/Code that gets you pwn(sl'd)/

Louis Nyffenegger
Louis@pentesterlab.com
@snyff/@PentesterLab

My job is to find, collect and study bugs to teach people how they can find, fix and exploit bugs.

```
func uploadFile(w http.ResponseWriter, r *http.Request) {
    file, handler, err := r.FormFile("file")
    if err != nil {
        fmt.Println("Error Retrieving the File")
        return
    }
    defer file.Close()
    tempFile, err := ioutil.TempFile("/tmp", handler.Filename)
    if err != nil {
        fmt.Println(err)
        return
    }
    fileBytes, err := ioutil.ReadAll(file)
    if err != nil {
        fmt.Println(err)
        return
    }
    tempFile.Write(fileBytes)
    defer tempFile.Close()
    fmt.Fprintf(w, "Successfully Uploaded File\n")
}
```

func TempFile

```
func TempFile(dir, pattern string) (f *os.File, err error)
```

TempFile creates a new temporary file in the directory dir, opens the file for reading and writing, and returns the resulting *os.File. The filename is generated by taking pattern and adding a random string to the end. If pattern includes a "*", the random string replaces the last "*". If dir is the empty string, TempFile uses the default directory for temporary files (see os.TempDir). Multiple programs calling TempFile simultaneously will not choose the same file. The caller can use f.Name() to find the pathname of the file. It is the caller's responsibility to remove the file when no longer needed.

```
ioutil.TempFile("/tmp","../../root/foo.*.suffix")
```

```
ioutil.TempFile("/tmp","../../root/foo.*.suffix")
```

```
=> /root/foo.917768646.suffix
```

```
ioutil.TempFile("/tmp","../../somewhere/else.*.suffix")
```

```
=> /root/foo.917768646.suffix
```

Reported and fixed in recent versions of Golang:

https://github.com/golang/go/issues/33920

https://go-review.googlesource.com/c/go/+/212597/

Python 3.8.1

```
>>> tempfile.NamedTemporaryFile(dir="/tmp",prefix="../../root/").name
'/root/jjq3h7bk'
```

Ruby 2.7

```
Tempfile.new('foo/../../root/',"/tmp/")
=> #<Tempfile:/tmp/foo....root20200323-7-1mzkbxy>
```

```
import urllib
import os
from flask import Flask,redirect,request
from secrets import token_hex
app = Flask(__name__)

@app.route('/fetch')
def fetch():
    url = request.args.get('url', '')
    if url.startswith("https://pentesterlab.com"):
      response = urllib.request.urlopen(url)
      html = response.read()
      return html
    return ""
```

startswith is bad....

https://trusted => https://trusted.pentesterlab.com

https://trusted => https://trusted@pentesterlab.com

https://trusted/jwks/ => https://trusted/jwks/../file_uploaded

https://trusted/jwks/ => https://trusted/jwks/../open_redirect

https://trusted/jwks/ => https://trusted/jwks/../header_injection

StartsWith is bad - C# edition

```
string s = "";
if (url.StartsWith("https://“)){
  using(WebClient client = new WebClient()) {
    s = client.DownloadString(url);
  }
  …
}
```

StartsWith is bad - C# edition

```
string s = "";
if (url.StartsWith("https://“)){
  using(WebClient client = new WebClient()) {
    s = client.DownloadString(url);
  }
  …
}
```

https://../../../../../../../../etc/passwd

```
func handler(w http.ResponseWriter, r *http.Request) {
    filename := path.Clean(r.URL.Query()["filename"][0])
    fd, err := os.Open(filename)
    defer fd.Close()
    if err != nil {
      http.Error(w, "File not found.", 404)
      return
    }
    io.Copy(w, fd)
}

func main() {
    http.HandleFunc("/", handler)
    log.Fatal(http.ListenAndServe(":8080", nil))
}
```

