/^[R][e][g][e][x]\s[F][o][r]\s[H][a][c][k][e][r][s]$/

By

Adam Langley & Ben Sadeghipour

@BuildHackSecure @NahamSec

hackinghub

Schedule

- Intro
- Basic String Matching
- Basic String Matching Limitations
- What is Regex?
  - Literal Matches
  - Exact String
  - Anchors
  - Character Sets
  - Quantifiers
  - Shortcuts
  - Wild Cards
  - Groups
- When Regex Goes Wrong
  - Post Message Origin Check
  - Open Redirects
  - CORS
  - SSRF
- OSINT Asset Discovery
- OSINT Content Discovery
- Regex For Data Dumps
- Code Reviews
- Tool Creation



About Adam

Experience:
Starting hacking in the late 90s (will exchange drinks for stories)
Worked In web development for 20+ years
Finally put them both together by creating CTF and learning labs
Content On: HackingHub, HackerOne, TryHackMe + Other CTF Events

Job:
CTO at HackingHub building core applications, courses, labs and CTFs!

Socials: @BuildHackSecure



- Regex For Hackers

About Ben

- 10+ Bug Bounty Experience
- $1.8M in bounties on HackerOne
- Recognized for Hacking Apple, Amazon, Facebook, and more
- Previously
  - Head of Hacker Education at HackerOne
- Currently:
  - Teaching the next generation of hackers on HackingHub.io
  - Content creator
  - Pentester / Bug Bounty Hunter

@NahamSec on YouTube, Twitter (X), Instagram, Twitch, [INSERT OTHER SOCIALS HERE]



- Regex For Hackers

Purpose Of this...

- Regex can look very overwhelming, but we’re here to help!



Purpose Of this...

- Regex can look very overwhelming, but we’re here to help!
- Help discover vulnerabilities and understand why they might occur.



Purpose Of this...

- Regex can look very overwhelming, but we’re here to help!
- Help discover vulnerabilities and understand why they might occur.
- Search better



Purpose Of this...

- Regex can look very overwhelming, but we’re here to help!
- Help discover vulnerabilities and understand why they might occur.
- Search better
- Discover secrets



Purpose Of this...

- Regex can look very overwhelming, but we’re here to help!
- Help discover vulnerabilities and understand why they might occur.
- Search better
- Discover secrets
- Build Cool Tools



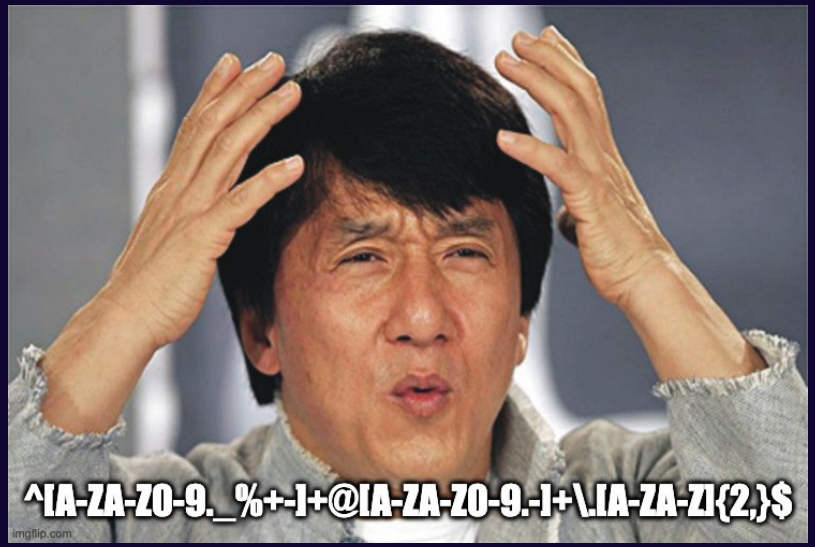
Purpose Of this…

- Regex can look very overwhelming, but we’re here to help!
- Help discover vulnerabilities and understand why they might occur.
- Search better
- Discover secrets
- Build Cool Tools
- In Summary….



Purpose Of this...

From This…





- Regex For Hackers

Purpose Of this...

To This…





- Regex For Hackers

Basic String Matching



- Regex For Hackers

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

username@acmecorp.ctf

I definitely wrote this without the help of AI — or maybe I didn’t?

Starts with

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

username@acmecorp.ctf

I definitely wrote this without the help of AI — or maybe I didn’t?

Starts with

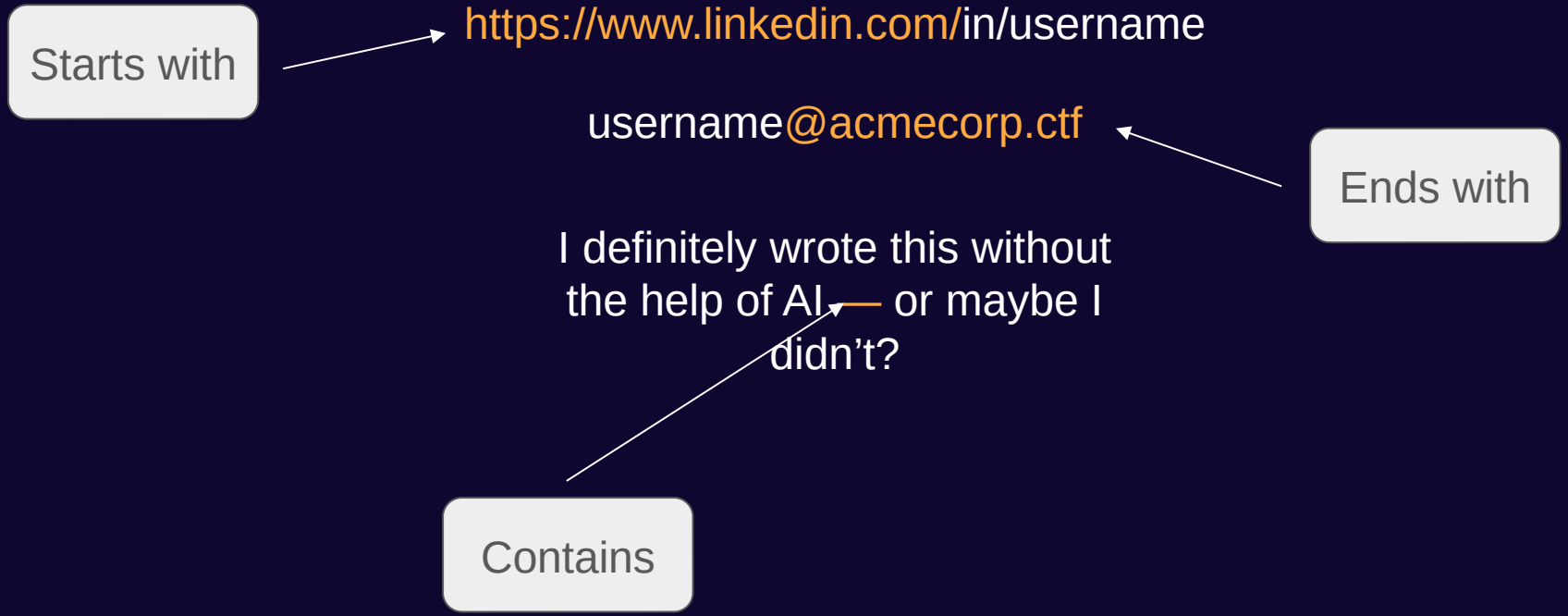
https://www.linkedin.com/in/username

username@acmecorp.ctf

Ends with

I definitely wrote this without the help of AI — or maybe I didn’t?




Starts with
https://www.linkedin.com/in/username
username@acmecorp.ctf
Ends with
I definitely wrote this without the help of AI — or maybe I didn't?
Contains



startswith()

endswith()

‘substr’ in string



- Regex For Hackers



startswith()

endswith()

‘substr’ in string



startsWith()

endsWith()

includes()



	startswith()	endswith()	‘substr’ in string
	startsWith()	endsWith()	includes()
	str_starts_with()	str_ends_with()	contains()

	startswith()	endswith()	‘substr’ in string
	startsWith()	endsWith()	includes()
	str_starts_with()	str_ends_with()	contains()
	startsWith()	endsWith()	contains()



	startswith()	endswith()	‘substr’ in string
	startsWith()	endsWith()	includes()
	str_starts_with()	str_ends_with()	contains()
	startsWith()	endsWith()	contains()
	start_with?()	end_with?()	include?()

	startswith()	endswith()	‘substr’ in string
	startsWith()	endsWith()	includes()
	str_starts_with()	str_ends_with()	contains()
	startsWith()	endsWith()	contains()
	start_with?()	end_with?()	include?()
	starts_with()	ends_with()	contains()

	startswith()	endswith()	‘substr’ in string
	startsWith()	endsWith()	includes()
	str_starts_with()	str_ends_with()	contains()
	startsWith()	endsWith()	contains()
	start_with?()	end_with?()	include?()
	starts_with()	ends_with()	contains()
	HasPrefix()	HasSuffix()	Contains()



- Regex For Hackers

Basic Match Limitations



Starts With Limitations



Starts With Limitations

192.168.1.0 - 192.168.10.254



Starts With Limitations

192.168.1.0 - 192.168.10.254

```
if (
    ip.startsWith("192.168.1.")
    || ip.startsWith("192.168.2.")
    || ip.startsWith("192.168.3.")
    || ip.startsWith("192.168.4.")
    || ip.startsWith("192.168.5.")
    || ip.startsWith("192.168.6.")
    || ip.startsWith("192.168.7.")
    || ip.startsWith("192.168.8.")
    || ip.startsWith("192.168.9.")
    || ip.startsWith("192.168.10.")
)
```



Starts With Regex Solution

```
^192\.168\.(10|[1-9])\.([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-4]?)$
```

Ends With Limitations



- Regex For Hackers

Ends With Limitations

@domain.co.uk

Ends With Limitations

@domain.co.uk

@domain.fr

Ends With Limitations

@domain.co.uk

@domain.fr

@domain.it



Ends With Limitations

@domain.co.uk

@domain.fr

@domain.it

```
if (
    email.endsWith(“@domain.co.uk”)
    || email.endsWith(“@domain.fr”)
    || email.endsWith(“@domain.it”)
)
```

