OWASP 2024
AppSec DAYS
SINGAPORE

Hunting for 0 & 1days by tracking Out of Bound Requests

Subhash Popuri

Agenda

- What's "Out of Bound/Band Requests"?
- Importance of OOB in manual and automated testing
- OOB Tools overview – Interact.sh, Burp Collaborator, etc.
- Why should you hunt for anomalous OOB requests?
- ***DEMO:*** Understanding "behaviour tracing via logs" in a web application
- ***DEMO:*** Adversary Infrastructure Hunting for Interact.sh
- ***DEMO:*** Hunting for unusual OOB requests – static v/s dynamic IoCs
- ***DEMO:*** Writing Detections
- ***DEMO:*** Baselining [dep on time]
- Go-Do's

About me

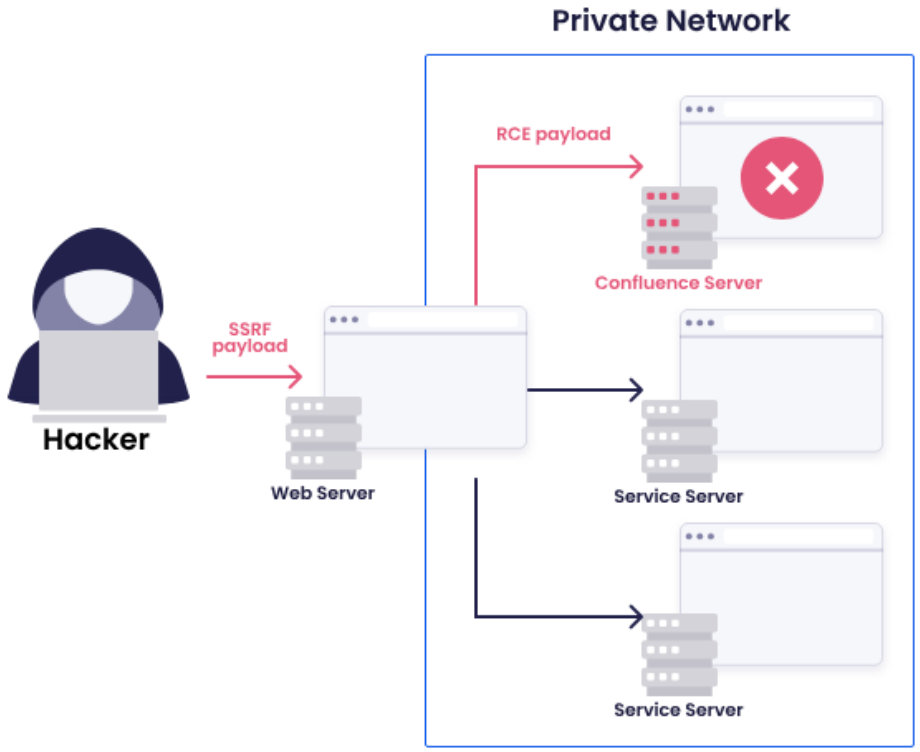
- Security Engineer at Microsoft; Blue Team, Threat Hunting & Incident Response
- Ex: Senior Security Consultant EY, PwC
- MTech from BITS Pilani in Computing Systems and Infra.
- Red Team and Incident response practitioner serving multiple clients (spec. BFSI, IT/ITeS, Manufacturing, etc.)
- ~5 years Industry experience & 10+ years of experience with Bug bounty & other aspects of Security research.
- Have experience with assisting and responding to medium to large scale cyber security incidents.
- Programmer, built multiple open-source toolkits, tools and solutions.
- Speaker at Bsides Singapore, ThreatCon, Black Hat Arsenal x2 and much more
- Contributed to multiple meetup groups like Null Hyderabad, Google Developer Group, Mozilla Hyderabad, etc.
- Trained executives (board level and otherwise) at multiple banks, IDRBT, RBI Academy and multiple governmental organisations.
- Represented country, state in cyber security startup space at multiple avenues (US govt delegation, UK govt delegation, national forums, etc.
- Talk to me about Movies, Geopolitics & Diplomacy, Badminton, Chess, Travelling and Engineering solutions to cool problems.

Found security issues in Google PayPal facebook yahoo!

And many more..

