BÀI 9.

AN TOÀN DỊCH VỤ WEB(3)

QUẢN LÝ PHIÊN

Bùi Trọng Tùng,
Viện Công nghệ thông tin và Truyền thông,
Đại học Bách khoa Hà Nội

1. QUẢN LÝ PHIÊN

Bùi Trọng Tùng,
Viện Công nghệ thông tin và Truyền thông,
Đại học Bách khoa Hà Nội

Phiên(session) là gì?

- Một chuỗi các thông điệp HTTP Request và HTTP Response được trao đổi giữa một trình duyệt và website
- Thường kéo dài trong một khoảng thời gian nào đó
- Quản lý phiên:
  - Người dùng chỉ đăng nhập một lần
  - Các thông điệp HTTP Request được gửi tiếp theo gắn liền trạng thái đã xác thực tài khoản người dùng

  →trạng thái của phiên cần được lưu trữ tại client và server

  → ứng dụng điển hình của cookie

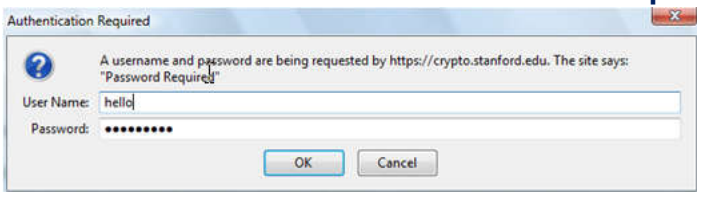
3

3

Sử dụng HTTP auth

- Sử dụng cơ chế HTTP auth
- HTTP request: GET /index.html
- HTTP response chứa:

WWW-Authenticate: Basic realm="Password Required"



- Các thông điệp HTTP Request sau đó chứa mã băm của mật khẩu

Authorization: Basic ZGFddfibzsdfgkjheczl1NXRleHQ=

4

4

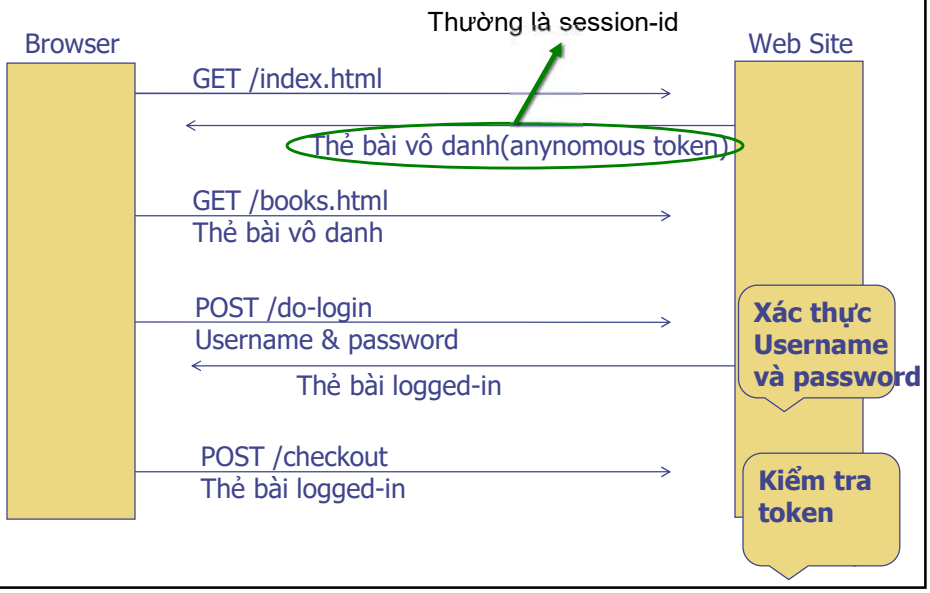
Hạn chế của HTTP auth

- Người dùng chỉ có thể đăng xuất bằng cách tắt cửa sổ trình duyệt.
- Thông báo có nội dung khó hiểu với người dùng
- Hộp thoại đăng nhập không thể tùy biến
- Trên các trình duyệt cũ: có thể đánh cắp cookie, mã băm của mật khẩu bằng cách lợi dụng HTTP TRACE Request
  - Hãy đọc thêm về lỗi cross-site tracing

5

5

Sử dụng thẻ bài (session token)



6

Lưu thẻ bài ở đâu?

- Trong cookie:

Set-Cookie: SessionToken=fduhye63sfdb

- Nhúng vào URL

https://site.com/checkout?SessionToken=kh7y3b

- Đặt trong thuộc tính ẩn

<input type="hidden" name="sessionid" value="kh7y3b">

- Đặt trong thuộc tính của DOM
- Hạn chế của mỗi phương pháp?

7

7

Lưu thẻ bài ở đâu?

- Trong cookie:

Mọi thông điệp HTTP Request gửi đi đều có giá trị thẻ bài → tấn công CSRF

- Nhúng vào URL

Lộ giá trị thẻ bài qua trường HTTP Referer

- Đặt trong thuộc tính ẩn

Chỉ áp dụng cho các phiên ngắn

- Đặt trong thuộc tính của DOM:

Lộ giá trị, chỉ áp dụng cho các phiên ngắn, không có tác dụng trên cửa sổ mới được mở ra

8

8

HTTP Referer

```
GET /wiki/John_Ousterhout HTTP/1.1
Host: en.wikipedia.org
Keep-Alive: 300
Connection: keep-alive
Referer: http://www.google.com/search?q=john+ousterhout&ie=utf-8&oe
```

