Anatomy of a yara Rule

Yara is a tool used to identify file, based on textual or binary pattern.

A rule consists of a set of strings and conditions that determine its logic.

Rules can be compiled with "yarac" to increase the speed of multiple Yara scans.

1 Import Module

Yara modules allow you to extend its functionality. The PE module can be used to match specific data from a PE:

- pe.number_of_exports
- pe.sections[0].name
- pe.imphash()
- pe.imports("kernel32.dll")
- pe.is_dll()

List of modules: pe, elf, hash, math, cuckoo, dotnet, time

2 Rule Name

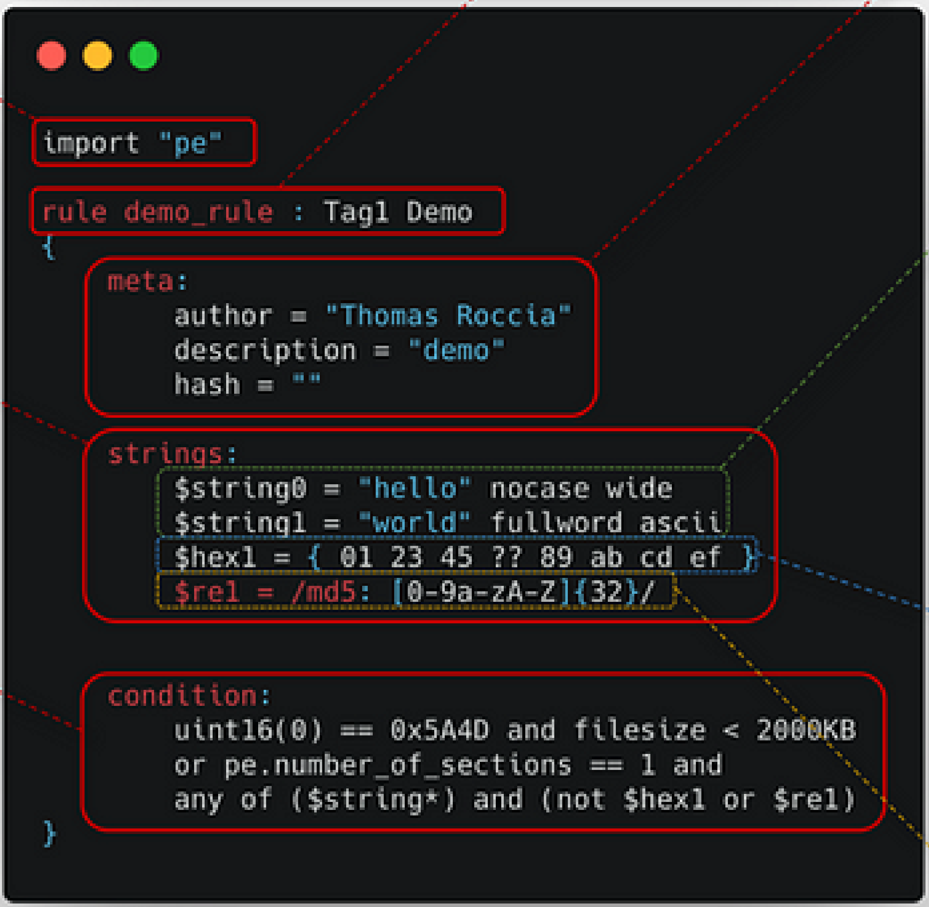
The rule name identifies your Yara rule. It is recommended to add a meaningful name. There are different types of rules:

- Global rules: applies for all your rules in the file.
- Private rules: can be called in a condition of a rule but not reported.
- Rule tags: used to filter yara's output.

3 Metadata

Rules can also have a metadata section where you can put additional information about your rule.

- Author
- Date
- Description
- Etc..



```
import "pe"

rule demo_rule : Tag1 Demo
{
    meta:
        author = "Thomas Roccia"
        description = "demo"
        hash = ""

    strings:
        $string0 = "hello" nocase wide
        $string1 = "world" fullword ascii
        $hex1 = { 01 23 45 ?? 89 ab cd ef }
        $re1 = /md5: [0-9a-zA-Z]{32}/


    condition:
        uint16(0) == 0x5A4D and filesize < 2000KB
        or pe.number_of_sections == 1 and
        any of ($string*) and (not $hex1 or $re1)
}
```

Text Strings

Text strings can be used with modifiers:

- nocase: case insensitive
- wide: encoded strings with 2 bytes per character
- fullword: non alphanumeric
- xor(0x01-0xff): look for xor encryption
- base64: base64 encoding

4 Strings

The field strings is used to define the strings that should match your rule. It exists 3 type of strings:

- Text strings
- Hexadecimal strings
- Regex

Hexadecimal

Hex strings can be used to match piece of code:

- Wild-cards: { 00 ?2 A? }
- Jump: { 3B [2-4] B4 }
- Alternatives: { F4 (B4 | 56) }

5 Condition

Conditions are Boolean expressions used to match the defined pattern.

- Boolean operators:
  - and, or, not
  - <=, >=, ==, <, >, !=
- Arithmetic operators:
  - +, -, *, \, %
- Bitwise operators:
  - &, |, <<, >>, ^, ~
- Counting strings:
  - #string0 == 5
- Strings offset:
  - $string1 at 100

Advanced Condition

- Accessing data at a given position: uint16(0) == 0x5A4D
- Check the size of the file: filesize < 2000KB
- Set of strings: any of ($string0, $hex1)
- Same condition to many strings: for all of them : (# > 3)
- Scan entry point: $value at pe.entry_point
- Match length: !re1[1] == 32
- Search within a range of offsets: $value in (0..100)

Regex

Regular expression can also be used and defined as text strings but enclosed in forward slash.

@FR0GGER_
Thomas Roccia