Ends With Regex Solution

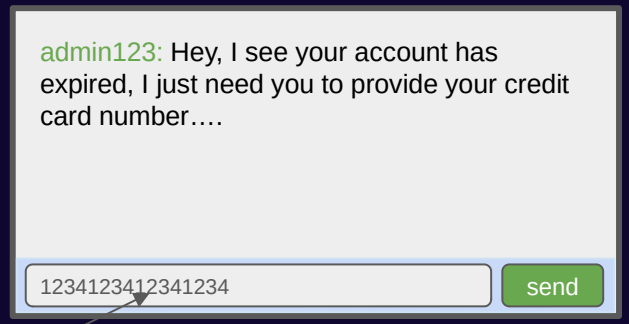
```
\@domain\.(co\.uk|fr|it)$
```



Contains Limitations

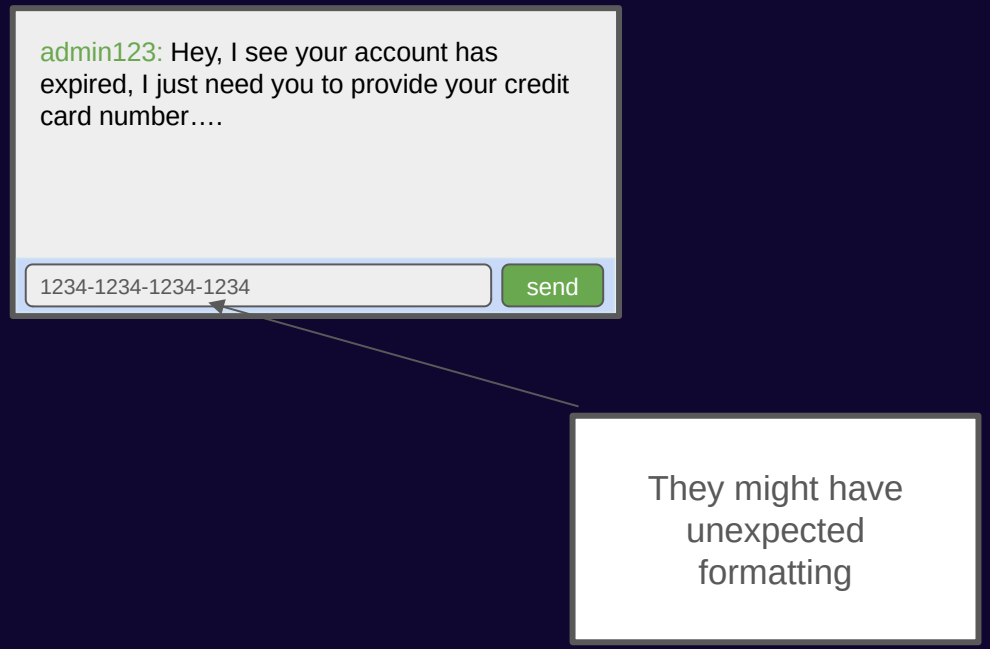


Contains Limitations

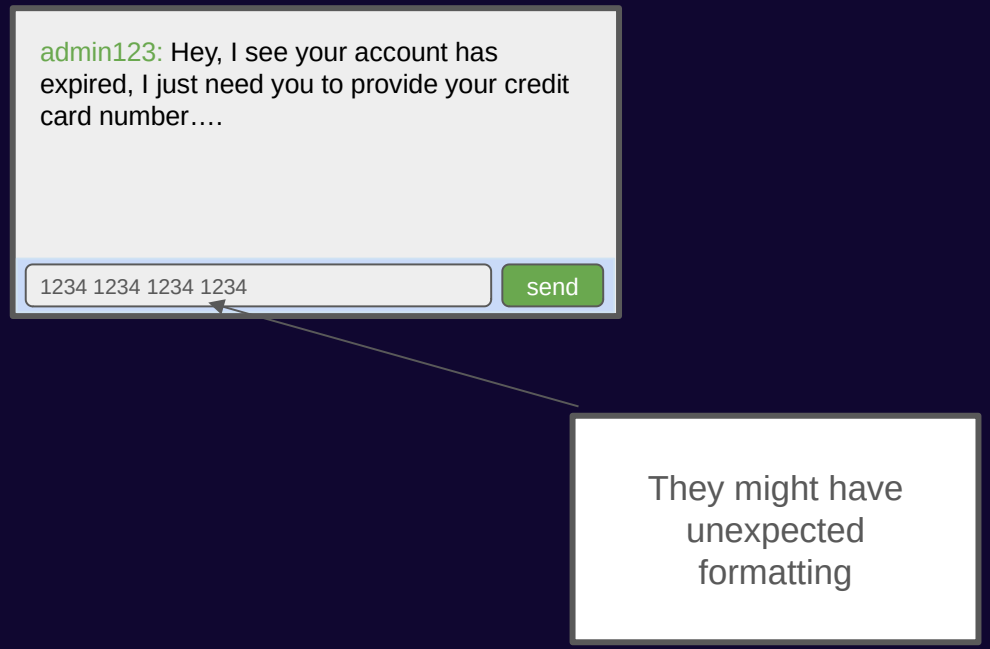


You can’t just search for numbers you don’t know!

Contains Limitations



Contains Limitations



Contains Regex Solution

```
(\d{4}(\s*|-)){3}\d{4}
```

What is Regex

What is Regex?

Regex is a search pattern language!
It lets you find, match, and extract specific text from strings.

Regex is like a state machine, it goes through your regular expression rules one by one looking for a match and then moving onto the next rule.

Think of it like a flow diagram!

Literal Matches



Oh how are you? Oh hello yes I’m fine

hello

Oh how are you? Oh hello yes I’m fine



Exact String



Would you like a supercomputer or computer



computer

Would you like a supercomputer or computer



\bcomputer\b

\b word boundaries

Would you like a supercomputer or computer



\bcomputer\b

Would you like a supercomputer or computer?

\b word boundaries
accept punctuation

\bcomputer\b

Would you like a supercomputer or computer.

\bcomputer\b

Would you like a supercomputer or computerx



Anchors
“Starts With”



https:\/\/

This is my url https://example.com



- Regex For Hackers

^https:\/\/

^ String must start with

This is my url https://example.com

^https:\/\/

https://example.com



Anchors
“Ends With”



@example.co

test@example.co.uk



- Regex For Hackers

@example.co$

$ String must end with

test@example.co.uk



@example.co$

test@example.co



@example.co$

. period is treated like a wildcard!

test@examplexco

@example\.co$

\. A backslash must be provided!

test@examplexco



- Regex For Hackers

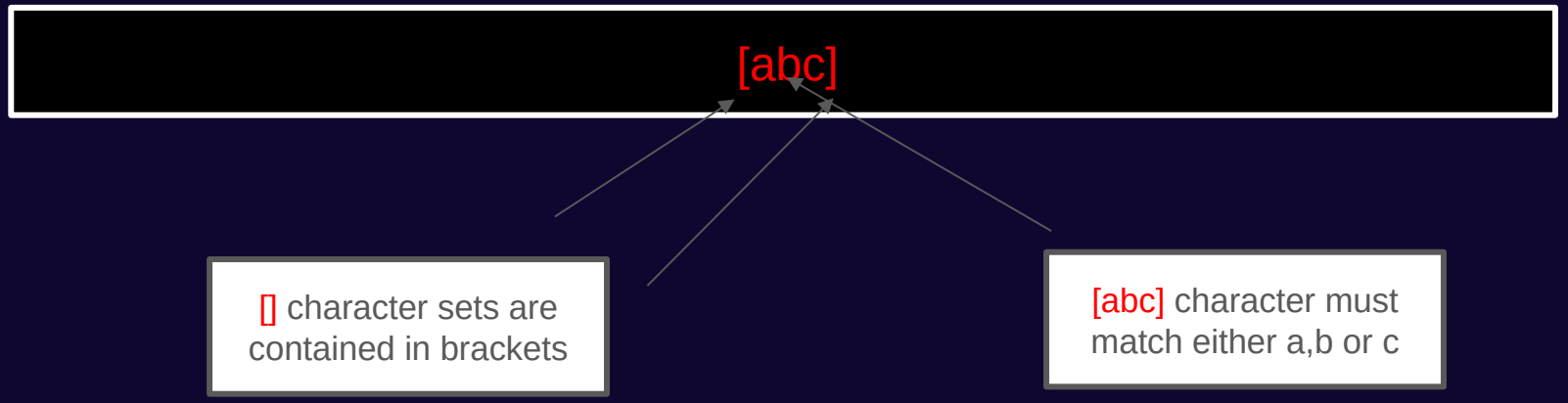
@example\.co$

test@example.co



- Regex For Hackers

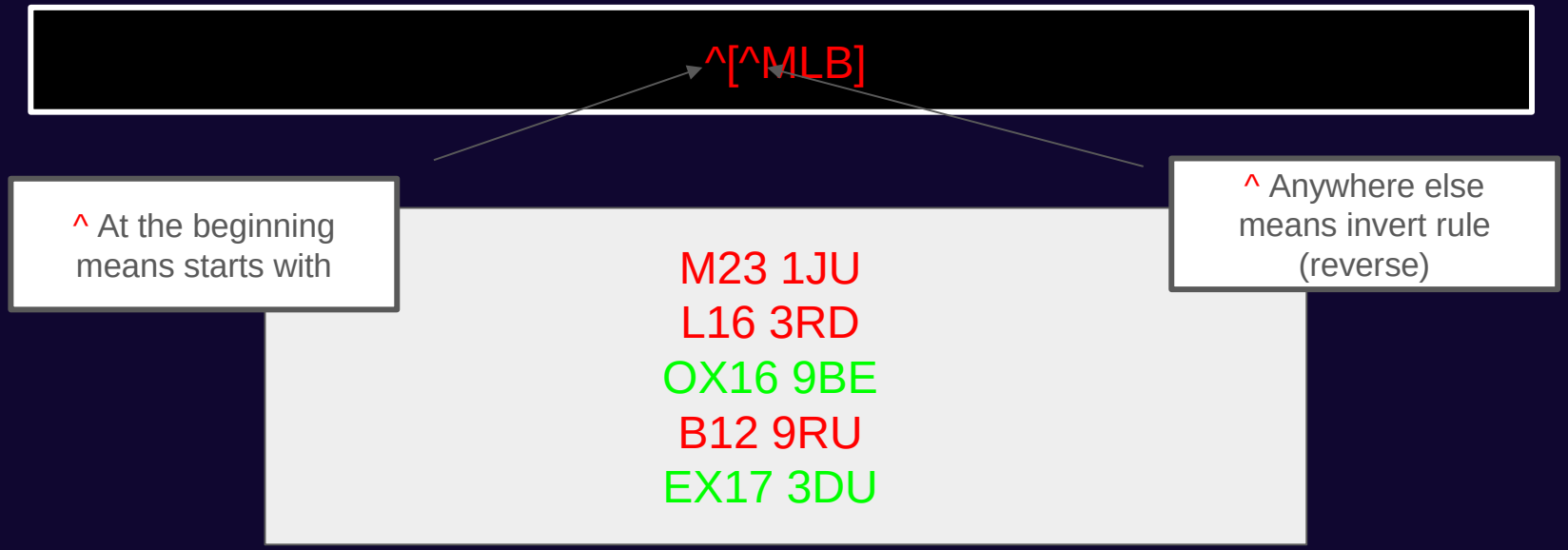
Character Sets


[abc]
[] character sets are contained in brackets
[abc] character must match either a,b or c

^[MLB]

M23 1JU
L16 3RD
OX16 9BE
B12 9RU
EX17 3DU

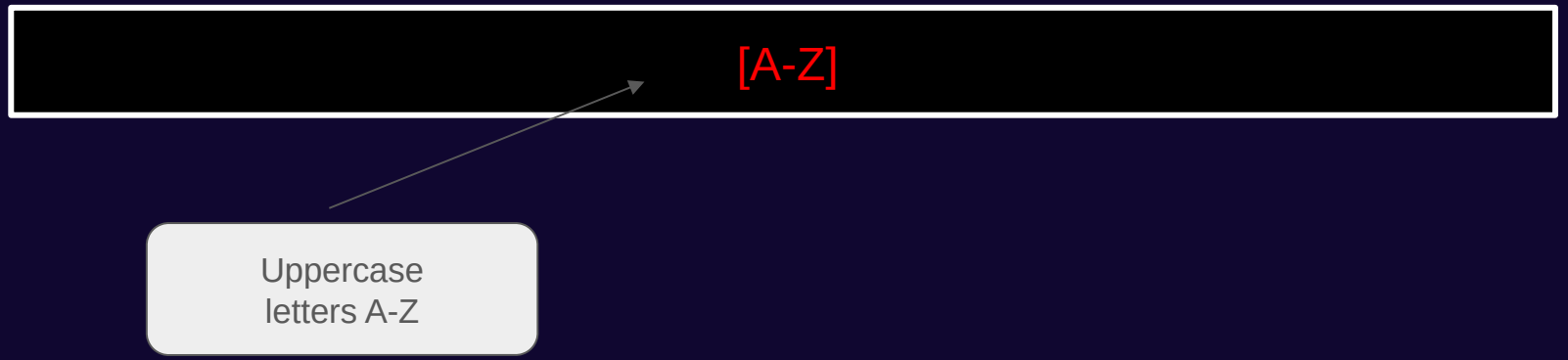


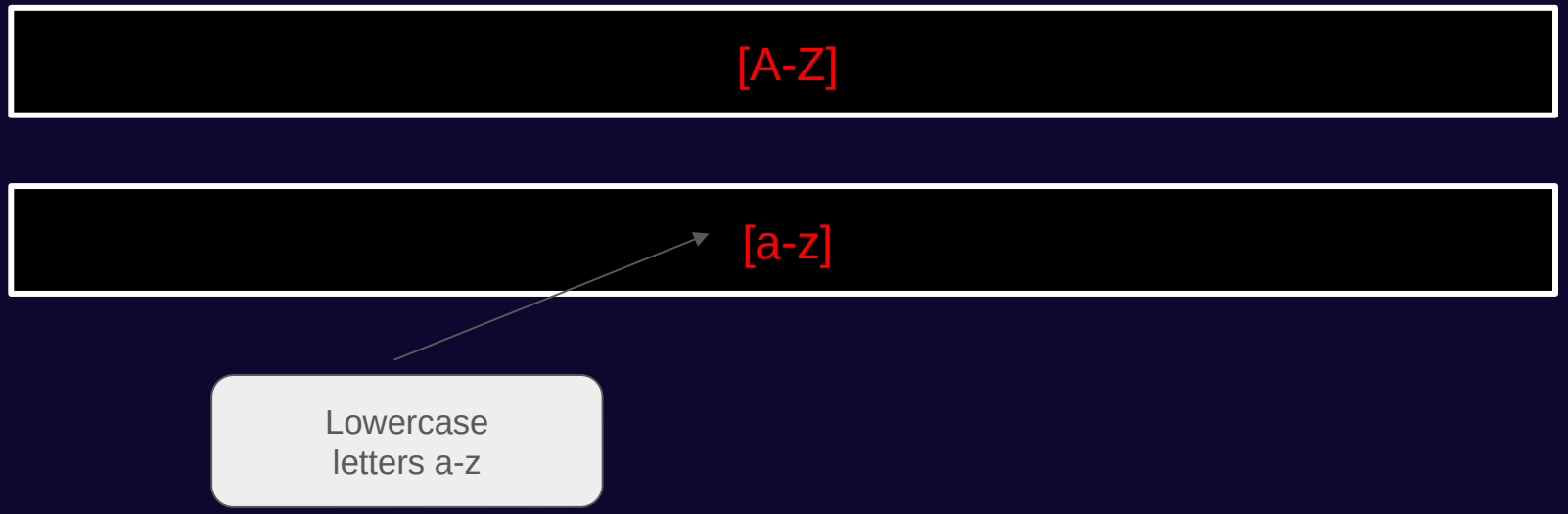

^[^MLB]
^ At the beginning means starts with
^ Anywhere else means invert rule (reverse)
M23 1JU
L16 3RD
OX16 9BE
B12 9RU
EX17 3DU

^[A-L]

You can even supply ranges!

