YaGo

*Parsing **Ya**ra rules like a **Go**pher.*

¿Quién soy?

Jaume Martín

Ingeniero Informático

Master en seguridad de la información

Miembro del proyecto YaraRules

Enamorado de ~~Python~~ Go

@Xumeiquer

xumeiquer@protonmail.com

¿Qué es YARA?

Regla Yara

```
rule php_anuna {
    meta:
        author = "Vlad https://github.com/vlad-s"
        date = "2016/07/18"
        description = "Catches a PHP Trojan"
    strings:
        $a = /<\?php \$[a-z]+ = '/
        $b = /\$[a-z]+=explode\(chr\(\([0-9]+[-+][0-9]+\)\)/
        $c = /\$[a-z]+=\([0-9]+[-+][0-9]+\)/
        $d = /if \(!function_exists\('[a-z]+'\)\)/
    condition:
        all of them
}
```

Regla Yara en JSON

```
{
        "private": false,
        "public": false,
        "rule": {
                "name": "php_anuna",
                "meta": {
                        "author": "Vlad https://github.com/vlad-s",
                        "date": "2016/07/18",
                        "description": "Catches a PHP Trojan"
                },
                "strings": {
                        "$a": "/<\?php \$[a-z]+ = '/"
                        "$b": "/\$[a-z]+=explode\(chr\(\([0-9]+[-+][0-9]+\)\)/"
                        "$c": "/\$[a-z]+=\([0-9]+[-+][0-9]+\)/"
                        "$d": "/if \(!function_exists\('[a-z]+'\)\)/"
                },
                "condition": "all of them"
        }
}
```

¿Por qué?

El proyecto YaraRules* tiene alrededor de 1326 reglas Yara.

- Complicado de gestionar y administrar
- Complicado controlar la calidad
- Difícil de catalogar (basado en carpetas)
- Difícil de buscar una regla según parámetros

*https://github.com/Yara-Rules/rules

¿Por qué?

Con las reglas en formato JSON es posible:

- Almacenar en bases de datos NoSQL
- Gestionar y administrar de forma cómoda
- Aplicar acciones sobre grupos de reglas
  - Desde un frontal web
- Generar paquetes de reglas según necesidades

¿Cómo se programa un parser y que implica?

Fases de un compilador

Análisis léxico

Fase encargada de leer el código fuente y **separarlo en *tokens*** para poder ser leído por el análisis sintáctico.

Análisis sintáctico

Fase que **evalúa los *tokens*** de código con el fin de que este cumpla con los requerimientos definidos por el lenguaje.

Análisis semántico

Esta fase comprueba que el código fuente cumpla con la semántica del lenguaje, es decir que el código este **correctamente escrito.**

Generación de código

Optimización de código

Generación de código intermedio

Fases de un compilador

Análisis léxico

Fase encargada de leer el código fuente y **separarlo en *tokens*** para poder ser leído por el análisis sintáctico.

Análisis sintáctico

Fase que **evalúa los *tokens*** de código con el fin de que este cumpla con los requerimientos definidos por el lenguaje.

Análisis semántico

Esta fase comprueba que el código fuente cumpla con la semántica del lenguaje, es decir que el código este **correctamente escrito.**

Generación de código

Optimización de código

Generación de código intermedio

Análisis léxico

- Identificar el alfabeto del leguaje
- Identificar las palabra clave (*keywords*)
- Identificar caracteres nulos o blancos
- Programar un *tokenizador*

Análisis léxico

Identificar el alfabeto del leguaje

␣ ! “ # $ % & ’ ( ) * + , - . / 0 1 2
3 4 5 6 7 8 9 : ; < = > ? @ A B C D E
F G H I J K L M N O P Q R S T U V W X
Y Z [\ ] ^ _ ‘ a b c d e f g h i j k
l m n o p q r s t u v w x y z { | } ~

Análisis léxico

Identificar las palabra clave (*keywords*)

all	and	any	ascii	at	condition	contains
entrypoint	false	filesize	fullword	for	global	in
import	include	int8	int16	int32	int8be	int16be
int32be	matches	meta	nocase	not	or	of
private	rule	strings	them	true	uint8	uint16
uint32	uint8be	uint16be	uint32be	wide		