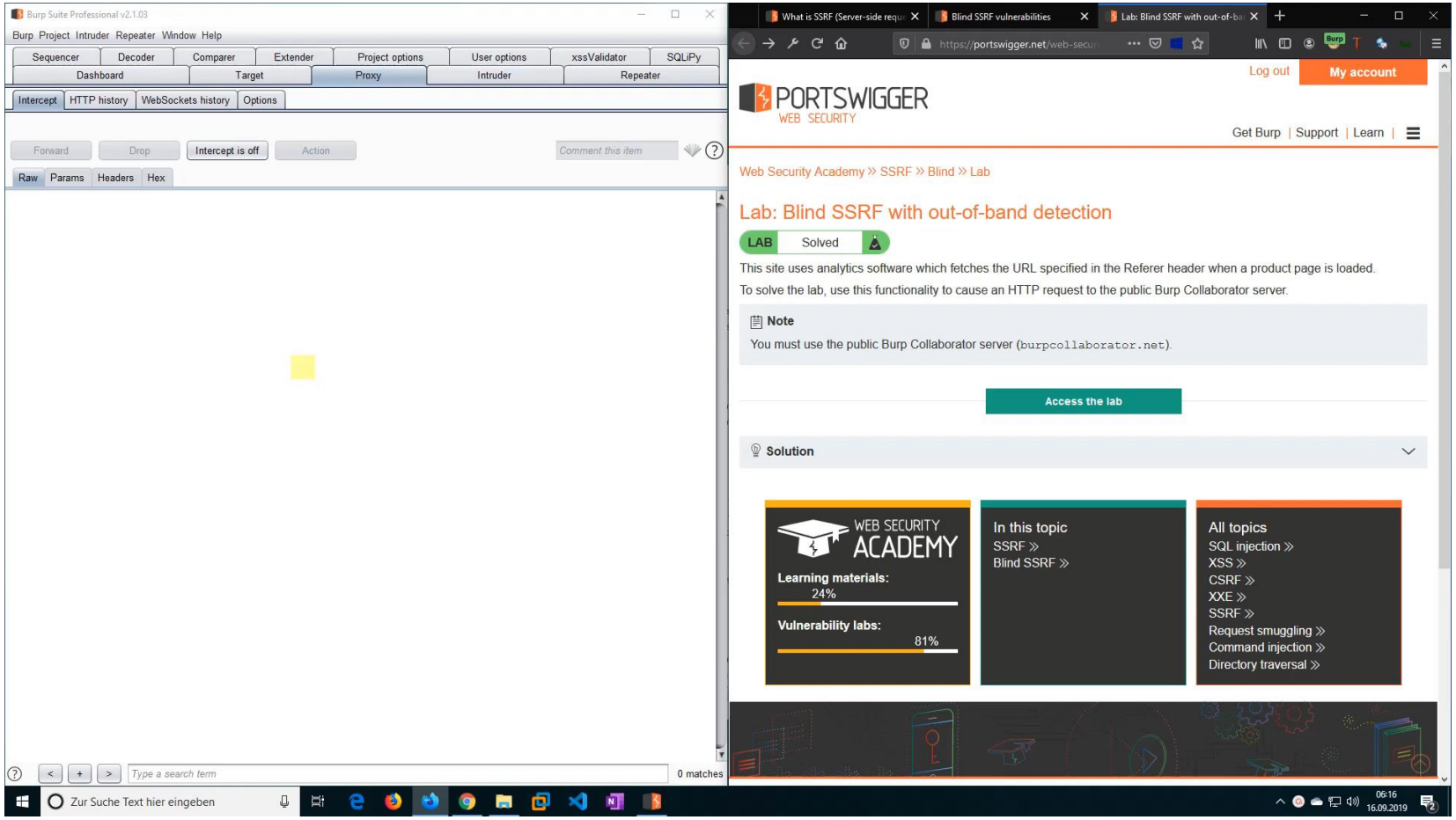
What's "Out of Bound/Band Requests"?

- Out of Band Requests are HTTP requests initiated from a host process egressing the system.
- Not all OOB requests are malicious. They can be legitimate. E.g. fetching stuff using an API request.
- OOB requests are often used by researchers to test various vulnerabilities that are "blind" in nature.
- OOB tools are "made available to the masses, just working out-of-the box with zero configuration"



Credit: Snyk Learn

What's "Out of Bound/Band Requests"?



