

Modern Webapp Penetration Testing

Hands-on Doing.

Brian (BB) King - @BBhacKing

1

1

About This Class

Objectives and Approach

2

2

Pentesting is Like Making Music

- Watch this on your own. How to play the flute. Seven seconds.
 - <https://www.youtube.com/watch?v=-1sM29zGxco>
- “Knowing” and “Doing” are very different things.

3

3

How to Play the Flute

“You blow there, and you move your fingers up and down here.”

He’s not wrong...



4

4

In theory, there is no difference
between *theory* and *practice*.
In practice, there is.

5

5

This Class is About Knowledge and Practice

- Lab-heavy
- Labs are focused on single issues
- Focus more on *why* and less on *how*
- Step-by-step walk-throughs provided
- VM includes much more than we cover in class
- Instructor here for you in Discord after class

6

6

Penetration Testing Is...

- Knowing how systems work
- Finding problems in systems
- Describing problems clearly
- Describing contributing factors
- Describing mitigating factors
- Offering solutions (when you can)

7

7

The Purpose of Pentesting

Testing Things Makes Things Better.

“Better” means regular people are safer while they computer.

8

8

Technology of Pentesting

- “I’ll never learn it all!!”

Correct!

A few foundational things go a long, long way.

There is no *advanced*.

There is only what you haven’t learned yet.

9

9

Methodology: How To Do It Well

10

Methodology: Which One is Right?

<ol style="list-style-type: none"> 1. Planning and Recon 2. Scanning 3. Gaining Access 4. Maintaining Access 5. Analysis and WAF Configuration 	<ol style="list-style-type: none"> 1. Passive Recon 2. Active Recon 3. Social Engineering 4. Exploitation 5. Post-Exploitation 6. Privilege Escalation 7. Lateral Movement 8. Maintain Access 9. Cover Tracks 10. Reporting
---	---

11

Methodology... Can Feel Overwhelming

12

OWASP Web Security Testing Guide

- 1337 Stars!
- 475 Pages

The Need for a Balanced Approach

With so many techniques and approaches to testing the security of web application which techniques to use or when to use them. Experience shows that there is no right of exactly which techniques should be used to build a testing framework. In fact, all all the areas that need to be tested.

13

13

Methodology: What's Your Goal?

Try to get something of value that should not be allowed.

- Attack the application and the server, of course.
- Attack *other users*, too.
- "Getting a shell" is pretty rare on webapp pentests.
- Invading other people's stuff? That's pretty common.

14

14

Methodology, Actually

1. Find out what it's supposed to do.
2. Find out what it *actually does*.
3. Report differences as you discover them.

15

15

Application Security is Situational Awareness

16

16

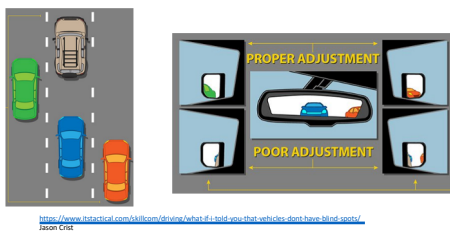
Driving: Blind Spots



17

17

Driving: Blind Spots



18

18

Adjustment and Awareness
Can Eliminate Blind Spots

- You'll notice things other people miss!
 - But still...
 - Check mirrors often
 - Look before changing lanes

19

19

Situational Awareness
for the Web

What's normal, what's common, what's possible.

20

20

Web Technologies

- HTTP
 - The "Web" part of the Internet
- URLs
 - The way to navigate the web
- HTML
 - The user interface
- Javascript
 - ...also the user interface.
 - ...and more.

