2010: and still bruteforcing OWASP Webslayer

Christian Martorella

July 18th 2010

Barcelona
Who am I

- Manager Auditoria
- CISSP, CISA, CISM, OPST, OPSA, CEH
- OWASP WebSlayer Project Leader
- FIST Conference, Presidente
- Edge-Security.com
Brute force attack

Is a method to determine an unknown value by using an automated process to try a large number of possible values.
BRUTE FORCE

If it doesn't work, you're just not using enough.
What can be bruteforced?

- Credentials (HTML Forms and HTTP)
- Session identifiers (session id’s)
- Predictable resource location (directories and files)
- Variable values
- Cookies
- WebServices methods (rest)
Where?

- Headers
- Forms (POST)
- URL (GET)
- Authentication (Basic, NTML)
How?

- Dictionary attack
- Search attack (all possible combinations of a character set and a given length)
- Rule based search attack (use rules to generate candidates)
Why 2010 and still bruteforcing?

In 2007 Gunter Ollmann proposed a series of countermeasures to stop automated attack tools.
Countermeasures

- Block HEAD requests
- Timeouts and thresholds
- Referer checks
- Tokens
Countermeasures

- Turing tests (captchas)
- Honeypot links
- One time links
- Custom messages
- Token resource metering (Hashcash)
# Countermeasures

<table>
<thead>
<tr>
<th>Technique</th>
<th>Tool Generation</th>
<th>Tool Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host Server Renaming</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>Blocking HEAD</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>REFERER Fields</td>
<td>**</td>
<td></td>
</tr>
<tr>
<td>Content-Type Manipulation</td>
<td>**</td>
<td></td>
</tr>
<tr>
<td>Client-side Redirection</td>
<td>**</td>
<td></td>
</tr>
<tr>
<td>HTTP Status Codes</td>
<td>**</td>
<td></td>
</tr>
<tr>
<td>Thresholds &amp; Timeouts</td>
<td>***</td>
<td></td>
</tr>
<tr>
<td>Onetime Links</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Honeypot Links</td>
<td>***</td>
<td></td>
</tr>
<tr>
<td>Turing Tests</td>
<td>***</td>
<td></td>
</tr>
<tr>
<td>Token Appending</td>
<td>***</td>
<td></td>
</tr>
<tr>
<td>Token Calculators</td>
<td>***</td>
<td></td>
</tr>
<tr>
<td>Token Resource Metering</td>
<td>***</td>
<td></td>
</tr>
</tbody>
</table>

**1st Generation** | **2nd Generation** | **2.5 Generation** | **3rd Generation** | **Web Spidering** | **CGI Scanning** | **Brute Forcing** | **Fuzzers** | **Vuln Scanning**

Key: [] No benefit, [*] Some benefit, [**] Noticeable Benefit, [***] Valuable Protection
Workarounds
Workarounds

Captcha breakers

Will break CAPTCHA for food.
Workarounds

Distributing scanning source traffic

- Attacker
- Proxy HTTP 1
- Proxy HTTP ...
- Proxy HTTP N
- Target
Workarounds

Distributing scanning on different targets
Workarounds

- **Diagonal scanning** (different username/password each round)
- **Horizontal scanning** (different usernames for common passwords)
- **Three dimension** (Horizontal, Vertical or Diagonal + Distributing source IP)
- **Four dimensions** (Horizontal, Vertical or Diagonal + time delay)
2010...

114,000 emails

https://dcp2.att.com/OEPClient/openPage?ICCID=NUMBER&IMEI=0
Access Any Users Photo Albums


aid=-3 (-3 for every public profile album)
id=0123456789
l=? (all we know is its 5 characters from the 0123456789abcdef range)
2010...