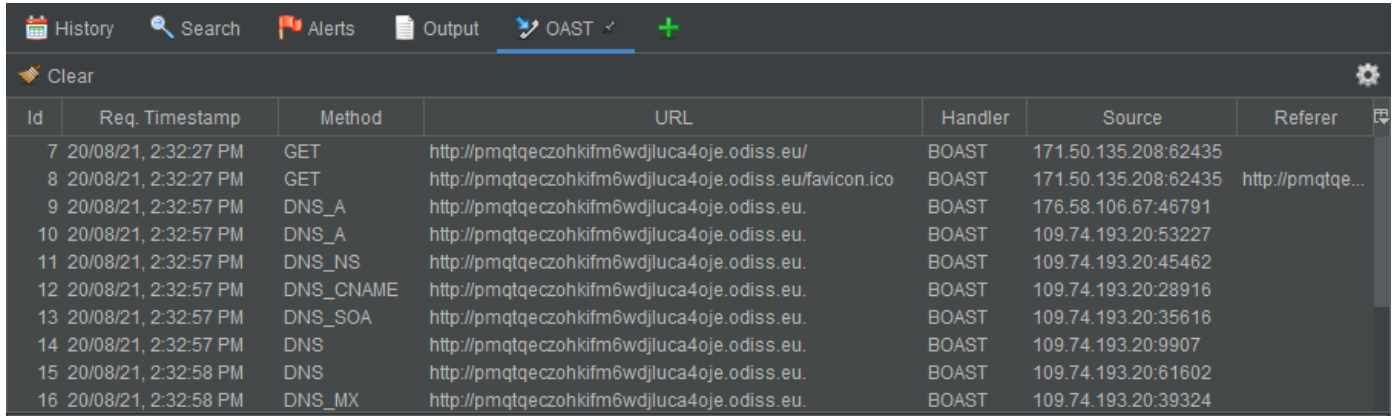
Credit: [Youtube – Michael Sommer]

OOB Tools Overview [Not an exhaustive list -]

S.No	Name of the platform	Dedicated for security testing?	Licensing	Self-Hosted Option	References	URL Pattern of OOB
1.	Security Scanners – Burp, Acuentix, Netsparker, AppCheck NG	✓	Commercial	✗	Burp Collaborator - Doc	*. burpcollaborator.net, *.r87.me, *.ptst.io, *.bxss.me
2.	Interact SH	✓	Freeware	✓	GitHub, App	*.oastify.*
3.	Canarytokens	✓	Freeware	✗	CanaryTokens Website	*.canarytokens.com/*
4.	Request Bin	✗	Freeware	✗	RequestBin	*.pipedream.net
5.	Dnslog.cn	✗	Freeware	✗	DnsLog	*.dnslog.cn
6.	TukTuk	✓	Freeware	✓	TukTuk GitHub	N/A
7	Boast	✓	Freeware	✓	Boast GitHub	N/A
6.	Custom HTTP listener (using Netcat, Python HTTP server, etc.)	✗	Freeware	✓	Custom HTTP server	N/A

Why to hunt for unusual OOB requests?

- OOB Tools are generally available to the masses, with zero configuration at times
- These are effectively used in testing and exploitation of blind SQL Injection, blind XSS, blind XXE, blind RCE, etc.
- When you send a DAST payload along with OOB domain and you get a successful pingback (DNS or otherwise), you can be sure the vulnerability exists.
- Any successful outbound interaction arising out of the malicious request is a potential vulnerability.
- While it's not clearly evident if it's an SSRF, it's clearly an indication of vulnerability being present.