Clean returns the shortest path name equivalent to path by purely lexical processing. It applies the following rules iteratively until no further processing can be done:

```
package main

import (
        "fmt"
        "path"
)

func main() {
        paths := []string{
                "a/c",
                "a//c",
                "a/c/.",
                "a/c/b/..",
                "/../a/c",
                "/../a/b/../././//c",
                "",
        }

        for _, p := range paths {
                fmt.Printf("Clean(%q) = %q\n", p, path.Clean(p))
        }

}
```

```go
package main

import (
        "fmt"
        "path"
)

func main() {
        paths := []string{
                "a/c",
                "a//c",
                "a/c/.",
                "a/c/b/..",
                "/../a/c",
                "/../a/b/../././/c",
                "",
        }

        for _, p := range paths {
                fmt.Printf("Clean(%q) = %q\n", p, path.Clean(p))
        }

}
```

```
Clean("a/c") = "a/c"
Clean("a//c") = "a/c"
Clean("a/c/.") = "a/c"
Clean("a/c/b/..") = "a/c"
Clean("/../a/c") = "/a/c"
Clean("/../a/b/../././/c") = "/a/c"
Clean("") = "."
```

```
path.Clean("a/c") = "a/c"
path.Clean("a//c") = "a/c"
path.Clean("a/c/.") = "a/c"
path.Clean("a/c/b/..") = "a/c"
path.Clean("/../a/c") = "/a/c"
path.Clean("/../a/b/../././//c") = "/a/c"
path.Clean("../../../a/b") = "../../../a/b"
```

```
<script>
  window.addEventListener("message", function(event){
    check = new RegExp(“.pentesterlab.com$”);
    if (check.test(event.origin)) {
        do_something_sensitive();
    } else {
    …
    }
  …
  }
</script>
```

Unescaped dot in regular expression

```
new RegExp(“.pentesterlab.com$”);
```

matches:

```
www.pentesterlab.com
wwwzpentesterlab.com
wwwzpentesterlabzcom
```

```
const express = require('express')
const app = express()

var allowedDomains = ['EXAMPLE.ORG', 'GMAIL.COM',
                      'GOOGLE.COM'];

function fetchContent(domain) {
 …
}

app.get('/fetch', (req, res) => {
    if (allowedDomains.includes(req.query.domain.toUpperCase())) {
      return res.send(fetchContent(req.query.domain))
    } else {
      res.status(403)
      return res.send("ACCESS DENIED")
    }
});
```

```
"ı".toUpperCase() == "I"

"gmaıl.com".toUpperCase() == "GMAIL.COM"
```

https://eng.getwisdom.io/hacking-github-with-unicode-dotless-i/

Uppercase

Char	Code Point	Output Char
ß	0x00DF	SS
ı	0x0131	I
ſ	0x017F	S
ff	0xFB00	FF
fi	0xFB01	FI
fl	0xFB02	FL
ffi	0xFB03	FFI
ffl	0xFB04	FFL
ſt	0xFB05	ST
st	0xFB06	ST

Source: https://eng.getwisdom.io/hacking-github-with-unicode-dotless-i/

This is crazy but this is the expected behaviour!

Figure 5-14. Uppercase Mapping for Turkish I

Normal			Turkish		
i 0069	↔	I 0049	i 0069	↔	İ 0130
ı 0131	→	I 0049	ı 0131	↔	I 0049
i + ◌̇ 0069 0307	↔	I + ◌̇ 0049 0307	i + ◌̇ 0069 0307	↔	İ + ◌̇ 0130 0307

Figure 5-15 shows the lowercase mapping for Turkish *i*.

Figure 5-15. Lowercase Mapping for Turkish I

Normal			Turkish		
I 0049	↔	i 0069	I 0049	↔	ı 0131
İ 0130	→	i + ◌̇ 0069 0307	İ 0130	↔	i 0069
I + ◌̇ 0049 0307	↔	i + ◌̇ 0069 0307	I + ◌̇ 0049 0307	→	i 0069

For those wanting to play @~

SCAN ME

http://capturethefl.ag:1234/

SCAN ME

http://capturethefl.ag:5678/

SO WE TALKED ABOUT REGEXP, WE TALKED ABOUT UNICODE... WHAT IF WE MIX THE TWO?

Python 3.8.1

```
>>> m = re.compile('i')
>>> print(mi.match("ı"))
None
```

Python 3.8.1

```
>>> m = re.compile('i', re.IGNORECASE)
>>> print(m.match("ı"))
<re.Match object; span=(0, 1), match='ı'>
```

Python 3.8.1

```
>>> m = re.compile('i', re.IGNORECASE)
>>> print(m.match("ı"))
<re.Match object; span=(0, 1), match='ı'>
```

```
>>> m = re.compile('S', re.IGNORECASE)
>>> print(m.match("ſ"))
<re.Match object; span=(0, 1), match='ı'>
```

```
>>> m = re.compile('K', re.IGNORECASE)
>>> print(m.match("K"))
<re.Match object; span=(0, 1), match='ı'>
```

