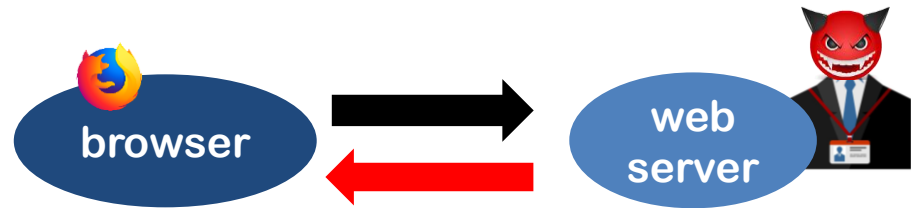**Web Security**

Server-side security risks

(esp. injection attacks)

Attacker models

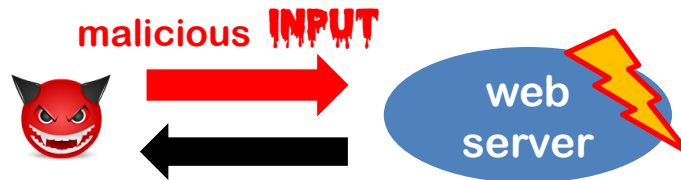1. **Man-in-the-Middle attacker**

2. **Spoofed/fake website**

3. **Attacks on web servers (this week)**

malicious INPUT

4. **Attacks on browsers & users**

    (next weeks)

Don't let the cute 😈 emoji fool you!

There can be really nasty people & organisations hiding behind it

One of the makes of the Mirai botnet

Omri Lacie & Shalev Hulio
of NSO Group

WANTED BY THE FBI

NOS Nieuws · Donderdag 1 juli 2021, 20:09

Celstraf voor man die software voor betaalfraude maakte en verkocht

Een man van 20 is veroordeeld tot twee jaar gevangenisstraf voor het maken en verkopen van computerprogramma's om betaalfraude te plegen. Na het uitzitten van zijn straf moet hij van de rechtbank in Den Haag meewerken aan Hack_Right, een programma waarin jonge internetcriminelen worden begeleid om hun ict-talent op legale wijze in te zetten.

United States and United Kingdom Sanction Members of Russia–Based Trickbot Cybercrime Gang

February 9, 2023

https://en.wikipedia.org/wiki/List_of_computer_criminals
https://www.fbi.gov/wanted/cyber
https://krebsonsecurity.com

Security concerns with static web pages

Security worries for *static HTML*

Recall the first stage of the evolution of the web: static HTML

Security risk:

- Accidentally exposing parts of the file system on the internet

    `http://www.cs.ru.nl/~erikpoll/websec/exam/exam2024.pdf`

- Such files can even be indexed by search engines

Countermeasures

- The OS (Operating System) imposes access control on the web server

- `.htaccess` file can be used to configure which files are exposed to the internet by the web server.

- Access restrictions for automated web crawlers, as used by search engines, can be specified in `robots.txt` files,