Análisis léxico

Identificar caracteres nulos o blancos

// Comentario

/* Comentario
Multiline */

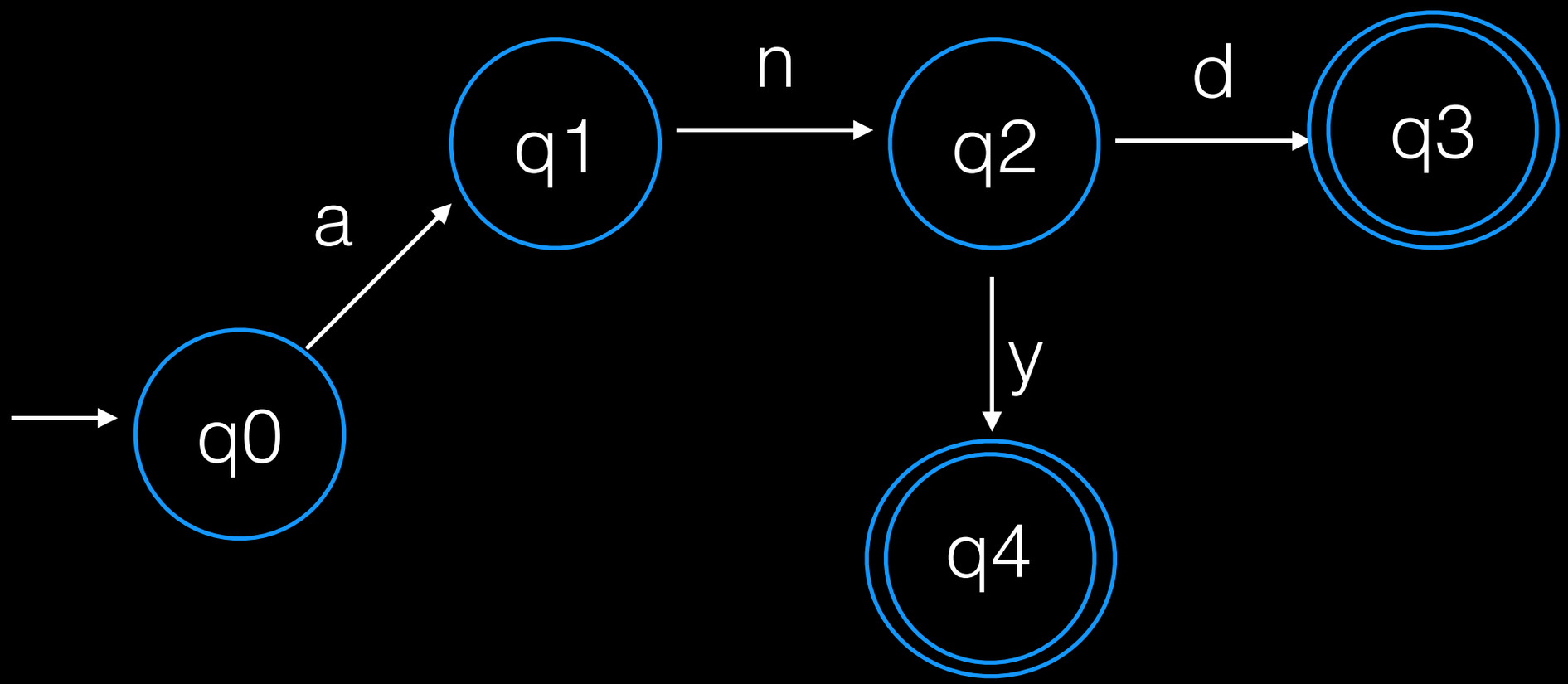
Espacios

Tabuladores
(\t)

Saltos de línea
(\n)