M23 1JU
L16 3RD
OX16 9BE
B12 9RU
EX17 3DU


[A-Z]
Uppercase
letters A-Z


[A-Z]
[a-z]
Lowercase
letters a-z

[A-Z]

[a-z]

[0-9]

Digits 0-9



[A-Z]

[a-z]

[0-9]

[A-L0-5]

Uppercase letters A-L and digits 0-5



Add character sets together to make stronger rules

```
^[A-Z][0-9][0-9][ ][0-9][A-Z][A-Z]$
```

M23 1JU
L16 3RD
OX16 9BE
B12 9RU
EX17 3DU
SK3 8D9

Quantifiers

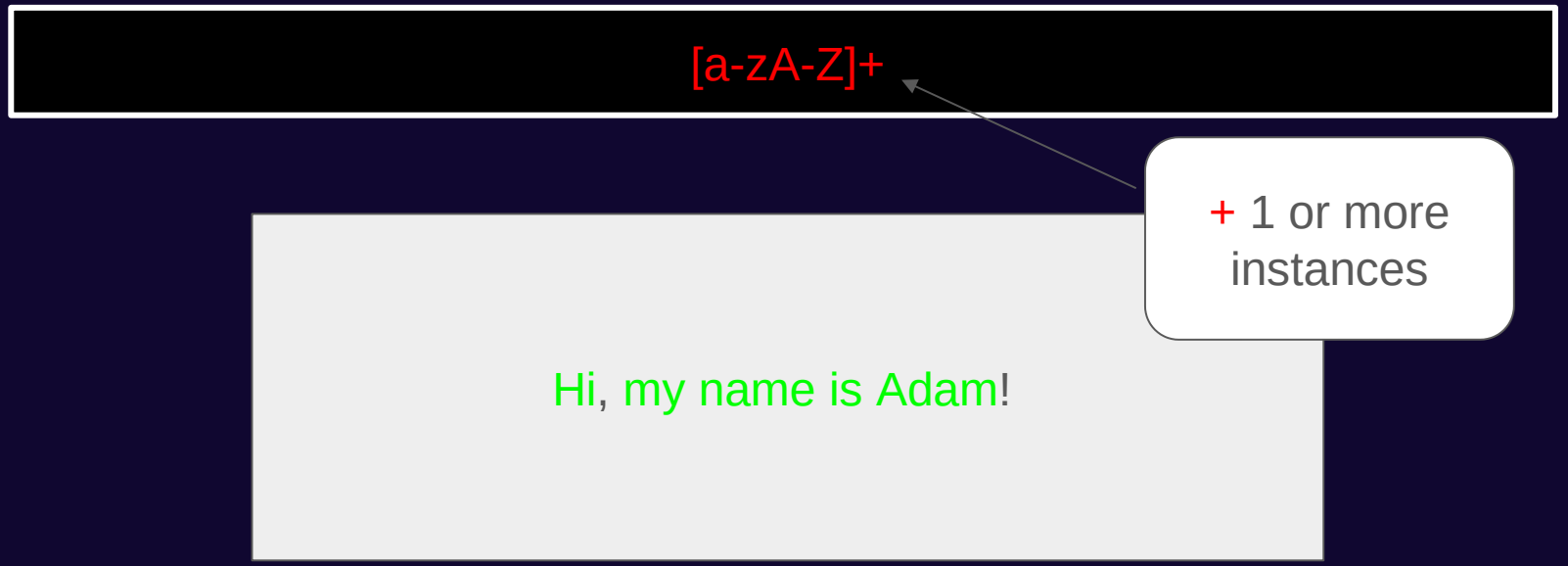


[a-z]+

+ 1 or more instances

Hi, my name is Adam!




[a-zA-Z]+
+ 1 or more instances
Hi, my name is Adam!

^http:\/\/

http://example.com
https://example.com
httpss://example.com



- Regex For Hackers

^https:\/\/

http://hackinghub.io
https://hackinghub.io
httpss://hackinghub.io



^https?:\/\/

? zero or 1 instance

http://hackinghub.io
https://hackinghub.io
httpss://hackinghub.io

\b[0-9]{16}\b

{n} n number of instances

Okay here's my credit card number 1234123412341234 thanks!!!



\b[0-9]{16,}\b

{n,} at least n number of instances

Okay here's my credit card number 1234123412341234123 thanks!!!



```
\b[0-9]{16,}\b
```

Checkout my binary string:
0100011100010100100101110010100101010110010010001010101001001001001010101001010101 :)



- Regex For Hackers

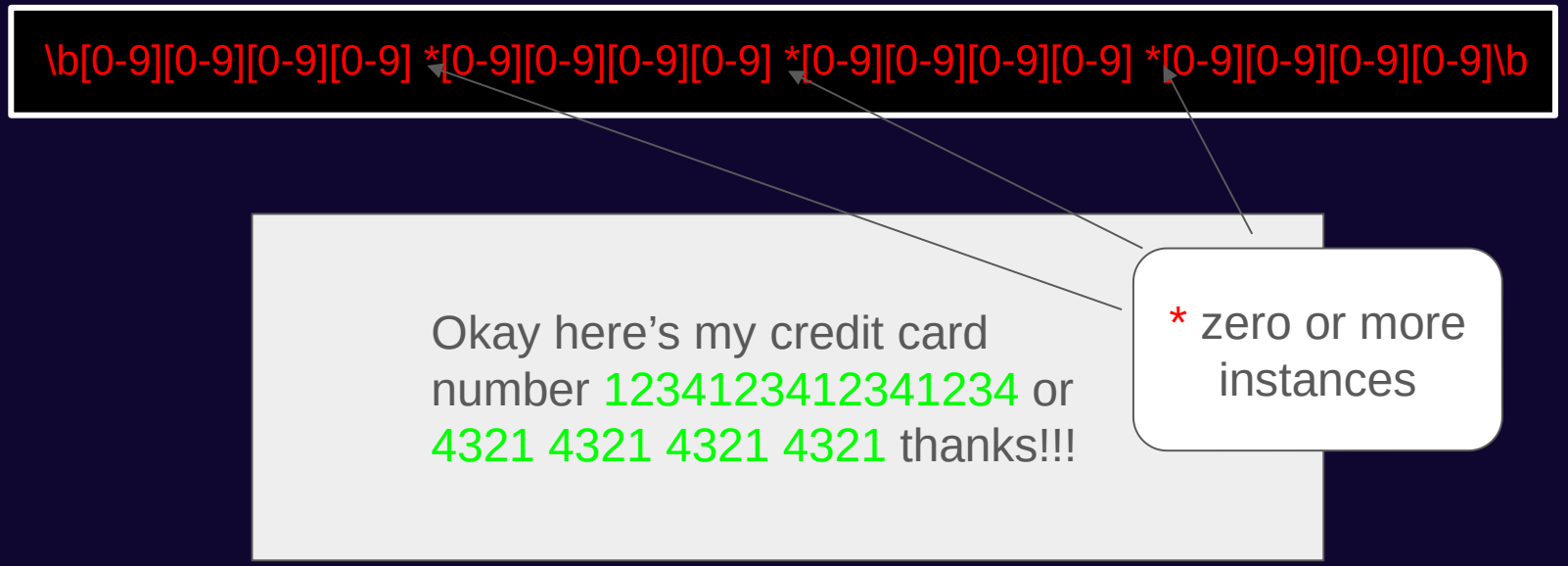
```
\b[0-9]{16,19}\b
```

{n,m} between n and m in length

Checkout my binary string:
0100011100010100100101110010100101010110010010001010101001001001001010101001010101 :)

\b[0-9]{16}\b

Okay here’s my credit card number 1234123412341234 or 4321 4321 4321 4321 thanks!!!


\b[0-9][0-9][0-9][0-9] *[0-9][0-9][0-9][0-9] *[0-9][0-9][0-9][0-9] *[0-9][0-9][0-9][0-9]\b
Okay here's my credit card number 1234123412341234 or 4321 4321 4321 4321 thanks!!!
* zero or more instances

Shortcuts

Digits

```
\b\d{4} *\d{4} *\d{4} *\d{4}\b
```

Okay here's my credit card number 1234123412341234 or 4321 4321 4321 4321 thanks!!!

\d any digits

Whitespace

```
\b\d{4}\s*\d{4}\s*\d{4}\s*\d{4}\b
```

Okay here's my credit card number 1234123412341234 or 4321 4321 4321 4321 thanks!!!

\s any whitespace character



Word Character

\b\w+\b

_adam_langley_123_

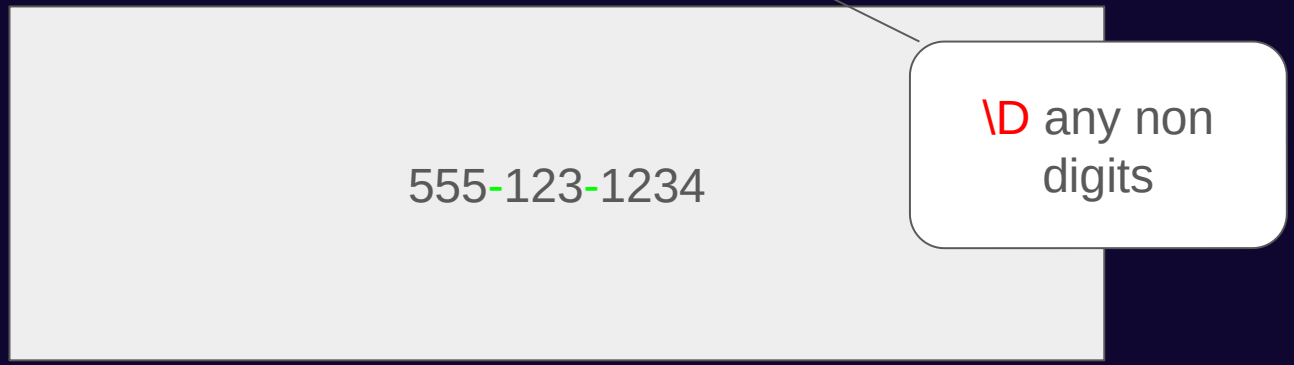
adam@email.com

\w characters
a-z A-Z 0-9 _



Non-Digits

\D



Non-Whitespace

\S

Password: “ “ - false

Password: “hunter 22” - true

\S any non whitespace character



Non-Words

\W

_adam_langley_123_

adam@email.com

\W non
a-z A-Z 0-9 _
characters



Wild Cards



```
href=".*"
```

. Wildcard character matches anything

```
<a href="https://www.example.com">
Visit My Website
</a>
```

href=".*"

Visit My Website

What if we just want to extract the link?