    – but it is up to the client to respect these - or not…

Security concerns with dynamically created web pages

Recall: dynamically created web pages

Most web pages you see are dynamically created

execution to dynamically create a webpage

browser ⬌ web server

dynamically generated HTML
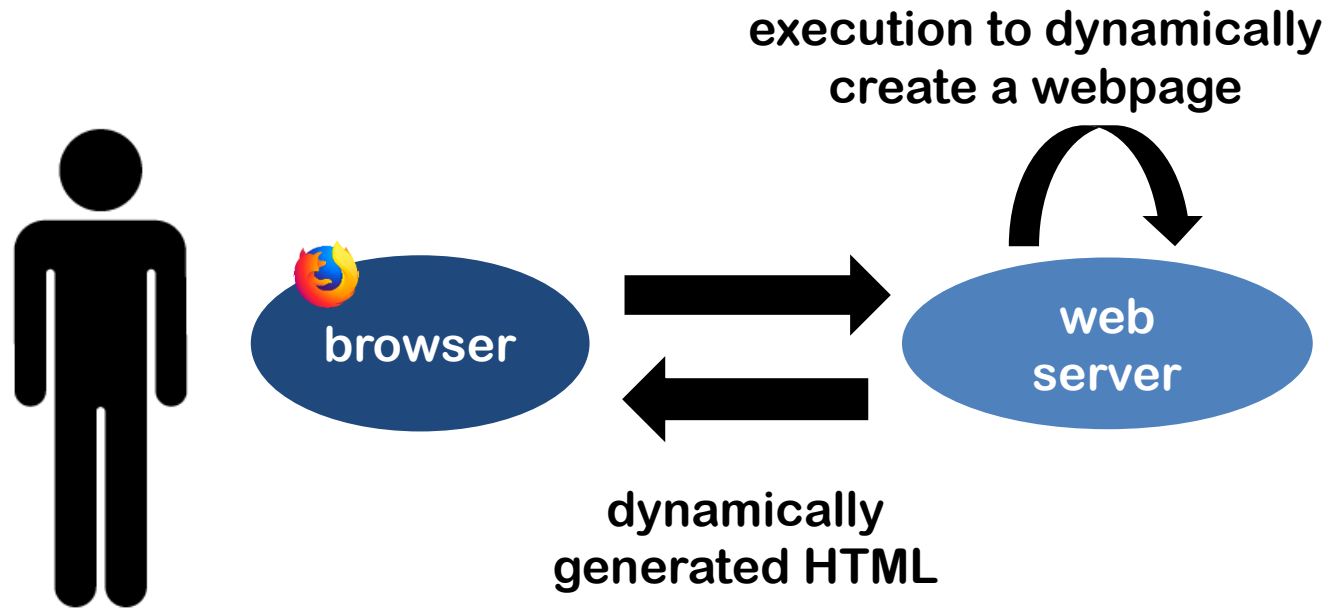
CGI (Common Gateway Interface)

Old-fashioned way to have dynamically generated web pages

Given an HTTP request to a cgi executable

`http://bla.com/cgi-bin/my_script?yr=2014&str=a%20name`

the web server executes the program `my_script`

passing parameters as input, and

returning the (HTML) output to client.

For the URL above, the web server would execute

`cgi-bin/my_script 2014 "a name"`

The executable `my_script` can be in *any* programming language

Example: CGI perl script

```perl
#!/usr/bin/perl
print "Content-type: text/html\n\n";


print <<HTML;
  <html>
  <head> <title>My first perl CGI script </title>
  </head>
  <body> <p>Hello World</p>
  </body>
  </html>
HTML
exit;
```

Languages & frameworks for the web

CGI is simple but very clumsy

Therefore people made:

- **dedicated programming languages for web applications**

  PHP, Ruby on Rails, Adobe ColdFusion, …

- **web frameworks offering a lot of standard software components**

  **Drupal (PHP), Spring (Java),
  React, Angular, AngularJS (JavaScript),
  ASP.NET (Microsoft CLR/.NET), …**

Example PHP script

```
<html> <title>A simple PHP script </title>
  <body>
    The number you choose was
        <?php echo $x = $_GET['number']; ?>
    This number squared plus 1 is
        <?php  $y = $x*$x; $y++; echo $y; ?>
    Btw, I know that your IP address is
        <?php echo $_SERVER['REMOTE_ADDR'];  ?>
      <script> alert('Hello World!'); </script>
  </body>
</html>
```

This looks just like an HTML page, with pieces of **PHP code** in it.

PHP code is executed *server-side* ; browser only sees the resulting HTML

JavaScript code in the HTML is executed *client-side*.

Security worries with dynamically created web pages

Command injection (in a CGI script)

A CGI bash script might contain

```
cat thefile | mail clientaddress
```

to email a file to a user-supplied email address.

How would you attack this?

```
erik@cs.ru.nl ; rm -fr /
```


What happens then ?

```
cat thefile | mail erik@cs.ru.nl ; rm -fr /
```

OS command injection

Any server-side code that uses client input to interact with the underlying OS might be used to inject commands to the OS.