21

21

HTTP 1.1

- Text-based protocol
- TCP for transport
- URLs
- Stateless
- Request and Response
- Defined in RFC 2626 (1999)
- Updated in RFCs 7230, 7321, 7233, 7234, 7235 (2014)

```
GET /index.html HTTP/1.1
Host: example.com

POST /login.php HTTP/1.1
Host: example.com
User-Agent: Mozilla 5.0 (more lies)
user=admin&password=admin123
```

22

22

HTTP Request Methods / Verbs

- GET: visit a link
- POST: submit a form
- PUT: create a resource
- PATCH: update a resource
- DELETE: delete a resource
- TRACE: echo the request
- CONNECT: establish a tunnel to target (e.g. proxy)
- OPTIONS: list supported verbs

23

23

HTTP Response Codes

- 100-series: Informational
 - 101: switching protocols (e.g. to WebSockets)
- 200-series: OK
 - 200 OK
- 300-series: redirects
 - 301, 302 Moved (permanent, temporary)
 - 304 Not Modified
- 400-series: problem with request
 - 400 Bad Request
 - 401 Unauthorized (prompt for auth)
 - 403 Forbidden (end of the line)
 - 404 Not Found
- 500-series: problem with server
 - 500 Internal Server Error
 - 502 Bad Gateway
 - 503 Service Unavailable



<https://httpstatuscodes.com/502-bad-gateway/>

24

24

HTTP 2.0

- HTTP/2
- Performance improvements
 - Address "head of line blocking"
- "Speculative server push"
- Encrypted connections only
 - ...required not by the spec, but by the browser makers
- RFC 7540 (2015)

25

25

HTTP 3.0

- HTTP/3
- Was "HTTP-over-QUIC"
- QUIC was "Quick UDP Internet Connections"
- Reduced latency
- RFC 7540 (2015)
- ...not finished yet.

26

26

HTTP 1.1

- HTTP 2.0 falls back to 1.1
- No differences at application / presentation layer
- For application testing ... not critical

Also: Testing tools are just starting to support HTTP/2

**Burp added "Experimental support" in July.*

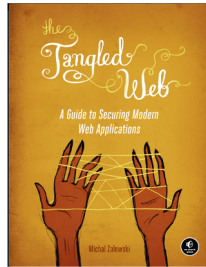
27

27

Fundamental Problem of Browsers

- Download random code
- ...from random places
- ...run it immediately
- Prevent dangerous operations
- Keep sites separate from each other
- But don't be too strict!
- And be quick about it!
- ...and don't inconvenience anyone!

<https://nostarch.com/tangledweb>
<https://www.html5rocks.com/en/tutorials/internals/howbrowserswork/>



28

28

HTML

- XML Document
 - Text
 - Tags `<h1>Welcome!</h1>`
 - Attributes `<h1 name="title">`
 - Replaced elements, non-replaced elements ``, `<p>hi</p>`
- Usually contains non-HTML bits
 - Javascript
 - CSS
 - SVG (whole 'nother XML document, really)
 - Inline images

29

29

HTML Rendering

- Several passes of *tokenizing and parsing*
- Tokenizing: identifying the "words"
 - aka "lexical analysis" or "lexing" done by the "lexer" (or tokenizer)
- Parsing: determining what the "words" mean

30

30

The Browser Starts Here

```

<!DOCTYPE HTML>
<HTML>
  <HEAD>
    <TITLE>Example</TITLE>
    <STYLE type="text/css">h1 {color: #0000ff}</STYLE>
  </HEAD><BODY>
    <H1>Example</H1>
    <!-- html comment -->
    <script>alert('Example!'); // js comment</script>
  </BODY>
</HTML>

```

31

31

Javascript

yes, Javascript.

Small s in the middle.

It is time.

...unless you also type *InterNet* and *MODEM* and *KeyBoard* in which case you have my blessing.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Language_Resources

32

32

Javascript

- Used to be just presentation, now everywhere.
- Often controls the UI
 - jQuery
 - Angular
 - React
 - Vue.js
 - Ember.js
- More later

33

33

URLs and URIs

- All URLs are URIs.
- A URI is a name.
- A URL is a name that also tells you where to find the thing it names.

