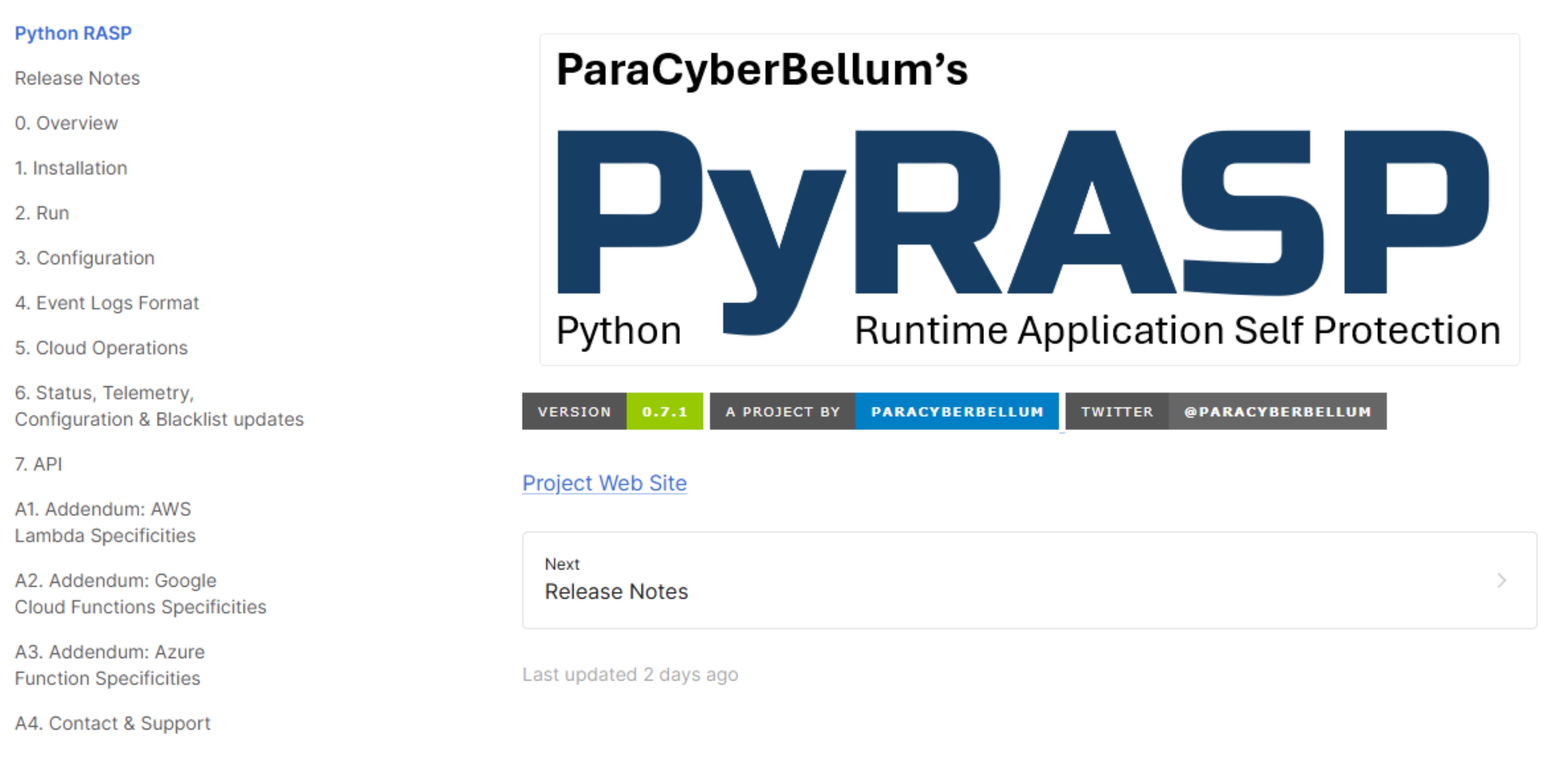
Environment Variables

- ⇨ Conf File Location
- ⇨ Configuration Server URL
- ⇨ Agent key

API

- ⇨ Agent Status
- ⇨ Agent Blacklist
- ⇨ Running Configuration
- ⇨ Set Configuration
- ⇨ Get Routes

https://rbidou.gitbook.io/pyrasp



Python RASP

- Release Notes
- 0. Overview
- 1. Installation
- 2. Run
- 3. Configuration
- 4. Event Logs Format
- 5. Cloud Operations
- 6. Status, Telemetry, Configuration & Blacklist updates
- 7. API
- A1. Addendum: AWS Lambda Specificities
- A2. Addendum: Google Cloud Functions Specificities
- A3. Addendum: Azure Function Specificities
- A4. Contact & Support

ParaCyberBellum's

PyRASP

Python Runtime Application Self Protection

VERSION 0.7.1 | A PROJECT BY PARACYBERBELLUM | TWITTER @PARACYBERBELLUM

Project Web Site

Next
Release Notes

Last updated 2 days ago

WOULD YOU LIKE TO KNOW MORE?

