Secure Programming

Mehmet Tahir SANDIKKAYA

Spring 2022

Istanbul Technical University
Computer Engineering Department

Syllabus

Week	Date	Rct	Covers	Subject	Announcement	Submit
1	22nd Feb		7	Fundamental concepts of security	TP-A	
2	01st Mar		1, 7	Compilation and Execution		
3	08th Mar		1, 7	Stack overflow and its mitigation	Asg1-A	
4	15th Mar		1, 7	Dynamic memory management		
5	22nd Mar	R	3, 7	Canonicalization attacks and mitigation	Asg2-A	Asg1-S
6	29th Mar		2, 7	Injection attacks	Asg1-G	
7	05th Apr	R	2, 7	Injection mitigation	Asg3-A	Asg2-S
8	12th Apr		4	Reverse engineering and obfuscation	Asg2-G	
9	19th Apr		5	Fundamental cryptography		Asg3-S
A	26th Apr	R	5	Principles of computer communication	Asg3-G	MT
-	03rd May			Spring break		
B	10th May		5	XSS & CSRF attacks and mitigation	Asg4-A	
C	17th May	R	5	Race conditions	MT-G	
D	24th May		6	Permission and authorization mechanisms in contemporary languages		Asg4-S
E	31st May		7	Test and static analysis tools	Asg4-G	TP-S

Recitation

Canonicalization and Command Injection

Recitation by Ayşe SAYIN covering canonicalization and command injections.

Matching Patterns

How could I match patterns?

Study regular expressions but do not rely on them! [Stackoverflow, 2014]

You cannot match a context-free language[a] with a regular expression!

[a] or context-sensitive or unrestricted

RegEx basics

- ✔ A literal matches a literal: `cat` matches `cat` but not `Cat`
- ✔ There are 12 *metacharacters*: `\^$.|?*+()[{`
- ✔ Escaping could be mind-blowing [xkcd, 1996]:
  e.g. Following SQL query in Java

```
SELECT * FROM table WHERE addr='http:\\'
```

could be matched by

```
SELECT * FROM table WHERE addr='http:\\\\\\\\\\\\\\\\'
```

How could I match patterns?

✔ Akin to dialects of a natural language, consider the RegEx flavou?r you use

✔ `\cM`, `\r`, `\x0D`, `\15`, and `\u000D` match a carriage return

✔ RegEx is eager and greedy to find the longest sequence:

```
catdog|cat|dog
cat dog catdog
cat|dog|catdog
```

✔ Characters (including metacharacters) apart from `]\^-` are regular characters inside character classes: `[.]` matches a .

✔ Character classes could be subtracted and intersected

Regular Expressions

RegEx	Match
`^`	Matches the start of the string
`$`	Matches the end of the string
`*`	Matches the preceding pattern zero or more times. Same as `{0,}`
`+`	Matches the preceding pattern one or more times. Same as `{1,}`
`?`	Matches the preceding pattern zero or one time. Same as `{0,1}`
`{n}`	Matches the preceding pattern exactly n times
`{n,}`	Matches the preceding pattern n or more times
`{,m}`	Matches the preceding pattern no more than m times
`{n,m}`	Matches the preceding pattern between n and m times
`.`	Most of the time matches any single character, except a newline

Regular Expressions

RegEx	Match
`cat|dog`	Matches `cat` or `dog`
`a(a|b)b`	Matches `aab` or `abb`
`(aa|bb)`	Matches `aa` or `bb`
`(?i)abc`	Switches off case sensitivity. Same as `(a|A)(b|B)(c|C)`
`ab\u{63}`	Matches abc (Unicode RegEx [Davis and Heninger, 2016])
`^Abc`	Matches any string starting with `Abc`
`\.`	Matches a .
`a-z`	Matches `a-z`
`.*`	Matches a line most of the time (Expands)
`.*?`	Prefers to match a zero length string (Lazy)
`.+?`	Matches any single character (Lazy)
`.??`	Prefers not to match anything. A zero-length match (Lazy)
`/abc/`	Matches `abc`. Many dialects prefer / to mark a RegEx