- You are a mammal
- But you'd probably call yourself "human" before "mammal"

- Same with URLs and URIs
- But don't get hung up on it.

<https://danielmiessler.com/study/difference-between-uri-url/>

34

34

URLs and HTTP Traffic

<https://example.com/search?q=bananas&lang=en#SomeHeading>

scheme & separator

"authority"

path

query string

fragment (aka "hash") -
NOT SENT TO SERVER

35

35

About that Hash...

- Originally for linking to sections in a document.
- Now, also used to pass variables to client-side scripts.

Original: <https://tools.ietf.org/html/rfc7519#section-5>

Now, also: <http://localhost:3000/#/login>

36

36

Lab #1:
Orient: A Tour of the VM

37

On The Virtual Machine

- Terminal [double-click the desktop icon]
- Juice Shop installed natively [double-click the desktop icon]
- Burp Suite Community [double-click desktop icon]
- Firefox web browser [double-click desktop icon]
 - Bookmarks Bar: Notes, Juice Shop, CyberChef, Regex101.com, Regexr.com

38

Alternative

- Deploy your own Juice Shop on Heroku
 - <https://elements.heroku.com/buttons/bkimminich/juice-shop>
 - Pick a name that won't be guessed, or others may be attacking, too.
 - <https://juice-shop.herokuapp.com/#/> (public version)
- Run your own Burp Suite anywhere
- ...but I can help you better with the VM.

39

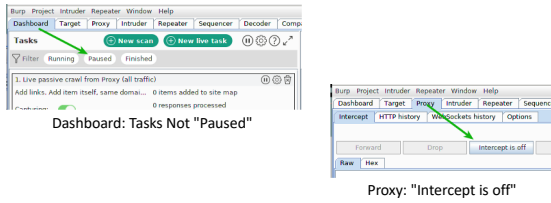
Intercepting HTTP Proxy

- Burp Suite Community Edition: Free. Good Enough.
 - We'll use this.
- ZAP: Also free. More complete than Burp Community
 - Not covered in detail in this class.
- Burp Suite Pro: Not free. Adds Scanner and Automation

40

40

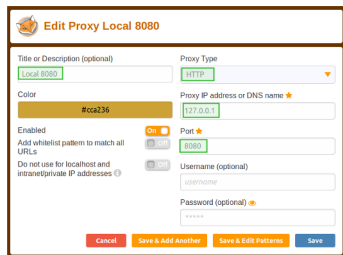
Start Up Burp Suite



41

41

Point Firefox At Burp Suite



42

42

Why Not Just Accept the Warnings?

- 1. Annoying
- 2. You don't always *get* a warning.

43

43

Import Burp's CA Certificate (This is already done on the VM)

Click "CA Certificate" Save the File cacert.der

Import the Burp CA Certificate about:preferences#privacy View Certificates...

44

44

Import Burp's CA Certificate (This is already done on the VM)

On Authorities tab, click "Import..."

Browse to saved "cacert.der" file

Trust it "to identify websites"

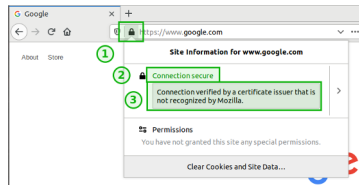
45

45

Make Sure It Works

With Firefox running through Burp, visit Google.

```
if (this_works){  
  you_are_ready;  
}
```



46

A Tour of Burp Suite

- Everyone uses it differently.
- Here are some of my habits.
 - startup
 - proxy history, reversed
 - scope and proxy history
 - filters
 - repeater
 - BApp Store
 - Pro vs Community

47

Lab #1 Complete:
Oriented.

48

Developer Tools

Let's just go take a look at all those things.

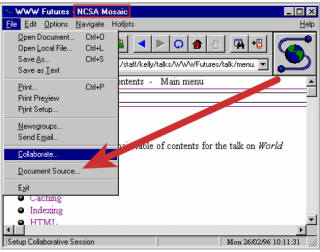
49

49

View Source and its Progeny

"View Source"
goes all the way back
to the beginning of the web.