Groups

href=".*"

```
<a href="https://www.example.com">
Visit My Website
</a>
```

```
Array(
    [0] => href="https://www.example.com"
)
```



href="(.*)"

```
<a href="https://www.example.com">
Visit My Website
</a>
```

```
Array(
    [0] => href="https://www.example.com",
    [1] => “https://www.example.com”
)
```



\b\d{4}\s*\d{4}\s*\d{4}\s*\d{4}\b

Vs

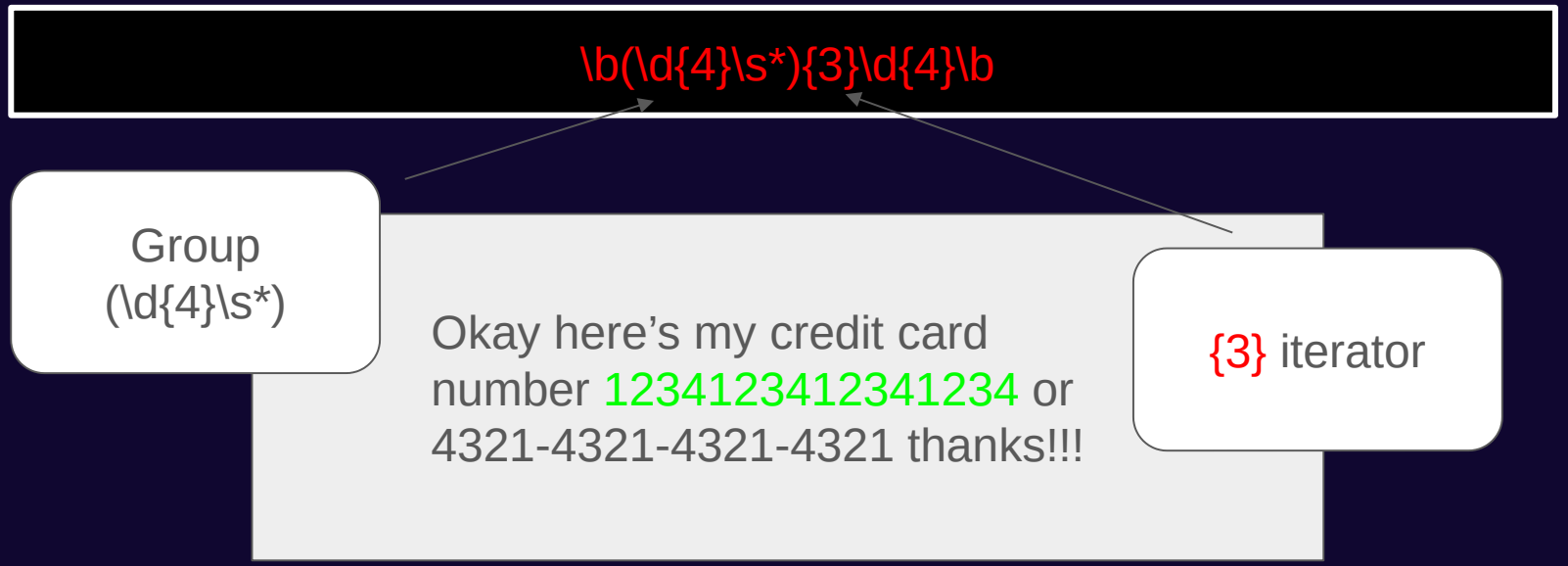
\b(\d{4}\s*){3}\d{4}\b

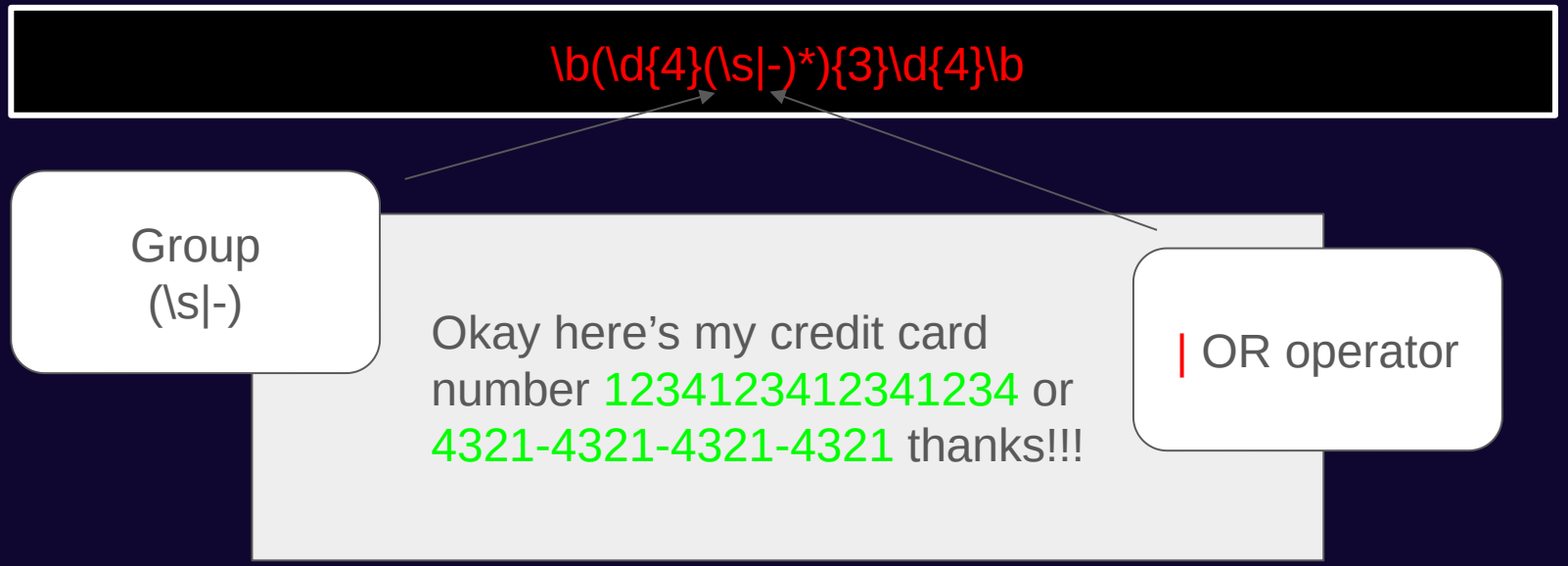
Group - parentheses (\d{4}\s*)

Okay here's my credit card number 1234123412341234 or 4321 4321 4321 4321 thanks!!!

{3} iterator




\b(\d{4}\s*){3}\d{4}\b
Group
(\d{4}\s*)
Okay here's my credit card number 1234123412341234 or 4321-4321-4321-4321 thanks!!!
{3} iterator


\b(\d{4}(\s|-)*){3}\d{4}\b
Group
(\s|-)
Okay here's my credit card number 1234123412341234 or 4321-4321-4321-4321 thanks!!!
| OR operator

When Regex Goes Wrong



When Regex Goes Wrong

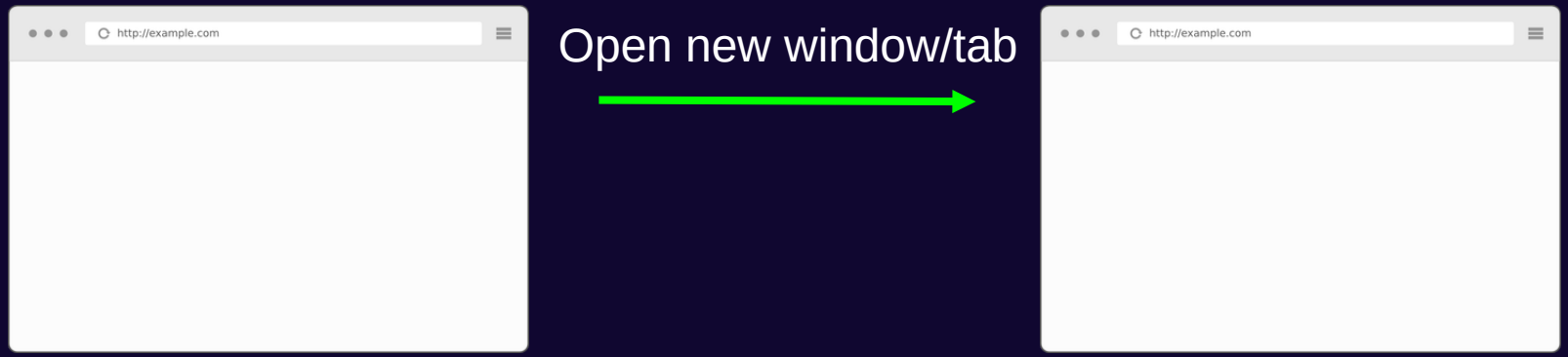
Black box?
No Code Access

White box?
Source Code Access /
or public/frontend code

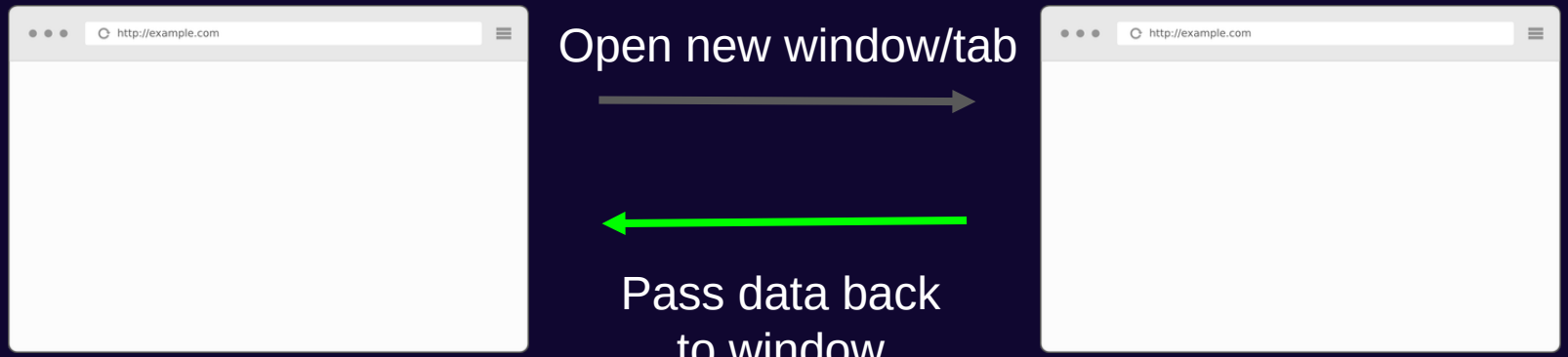


PostMessage

PostMessage



PostMessage



PostMessage

```
window.addEventListener("message", function(event) {
    if (/https:\/\/www.trusted.com/.test(event.origin)) {
        if( event.data.type === ‘error’ ){
            div.innerHTML(event.data.error.message);
        }
        // other code….
    }
});
```



```
{
    “type”: “error”,
    “error”: {
        “message”: “<img src=’x’ onerror=’alert(1)’>”
    }
}
```



- Regex For Hackers

https:\/\/www.trusted.com



Missing ends with anchor $



https:\/\/www.trusted.com

https://www.trusted.com.attacker.com



https:\/\/www.trusted.com$



- Regex For Hackers

. wildcard, requires a backslash



https:\/\/www.trusted.com$

https://wwwxtrusted.com



- Regex For Hackers

PostMessage

```
window.addEventListener("message", function(event) {
    if (/https:\/\/www\.trusted\.com$/.test(event.origin)) {
        if( event.data.type === ‘error’ ){
            div.innerHTML(event.data.error.message);
        }
        // other code….
    }
});
```



- Regex For Hackers

Open Redirects



- Regex For Hackers

Open Redirect

```
if (/https:\/\/.*.trusted.com/.test(url)) {
    // redirect to website if matches
}
```



- Regex For Hackers

. wildcard, requires a backslash

https:\/\/.*.trusted.com



- Regex For Hackers

https:\/\/.*.trusted.com

https://wwwxtrusted.com



- Regex For Hackers

.* wildcard
subdomain accepts
any characters



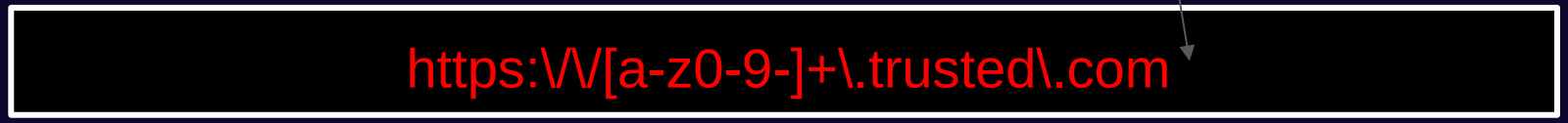
https:\/\/.*.trusted.com

https://qa.prod1.trusted.com



- Regex For Hackers

$ no ends with anchor



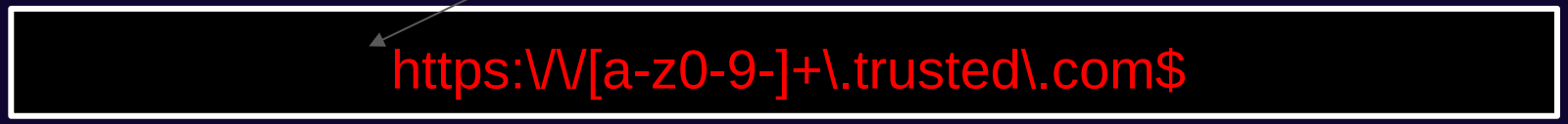
https:\/\/[a-z0-9-]+\.trusted\.com

https://www.trusted.com.evil.com



- Regex For Hackers

^ no starts with anchor



```
https:\/\/[a-z0-9-]+\.trusted\.com$
```

https://evil.com/?x=https://www.trusted.com

Open Redirect

```
if (/^https:\/\/[a-z0-9-]+\.trusted\.com$/.test(url)) {
    // redirect to website if matches
}
```