Regular Expressions

RegEx	Match
[.]	A character class that matches only ..
[abc]	A character class that matches one of the enclosed characters.
[^abc]	A character class that matches none of the enclosed characters.
[a-z]	A character class that matches the range from a to z.
[A-Z]	A character class that matches uppercase letters.
[^0-9]	A character class that matches anything but the digits.
[?.]	A character class that matches either . or ?.
[a^b]	A character class that matches one of the enclosed characters.
[-ab]	A character class that matches one of the enclosed characters.
[c|cat]	May match c. Never matches cat.

Regular Expressions

RegEx	Match
\r	Matches a carriage return
\n	Matches a newline
\t	Matches a tab
\b	Matches the position between a word and a space
\B	Matches a non-word boundary
\d	Matches a digit, same as [0-9]
\D	Matches a non-digit, same as [^0-9]
\s	Matches a white-space character; same as [\f\n\r\t\v\h]
\S	Matches a non-white-space character; same as [^ \f\n\r\t\v\h]
\w	Matches a word character; same as [a-zA-Z0-9_]
\W	Matches a non-word character; same as [^a-zA-Z0-9_]
\w{3}	Matches three word characters; same as \w\w\w

Regular Expressions

RegEx	Match
(a*)x\1	Matches equal number of a's around x
([cat]+)\1	Matches catcat, or cc, or caca
([cat])+\1	Matches catt, or catcat, or aa
aca(?=b)	Matches aca in acab
(?<!a)b	Matches b that is not preceded by a
(?!foo).{3}	Matches any three character word that is not foo
(?!.*)	Does not matches anything

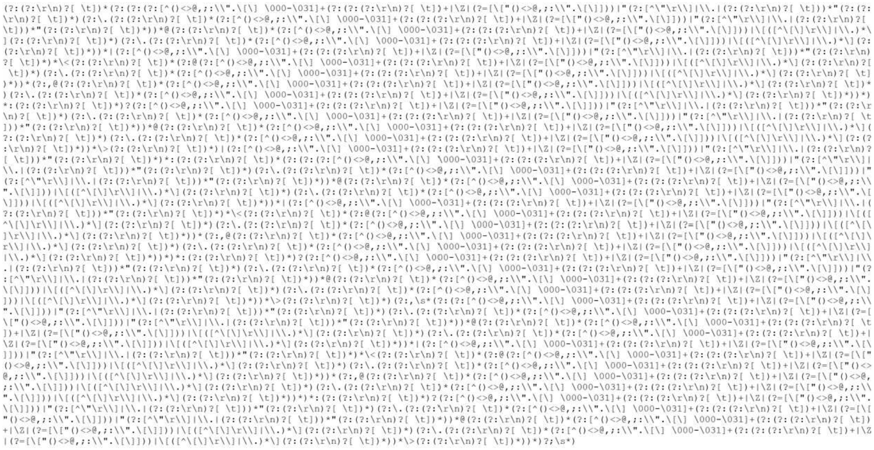
An example: e-mail

See [Schaumann, 2022]

- ✔ @1st.relay,@2nd.relay:sandikkaya@itu.edu.tr
- ✔ !google.com!google.com.tr!metu.edu.tr!sandikkaya@itu.edu.tr
- ✔ !sandikkaya%itu.edu.tr@relay.through
- ✔ '*+-/=?^_`{|}~#$@itu.edu.tr
- ✔ . and + are not special: sandik+kaya@itu.edu.tr sandik.kaya@itu.edu.tr
- ✔ " "@itu.edu.tr, "<>"@itu.edu.tr, "even a newline\^M <--"@itu.edu.tr
- ✔ sandikkaya@itu.edu.tr and SANDIKKAYA@itu.edu.tr are different
- ✔ Unicode is valid áçãb̈@itu.edu.tr
- ✔ sandikkaya@[160.75.1.1]