Obsoleted (almost) by Javascript



The screenshot shows the NCSA Mosaic browser interface. The 'View Source' menu option is highlighted with a red arrow. The browser window displays a page with a table of contents for a talk on World.

50

50

Use Developer Tools For...

- Situational Awareness!
- Understanding how a page is built.
- Seeing what hosts it's talking to, what content it's bringing in.
- Extracting useful information.
- Changing the app's behavior.
- Different browsers have similar tools.
- Firefox docs: <https://developer.mozilla.org/en-US/docs/Tools>
- Chrome docs: <https://developers.google.com/web/tools/chrome-devtools/>
 - Chrome Dev tools for ... actual developers...
 - <https://bit.ly/wickedweapons> (Greg Malcolm's "Raiding the Armory" talk and tools)

51

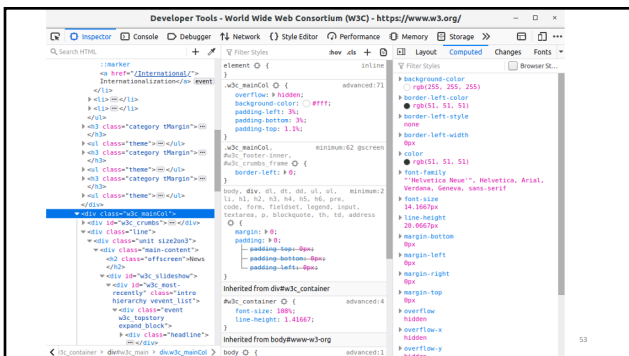
51

Getting Into Them

- Play Along in your VM!
 - <https://developer.mozilla.org/en-US/docs/Tools>
- F12 ... Ctrl-Shift-i ... Cmd-Opt-i
- Console: what's going on.
- Inspector: how it's put together.
- DOM: top-down view.
- Storage: little databases...
- Network: where's it all coming from.
- Debugger: do that again, but slower.

52

52



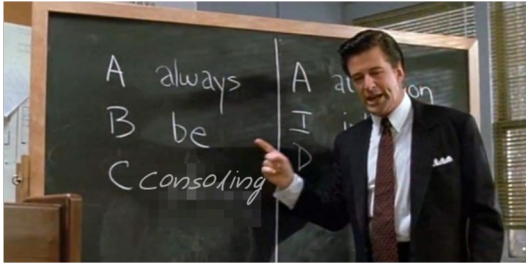
53

Display Area is Limited so...



54

Keep the Console Visible!



55

Console...

Console: *One of the tools*
Tap 'esc' to open Console
along with any other tool



56

56

Console

- HTTP Requests (and responses)
 - Summary and detail
 - References for status codes &
 - HTTP version
- Calls to console.log();
- !! \o/ !!
- Browser errors & warnings

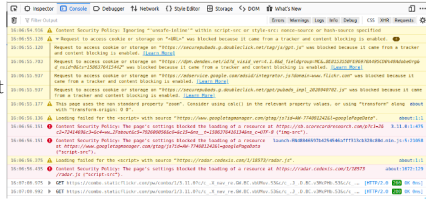


57

57

What Kinds of Warnings?

- CSP
- Privacy Blocks
- Mixed Content
- TLS Errors
- Deprecation
- "Learn More" !



58

Network

- Every request & response
- ...kind of like the console on first look
- Still has links to header documentation
- Adds filters by content-type
- Adds freeform text filter box
- Adds "Edit and Resend" box!