- The 500 worst passwords list
- Alyssa banned passwords list
- Cain’s list of passwords
- Conficker’s list
- The English dictionary
- Faithwriters banned passwords list
- Hak5’s list
- Hotmail’s banned passwords list
- Myspace’s banned passwords list
- PHPbb’s compromised list
- RockYou’s compromised list
- Twitter’s banned passwords list
Webservices

http://l33.login.scd.yahoo.com/config/isp_verify_user?
l=USERNAME&p=PASSWORD

OK:0:username

ERROR:101:Invalid Password

ERROR:102:Invalid Login
2010...

Password bruteforce

946 tries

```
python wfuzz.py -c -z file -f wordlists/common.txt --hc 200 -
d"email=securik@gmail.com&input_password=FUZZ&timezone=1" "https://www.tuenti.com/?m=Login&func=do_login"
```
Automated scanning tools are designed to take full advantage of the state-less nature of the HTTP protocol and insecure development techniques.
Tools

WEBSLAYER
Edge-Security
AN OWASP PROJECT

Evolution of WFUZZ
The main objective is to provide to the security tester a tool to perform highly customized brute force attacks on web applications, and a useful results analysis interface. It was designed thinking in the professional tester.
Webslayer
Webslayer

- Predictable credentials (HTML Forms and HTTP)
- Predictable sessions identifier (cookies, hidden fields, url)
- Predictable resource location (directories and files)
- Variables values and ranges
- Cookies
- WebServices methods
- Traversals, Injections, Overflows, etc
Webslayer

- **Encodings**: 15 encodings supported
- **Authentication**: supports Ntlm and Basic (known or guess)
- **Multiple payloads**: you can use 2 payloads in different parts
- **Proxy support**: (authentication supported)
- **Multithreads**
- **Multiple filters** for improving the performance and for producing cleaner results
Webslayer

- Predictable resource location: Recursion, common extensions, non standard code detection, (Huge collection of dictionaries)
- Advanced payload generation
- Live filters
- Session saving/restoring
- Integrated browser (webKit)
- Full page screenshot
Resource location prediction

- Based on the idea of Dirb (Darkraver)
- Custom dictionaries of known resources or common passwords
  - **Servers**: Tomcat, Websphere, Weblogic, Vignette, etc
  - **Common words**: common (950), big (3500), spanish
  - **CGIs (vulnerabilities)**
  - **Webservices**
  - **Injections** (SQL, XSS, XML, Traversals)
<table>
<thead>
<tr>
<th>Timer</th>
<th>Code</th>
<th>Lines</th>
<th>Words</th>
<th>Chars</th>
<th>MDS</th>
<th>Payload</th>
<th>Cookie</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.111258</td>
<td>403</td>
<td>44</td>
<td>108</td>
<td>1173</td>
<td>59b9c4dd9...</td>
<td>manual</td>
</tr>
<tr>
<td>5</td>
<td>0.093343</td>
<td>301</td>
<td>20</td>
<td>241</td>
<td>c99d4a10...</td>
<td>secured</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.130115</td>
<td>200</td>
<td>103</td>
<td>292</td>
<td>5a080b1d...</td>
<td>cart - php</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.280272</td>
<td>200</td>
<td>107</td>
<td>308</td>
<td>57d0188bc...</td>
<td>guestbook - php</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.140843</td>
<td>200</td>
<td>102</td>
<td>288</td>
<td>0441f30c2...</td>
<td>index - php</td>
<td></td>
</tr>
</tbody>
</table>
Webslayer OWASP 2010

Base 64: VZvkZxheWVvIE9QYQoQ1qWNA=

Uri Hexadecimal: 57%56%52%73M%56%36%79%65%72%20%64%61%53%50%20%32%30%32%30%31%30

Random upper: WebsLayer OWASP 2010

Double nibble hexa:
%35%37%36%35%30%30%32%30%32%31%36%39%33%33%32%33%31%30%30%30%31%33%33%34%30

SHA1: ea95025740cede6a90e12412810b73936edc6d

MD5: 2d1e17b21d0464ac322e5f22469258d

Binary Ascii: 576562736c61796572204f574135302032303330

Html Decimal: &amp;#87;&amp;#101;&amp;#98;&amp;#115;&amp;#108;&amp;#97;&amp;#121;&amp;#101;&amp;#114;&amp;#32;&amp;#83;&amp;#81;&amp;#80;&amp;#32;&amp;#50;&amp;#48;&amp;#49;&amp;#48;

