Web Application Firewall Bypassing – how to defeat the blue team

KHALIL BIJJOU
CYBER RISK SERVICES
DELOITTE

29th Octobre 2015
STRUCTURE

• Motivation & Objective
• Introduction to Web Application Firewalls
• Bypassing Methods and Techniques
• Approach for Penetration Testers
• The Tool WAFNinja
• Results
• Conclusion
Motivation & Objective
MOTIVATION AND THESIS OBJECTIVE (I)

MOTIVATION

• Number of deployed Web Application Firewalls (WAFs) is increasing

• WAFs make a penetration test more difficult

• Attempting to bypass a WAF is an important aspect of a penetration test
OBJECTIVE

Provide a practical approach for penetration testers which helps to ensure accurate results
Introduction to Web Application Firewalls
INTRODUCTION TO WEB APPLICATION FIREWALLS (I)

OVERVIEW

- Protects a web application by adding a security layer
- Stands between a user and a web server
- Understands HTTP traffic better than traditional firewalls
- Checks for malicious traffic and blocks it
INTRODUCTION TO WEB APPLICATION FIREWALLS (IV)
FUNCTIONALITY

- Pre-processor:
  Decide whether a request will be processed further

- Normalization:
  Standardize user input

- Validate Input:
  Check user input against policies
INTRODUCTION TO WEB APPLICATION FIREWALLS (V)
NORMALIZATION FUNCTIONS

• Simplifies the writing of rules
• No Knowledge about different forms of input needed

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>compressWhitespace</td>
<td>converts whitespace chars to spaces</td>
</tr>
<tr>
<td>hexDecode</td>
<td>decodes a hex-encoded string</td>
</tr>
<tr>
<td>lowercase</td>
<td>converts characters to lowercase</td>
</tr>
<tr>
<td>urlDecode</td>
<td>decodes a URL-encoded string</td>
</tr>
</tbody>
</table>
• Security Models define how to enforce policies
• Policies consist of regular expressions
• Three Security Models:
  1. Positive Security Model
  2. Negative Security Model
  3. Hybrid Security Model
### INTRODUCTION TO WEB APPLICATION FIREWALLS (VII)

**INPUT VALIDATION**

<table>
<thead>
<tr>
<th>Positive Security Model (Whitelist)</th>
<th>Negative Security Model (Blacklist)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deny all but known good</td>
<td>Allow all but known bad</td>
</tr>
<tr>
<td>Prevents Zero-day Exploits</td>
<td>Shipped with WAF</td>
</tr>
<tr>
<td>More secure than blacklist</td>
<td>Fast adoption</td>
</tr>
<tr>
<td>Comprehensive understanding of application is needed</td>
<td>Little knowledge needed</td>
</tr>
<tr>
<td>Creating policies is a time-consuming process</td>
<td>Protect several applications</td>
</tr>
<tr>
<td></td>
<td>Tends to false positives</td>
</tr>
<tr>
<td></td>
<td>Resource-consuming</td>
</tr>
</tbody>
</table>
Bypassing Methods and Techniques
**BYPASSING METHODS AND TECHNIQUES (I)**

**OVERVIEW**

**Pre-processor Exploitation:**
Make WAF skip input validation

**Impedance Mismatch:**
WAF interprets input differently than back end

**Rule Set Bypassing:**
Use Payloads that are not detected by the WAF
Pre-processor Exploitation
BYPASSING METHODS AND TECHNIQUES (II)
BYPASSING PARAMETER VERIFICATION

• PHP removes whitespaces from parameter names or transforms them into underscores

  http://www.website.com/products.php?%20productid=select 1,2,3

• ASP removes % character that is not followed by two hexadecimal digits

  http://www.website.com/products.aspx?%productid=select 1,2,3

• A WAF which does not reject unknown parameters may be bypassed with this technique.
BYPASSING METHODS AND TECHNIQUES (III)
PRE-PROCESSOR EXPLOITATION EXAMPLE

X-* Headers

• WAF may be configured to trust certain internal IP Addresses
• Input validation is not applied on requests originating from these IPs
• If WAF retrieves these IPs from headers which can be changed by a user a bypass may occur
• A user is in control of the following HTTP Headers:
  ▪ X-Originating-IP
  ▪ X-Forwarded-For
  ▪ X-Remote-IP
  ▪ X-Remote-Addr
BYPASSING METHODS AND TECHNIQUES (IV)
MALFORMED HTTP METHOD

• Misconfigured web servers may accept malformed HTTP methods

• A WAF that only inspects GET and POST requests may be bypassed
BYPASSING METHODS AND TECHNIQUES (V)