59

Network Tab: List

Sta...	M...	Domain	File	Ca...	Type	Transferred	Size
200	GET	developer.mozilla.org	file:7ac510b78865.svg	img	svg	cached	339 B
200	GET	developer.mozilla.org	clock.661d537f6023.svg	img	svg	cached	409 B
200	GET	developer.mozilla.org	web-docs-sprite.22a6a085cf14.svg	img	svg	cached	10.09 KB
200	GET	developer.mozilla.org	Favicon144.e7e21ca263ca.png	img	png	cached	1.30 KB
200	GET	developer.mozilla.org	Favicon32.7f3da72dcea1.png	img	png	cached	441 B
200	GET	www.google-an...	collect?v=1&_vs=81&ip=1&a=1751...	img	gif	598 B	35 B

60

60

Network Tab:
Details

Live help links
ALL OVER THE PLACE

61

To the Tools!
Network

62

Storage

- Cache
 - ...not the disk cache: the Cache interface from the Web APIs
 - usually empty
- Cookies (add, view, edit, delete)
 - 4kb per cookie, 20 or so per origin
- Indexed DB
 - At least 10MB and no more than 2GB (or 10% of hard disk).
 - Database. Structured data. (also usually empty)
- Local Storage (view and edit)
 - 5MB per origin. Persistent.
- Session Storage
 - 5MB per origin. Purged when browser closes
 - ... (as with "session cookies")
- This stuff won't show up in your proxy!
 - Well, the cookies **will** might.

63

Storage: Cookies

- Add, Edit, Delete.
- No browser extension needed.

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly
_gat	1	.mozilla.org	/	Tue, 21 Apr 2020 2...	5	false
_ga	GA1.2.646147...	.mozilla.org	/	Thu, 21 Apr 2022 2...	29	false
_gid	GA1.2.272002...	.mozilla.org	/	Wed, 22 Apr 2020 ...	30	false
dnt_sg...	False	.developer...	/	Thu, 21 May 2020 ...	27	false
lux_uid	158750138571...	developer...	/	Tue, 21 Apr 2020 2...	25	false

64

To the Tools!
Storage

65

Inspector (and the Console)

- Like "View Source" (Ctrl-U) but better
- Same as right-click, inspect element
 - shows current state, not just "as-received" from server
 - live view (mouseover to get highlights)
 - live view (can edit: edits show in rendered view)
 - This is a great tool for practical jokes, by the way.

66

Inspector & Console for Useful Things

- Reveal text in "password" fields.
- Execute functions... See: <https://kriszyp.github.io/docs/>

```

AES Encryption
Plain text encryption
var CryptoJS = require("crypto-js");
// Encrypt
var ciphertext = CryptoJS.AES.encrypt('my message', 'secret key 123').toString();
// Decrypt
var bytes = CryptoJS.AES.decrypt(ciphertext, 'secret key 123');
var originalText = bytes.toString(CryptoJS.enc.Utf8);
console.log(originalText); // 'my message'

```

67

To the Tools! Inspector

68

Customizing the Tools

- F1 (settings)
 - Enable and disable panes (add the DOM pane)
 - Dark mode
 - Screenshot button!

69

Console Tricks (Javascript here)

- What can I ask the console about?
- console.log(top);
- top === window
- top === document
- window === document

```

document
  <- HTMLDocument https://www.flickr.com/about
  URL: "https://www.flickr.com/about"
  _uid: "yui_3_11_0_3_1586379241484_3"
  > activeElement: <body class="zeus en-us new-footer ne.a-linux-gecko-74 liquid">
  > alt: HTMLCollection (0)
  > all: HTMLCollection (0)
  > anchors: HTMLCollection (1)
  > applets: HTMLCollection (length: 0)
  > baseURI: "https://www.flickr.com/about"
  > bgColor: ""
  > body: <body class="zeus en-us new-footer ne.a-linux-gecko-74 liquid">
  > characterSet: "UTF-8"
  > charset: "UTF-8"
  > childElementCount: 1
  > children: NodeList (1)
  > children: HTMLCollection (1)
  > compatMode: "CSS1Compat"
  > contentType: "text/html"
  > cookie: "yd=668181; vp=122292c118992c1b2c12; s.cctrue; _sid=687c94683862626d"
  
```

70

Console Tricks (Javascript here)

Get every anchor link on a page.

```

var links = document.getElementsByTagName("A");
const links = document.links;
document.links; // ...actually...
  
