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* (`~`) 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:

```
import "pe"
import "math"

private rule DEMO1
{
  meta:
    desc = "Demo1 rule. Hex alternatives, wildcards, not, jumps, string modifiers"
  strings:
    $h1 = { 90 ~F? [6] ?? ( 4F?1 82 | A0 CE~03) [2-5]DE}
    $s2 = "line1\nline2\t\"quote\"" xor(0x01-0xff) ascii private
  condition:
    #h1 in (0..100) >= 1 and
    ($s2 at pe.entry_point) or (any of ($s2, $h1) in (0..512))
}

rule DEMO2 : trojan loader
{
  meta:
    desc = "Demo2 showcasing condition features, references Demo1 rule"
    num1 = 34
    num2 = 009436
    num3 = 000
  strings:
    $http  = "http://" nocase
    $sig   = { 6A ?? 68 }
    $magic   = { 4D 5A }
  condition:
    uint16(0) == 0x5A4D and pe.is_pe and math.entropy(0, filesize) > 6.5 and
    (1 of ($http, $sig) or any of ($http, $sig) in (0..filesize\2)) and
    #sig in (0..filesize) >= 1 and $sig in (100..2000) and $magic at 0 and
    for any i in (0..#sig - 1) : (@sig[i] < filesize) and
    for any sec in pe.sections : (sec.virtual_size > 1000 and
    (sec.characteristics & 0x20000000) != 0) and DEMO1
}
```

Výstup parsera pri spracovaní definície `$h1`:

example.yar

```
...
private rule DEMO1
{
  ...
  strings:
    $h1 = { 90 ~F? [6] ?? ( 4F?1 82 | A0
   CE~03) [2-5]DE}
  ...
}
```

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

Výstup parsera pri spracovaní sekcie `condition`:

example.yar

```
...
private rule DEMO1
{
  ...
  condition:
    #h1 in (0..100) >= 1 and
    ($s2 at pe.entry_point) or (any of (
   $s2, $h1) in (0..512))
  ...
}
```

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