Id	Req. Timestamp	Method	URL	Handler	Source	Referer
7	20/08/21, 2:32:27 PM	GET	http://pmqtqeczohkifm6wdjluca4oje.odiss.eu/	BOAST	171.50.135.208:62435	
8	20/08/21, 2:32:27 PM	GET	http://pmqtqeczohkifm6wdjluca4oje.odiss.eu/favicon.ico	BOAST	171.50.135.208:62435	http://pmqtqe...
9	20/08/21, 2:32:57 PM	DNS_A	http://pmqtqeczohkifm6wdjluca4oje.odiss.eu.	BOAST	176.58.106.67:46791	
10	20/08/21, 2:32:57 PM	DNS_A	http://pmqtqeczohkifm6wdjluca4oje.odiss.eu.	BOAST	109.74.193.20:53227	
11	20/08/21, 2:32:57 PM	DNS_NS	http://pmqtqeczohkifm6wdjluca4oje.odiss.eu.	BOAST	109.74.193.20:45462	
12	20/08/21, 2:32:57 PM	DNS_CNAME	http://pmqtqeczohkifm6wdjluca4oje.odiss.eu.	BOAST	109.74.193.20:28916	
13	20/08/21, 2:32:57 PM	DNS_SOA	http://pmqtqeczohkifm6wdjluca4oje.odiss.eu.	BOAST	109.74.193.20:35616	
14	20/08/21, 2:32:57 PM	DNS	http://pmqtqeczohkifm6wdjluca4oje.odiss.eu.	BOAST	109.74.193.20:9907	
15	20/08/21, 2:32:58 PM	DNS	http://pmqtqeczohkifm6wdjluca4oje.odiss.eu.	BOAST	109.74.193.20:61602	
16	20/08/21, 2:32:58 PM	DNS_MX	http://pmqtqeczohkifm6wdjluca4oje.odiss.eu.	BOAST	109.74.193.20:39324	

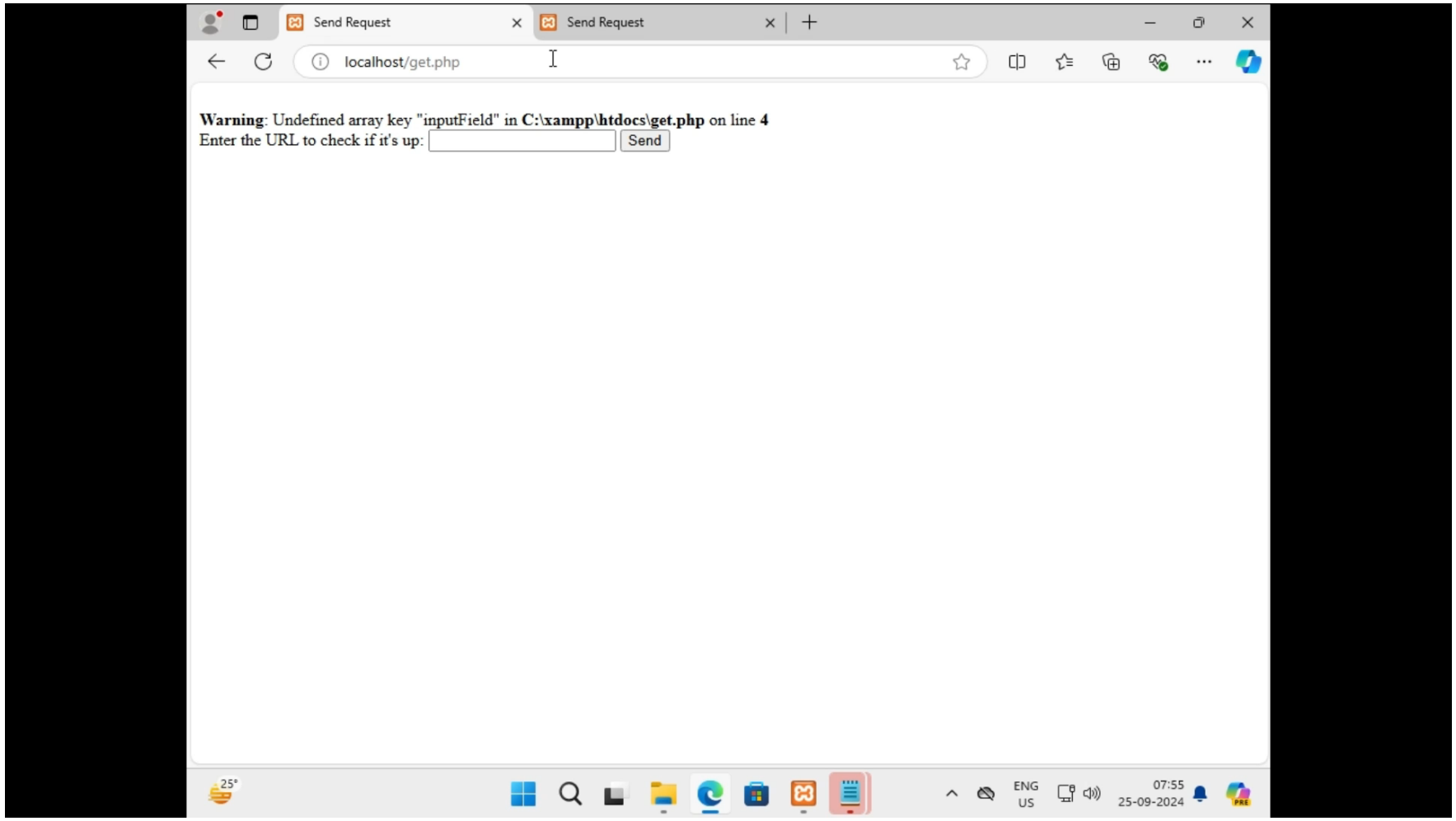
How to trace application behaviour through logs [anatomy of an access log]