```

Get every image on a page.

```

document.images;
  
```

71

Console Tricks - Modify Link Targets

Add an "admin=true" to all the links.
That could work, right?

```

for(link of document.links) {
  link.href = link.href + "&admin=true";
  link.style = "border: 5px solid blue";
}
  
```

72

Console Tricks - Collect Identifiers

Get the URL of Every Image

```

const pic_urls = [];
for(pic of document.images){
  pic_urls.push(pic.src);
}
console.log(pic_urls);

```

```

>> const pic_urls = [];
for(pic of document.images){
  pic_urls.push(pic.src);
}
console.log(pic_urls);
← undefined
(94) [1]
0: "https://combo.staticflickr.com/pw/images/tour/en-us/create-acco
1: "https://farm3.staticflickr.com/2853/buddyicons/60852878000_r.jp
2: "https://farm3.staticflickr.com/8322/buddyicons/365119815430481_r_
3: "https://farm3.staticflickr.com/7317/buddyicons/1571381800_r.jp
4: "https://farm3.staticflickr.com/8863/buddyicons/281447870807_r.jp
5: "https://farm3.staticflickr.com/892/buddyicons/11787544005_r.jpg
6: "https://farm3.staticflickr.com/8937/buddyicons/78522878000_r.jp
7: "https://farm3.staticflickr.com/8725/buddyicons/91728078002_r.jpg
8: "https://farm3.staticflickr.com/7386/buddyicons/63652802802_r.jp
9: "https://farm3.staticflickr.com/3718/buddyicons/267946078082_r.jp
10: "https://farm3.staticflickr.com/6689/buddyicons/18649878000_r.jp

```

(or, filter the "Network" tab by "Images")

73

73

What All's Out There

Starting somewhere is better than not starting.

74

74

Thank You, OWASP

Who else can you trust to pay attention to all this stuff?

- Open Web Application Security Project, since 2001
 - Security of applications delivered over the Web
 - Vendor neutral
- OWASP Top Ten, yes.
- Tons of support for developers and defenders
 - 18 "Flagship" projects
 - 15 "Lab" projects
 - 33 "Incubator" projects

Tools

- OWASP Zed Attack Proxy
- OWASP Web Testing Environment Project
- OWASP OWTF
- OWASP Dependency Check
- OWASP Security Shepherd
- OWASP Defectdojo Project
- OWASP Juice Shop Project
- OWASP Security Knowledge Framework
- OWASP Dependency Track Project

Code Health Check January 2017

- OWASP ModSecurity Core Rule Set Project
- OWASP CSRFGuard Project

Documentation

- OWASP Application Security Verification Standard Project
- OWASP Software Assurance Maturity Model (SAMM)
- OWASP Assessor Project
- OWASP Top Ten Project
- OWASP Testing Project
- OWASP Cheat Sheet Series
- OWASP Mobile Security Testing Guide

75

75

Getting Involved

1. Find your local chapter
 - a. <https://owasp.org/chapters/>
2. Go To A Meeting
 - a. Say, "Hi. I'm new here. I took a webapp pentesting class and this all looks like a truckload of fun. What do you all do here?"
 - b. Engage in active listening
 - c. Follow up in some tiny way
3. GOTO 2

76

76

Gratitude for OWASP Juice Shop

Björn Kimminich: OWASP. Prolific.

- <https://hub.docker.com/r/bkimminich/juice-shop/>

Solution Guide!

- <https://bkimminich.gitbooks.io/owning-owasp-juice-shop/content/>



77



77

Digi Ninja

<https://digi.ninja/labs.php>

TONS of free labs here.

