

Ročníkový projekt (ZS)

1 Práca počas ZS

Počas zimného semestra som robil prvú časť projektu, tvorba gramatiky pre syntax YARA pravidiel. Toto bol potrebný krok pred začatím vývoja samotného editora, lebo korektná gramatika neexistovala a bez nej nie je možné robiť kontrolu syntaxe.

2 Postup

Hlavnou prekážkou bola absencia akejkoľvek presnej oficiálnej špecifikácie syntaxe.

Počas hľadania či existuje nejaké open-source riešenie s využitím Tree-sitter, ktoré korektne popisuje gramatiku, som našiel jednu implementáciu. Tá však pokrývala len časť syntaxe a gramatiku bolo potrebné rozšíriť, aby rešpektovala pravidlá syntaxe aktuálnej verzie Yara pravidiel.

Ako najpriateľnejšie riešenie sa javilo ručne odvodiť gramatiku z tutoriálu Writing YARA rules. Toto riešenie sa nakoniec ukázalo ako celkom validné.

Tree-sitter dokáže ako vstup zobrať gramatiku napísanú v špeciálnom DSL a ako výstup dať parser, ktorý následne vie vypočítať syntaktický strom pre nejaký zdrojový súbor. Počas písania gramatiky som parser testoval na príkladoch z tutoriálu, aby som overil správnosť písaných pravidiel. Po prechode celého tutoriálu som parser otestoval aj na dlhších a zložitejších pravidlách, presnejšie na všetkých pravidlách z repozitáru Yara-Rules.

3 Výsledok

Oficiálna špecifikácia Yara syntaxe neexistuje, a preto parser určite nešpecifikuje úplnú Yara syntax. Môj repozitár obsahuje dosiahnutý stav parsera a gramatiky definovanej pomocou Tree-sitter JavaScript DSL.

4 Moje príspevky

Zdrojové súbory Yara pravidiel obsahujú rôzne sekcie. Na existujúcej gramatike som vykonal mnoho opráv i rošírení, ale väčšina z nich sa týkala najmä sekcii *Strings* a *Condition*.

4.1 Rozšírenia syntaxe pre *Strings* sekciu

Pridaná syntax pre:

- sekvencie bytov v hex string alternatívach
- nibble-wise wildcard v hex string
- operátor *not* (\sim) v hex string
- skoky (*jumps*) v hex string
- rozsahy (*ranges*) ako parameter *xor* modifikátora
- riadiace/špeciálne znaky (*escape sequence*) v text string
- *private* modifikátor

4.2 Rozšírenia syntaxe pre *Condition* sekciu

Pridaná syntax pre:

- počet výskytov (*count*) stringov v určitom rozsahu adries
- výskyt stringu na určitej adrese (**at** operátor)
- výskyt stringu v určitom rozsahu adries (**in** operátor)
- premenné z modulov
- volania funkcií čítajúcich celočíselné hodnoty z adresy
- volania funkcií z modulov
- množiny stringov
- špeciálne výrazy typu **of** a **for...of**
- iterátory:
`for <quantifier> <variables> in <iterable> : (<podmienka používajúca variables>)`
- necelé čísla (*float*)
- referencie na iné Yara pravidlá
- bitwise operátory AND (**&**), OR (**|**), XOR (**^**), NOT (**~**), shift (**<<**, **>>**)

Odstránená syntax pre:

- nepodporované kľúčové slovo **entrypoint**

4.3 Iné vylepšenia

Napríklad:

- lepšie pomenovanie pravidiel v gramatike
- správna priorita operátorov
- lepšie spracovanie komentárov
- oddelenie podmnožiny výrazov ako číselné výrazy, string výrazy
- podpora pre anonymné mená stringov (**\$** namiesto **\$a**)

5 Ukážka

Nižšie je ukážka súboru, ktorý využíva veľkú časť spomenutých rozšírení syntaxe.