This is possibly in any programming language.
Dangerous things to look out for
- **C/C++** `system(), execvp(), ShellExecute(), ..`
- **Java** `Runtime.exec(), ...`
- **Perl** `system, exec, open, `, /e, ...`
- **Python** `exec, eval, input, execfile, ...`

How would you prevent this or mitigate the potential impact?
1. input sanitisation: check for malicious inputs
   - easier said than done…
2. the server should run with minimal rights
   - eg. you don't want to run it as super-user/admin

How would you attack this?

Suppose a website contains a link

`http:/somesite.com/get-files.php?file=exam2023.pdf`

`exam2023.pdf` looks like a filename...

You can try any other filename, e.g. `exam2024.pdf`

Or even any other path name, e.g. `../../../etc/passwd`

Known as path traversal or directory traversal attack

Directory traversal aka path traversal

Consider PHP code below,

which uses PHP string concatenation operator .

```
$base_dir = "/usr/local/clientdata/";

echo file_get_contents($base_dir .

                       $_GET['username']);
            // concatenates base_dir and username
```

This can be attacked in the same way.

DoS by directory traversal

Directory traversals can also cause Denial-of-Service, if you access

- **a file or directory that does not exists**
  - **This may crash a web application, though it's unlikely**

- **device files, ie pseudo-files that provide interfaces to devices**
  - **`/var/spool/printer`**

    This printer queue cannot be opened for reading, only for writing. Opening it for reading may cause web application to hang.
  - **`/dev/urandom`**

    The random number generator that provides infinite stream of random numbers

Real life example

> **Page in Dutch, based on IP address or language settings of browser/OS**

https://instagram.com
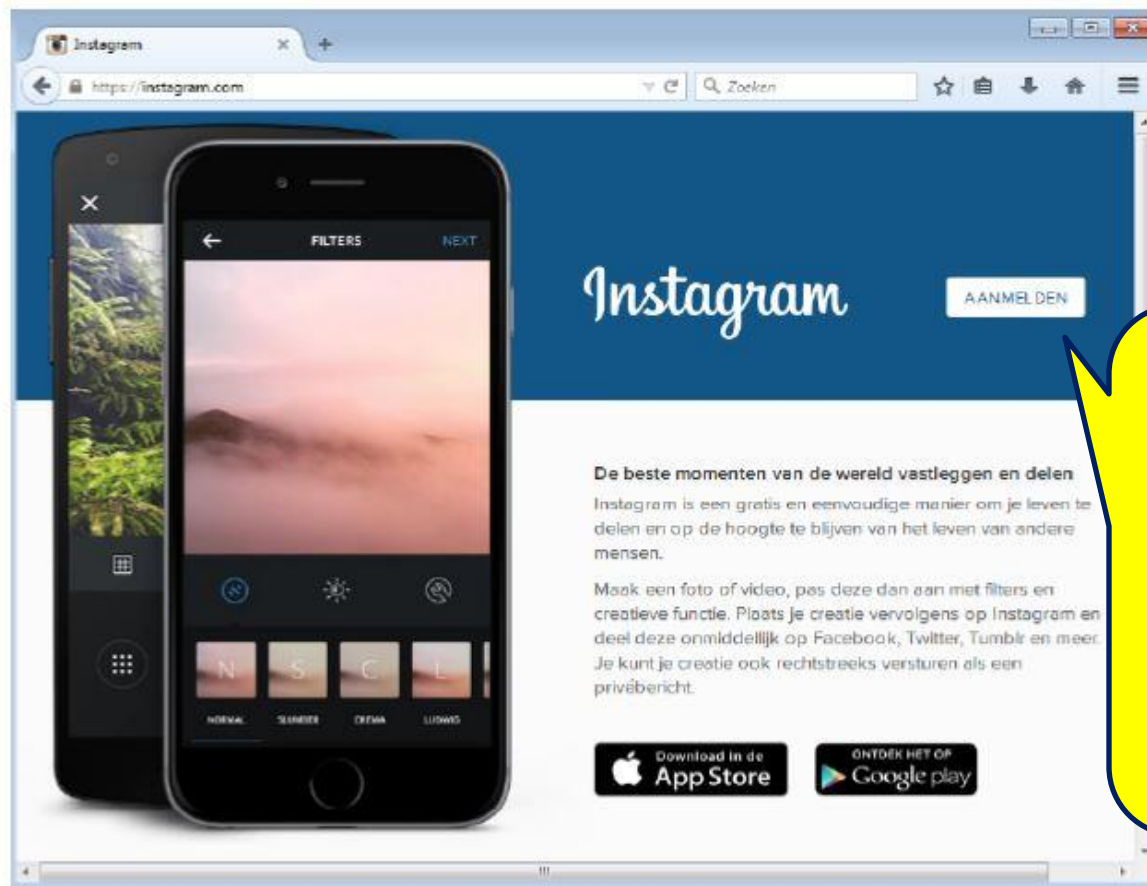
Thanks to Arne Swinnen. See his blog at http://www.arneswinnen.net.