LABS

Every now and then I come across a topic that is too complicated to try to learn or explain through a single blog post, or those instances I like to build up a lab to play around in. Here are a bunch of these labs, the ones that are safe to put online.

If you like these labs, and are interested in using them in your training or would like some bespoke labs created for your organization, please [get in touch](mailto:bjk@kimminich.net).

Auth0BrokenIDP
Play with various broken authentication systems.

The CORS Demos
A set of CORS requests and responses to demonstrate all the different permutations.

OWASP Juice Shop
A lab to help understand and then attack a GraphQL based application.

SocketCAN
A lab to play with web sockets.

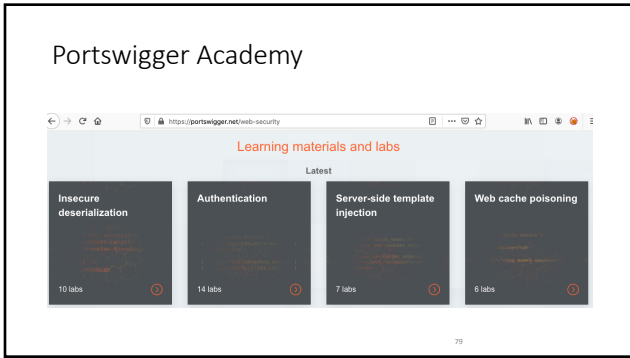
OWASP Defence Scanner
Did you know that OWASP ZAP can contain JavaScript which can be used for Cross Site Scripting? This lab demonstrated this, and shows how it can be defended against.

LDAP
An LDAP based vulnerable web application.

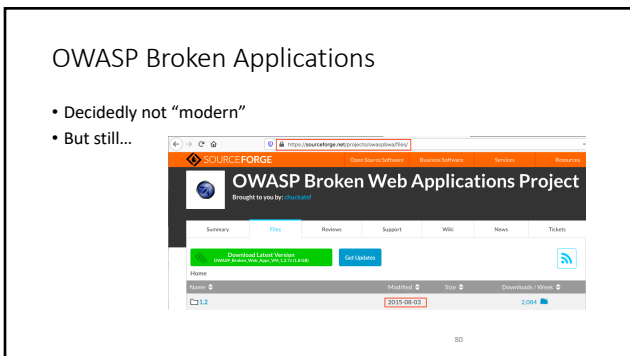
Web Cache Poisoning
The implementation of some of the web cache poisoning issues identified by James Kettle.

78

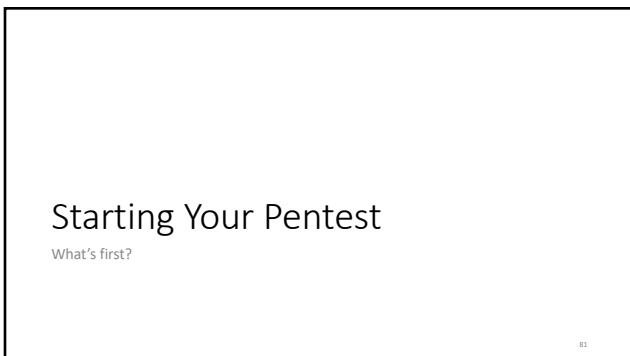
78



79



80



81

Learn the Target

Mistake #1: Jumping into "testing" before you know the application.
Jumping straight to testing is what vulnerability scanners do.
You're smarter than a vulnerability scanner.

You're slower, and more easily distracted, too ...
"smart" isn't everything, you know.

82

82

Bring Your Brain

Curiosity.
Persistence.
Attention to Detail.

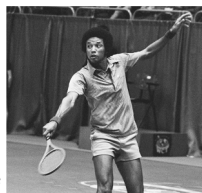
These are the keys to a good webapp^W pentest.

83

83

Start where you are.
Use what you have.
Do what you can.