127.0.0.1 - Scott [10/Dec/2019:13:55:36 -0700] "GET /server-status HTTP/1.1" 200 2326

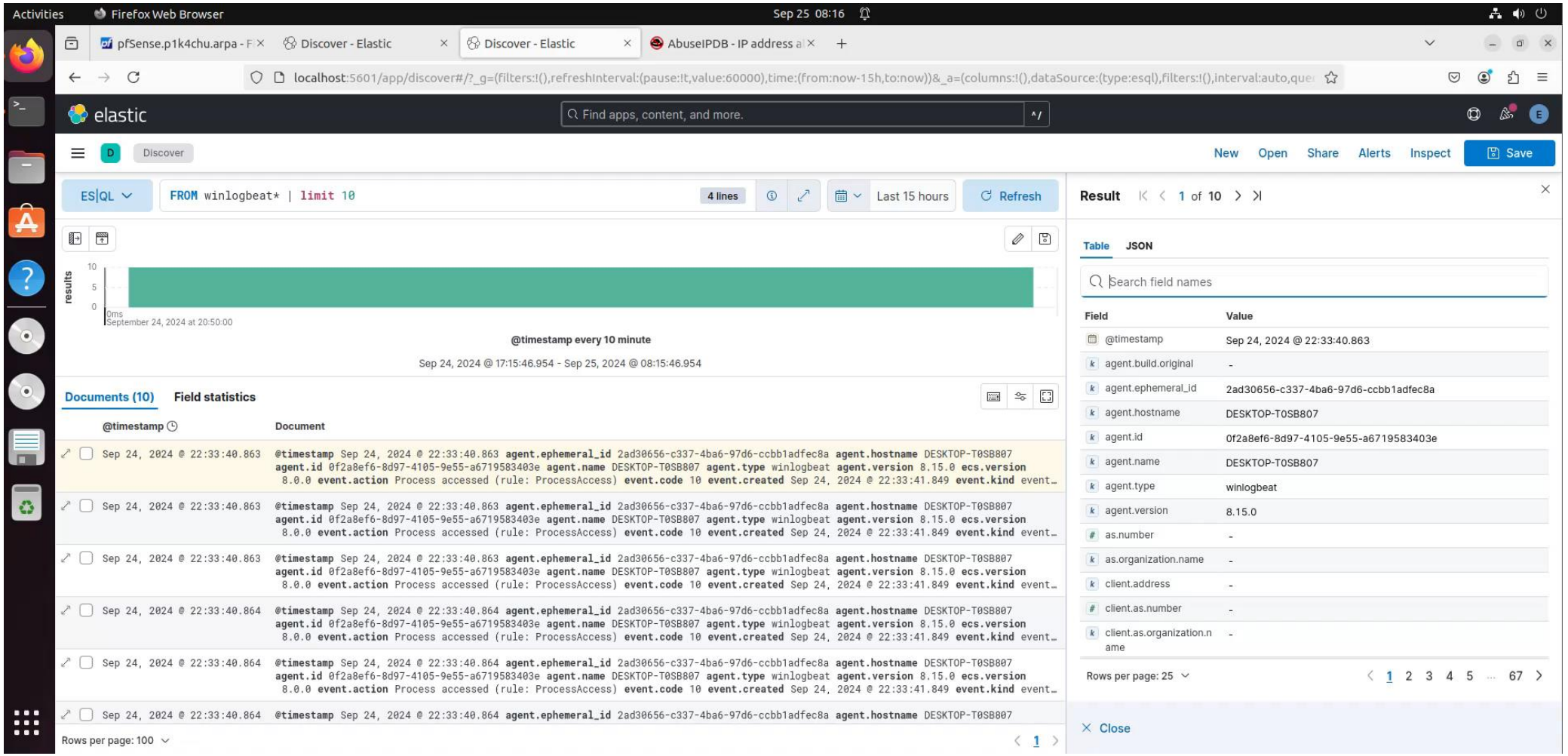
The fields in the above sample record represent the following:

- **127.0.0.1** - IP address of the client that made the request;
- The **hyphen** defining the second field in the log file is the identity of the client. This field is often returned as a hyphen and Apache's HTTP server documentation recommends that this particular field not be relied upon except in the case of a controlled internal network.
- **Scott** - userid of the person requesting the resource;
- **[10/Dec/2019:13:55:36 -0700]** - date and time of the request;
- **"GET /server-status HTTP/1.1"** - request type and resource being requested;
- **200** - HTTP response status code;
- **2326** - size of the object returned to the client.

How to trace application behaviour through logs [through application access log]



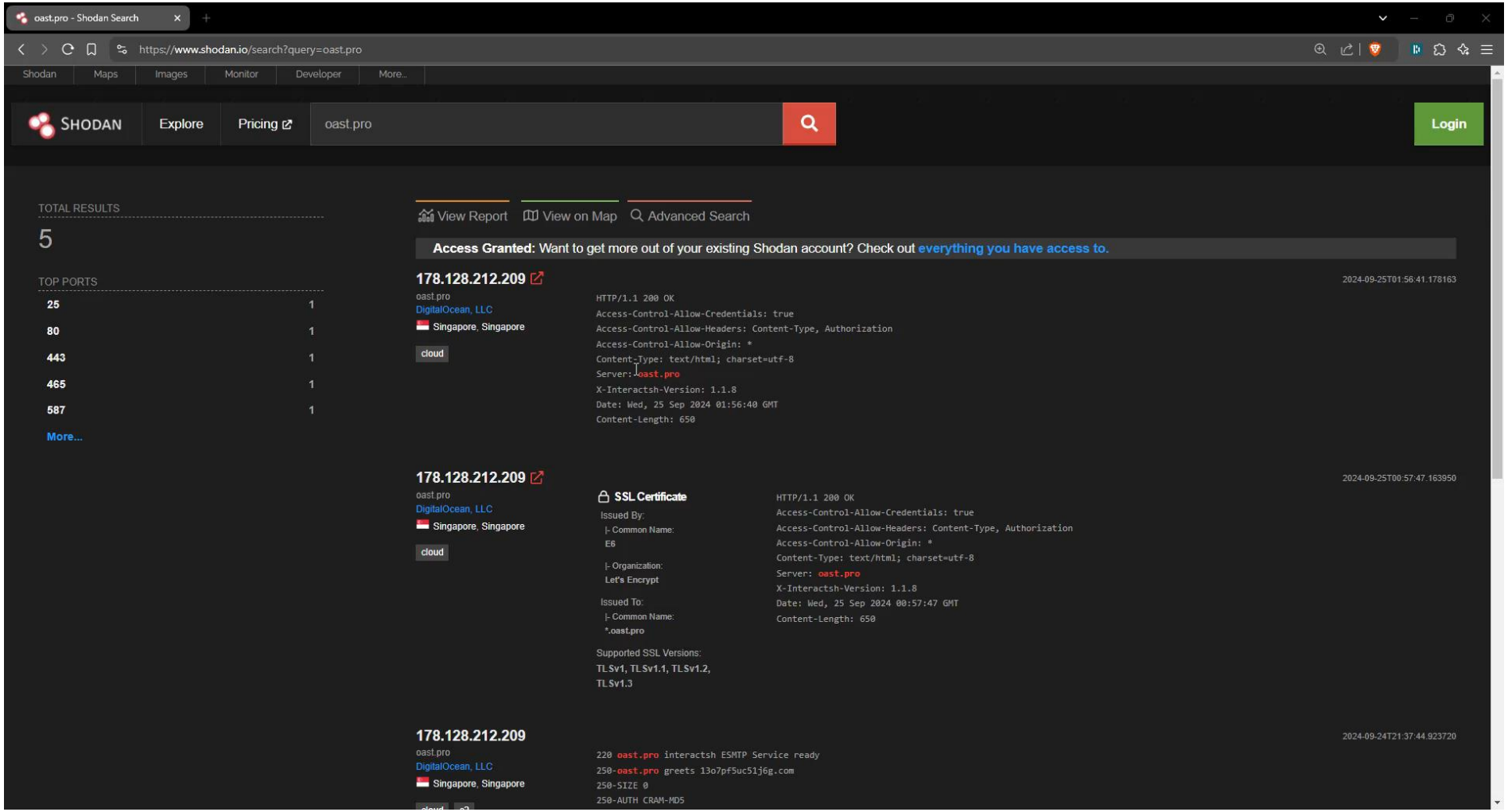
How to trace application behaviour through logs [through host security log]