https://instagram.com/?hl=en

No error message: .**/en** gives same result as **en**
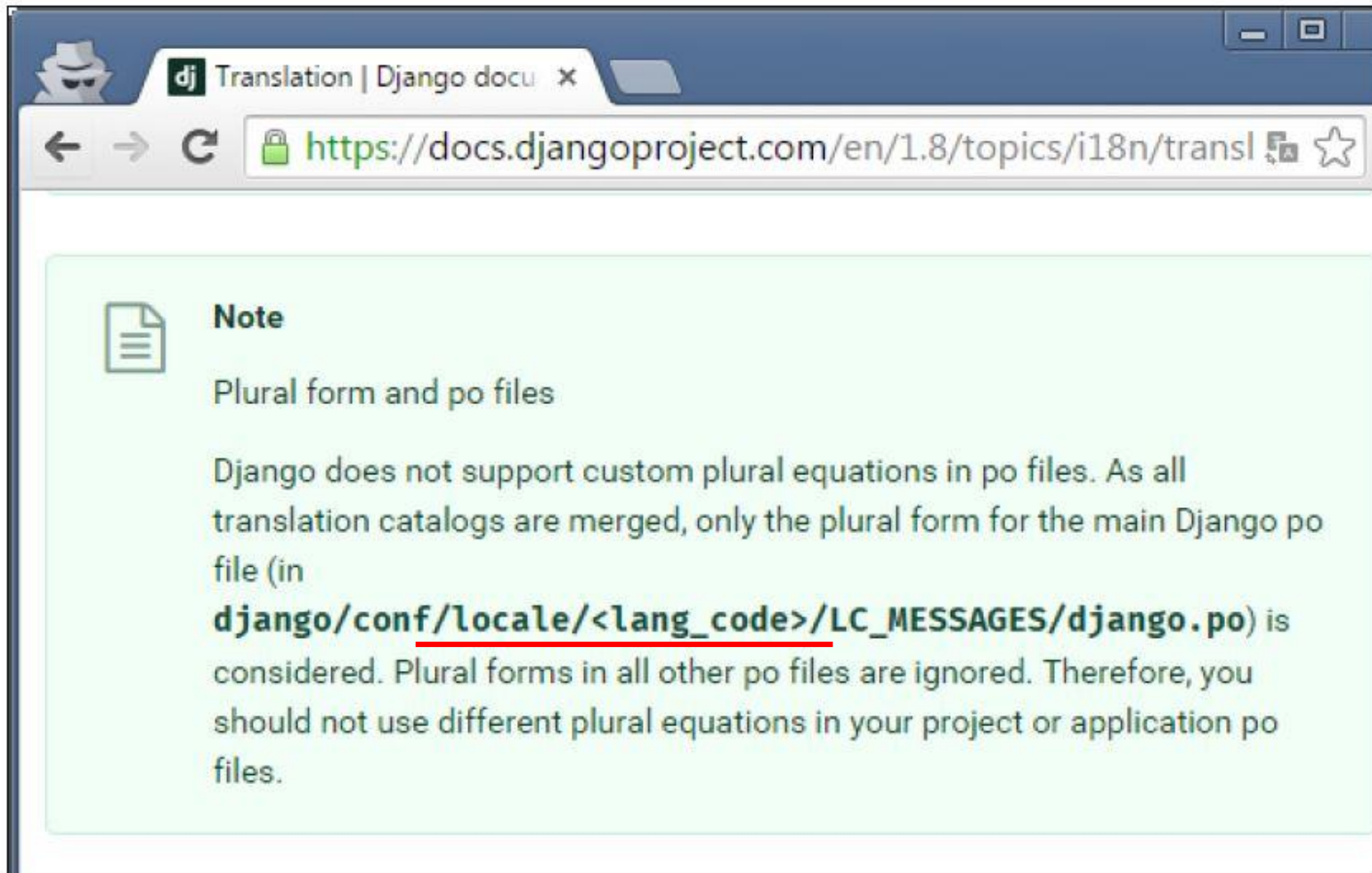
https://instagram.com/?hl=./en