- Regex For Hackers

Open Redirect

```
if (/^https:\/\/(admin|www|download)\.trusted\.com$/.test(url)) {
    // redirect to website if matches
}
```



Open Redirect Bounty

```
https://myaccounts.redacted.com/partner-site/?sitename=someapplication&path=/auth
```



```
https://someapplication.redacted.com/auth/?auth_token=xxxxx
```



- Regex For Hackers

Open Redirect Bounty

```
https://myaccounts.redacted.com/partner-site/?sitename=someapplication&path=.evil.com
```



```
https://someapplication.redacted.com.evil.com/?auth_token=xxxxx
```



- Regex For Hackers

Open Redirect Bounty

```
https://myaccounts.redacted.com/partner-site/?sitename=someapplication&path=.evil.com
```

$14,000

```
https://someapplication.redacted.com.evil.com/?auth_token=xxxxx
```

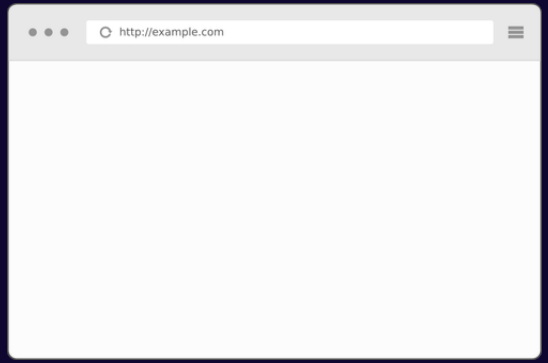


- Regex For Hackers

CORS



CORS - PreFlight Check



I’m about to make an AJAX Request to an external domain but want to send cookies with the request….

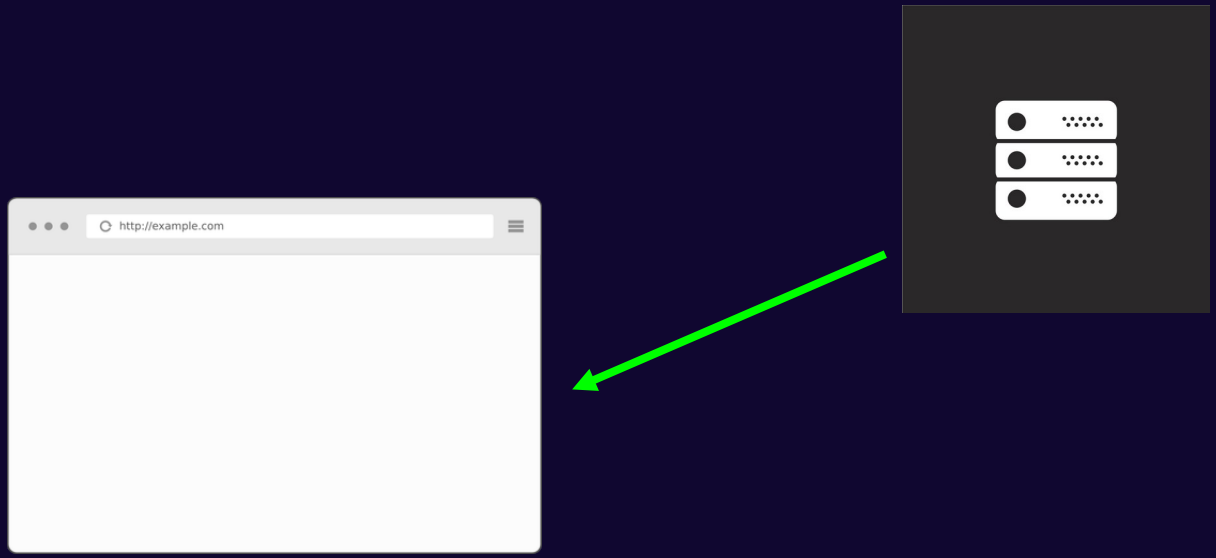
- Regex For Hackers

CORS - PreFlight Check



First send a Preflight (HTTP OPTIONS) request to make sure I’m allowed. I’ll send my domain name and protocol in the request

CORS - PreFlight Check



External server responds back either allowing or denying the request through HTTP headers.

Request

```
OPTIONS /api/me HTTP/2
Host: api.trusted.com
Origin: https://admin.trusted.com
Connection: keep-alive
```

Response

```
HTTP/2 200
Access-Control-Allow-Origin: https://admin.trusted.com
Access-Control-Allow-Credentials: true
Content-Type: application/json
```



- Regex For Hackers

CORS

```
if (/https:\/\/.*.trusted.com/.test(req.origin)) {
    header('Access-Control-Allow-Origin',req.origin)
}
```



- Regex For Hackers

. wildcard, requires a backslash

https:\/\/.*.trusted.com



- Regex For Hackers

https:\/\/.*.trusted.com

https://wwwxtrusted.com



- Regex For Hackers

.* wildcard
subdomain accepts
any characters



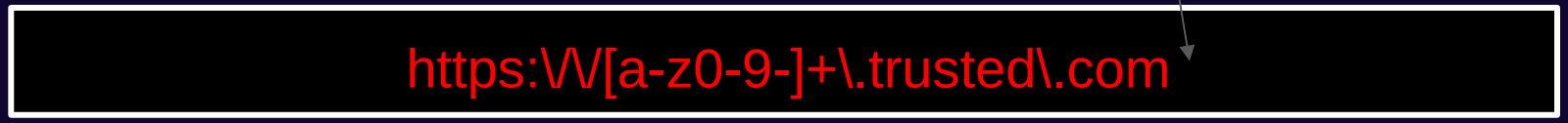
https:\/\/.*.trusted.com

https://qa.prod1.trusted.com



- Regex For Hackers

$ no ends with anchor



https:\/\/[a-z0-9-]+\.trusted\.com

https://www.trusted.com.evil.com



- Regex For Hackers

CORS

```
if (/https:\/\/[a-z0-9_]+*\.trusted\.com/.test(req.origin)) {
    header('Access-Control-Allow-Origin',req.origin)
}
```



- Regex For Hackers

SSRF



- Regex For Hackers

SSRF

```
if (/https:\/\/www.linkedin.com/.test(url)) {
    // get URL and convert to PDF
}
```



- Regex For Hackers

. wildcard, requires a backslash

https:\/\/www.linkedin.com



https:\/\/www.linkedin.com

https://wwwxlinkedin.com

DNS: A 127.0.0.1



- Regex For Hackers

/ no forwards slash at the end



https:\/\/www\.linkedin\.com

https://www.linkedin.com.evil.com

DNS: A 127.0.0.1



^ no starts with anchor

https:\/\/www\.linkedin\.com\/



https:\/\/www\.linkedin\.com\/

https://evil.com/?x=https://www.linkedin.com/

DNS: A 127.0.0.1



No exact format check



^https:\/\/www\.linkedin\.com\/

https://www.linkedin.com/redirect?url=https://evil.com

DNS: A 127.0.0.1



SSRF

```
if (/^https:\/\/www\.linkedin\.com\/in\/\w+$/.test(url)) {
    // get URL and convert to PDF
}
```

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



- Regex For Hackers

SSRF In Bug Bounty



- Regex For Hackers

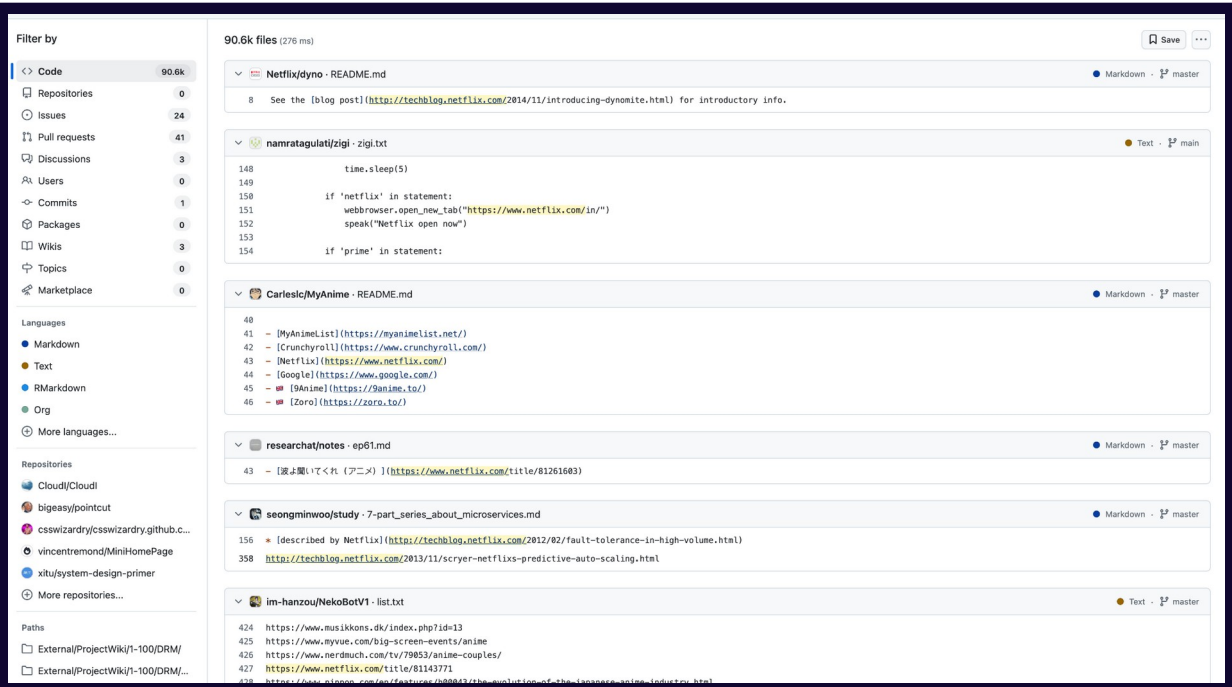
OSINT
Asset Discovery



- Regex For Hackers

Discover Netflix domains on different TLDs

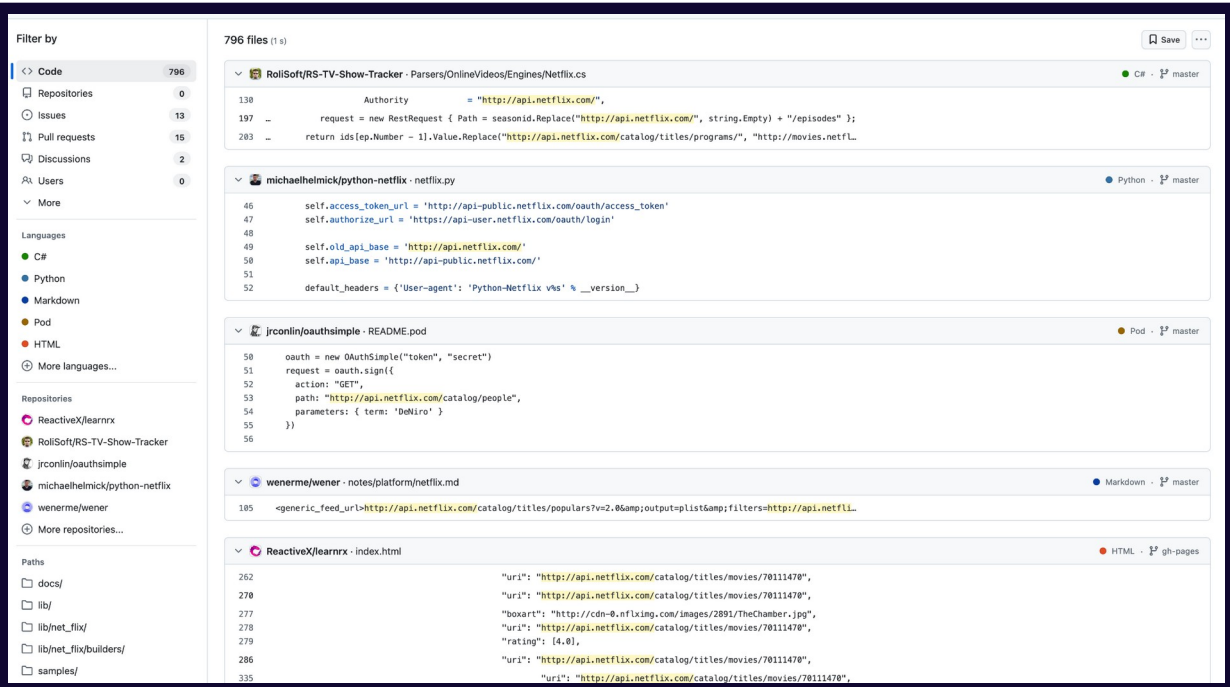
```
/https?:\/\/[a-z0-9\.-]+\.netflix.[a-z]+\//
```