Análisis léxico

Programar un *tokenizador*



Análisis léxico

Programar un *tokenizador*

```
Algoritmo siguiente;                    // Con control de flujo
 q := q0;                               // Estado actual
 l := λ;                                // Lexema o token
 uf:= indefinido;                       // Último estado final
 ul:= λ;                                // Lexema leído hasta el último final
 mientras 0 = 1 – 1 hacer
        c := siguiente_carácter;
        l := l · c;
        opción q
            q0:                         // código del estado q0
            q1:                         // código del estado q1
             …
            qn:                         // código del estado qn
        fin opción
    fin mientras
fin siguiente
```

Análisis léxico

Programar un *tokenizador I*

```
func lexText(l *Lexer) stateFn {
   r := l.next()
    for !isEOF(r) {
        switch {
        case isEOF(r):
            l.emit(ItemEOF)
            return nil
        case isBlank(r):
            l.acceptRun(blankChars)
            l.ignore()
        case isSlash(r):
            return scanCommentOrRegex
        case isDolar(r):
            return scanVariable
        case isQuote(r):
            l.ignore()
            return scanQuote
        case isEqual(r):
            return scanEqual
        case isAlphaNumeric(r):
            return scanKeyword
```

```
        case isColon(r):
            return scanColon
        case isLeftCurly(r):
            return scanLeftCurly
        case isRigthCurly(r):
            return scanRightCurly
        case isLeftSqrt(r):
            return scanLeftSqrt
        case isRigthSqrt(r):
            return scanRightSqrt
        case isLeftBra(r):
            return scanLeftBra
        case isRigthBra(r):
            return scanRightBra
        case isWildCard(r):
            return scanWildCard
        case isDash(r):
            return scanDash
        case isPipe(r):
            return scanPipe
        case isHash(r):
            return scanHash
```

```
        case isDot(r):
            return scanDot
        case isStar(r):
            return scanStar
        case isCaret(r):
            return scanCaret
        case isPlusOrMinus(r):
            return scanPlusOrMinus
        case isComma(r):
            return scanComma
        case isAt(r):
            return scanAt
        case isGratherThan(r):
            return scanGratherThan
        case isLessThan(r):
            return scanLessThan
        }

        r = l.next()
    }
    l.emit(ItemEOF)
    return nil
}
```

Análisis léxico

Programar un *tokenizador II*

```
func scanKeyword(l *Lexer) stateFn {
    r := l.next()
    length := 1
    for !isBlank(r) && isAlphaNumeric(r) && length <= maxKeywordLength {
        length++
        if keyword[l.scanned()] > ItemKeyword {
            l.emit(keyword[l.scanned()])
            return lexText
        }
        r = l.next()
    }
    if isBlank(r) || !isAlphaNumeric(r) {
        l.backup()
    }
    for !isBlank(l.peek()) && isAlphaNumeric(r) {
        l.next()
    }
    if _, err := strconv.Atoi(l.scanned()); err == nil {
        l.emit(ItemNumber)
        return lexText
    }
    l.emit(ItemIdentifier)
    return lexText
}
```

Análisis léxico

Programar un *tokenizador III*

```
func scanQuote(l *Lexer) stateFn {
	r := l.next() // We've aready got the "
	for !isQuote(r) {
		if r == '\\' {
			if l.peek() == '"' {
				r = l.next() // Read the "
				r = l.next() // Prepare r for the next iteration
			} else if l.peek() == '\\' {
				r = l.next() // Read the \
				r = l.next() // Prepare r for the next iteration
			} else if l.peek() == 't' {
				r = l.next() // Read the t
				r = l.next() // Prepare r for the next iteration
			} else if l.peek() == 'n' {
				r = l.next() // Read the n
				r = l.next() // Prepare r for the next iteration
			} else if l.peek() == 'x' {
				r = l.next() // Read the x
				// It needs a two hex characters
				if isHexChar(l.peek()) {
					r = l.next() // Read first hex char
					if isHexChar(l.peek()) {
						r = l.next() // Read first hex char
						r = l.next() // Prepare r for the next iteration
					} else {
						return l.errorf("Line %d: illegal escape sequence", l.Line)
					}
				} else {
					return l.errorf("Line %d: illegal escape sequence", l.Line)
				}
			} else {
				return l.errorf("Line %d: illegal escape sequence", l.Line)
			}
		}
		r = l.next()
	}
	l.backup() // Remove the last "
	l.emit(ItemString)
	l.next()   // Consume the "
	l.ignore() // Just ignore it
	return lexText
}
```