OVERLOADING THE WAF

• A WAF may be configured to skip input validation if performance load is heavy

• Often applies to embedded WAFs

• Great deal of malicious requests can be sent with the chance that the WAF will overload and skip some requests
Impedance Mismatch
BYPASSING METHODS AND TECHNIQUES (VI)
HTTP PARAMETER POLLUTION

- Sending a number of parameters with the same name
- Technologies interpret this request differently:

```plaintext
http://www.website.com/products/?productid=1&productid=2
```

differently:

<table>
<thead>
<tr>
<th>Back end</th>
<th>Behavior</th>
<th>Processed</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASP.NET</td>
<td>Concatenate with comma</td>
<td>productid=1,2</td>
</tr>
<tr>
<td>JSP</td>
<td>First Occurrence</td>
<td>productid=1</td>
</tr>
<tr>
<td>PHP</td>
<td>Last Occurrence</td>
<td>productid=2</td>
</tr>
</tbody>
</table>
The following payload can be divided:

- **WAF** sees two individual parameters and may not detect the payload
- **ASP.NET** back end concatenates both values

```
?productid=select 1,2,3 from table
```

```
?productid=select 1&productid=2,3 from table
```
BYPASSING METHODS AND TECHNIQUES (VIII)
HTTP PARAMETER FRAGMENTATION

• Splitting subsequent code between different parameters

• Example query:

```sql
sql = "SELECT * FROM table WHERE uid = "$_GET['uid']" and pid = "$_GET['pid']"
```

• The following request:

```http
http://www.website.com/index.php?uid=1+union/*&pid=*/select 1,2,3
```

would result in this SQL Query:

```sql
sql = "SELECT * FROM table WHERE uid = 1 union/* and pid = */select 1,2,3"
```
BYPASSING METHODS AND TECHNIQUES (IX)  
DOUBLE URL ENCODING

• WAF normalizes URL encoded characters into ASCII text
• The WAF may be configured to decode characters only once
• Double URL Encoding a payload may result in a bypass

's' -> %73 -> %25%37%33

• The following payload contains a double URL encoded character

1 union %25%37%33lect 1,2,3
Rule Set Bypassing
BYPASSING METHODS AND TECHNIQUES (X)

BYPASS RULE SET

• Two methods:
  ▪ Brute force by enumerating payloads
  ▪ Reverse-engineer the WAFs rule set
APPROACH FOR PENETRATION TESTERS
APPROACH FOR PENETRATION TESTERS (I)

OVERVIEW

• Similar to the phases of a penetration test
• Divided into six phases, whereas Phase 0 may not always be possible
APPROACH FOR PENETRATION TESTERS(II)

PHASE 0

Identifying vulnerabilities with a disabled WAF

Objective: find security flaws in the application more easily

- assessment of the security level of an application is more accurate
- Allows a more focused approach when the WAF is enabled
- May not be realizable in some penetration tests
APPROACH FOR PENETRATION TESTERS (III)

PHASE 1

Reconnaissance

Objective: Gather information to get a good overview of the target

• Basis for the subsequent phases

• Gather information about:
  ▪ web server
  ▪ programming language
  ▪ WAF & Security Model
  ▪ Internal IP Addresses
APPROACH FOR PENETRATION TESTERS (IV)
PHASE 2

Attacking the pre-processor

Objective: make the WAF skip input validation

• Identify which parts of a HTTP request are inspected by the WAF to develop an exploit:
  1. Send individual requests that differ in the location of a payload
  2. Observe which requests are blocked
  3. Attempt to develop an exploit
APPROACH FOR PENETRATION TESTERS (V)

PHASE 3

Attempting an impedance mismatch

Objective: make the WAF interpret a request differently than the back end and therefore not detecting it

• Knowledge about back end technologies is needed
APPROACH FOR PENETRATION TESTERS (VI)

PHASE 4

Bypassing the rule set

Objective: find a payload that is not blocked by the WAFs rule set

1. Brute force by sending different payloads

2. Reverse-engineer the rule set in a trial and error approach:
   1. Send symbols and keywords that may be useful to craft a payload
   2. Observe which are blocked
   3. Attempt to develop an exploit based on the results of the previous steps
Identifying miscellaneous vulnerabilities

Objective: find other vulnerabilities that can not be detected by the WAF

• Broken authentication mechanism
• Privilege escalation
APPROACH FOR PENETRATION TESTERS (VIII)
PHASE 6

Post assessment

Objective: Inform customer about the vulnerabilities

• Advise customer to fix the root cause of a vulnerability

• For the time being, the vulnerability should be virtually patched by adding specific rules to the WAF