-- Arthur Ashe



84

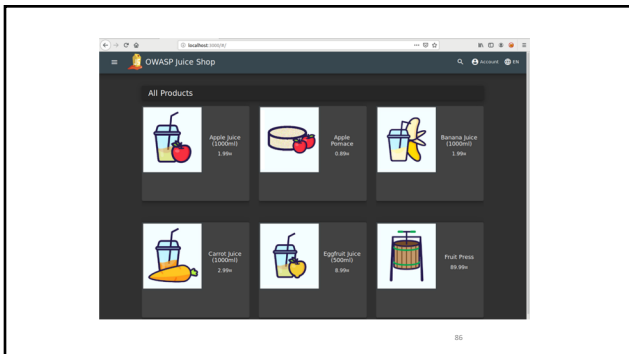
84

Start Where You Are

- We are in the web browser. Let's start here.
- Browse the site, while Burp records everything.
 - What does the site *do*?
 - Who are its *intended users*?
 - Is there a login page?
 - Does it do anything "risky" or "expensive"?
 - What INPUTS does it ask for?
 - Does it show user-generated content?
- Click every link, submit every form.
 - When I say, "every..."

85

85



86

86

"Post-Recon"

You'll know you're done when the target sitemap has no gray left.

When I say, "done..."

You're never "done"

For today ... why don't you see all the unvisited links in the Burp Sitemap?

87

87

Lab #2:
Learn the Target

88

Lab #2: Understand The Shop.

- Double-click "start juiceshop" on the desktop, then visit <http://localhost:3000> in browser.
- Browse the site, while Burp records everything.
 - What does the site *do*?
 - Who are its *intended users*?
 - Does it do anything "risky" or "expensive"?
 - What inputs does it ask for?
 - Does it show user-generated content?
- Click every link, submit every form.
 - "every..."
- **Register an Account. Buy Something.**
- Keep the Console open. Watch for anything interesting.
- Use the Inspector to get details about something interesting.

89


What We've Learned

It's a store. You can self-register, then buy things
What steps were needed to buy something?
 You can write reviews, edit reviews & read others' reviews
 You can purchase things
 You can see order status
 You can save credit card info and addresses
 There are coupons
 There's *Customer Feedback and Complaints*

90

Lab #2 Complete:
Learn the Target

91



What's Your
Takeaway from
the Lab?

92

Hidden or "Unlinked" Content

- Different roles...
 - ...different menus
 - Administrator? Here's the "add users" item
 - Manager? Here's the "view users" item
 - Visitor? No "user" item at all
- Different users...
 - ...different content
 - I see my documents
 - You see yours

93

93

Securing Hidden or "Unlinked" Content

Safest way to make sure people only see what they should see?

- A. Hide links to what the current user is allowed to see
- B. Check "Authorization" of each request on the server
- C. Both

...now, what's the *easiest* way?

94

Lab #3:
Find the Scoreboard

95

Before You Start...

The Trouble Writing a CTF:

- What's too clever?
- What's too obvious?
- When does sequencing get in the way?
- Remember the thing about puzzles?
- In a real pentest, there is no scoreboard
- But there may be hidden resources.

96

Hint

How does this app render the UI? How does the # work?
Where might you look in responses for references to a score board?
What tools might help? Developer Tools? Burp Target Sitemap?
What descriptive files (might) exist on any web server?
There's one for search engines... (robots.txt^[1])
And another for security people... (<https://securitytxt.org/>)

[1] <https://webmasters.googleblog.com/2019/07/rep-id.html>

97

97

Lab #3 Complete:
Find the Scoreboard

98

98



What's Your
Takeaway from
the Lab?

99

99

Day 1 Recap

- Doing and Knowing complement each other
- Situational Awareness is KEY
- HTTP 1.1 is Good Enough (for application testing)
- Developer Tools are Awesome
 - Always Be Console-ing
- First Task in a Pentest: Learn the Target
 - First: What's it meant to do?
 - Then: What's it actually do?
- Found the Scoreboard!

100

100