But there's a challenge

S.No	Name of the platform	Dedicated for security testing?	Licensing	Self-Hosted Option	References	URL Pattern of OOB
1.	Security Scanners – Burp, Acuentix, Netsparker, AppCheck NG	✓	Commercial	✗	Burp Collaborator - Doc	*. burpcollaborator.net
2.	Interact SH	✓	Freeware	✓	GitHub, App	*.oastify.*
3.	Canarytokens	✓	Freeware	✗	CanaryTokens Website	*.canarytokens.com/*
4.	Request Bin	✗	Freeware	✗	RequestBin	*.pipedream.net
5.	Dnslog.cn	✗	Freeware	✗	DnsLog	*.dnslog.cn
6.	TukTuk	✓	Freeware	✓	TukTuk GitHub	N/A
7	Boast	✓	Freeware	✓	Boast GitHub	N/A
6.	Custom HTTP listener (using Netcat, Python HTTP server, etc.)	✗	Freeware	✓	Custom HTTP server	N/A

Hunting for adversary infrastructure – feat. Interact sh



Hunting for unusual OOB requests static v/s dynamic

- Static detection means taking IoC Hits from the previous step (Infra hunting) + some known OOB targets & hunt for them.
- Leverage SOAR or other orchestration tools (such as Jupyter Notebooks) for capturing & storing IoCs
- Purely static v/s dynamic indicators

Completely static

```
DeviceNetworkEvents
| where RemoteUrl has_any ("*.oastify.*","*.burpcollaborator.net","*.pipedream.net")
```

Dynamic IoC

```
DeviceNetworkEvents
| where RemoteIP in ("146.190.116.224","137.184.84.19") or RemoteUrl has_any ("oobd.ru","caeou.com")
```

Hunting for unusual OOB requests – baselining

- Static detection means taking IoC Hits from the previous step (Infra hunting) + some known OOB targets & hunt for them.
- Leverage SOAR or other orchestration tools (such as Jupyter Notebooks) for capturing & storing IoCs

Steps for baselining your application traffic:

- Check the traffic originating from the respective server processes
- Identify if it's legitimate API request or if the application has legitimate functionality that allows for arbitrary network requests.
- Check if the application is spawning any "LOLBAS" binaries such as "curl.exe" or "wget.exe", etc and that binary is creating any traffic.
- Identify the IP addresses which are egressing and check if there's a legitimate reasoning for it [such as API, etc]

The big picture – Hunting for 0days and 1day exploitation across *any* platform?

- Welcome Sigma - https://github.com/SigmaHQ/sigma

Go-Do’s

- Application logs [access logs & application logs] are very important.
- Integrate hunting methodology to identify malicious OOB requests in your environment.
- Start with basic stuff such as static IOCs, dynamic IOCs
- Baselining is a must for organisations moving towards maturity.
- Start using the sample detection rule & modify accordingly.


OWASP 2024
AppSec DAYS
SINGAPORE
THANK
YOU!