example.yar:

```
1 import "pe"
2 import "math"
3
4 private rule DEM01
5 {
6     meta:
7         desc = "Demo1 rule. Hex alternatives, wildcards, not, jumps, string modifiers"
8     strings:
9         $h1      = { 90 ~F? [6] ?? ( 4F?1 82 | A0 CE~03) [2-5]}
10        $s2       = "line1\nline2\t\"quote\"" xor(0x01-0xff) ascii private
11    condition:
12        #h1 in (0..100) >= 1 and
13        ($s2 at pe.entry_point) or (any of ($s2, $h1) in (0..512))
14 }
15
16 rule DEM02 : trojan loader
17 {
18     meta:
19         desc = "Demo2 showcasing condition features, references Demo1 rule"
20         num1 = 34
21         num2 = 009436
22         num3 = 000
23     strings:
24         $http    = "http://" nocase
25         $sig      = { 6A ?? 68 }
26         $magic    = { 4D 5A }
27     condition:
28         uint16(0) == 0x5A4D and pe.is_pe and math.entropy(0, filesize) > 6.5 and
29         (1 of ($http, $sig) or any of ($http, $sig) in (0..filesize\2)) and
30         #sig in (0..filesize) >= 1 and $sig in (100..2000) and $magic at 0 and
31         for any i in (0..#sig - 1) : (@sig[i] < filesize) and
32         for any sec in pe.sections : (sec.virtual_size > 1000 and
33         (sec.characteristics & 0x20000000) != 0) and DEM01
34 }
```

Výstup parsera pri spracovaní definície \$h1:

example.yar

```
1 ...
2 private rule DEM01
3 {
4     ...
5     strings:
6         $h1 = { 90 ~F? [6] ?? ( 4F?1 82 | A0
7             CE~03) [2-5]DE}
8         ...
9     }
```

```
1 (strings_section [7, 2] - [9, 64]
2   (string_definition [8, 4] - [8, 56]
3     name: (string_identifier [8, 4] - [8, 7])
4     value: (hex_string [8, 10] - [8, 56]
5       (hex_seq [8, 12] - [8, 18])
6       (hex_jump [8, 19] - [8, 22]
7         (integer_decimal_positive [8, 20] - [8, 21])
8       )
9       (hex_seq [8, 23] - [8, 25])
10      (hex_alternative [8, 26] - [8, 47]
11        (hex_seq [8, 28] - [8, 35])
12        (hex_seq [8, 38] - [8, 46]))
13      (hex_jump [8, 48] - [8, 53]
14        (integer_decimal_positive [8, 49] - [8, 50])
15        (integer_decimal_positive [8, 51] - [8, 52])
16      )
17    )
18  )
```

Výstup parsera pri spracovaní sekcie `condition`:

example.yar

```
1  ...
2  private rule DEM01
3  {
4      ...
5      condition:
6          #h1 in (0..100) >= 1 and
7          ($s2 at pe.entry_point) or (any of (
8              $s2, $h1) in (0..512))
9  }
```

```
1  (condition_section [10, 2] - [12, 62]
2      (binary_expression [11, 4] - [12, 62]
3          left: (binary_expression [11, 4] - [12, 27]
4              left: (binary_expression [11, 4] - [11, 24]
5                  left: (string_count [11, 4] - [11, 19]
6                      (range [11, 11] - [11, 19]
7                          (integer_zero [11, 12] - [11, 13])
8                          (integer_decimal_positive [11, 15] -
9                              [11, 18]))))
10                 right: (integer_decimal_positive [11, 23] -
11                     [11, 24]))
12                 right: (parenthesized_expression [12, 4] -
13                     [12, 27]
14                         (string_at_offset [12, 5] - [12, 26]
15                             (string_identifier [12, 5] - [12, 8])
16                             (module_var_or_func [12, 12] - [12, 26]
17                                 (module_identifier [12, 12] - [12, 14])
18                                 (identifier [12, 15] - [12, 26])))))
19             right: (parenthesized_expression [12, 31] - [12,
20                 62]
21                     (string_at_range [12, 32] - [12, 61]
22                         (of_expression [12, 32] - [12, 49]
23                             (quantifier [12, 32] - [12, 35])
24                             (string_set [12, 39] - [12, 49]
25                                 (string_identifier [12, 40] - [12, 43])
26                                 (string_identifier [12, 45] - [12, 48]))
27                             )
28                         (range [12, 53] - [12, 61]
29                             (integer_zero [12, 54] - [12, 55])
30                             (integer_decimal_positive [12, 57] - [12,
31                                 60]))))))))
```