Html Hexadecimal: 84%57%65%62%5c%73%6c%61%79%65%72%20%4f%57%41%35%30%20%32%30%33%30

UTF-8 Binary: 1b57%65%62%73%6c%61%79%65%72%20%4f%57%41%35%30%20%32%30%33%30

UTF-8: \u0065\u0062\u0073\u006c\u0061\u0079\u0065\u0072\u0020\u004f\u0057\u0035\u0030\u0032\u0030\u0033\u0030


Mysql char:

*CHAR(87)+CHAR(101)+CHAR(98)+CHAR(115)+CHAR(108)+CHAR(97)+CHAR(121)+CHAR(101)+CHAR(114)+CHAR(32)+CHAR(79)+CHAR(87)+CHAR(65)+CHAR(83)+CHAR(80)+CHAR(10)+CHAR(50)+CHAR(48)+CHAR(49)+CHAR(48)

***ALL encoding end***

All: ------------------------------
Payload Generation

- Payload generator:
  - Usernames
  - Credit Card numbers
  - Permutations
  - Character blocks
  - Ranges
  - Files
  - Pattern creator and regular expression (encoders)
Potential usernames:

Given 2 words will create combinations like: JOHN DOE = JOHDOE,JDOE,JOHNDOE,JOHND,JDOEJOHN,DOE, etc... Great for usernames lists
WebSlayer

Credit card type: VISA 13 Digits

Numbers:
5255855730075981
5157257720549951
5303634089378391
5519532744556445
5579887025546562
5256321792251293
540640593110222
5493155760187968
516627367217461
5240156999123526

Credit Cards numbers:
You can create valid credit card numbers for testing applications that require these kind of numbers, there are not valid credit card numbers, there are well formed numbers for each brand.

FINAL PAYLOAD:
- ace - 5166273673
- ace - 5240155991
- ace - 5255855730
- ace - 51572577205
- ace - 53036340893
- ace - 55195327445
- ace - 55798870255
- ace - 54064059311
- ace - 51662736721
- ace - 5240156999123526

Payload Creator

Pattern: @FPerm02@ - @PCred04@

Generate PAYLOAD

Results loaded

Add from file
Save file
Drop Payload
Delete selection
Demo

login page

http://test.acunetix.com/login.php

TEST and Demonstration site for Acunetix Web Vulnerability Scanner

home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo | Logout admin

search art

Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo
Logout

Links
Security art
Fractal Explorer

If you are already registered please enter your login information below:

Username: admin
Password: [input field]

login

You can also signup here.

About Us | Privacy Policy | Contact Us | ©2006 Acunetix Ltd
Advanced uses

Sweep an entire range with a common dictionary

HTTP://192.168.1.FUZZ/FUZ2Z

FUZZ: RANGE [1-254]

FUZ2Z: common.txt
Advanced uses

Scanning through proxies

```
me ----> Server w/proxy ---->LAN
```

```
wfuzz -x serverip:53 -c -z range -r 1-254 --hc XXX -t 5 http://10.10.1.FUZZ
```

-x set proxy
--hc is used to hide the XXX error code from the results, as machines w/o webserver will fail the request.
Future features

- Time delay between request
- Multiple proxies (distribute attack)
- Diagonal scanning (mix dictionaries)
Contact

- cmartorella__at_s21sec.com
- cmartorella_at_edge-security.com
- http://twitter.com/laramies
- http://laramies.blogspot.com
- http://www.edge-security.com
References

- http://projects.webappsec.org/Predictable-Resource-Location
- http://projects.webappsec.org/Credential-and-Session-Prediction
- http://projects.webappsec.org/Brute-Force
- http://gawker.com/5559346/
- http://nukeit.org/facebook-hack-access-any-users-photo-albums/