- Regex For Hackers

Netflix subdomains containing api

```
/https?:\/\/([a-z0-9-]{1,}[\.])*api\.([a-z0-9-]{1,}[\.])*netflix\.[a-z\.]+\//
```



- Regex For Hackers

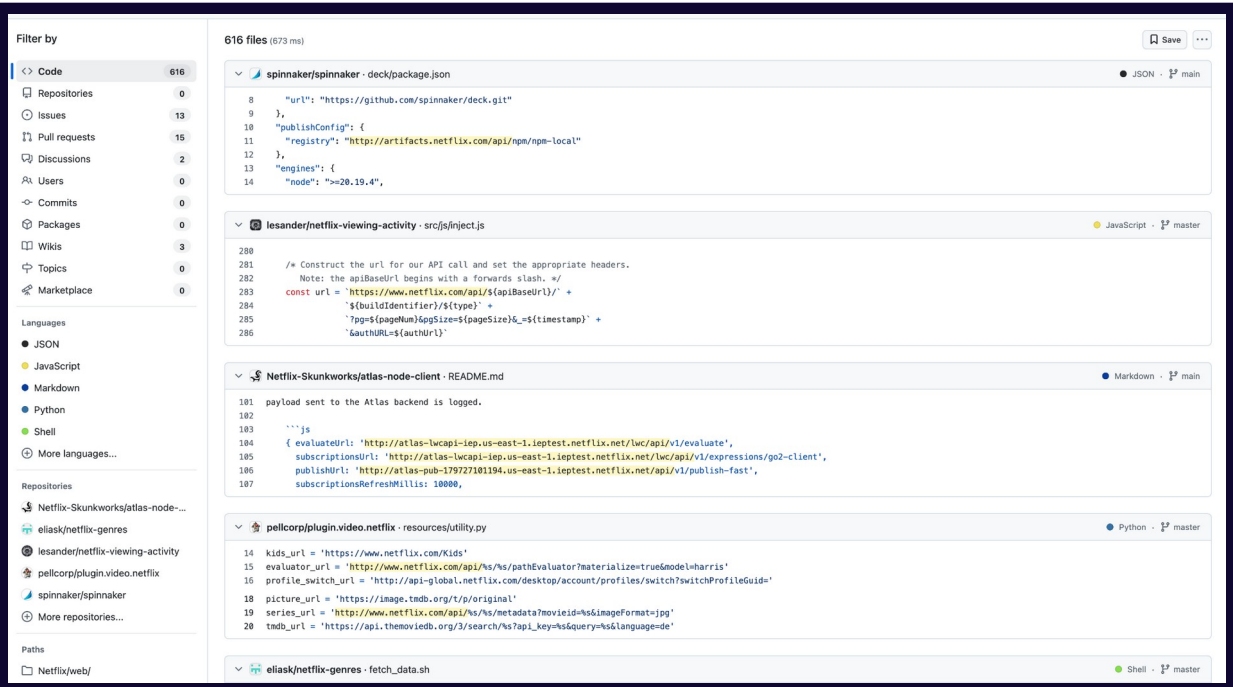
OSINT
Content Discovery



- Regex For Hackers

Netflix URLs containing */api/*

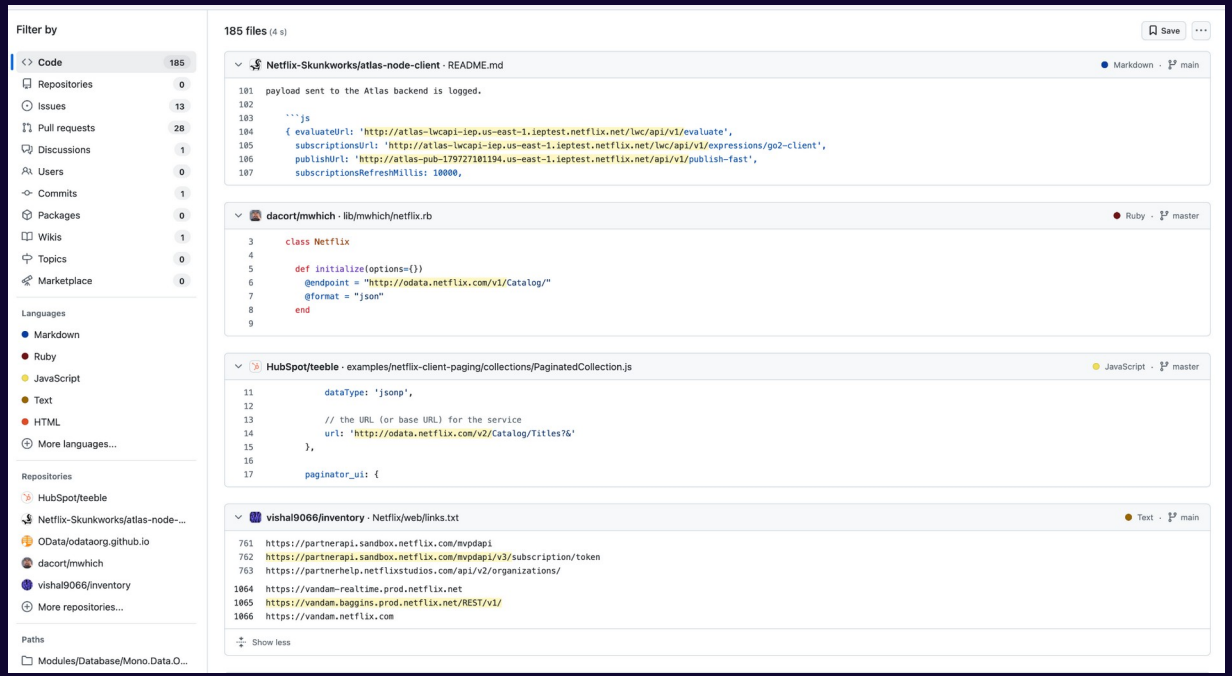
```
/https?:\/\/[a-z0-9\.-]+\.netflix.[a-z]+\/([a-z0-9-_]+\/)*api\//
```



- Regex For Hackers

Netflix URLs containing */v*/*

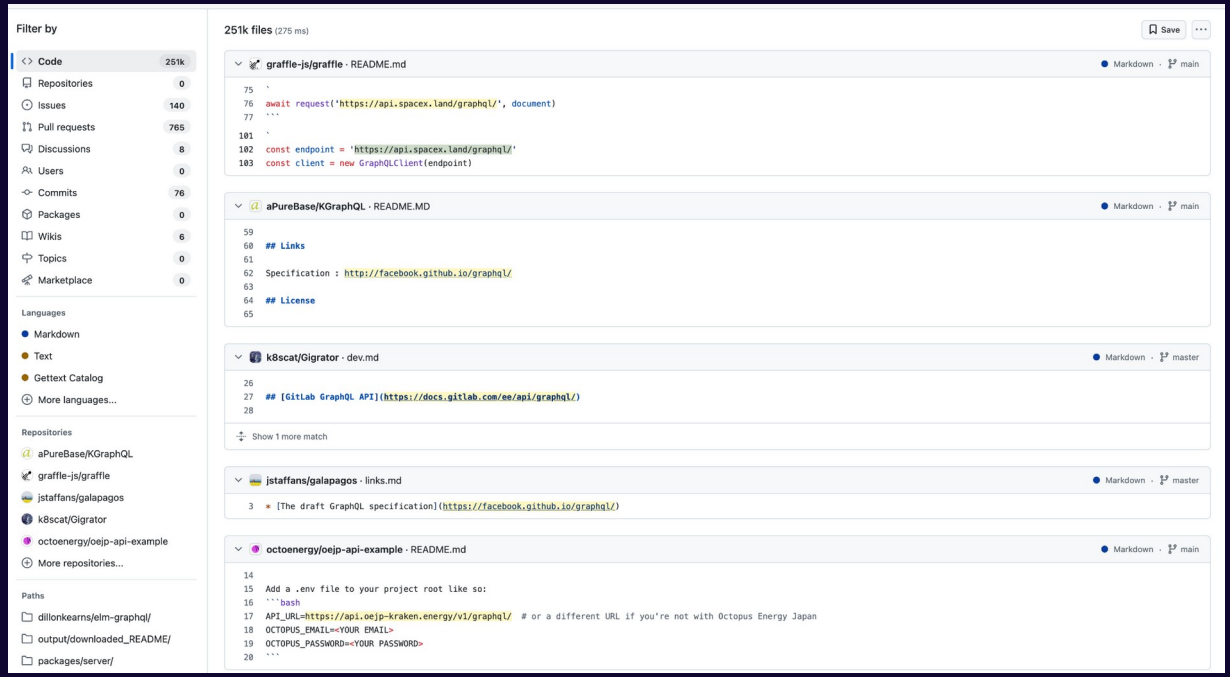
```
/https?:\/\/[a-z0-9\.-]+\.netflix.[a-z]+\/([a-z0-9-_]+\/)*v[0-9]+\//
```



- Regex For Hackers

URLs containing /graphql/

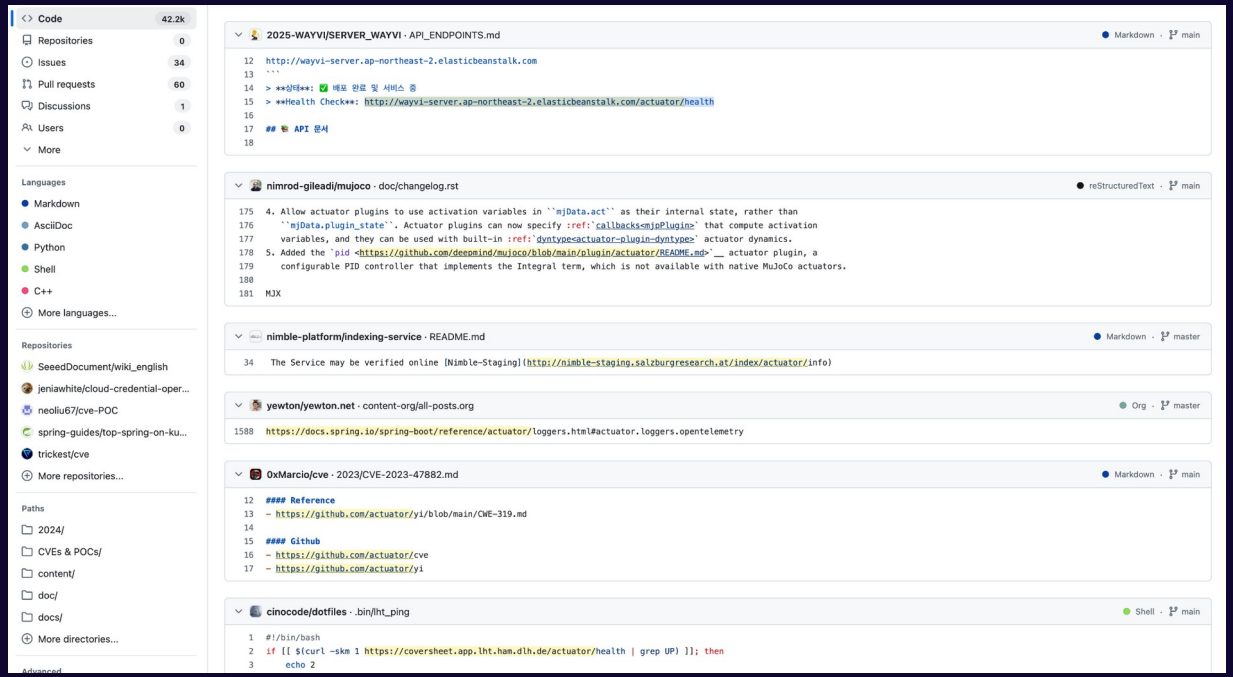
```
/https?:\/\/[a-z0-9\.-]{3,}\/([a-z0-9-_]+\/)*graphql\//
```