• Explain that the WAF can help to mitigate a vulnerability, but can not thoroughly fix it
WAFNINJA (I)
OVERVIEW

• CLI Tool written in Python
• Automates parts of the approach
• Already used in several penetration tests
• Supports
  • HTTPS connections
  • GET and POST parameter
  • Usage of cookies
**WAFNINJA (II)**  
**MOST IMPORTANT FUNCTIONS**

<table>
<thead>
<tr>
<th><strong>Fuzz</strong></th>
<th><strong>Bypass</strong></th>
</tr>
</thead>
</table>
| • Reverse-engineer a WAFs rule set by sending different symbols and keywords  
• Analyzes the response of every request  
• Results are displayed in a clear and concise way  
• Fuzzing strings can be extended with the `insert-fuzz` function | • Brute forcing the WAF by enumerating payloads and sending them to the target  
• Analyzes the response of every request  
• Results are displayed in a clear and concise way  
• Payloads can be extended with the `insert-bypass` function |
RESULTS
RESULTS (I)

OVERVIEW

• Results of using WAFNinja to attempt to bypass three WAFs in a test environment
• Deployed WAFs used the standard configuration
• Two vulnerable web applications behind every WAF
RESULTS (II)
COMODO WAF

• Most intelligent rule set of the three tested WAFs
• SQL Injection payload found:
  
  0 union/**/select 1,version(),@@datadir

• Disclosure of sensitive information:

  SQLI DUMB SERIES-2
RESULTS (III)
MODSECURITY WAF

• Highly restrictive rule set
• SQL Injection payload found:

\[1+uni%0Bon+se%0Blect+1,2,3\]

but was not processed by the back end
RESULTS (IV)

AQTRONIX WEBKNIGHT WAF

• Most vulnerable rule set of all three WAFs

• SQL Injection payload found:

```
0 union(select 1,@@hostname,@@datadir)
```

• Disclosure of sensitive information:
RESULTS (V)
AQTRONIX WEBKNIGHT

• SQL Injection payload found:

```
0 union(select 1,username,password from(users))
```

• Disclosure of personal data:

```
Welcome   Dhakkan
Your Login name:Dumb
Your Password:Dumb
```
RESUL TS (VI)
AQTRONIX WEBK NIGH T

• XSS payload found:

```html
<img src=x onwheel=prompt(1)>
```

• “onwheel” replaced an old JavaScript event handler
CONCLUSION
CONCLUSION (I)

• Different Bypass Methods and Techniques have been gathered and categorized
• Based on these techniques a practical approach is described
• A tool which facilitates this approach was developed
• The tool’s results contributed to finding several bypasses
CONCLUSION (II)

• The given approach can improve the accuracy of penetration test results

• The listing of bypassing techniques can be used by vendors to improve their WAFs

• WAF vulnerabilities found were reported to the particular WAF vendors

• Ultimately: WAFs make exploiting vulnerabilities more difficult, but do not guarantee that a security breach will not happen
CONCLUSION (III)

WebKnight Downloads

Download WebKnight 4.3 (only for support contracts) Changelog

This is a feature release focused on improving our scanning engine and related bug fixes.

- Added a lot of new signatures to detect remote file inclusion and PHP exploits.
- Improved SQL injection scanning. Special thanks to Khalil Bijjou for reporting some bypasses and suggesting improvements.
- Forms Authentication scanning.
- Detect parameter pollution attacks.
- Added new XSS keywords for mobile devices, animations...
- Deny payloads (post data) for certain methods.
- Fixed mp3/mp4 files not playing in Chrome/IE.
- Fixed OnUrlMap race condition between IIS 8 and WebKnight.
- IIS Authentication notification can be disabled. This fixes the issue in KB 2605401.
CONCLUSION (III)

Transaktionsdetails

Zahlung erhalten (Transaktionscode)

Absender: (Der Absender dieser Zahlung ist Nicht-US-verifiziert.)
E-Mail-Adresse des Käufers:
Zahlung gesendet an:

Gesamtbetrag: €150,00 EUR
Gebühr: €60,00 EUR
Nettobetrag: €150,00 EUR

Rückzahlung senden
Innerhalb von 60 Tagen können Sie eine Rückzahlung senden.

Datum: 15. Okt 2015
Zeit: 23:01:58 MESZ
Status: Abgeschlossen

Betreff: Thank you for reporting WebKnight bypasses and suggesting improvements.
Zahlungsart: Sofort

Yay!
THANK YOU FOR YOUR ATTENTION!

E-Mail: kbijjou@deloitte.de
Xing: Khalil Bijjou