Ruby 2.7.0

```
puts ("ı" =~ /i/i).inspect
=> nil


puts ("ı" =~ /I/i).inspect
=> nil


puts ("ı".upcase)
=> I
```

Ruby 2.7.0

```
puts ("ı" =~ /i/i).inspect
=> nil
```

```
puts ("K" =~ /K/i).inspect
=> 0
```

Kelvin Sign

```
puts ("K" =~ /k/i).inspect
=> 0
```

Kelvin Sign

Ruby 2.7.0

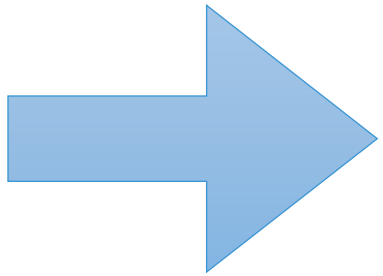
```
puts ("ı" =~ /i/i).inspect
=> nil
```

```
puts ("ſ" =~ /S/i).inspect
=> 0
```

```
puts ("ſ" =~ /s/i).inspect
=> 0
```

Golang 1.13.8

```
match, _ := regexp.MatchString("(?i)i", "ı")
fmt.Println(match)
match2, _ := regexp.MatchString("(?i)k", "K")
fmt.Println(match2)
match3, _ := regexp.MatchString("(?i)s", "ſ")
fmt.Println(match3)
```



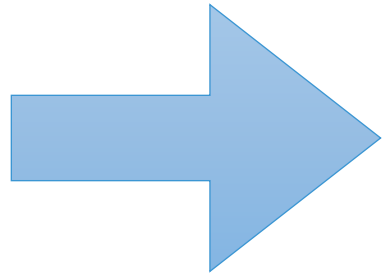
```
false

true

true
```

C#

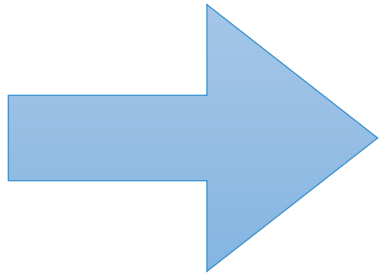
```
Regex.IsMatch("ı", "i", RegexOptions.IgnoreCase);
Regex.IsMatch("K", "k", RegexOptions.IgnoreCase);
Regex.IsMatch("ſ", "s", RegexOptions.IgnoreCase);
```



```
False
True
False
```

C#

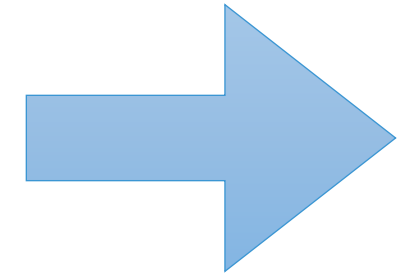
```
String s = "test.ınc";
s.EndsWith("inc");
s.EndsWith("inc", StringComparison.InvariantCultureIgnoreCase);
s.EndsWith("inc", StringComparison.OrdinalIgnoreCase);
s.EndsWith("inc", StringComparison.CurrentCultureIgnoreCase);
```



```
False
False
True
False
```

openjdk 13.0.2

```
"ı".equalsIgnoreCase("i");
"i".equalsIgnoreCase("ı");
"K".equalsIgnoreCase("k");
"k".equalsIgnoreCase("K");
"ſ".equalsIgnoreCase("s");
"s".equalsIgnoreCase("ſ");
```



```
True
True
True
True
True
True
```

openjdk 13.0.2

```
Pattern p = Pattern.compile("i", Pattern.CASE_INSENSITIVE);
System.out.println(p.matcher("ı").find());

Pattern ps = Pattern.compile("s", Pattern.CASE_INSENSITIVE);
System.out.println(ps.matcher("ſ").find());

Pattern pk = Pattern.compile("k", Pattern.CASE_INSENSITIVE);
System.out.println(pk.matcher("K").find());
```

```
False

False

False
```

Examples

If domain matches

- /something-with-a-i\.TLD$/i
- /something-with-a-s\.TLD$/i
- (/something-with-a-k\.TLD$/i)

If email address ends with:

- /something-with-a-i\.TLD$/i
- /something-with-a-s\.TLD$/i
- (/something-with-a-k\.TLD$/l)

Conclusion

Computers are hard?

Devil is in the details?

Don't make assumptions?

Thanks for your time!
Any questions?

@snyff
@PentesterLab