- Regex For Hackers

URLs containing /actuator/

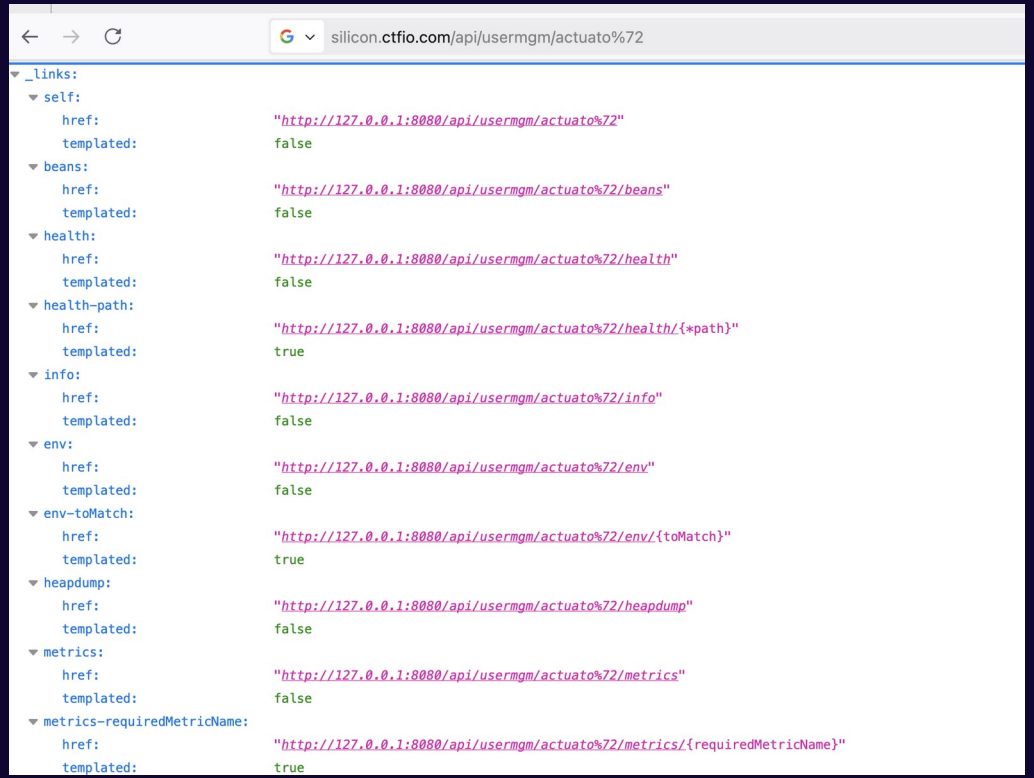
```
/https?:\/\/[a-z0-9\.-]{3,}\/([a-z0-9-_]+\/)*actuator\//
```



Parsing Data Dumps

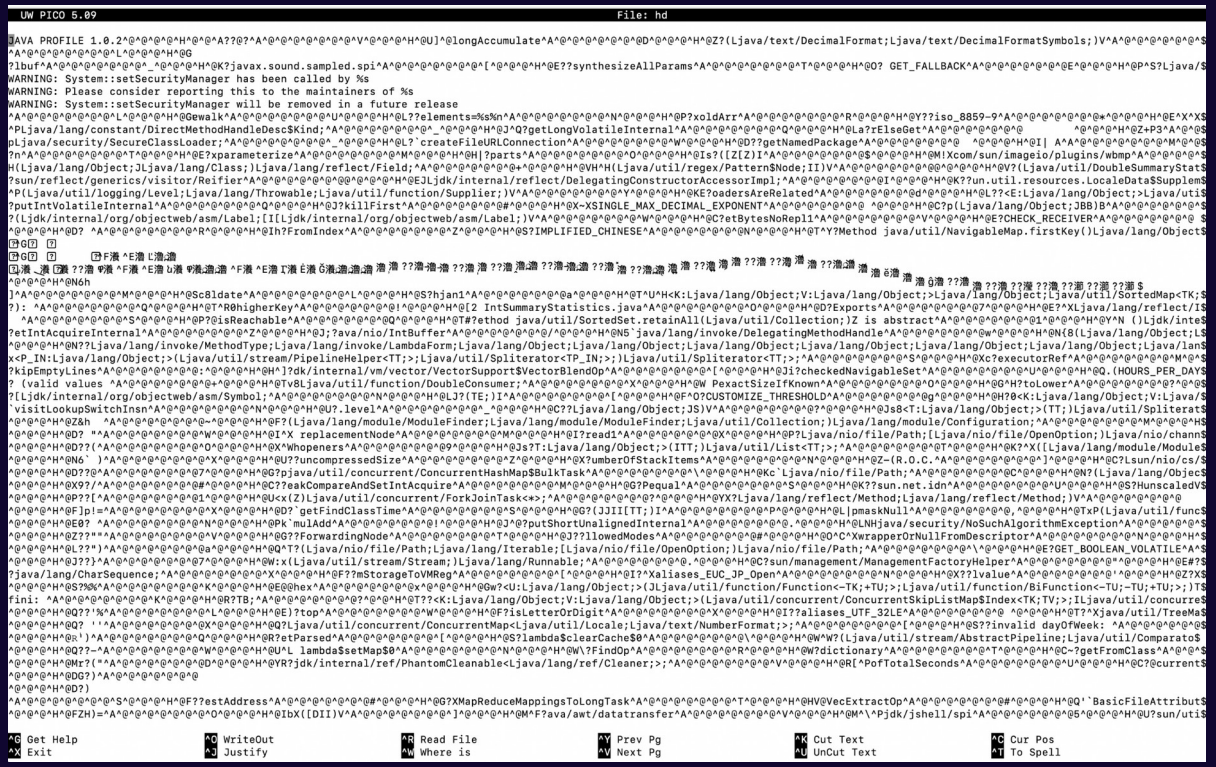


Actuator Heap Dump



- Regex For Hackers

Actuator Heap Dump



- Regex For Hackers

Actuator Heap Dump

```
adamlangley@Adams-MacBook-Pro ~ % strings hd | grep -on '[a-zA-Z0-9]\{15,\}[=]\{1,2\}'
2667:DumpLoadedClassList=
7221:cachedFixedDate=
11275:currentStandardOffset=
16666:SharedClassListFile=
18920:areAllFieldsSet=
22485:ArchiveClassesAtExit=
32166:SharedArchiveFile=
41189:targetRequestPath=
42812:parameterDescriptors=
50118:threadContextMap=
52564:interfaceClassName=
54107:propertySources=
58623:proxyTargetClass=
59180:J2EEApplication=
59384:targetRequestParams=
61700:ignoreInvalidFields=
72808:enableLoggingRequestDetails=
73149:percentileValues=
74137:defaultProfiles=
74888:ignoreUnknownFields=
86090:destroyMethodName=
88572:matrixVariables=
```



- Regex For Hackers

Actuator Heap Dump

```
132949:cachedFixedDate=
133356:SharedClassListFile=
133839:currentStandardOffset=
135582:areAllFieldsSet=
138250:ArchiveClassesAtExit=
144141:igsvresleciedaoaj=
162507:dependencyCheck=
162508:autowireCandidate=
162510:factoryBeanName=
162511:factoryMethodName=
162513:destroyMethodName=
175584:J2EEApplication=
177753:J2EEApplication=
183442:J2EEApplication=
185058:J2EEApplication=
185073:J2EEApplication=
185074:J2EEApplication=
185120:J2EEApplication=
185414:J2EEApplication=
185415:J2EEApplication=
189282:interfaceClassName=
189614:interfaceClassName=
192189:YWRtaW46YWRtaW4==
```

This looks interesting…



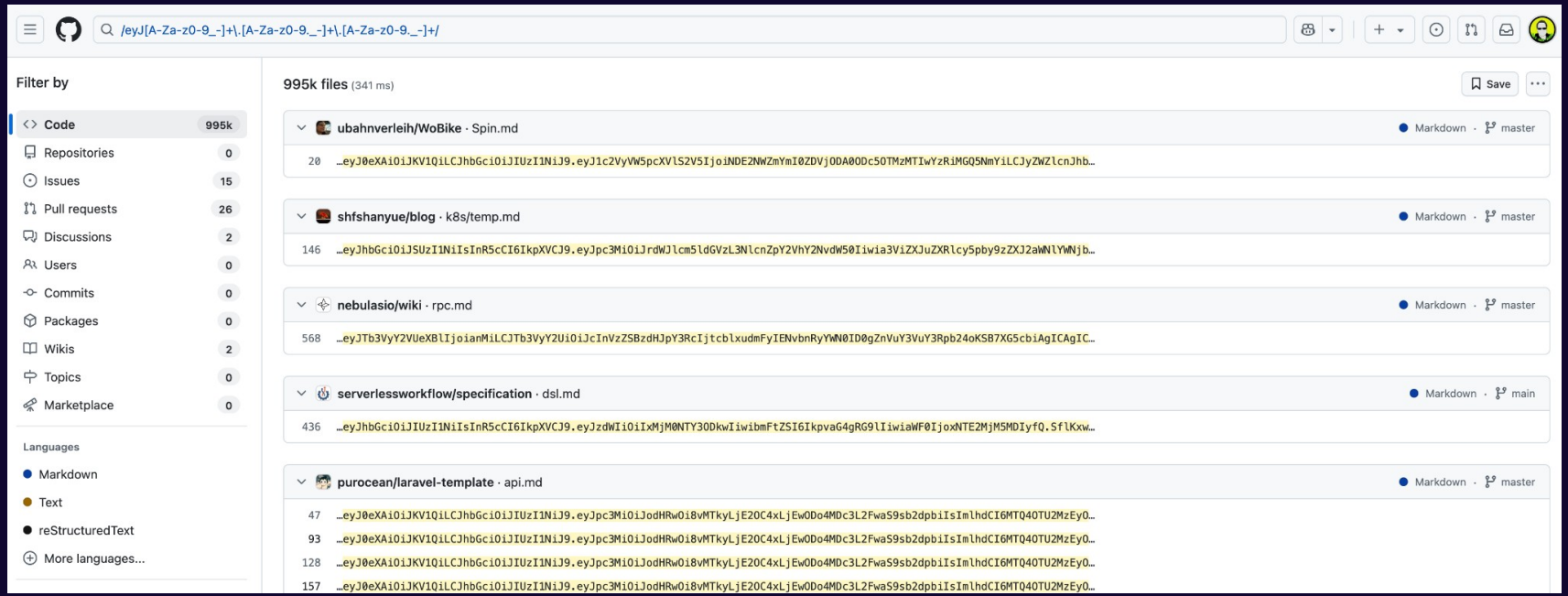
- Regex For Hackers

OSINT
Saved Secrets



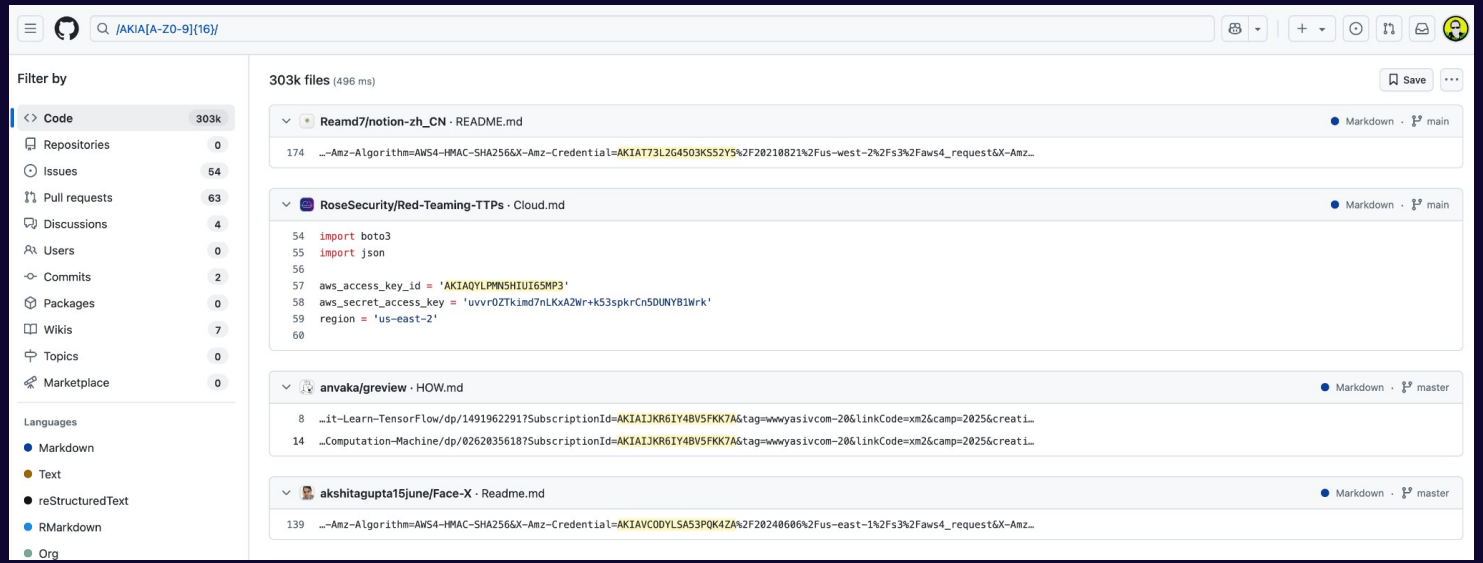
- Regex For Hackers

```
eyJ[A-Za-z0-9_-]+\.[A-Za-z0-9._-]+\.[A-Za-z0-9._-]+
```