Strange input leads to the Dutch page. Why?

Presumably the page reverts to the default language if value of hl gives an error
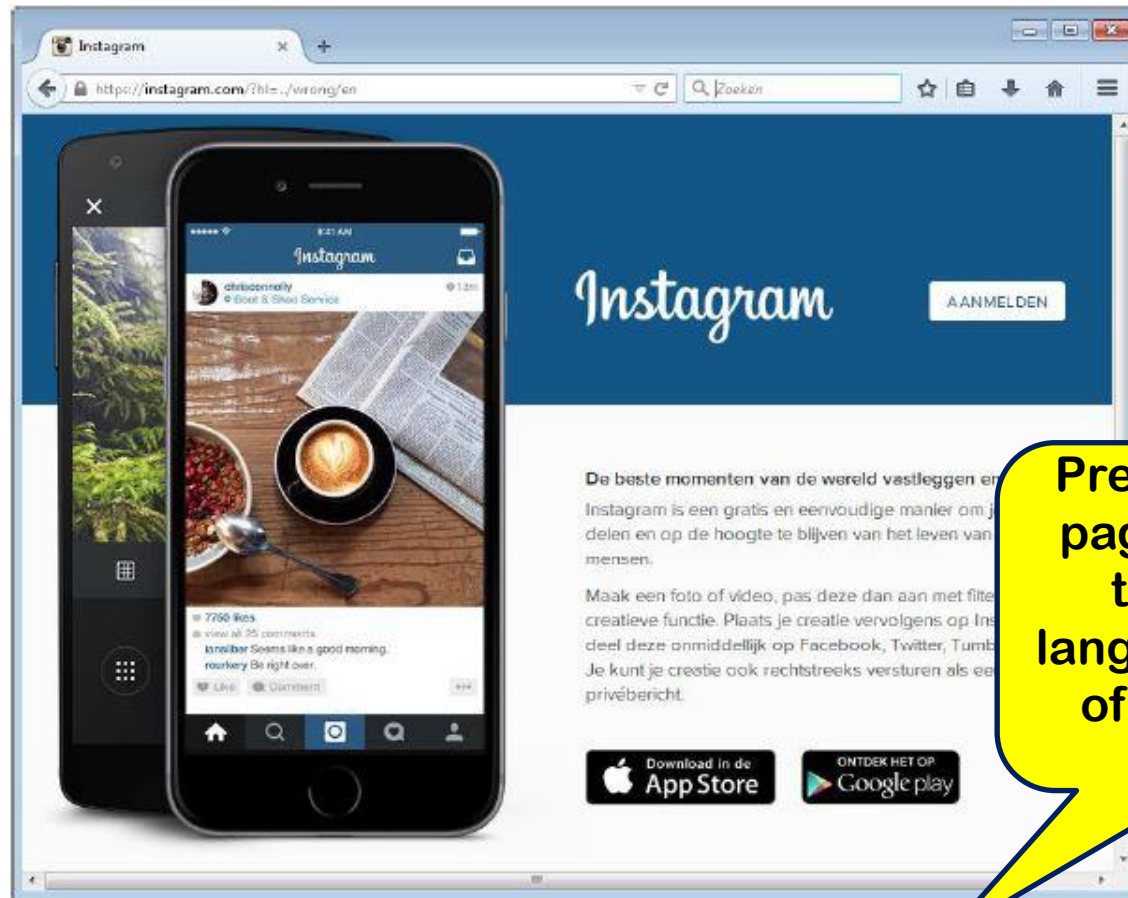
https://instagram.com/?hl=../wrong/en

