

# Writing YARA rules

An introduction to YARA for AIL usage



**CIRCL**

Computer Incident  
Response Center  
Luxembourg

Alexandre Dulaunoy

[alexandre.dulaunoy@circl.lu](mailto:alexandre.dulaunoy@circl.lu)

Jean-Louis Huynen

[jean-louis.huynen@circl.lu](mailto:jean-louis.huynen@circl.lu)

[info@circl.lu](mailto:info@circl.lu)

November 25, 2022

# Links

---

- ALL project: <https://github.com/ail-project>
- ALL framework:  
<https://github.com/ail-project/ail-framework>
- Training materials:  
<https://github.com/ail-project/ail-training>
- YARA doc: <https://yara.readthedocs.io/en/stable/>
- YARA download: <http://virustotal.github.io/yara/>

## What's YARA?

---

- *The pattern matching swiss knife for malware researchers (and everyone else);*
- It's an improved **grep** to create pattern matching rule to search for **strings, binary patterns, regular expressions;**
- A YARA rule can be contextualised with metadata and tags describing a specific set of pattern matching rules.
- Easier definition of conditions compared to regex.

## A sample rule - disneyplus.yara

---

```
1 rule disney_plus : credential_leak
2 {
3     meta:
4         description = "Finding list of credentials for
5         Disney Plus"
6         leak = 1
7     strings:
8         $a = "gmail.com:"
9         $b = "DISNEY_PLUS"
10        $c = "Disney Plus"
11    condition:
12        $a and ($b or $c)
13 }
```

## Calling yara from command line

---

- Searching a single file

```
1 yara disneyplus.yara /home/adulau/dataset/2021/09/01/  
   nv6RsKFm  
2
```

- Searching a directory

```
1 yara disneyplus.yara -r /home/adulau/dataset  
   /2021/09/01/  
2
```

# Regular Expressions

---

- Regular Expressions (Regex) are extremely useful in extracting information from text
- A regex is a sequence of characters that specifies a search pattern
- They can be used to match, locate extract and replace text

# Regular Expressions

---

You can search for simple letters and specify repetition or existence

```
1 String: 'Cookie'
2
3 // '.' Any single character
4 $re1: /Co.kie/
5
6 // '*' Zero or more for the previous sequence
7 $re2: /Co*kie/
8
9 // '+' One or more for the previous sequence
10 $re3: /Co+kie/
```

# Regular Expressions

---

```
1 // '{2,3}' Between 2 and 3
2 $re4: /Co{2,3}kie/
3
4 // '[a-zA-Z]' Any letter between 'a' and 'Z'
5 $re5: /Co[a-zA-Z]kie/
```



# Regular Expressions

---

Usecase: Email addresses

```
1 $re1: /.+@.+\...+/  
2 // The address '1@.' is valid  
3  
4  
5 $re2: /.+@.+\. [a-zA-Z]{2,4}/  
6 // We enforce a correct TLD (i.e. '.com')
```

# Regular Expressions

## Usecase: Email addresses

```
1 $re2: /.+@[a-zA-Z0-9_.-]+\.[a-zA-Z]{2,4}/
2 // We enforce a correct domain (i.e. 'gmail' or 'hotmail
   // ')
3
4 $re3: /[a-zA-Z0-9_%.%+-]+@[a-zA-Z0-9_.-]+\.[a-zA-Z]{2,4}/
5 // We enforce a correct email format
6 // '!john@doe~@gmail.com' is not valid anymore
```

The screenshot shows a regular expression testing interface. The regular expression is `/([\w_%.%+-]+)([\w.-]+\.[a-zA-Z]{2,4})/m`. The test string is `john.doe@gmail.com`. The tool reports 1 match in 20 steps, 0.0ms. The match information table is as follows:

MATCH INFORMATION		
Match 1	0-18	john.doe@gmail.com
Group 1	0-8	john.doe
Group 2	9-14	gmail
Group 3	15-18	com

# Fun with Regular Expressions

---

<https://regexcrossword.com/>

	(FY F RG)+	[NODE]+	(.)[IF]+	(YE OT)K	(FI A)+
(Y F)(.)\2[DAF]\1					
(U O I)*T[FRO]+					
[KANE]*[GIN]*					

# Fun with Regular Expressions

---

<https://regexcrossword.com/>

	(FI A)+	(YE OT)K	(.) [IF]+	[NODE]+	(FY F RG)+
(Y F)(.)\2[DAF]\1	F	O	O	D	F
(U O I)*T[FR0]+	I	T	F	O	R
[KANE]*[GIN]*	A	K	I	N	G



## Searching in binaries

---

```
1 rule torcryptomining
2 {
3     strings:
4         $supx_erase = {(00 FF 99 41|DF DD 30 33)}
5     condition:
6         $supx_erase at 236
7 }
8
```