- Regex For Hackers

AKIA[A-Z0-9]{16}





- Regex For Hackers

JWT Tokens + netflix domains

/eyJ[A-Za-z0-9_-]+\.[A-Za-z0-9._-]+\.[A-Za-z0-9._-]+/ and /netflix\.[a-z]+/

```
44  - Supervised Learning + 比較研究 ("因果効果の推定"): [Applied Causal Inference Powered by ML and AI](https://causalml-book.org/)
54  …eyJ2IjoiMSJ9.8xEhmFZMbM8Lp9oFm6XWIaOF2P9Sdnzkqz6iEWzKq0ACvKLX9L6lymL7M2hmzy2YGKYxAISncfbB-G6cZr5ecZ2RlEEnKnkh-jd148a4U…
60      - [Understanding Machine Learning: From Theory to Algorithms](https://www.cs.huji.ac.il/~shais/UnderstandingMachine…
93  - 機械学習 $+$ "因果効果"に対して、学際的な関心 [@athey2019machine; @grimmer2021machine; @brand2023recent]
95  …com/en-us/research/group/alice/), [netflix](https://research.netflix.com/), [サイバーエージェント](https://cyberagent.ai/ailab/))
105 - 予測の発展: Random forest, Boosting, Stackingなど
292     Estimand[label = "Estimand"];
300     Researcher -> Estimand[style=dashed, color=grey, label = "Image"]

Show 3 more matches
```



- Regex For Hackers

Code Review

Finding reflected values for XSS

```
echo\s*\$_(GET|POST)\[
```

```
<?php
echo $_GET[“username”];
```



- Regex For Hackers

Finding reflected values for XSS

```
<\?=\s*\$_(GET|POST)\[
```

```
<div>
<p>Hello, <?= $_GET[“username”]; ?></p>
</div>
```

Finding system commands

```
system\s*\(
```

```
<?php
system("ping 8.8.8.8");
```



Finding regex’s that could be insecure

```
(/[^/\n]+/[gimsuy]*)|(re\.(search|match|compile)|preg_match|new RegExp)\s*\(\s*["'`].+?["'`]
```

The above regex checks for usage of regex in code for PHP, JavaScript and Python.

Finding Basic Auth

```
[A-Za-z0-9+/]{20,}={0,2}
```

server = “https://admin.example.com”
auth = “dXNlcm5hbWU6c2VjcmV0cGFzc3dvcmQ=”



- Regex For Hackers

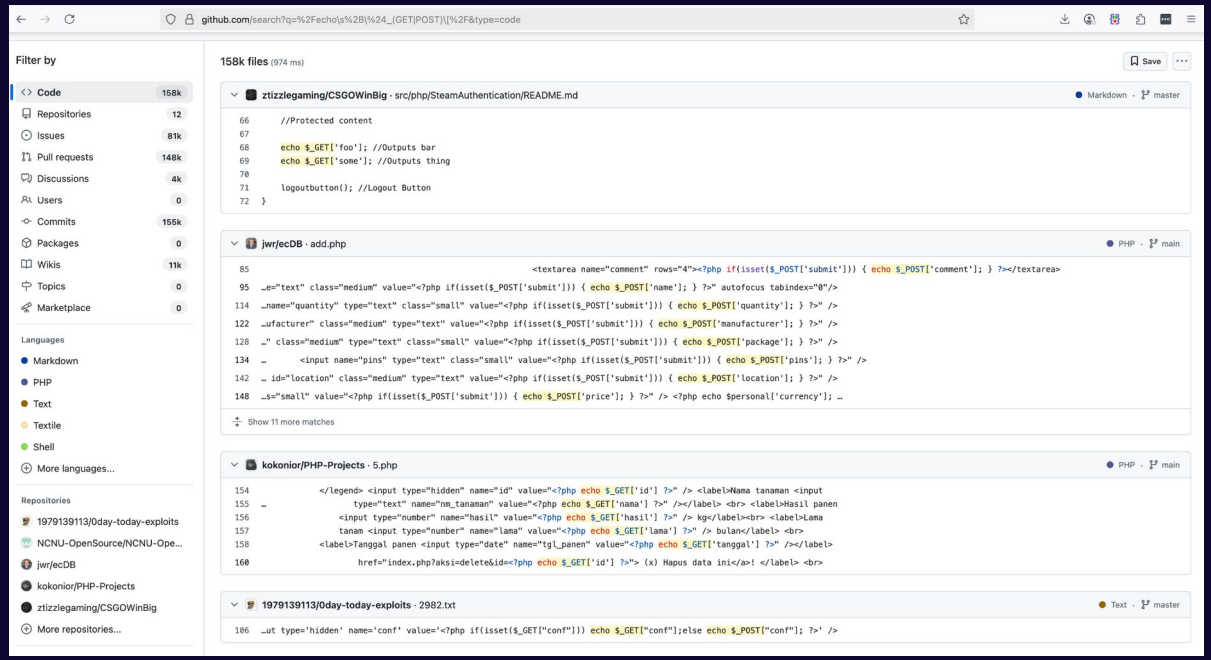
Finding JWT Tokens

```
eyJ[A-Za-z0-9_-]+\.[A-Za-z0-9._-]+\.[A-Za-z0-9._-]+
```

```
<?php
$token =
“eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWUsImlhdCI6MTUxNjIzOTAyMn0.KMUFsIDTnFmyG3nMiGM6H9FNFUROf3wh7SmqJp-QV30”;
$http_request->header(“Authorization: Bearer ”.$token);
```

Finding XSS on GitHub

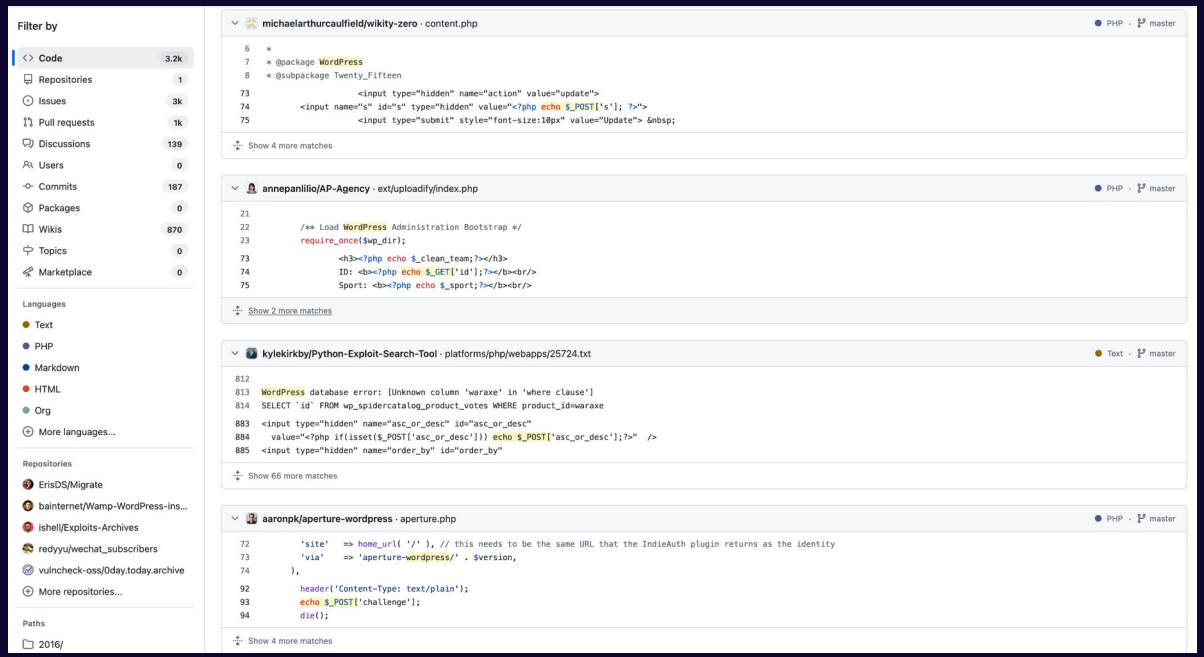
```
/echo\s+\$_(GET|POST)\[/
```



- Regex For Hackers

Finding Wordpress XSS on GitHub

/echo\s+\$_(GET|POST)\[/ AND wordpress



- Regex For Hackers

Finding RCE on GitHub

```
/system\(\s*('|")[^"']*\$_(GET|POST|REQUEST|SERVER)\[/ path:.php
```

```
5       }
6   }
7   if ($_GET['act'] == 'delete') {
8       system("cancel {$_GET['jobid']}");
9       Header("Location:index.php");
0       //echo "<script type 'text/javascript'>location.href='index.php'</script>";
1   }
```



- Regex For Hackers

Finding RCE on GitHub

```
/system\(\s*('|")[^"']*\$_(GET|POST|REQUEST|SERVER)\[/ path:.php
```

```
1  GIF8
2  <?php system('$_GET["rce"]'); ?>
```



- Regex For Hackers

Finding RCE on GitHub

```
/system\(\s*('|")[^"']*\$_(GET|POST|REQUEST|SERVER)\[/ path:.php
```

```
1   GIF8
2   <?php system('$_GET["rce"]'); ?>
```



Building Tools



Creatures Of Habit

Examine all public repos of a user or organisation for web routes in various frameworks…



- Regex For Hackers

Creatures Of Habit

Python - Flask

```
\.route\(\s*('|")([^"']{1,})('|")\)
```

```
@app.route('/admin')
def home():
    return "Hello, world!"
```



- Regex For Hackers

Creatures Of Habit

PHP - Laravel

```
Route::([a-z]{1,})\((\s*('|")([^"']{1,})('|"))
```

```
Route::get('/admin-logs', function () {
    return 'Hello, world!';
});
```



- Regex For Hackers

Creatures Of Habit

JavaScript - Express JS

```
\.(get|post|put|delete|patch)\(\s*('|")([^"']{1,})('|\")
```

```
app.get('/private', (req, res) => {
  res.send('Hello, world!');
});
```



- Regex For Hackers

Creatures Of Habit

Python - Django

```
path\(\s*('|")([^"']{1,})('|")
```

```
urlpatterns = [
    path('/admin-home', views.home, name='home'),
]
```



- Regex For Hackers

Creatures Of Habit

ASP.NET

```
\.Map(Get|Post|Put|Patch|Delete)\(\s*('|")([^"']{1,})('|")
```

```
app.MapGet("/secret", () => "Hello, world!");
```



- Regex For Hackers

Creatures Of Habit

Ruby On Rails

```
get\s+('|")([^"']{1,})('|")
```

```
Rails.application.routes.draw do
  get '/daily-backups', to: 'backups#daily'
end
```

Creatures Of Habit

GO! HTTP

```
\.(HandleFunc|Handle)\(\s*('|")([^"']{1,})('|")
```

```
func main() {
    http.HandleFunc("/admin-only",
dailyBackupsHandler)
    http.ListenAndServe(":8080", nil)
}
```



- Regex For Hackers

Git Scraper

Examines Repo PHP code for:

Parameters in query strings or http body

HTTP Headers

Methods

Git Scraper

Parameters

```
\$_(GET|POST)\[\s*('|")([^"']{1,})('|")\]
```

```
<?php
$var = $_GET["file_name"];
$var2 = $_POST["user_id"];
```

Git Scraper

HTTP Headers

```
\$_(SERVER)\[\s*('|")([^"']{1,})('|")\]
```

```
<?php
$userId = $_SERVER["HTTP_USER_ID"];
```



- Regex For Hackers

Git Scraper

Get Method Names

```
function\s*([a-zA-Z_]\w+)\(
```

```
<?php
public function getUser(int $id) : int {
    //do code….
}
```



- Regex For Hackers

Thank You!

Any Questions?