Looking up some documentation
(for Django framework used by Instagram)
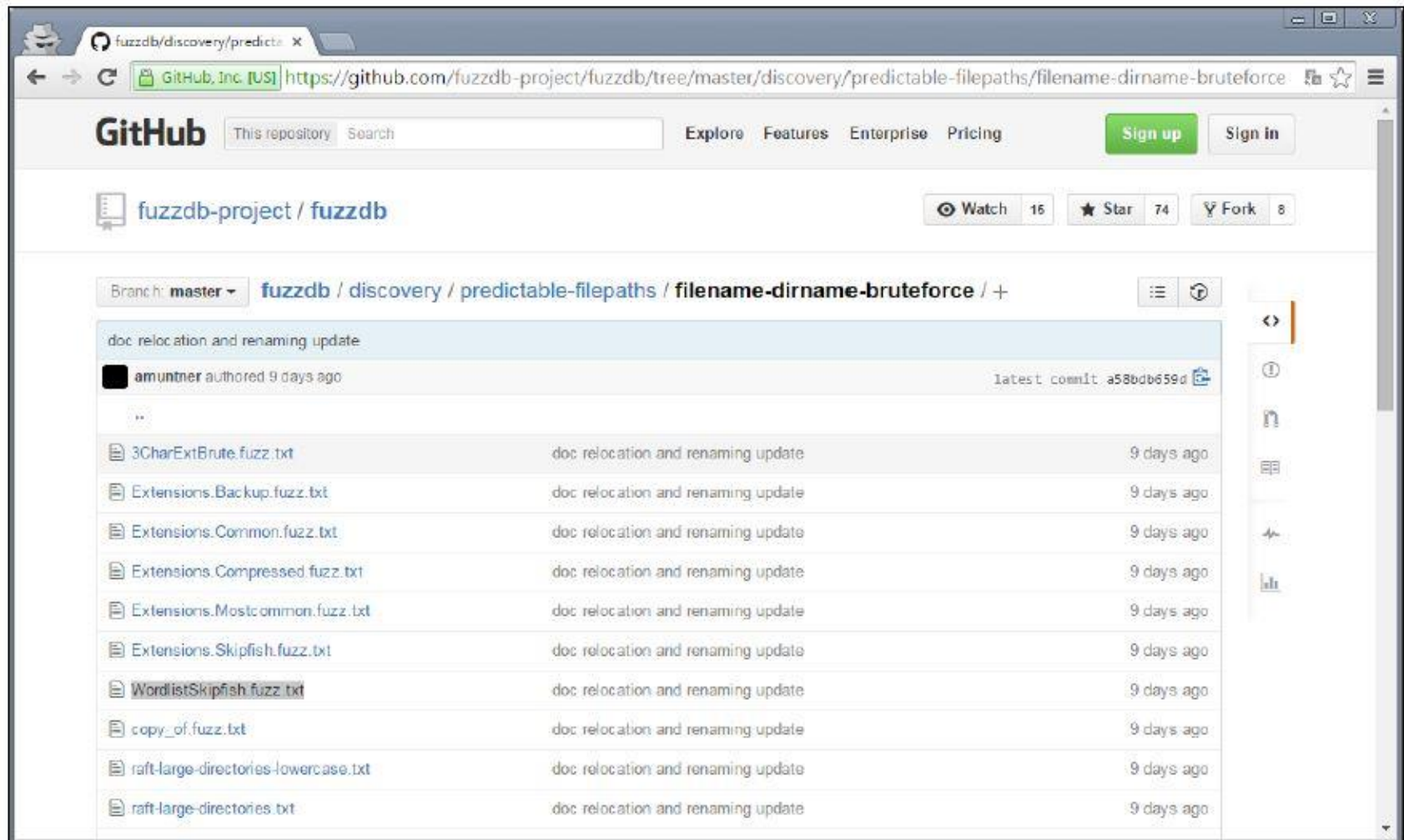
Webpage in English, so ../locale/en exists