Análisis sintáctico

- Mantener la tabla de reglas
- Mantener la tabla de símbolos
- Mantener la tabla de imports
- ¿Cumple con la gramática?
- Reportar errores

Análisis sintáctico

¿Cuál es el *input* para el análisis sintáctico?

```
rule php_anuna {
    meta:
        author = "Vlad https://github.com/vlad-s"
        date = "2016/07/18"
        description = "Catches a PHP Trojan"
    strings:
        $a = /<\?php \$[a-z]+ = '/
        $b = /\$[a-z]+=explode\(chr\(\([0-9]+[-+][0-9]+\)\)/
        $c = /\$[a-z]+=\([0-9]+[-+][0-9]+\)/
        $d = /if \(!function_exists\('[a-z]+'\)\)/
    condition:
        all of them
}
```

Análisis sintáctico

Mantenimiento de tablas y estructuras

```
type Parser struct {
   Name      string
   Lex       *lexer.Lexer
   LastItem lexer.Item
   Imports   []string
   Rules     []RuleDef
   log       *logrus.Logger
}
```

```
type RuleDef struct {
   Name       string
   Global     bool
   Private    bool
   Tags       []string
   Meta       map[string]string
   Strings    []StringDef
   Condition string
}
```

```
type StringDef struct {
   Name      string
   Value     string
   Modifiers []string
}
```

Análisis sintáctico

¿Cumple con la gramática?

```
func (p *Parser) parse() {
    private := false
    global := false
    for item := range p.Lex.Items {
        switch {
        case checkItemType(item, lexer.ItemImport):
            p.processImport()
            break
        case checkItemType(item, lexer.ItemPrivate):
            private = true
            break
        case checkItemType(item, lexer.ItemGlobal):
            global = true
            break
        case checkItemType(item, lexer.ItemRule):
            p.processRule(global, private)
            private = false
            global = false
            break
        }
    }
}
```

Gramatica para reglas Yara

https://github.com/VirusTotal/yara/blob/master/libyara/grammar.y

Uso de YaGo

```
./build/yago
  -fileName string
        Yara file you want to parse
  -format string
        Format you want the Yara rule to be parsed (default "json")
```

```
./build/yago -fileName test/rule.yar
```

```
./build/yago -fileName test/rule.yar | \
    mongoimport --db yararules --collection rules --type json
```

Uso de YaGo

```
import "pe"

// Comentario

private rule dummy : tag1 tag2 {
    meta:
        autor = "Jaume Martin"
        score = 40
    strings:
        $ping = "pong\"asd\xaang\n mam" nocase wide
        $re1 = /md5: \q[0-9a-zA-Z]{32}/
        $my_hex_string = { E2 FB [2-3] 44 ( ?? | 00 ) }
    condition:
        $ping or $my_hex_string or $re1
}
```


```
{"imports":["pe"],"rules":[{"name":"dummy","global":false,"private":true,"tags":
["tag1","tag2"],"meta":{"autor":"Jaume Martin","score":"40"},"strings":
[{"name":"$ping","value":"pong\\\"asd\\xaang\\n mam","modifers":
["nocase","wide"]},{"name":"$re1","value":"/md5: \\q[0-9a-zA-Z]
{32}/","modifers":null},{"name":"$my_hex_string","value":"{E2FB[2-3]44(??|
00)}","modifers":null}],"condition":"$ping or $my_hex_string or $re1"}]}
```

¿Vemos como funciona?

Mejoras

- Control de strings en condition
- Control de modulos importados
- Parerar un directorio de reglas Yara
- Pasear un index de reglas Yara
- Poder generar un fichero de reglas
- Y más …

¿Preguntas?

Gracias