Coming Soon

Roadmap

v0.8.0

Zero-Trust Chromium Extension

July 2024

v0.9.0

Release Candidates

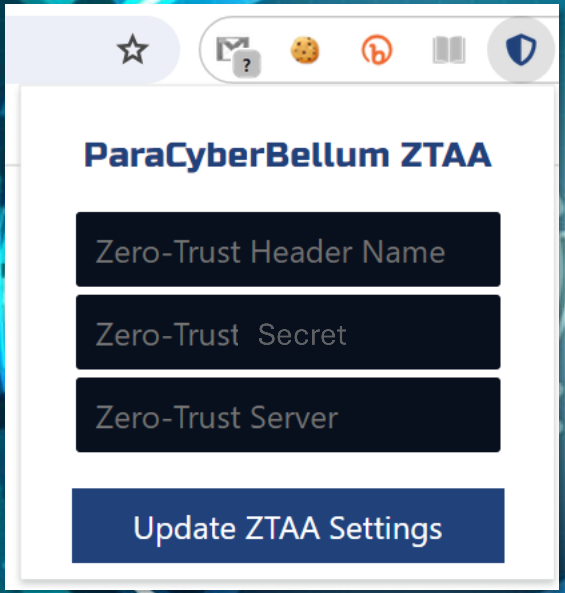
September 2024

v1.0

Release

Q4 2024

Zero-Trust Application Access



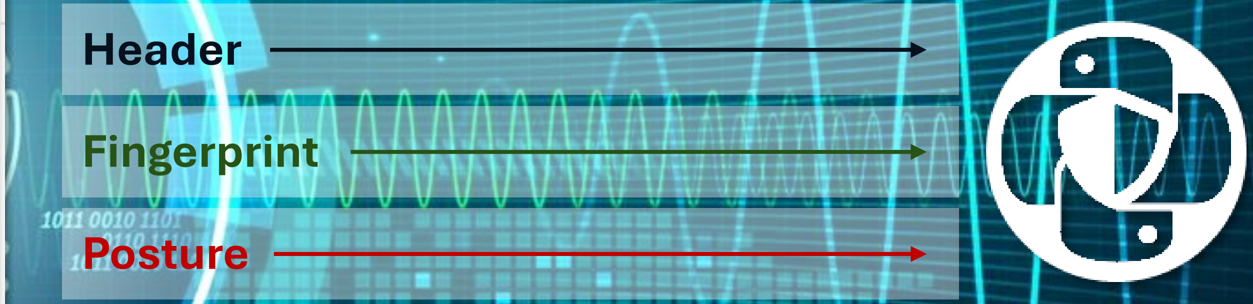
ParaCyberBellum ZTAA

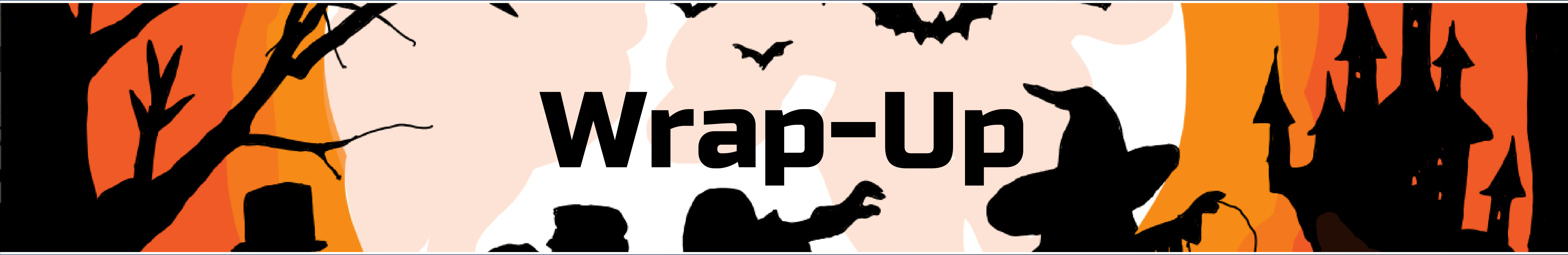
Zero-Trust Header Name

Zero-Trust Secret

Zero-Trust Server

Update ZTAA Settings





Wrap-Up

PyRASP

1. Designed for Real Needs
2. Security First
3. Minimal Management
4. Runs in Production

Resources

https://pyrasp.paracyberbellum.io

https://rbidou.gitbook.io/pyrasp

https://pypi.org/project/pyrasp/

@ParaCyberBellum

https://github.com/rbidou/pyrasp

renaud@paracyberbellum.io

https://paracyberbellum.io

Pass
The SALT

Si vis cyber pacem

ParaCyberBellum

PyRASP Project

Thank You