https://instagram.com/?hl=../locale/en

Using fuzzdb to fuzz common file names

Success!

Fuzzdb finds 42 hits for `../<GUESS>/../locale/nl/`

Facebook's bug bounty program paid Arne 500$

Trying out

```
https://instagram.com/?hl=../../../../../../../../../../../../../dev/random%00
https://instagram.com/?hl=../../../../../../../../../../../../../dev/urandom%00
```

could have caused serious damage

The NULL trick

https://instagram.com/?hl=../../../../../../../../../../../../../../dev/random%00
https://instagram.com/?hl=../../../../../../../../../../../../../../dev/urandom%00

If the attacker's input ends up in the middle of a concatenation, this limits the scope of the attack.

For instance, by supplying malicious `<INPUT>` to

      `/usr/local/web/conf/<INPUT>.html`

then attacker can only access files with `.html` extensions

But: with **NULL** character, URL-encoded as `%00`, at the end of `<INPUT>`, the web server may ignore the rest of the string

More recent example

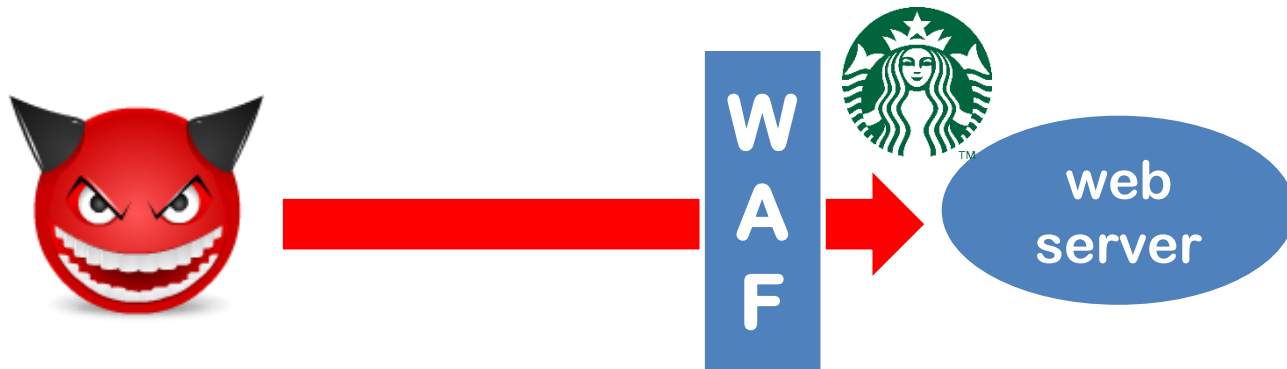## Security researcher earns $4k bug bounty after hacking into Starbucks database

John Leyden 22 June 2020 at 14:18 UTC

Path traversal weakness in a back-end API

Explanation at https://www.youtube.com/watch?v=sjvW79tjWoM

Fooling Starbuck's Web Application Firewall (WAF)

Starbuck's WAF disallows multiple . .

So you cannot include . . / . . in your malicious input ☹

How would you circumvent this?

    Type . . / . / . . instead ☺

A WAF (Web Application Firewall) sits in front of the web server and tries to filter very generic malicious inputs. Some WAFs are pretty crappy…