An example: e-mail

- ✔ Incomplete e-mail regex is 4724 characters long
- ✔ Optimized e-mail regex is 6598 character long
- ✔ They are both known to be incomplete — "what could you do (sometimes...)"@domain.com



```
(?:(?:\r\n)?[ \t])*(?:(?:(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:
\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[
\t]))*"(?:(?:\r\n)?[ \t])*))*@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\
](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:
(?:\r\n)?[ \t])*))*|(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)
?[ \t])*)*\<(?:(?:\r\n)?[ \t])*(?:@(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[
 \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t]
)*))*(?:,@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*
)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*)
*:(?:(?:\r\n)?[ \t])*)?(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r
\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t
]))*"(?:(?:\r\n)?[ \t])*))*@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](
?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?
:\r\n)?[ \t])*))*\>(?:(?:\r\n)?[ \t])*)|(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?
[ \t]))*"(?:(?:\r\n)?[ \t])*)*:(?:(?:\r\n)?[ \t])*(?:(?:(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|
\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"
(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*))*@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\
".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[
\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*|(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(
?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)*\<(?:(?:\r\n)?[ \t])*(?:@(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([
^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\
]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*(?:,@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\
r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]
|\\.)*\](?:(?:\r\n)?[ \t])*))*)*:(?:(?:\r\n)?[ \t])*)?(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\
.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?
:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*))*@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".
\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]
]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*\>(?:(?:\r\n)?[ \t])*)(?:,\s*(?:(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\
".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[
\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*))*@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t
])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|
\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*|(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\
]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)*\<(?:(?:\r\n)?[ \t])*(?:@(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["
()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>
@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*(?:,@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,
;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\
".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*)*:(?:(?:\r\n)?[ \t])*)?(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".
\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\[
"()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*))*@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])
+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z
|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*\>(?:(?:\r\n)?[ \t])*))*)?;\s*)
```

Remarks on RegExes

- ✔ Consider ^x|y$ and ^(x|y)$
- ✔ Another kind of attack: reDoS
- ✔ Consider ^([a-zA-Z]+)*$
- ✔ Now, try to match aaaaaaaaaaaaaaaa1
- ✔ How many unchecked branches you have before you decide this pattern does not match?[a]
- ✔ Thus, first convert all NFA's to DFA's before writing your RegExes (so, no backtracking)

[a] You may want to check Turing's undecidability theorem [Turing, 1936].

Do these till next week...

- ✔ Check input validation on [Wheeler, 2015]
- ✔ Get used to regular expressions [Kleene, 1951], [Goyvaerts, 2020] is a nice tutorial to learn.

Bibliography

References

[Davis and Heninger, 2016] Davis, M. and Heninger, A. (2016). Unicode Regular Expressions. `http://unicode.org/reports/tr18/`.

[Goyvaerts, 2020] Goyvaerts, J. (2020). Regular expression tutorial. `https://www.regular-expressions.info/tutorial.html`.

[Kleene, 1951] Kleene, S. C. (1951). Representation of events in nerve nets and finite automata. Technical report, RAND PROJECT, AIR FORCE, SANTA MONICA, CA.

[Schaumann, 2022] Schaumann, J. (2022). *Look, your email validation logic is very, very likely wrong.* Accessed on 18.02.2022, `https://mobile.twitter.com/jschauma/status/1378172844169961477`.

[Stackoverflow, 2014] Stackoverflow (2014). RegEx match open tags except XHTML self-contained tags. `https://stackoverflow.com/questions/1732348/regex-match-open-tags-except-xhtml-self-contained-tags/`.

[Turing, 1936] Turing, A. M. (1936). On computable numbers, with an application to the entscheidungsproblem. *J. of Math*, 58(345–363):5.

[Wheeler, 2015] Wheeler, D. A. (2015). *Secure programming for Linux and Unix HOWTO.* `http://www.dwheeler.com/secure-programs`.

[xkcd, 1996] xkcd (1996). Backslashes. `https://xkcd.com/1638/`.