- Trường Referer có thể làm lộ cookie cho bên thứ 3
- Che giấu cookie khi chuyển từ trang HTTPS sang trang sử dụng HTTP:
  - HTML5: <a **rel="noreferrer"** href=www.example.com>

9

9

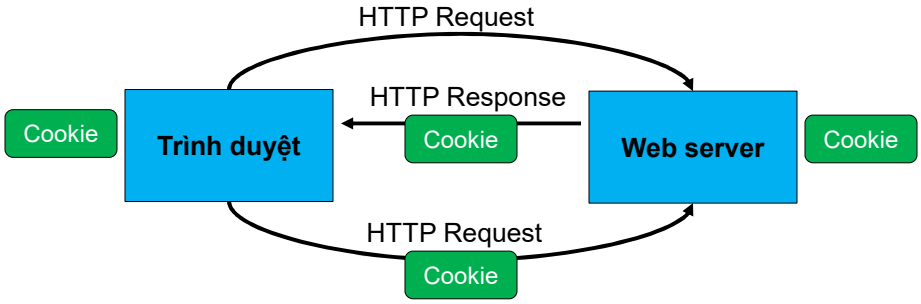
2. ỨNG DỤNG COOKIE TRONG QUẢN LÝ PHIÊN

Bùi Trọng Tùng,
Viện Công nghệ thông tin và Truyền thông,
Đại học Bách khoa Hà Nội

10

10

HTTP Cookie



- Cookie: dữ liệu do Web server tạo ra, chứa thông tin trạng thái của phiên làm việc
  - Server có thể lưu lại cookie(một phần hoặc toàn bộ)
- Sau khi xử lý yêu cầu, Web server trả lại thông điệp HTTP Response với coookie đính kèm
  - Set-Cookie: key = value; options;
- Trình duyệt lưu cookie
- Trình duyệt gửi HTTP Request tiếp theo với cookie được đính kèm

11

11

Các vấn đề bảo mật của cookie

- Server
  - Không đọc được một số thuộc tính của cookie
  - Không "nhớ" cookie được thiết lập cho scope nào
  - Không kiểm tra được tính toàn vẹn của cookie
- Client: có thể đọc, thiết lập tùy ý
  - Firefox: cookies.sqlite
- Cookie có thể bị thay đổi khi truyền:
  - Firefox add-on: TamperData
  - Web proxy: Burp suite, ZAP…
- Cookie có thể bị đánh cắp

12

12

Ví dụ 1:

- Alice đăng nhập trên trang login.site.com
  - Một cookie được thiết lập với *session-id* cho site.com
  - Lưu ý: cookie này được sử dụng cho mọi trang có tên miền đuôi site.com
- Alice truy cập vào một trang bị chèn mã độc evil.site.com
  - Ghi đè cookie trên với user là attacker
- Alice truy cập vào other.site.com
  - Nguy cơ?
- Nguyên nhân?

13

13

Ví dụ 2: HTTPS cookie

- Alice đăng nhập tại **https**://www.google.com/accounts

set-cookie: **SSID**=A7_ESAgDpKYk5TGnf; Domain=.google.com; Path=/ ;
Expires=Wed, 09-Mar-2026 18:35:11 GMT; **Secure; HttpOnly**
set-cookie: **SAPISID**=wj1gYKLFy-RmWybP/ANtKMtPIHNambvdI4; Domain=.google.com;
Path=/ ; Expires=Wed, 09-Mar-2026 18:35:11 GMT; **Secure**

- Alice truy cập **http**://www.google.com
  - HTTP Response có thể bị chèn cookie như sau:

    Set-Cookie: SSID=attacker; secure
  - →HTTPS cookie vẫn có thể bị ghi đè
  - →HTTPS không đảm bảo tính toàn vẹn cho cookie

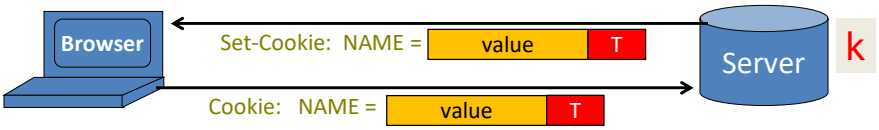
14

14

Xác thực cookie

- Server sử dụng khóa bí mật K, không chia sẻ

Sinh tag: T ← $HMAC_{sign}$ (K, SID ll name ll value)



Verify tag: $HMAC_{verify}$ (k, SID ll name ll value,)

15

15

Ví dụ: ASP .Net

- Thiết lập khóa bí mật:

System.Web.Configuration.MachineKey

- Tạo và mã hóa-xác thực cookie

```
HttpCookie   cookie = new HttpCookie(name, val);
HttpCookie   encodedCookie =
                    HttpSecureCookie.Encode (cookie);
```

- Giải mã và kiểm tra

HttpSecureCookie.Decode (cookie);

16

16

Session Hijacking

- Kẻ tấn công đánh cắp Session Token của người dùng và chiếm (hijack) phiên làm việc

→gửi yêu cầu mạo danh người dùng

- Session Token trong cookie có thể bị bộc lộ như thế nào?
  - Chặn bắt gói tin nếu kênh truyền không được mã hóa
  - Tấn công đầu độc DNS khiến trình duyệt truy cập vào website giả mạo
  - Đoán nhận nếu token không được sinh ngẫu nhiên
  - Trình duyệt/máy chủ bị tấn công .Ví dụ: tấn công XSS, tấn công bằng phần mềm mã độc

17

17

Session Hijacking – Ví dụ

- Tấn công vào lỗ hổng của Twitter năm 2013
- Twitter sử dụng auth_token để xác thực:
  - Được sinh từ username và password
- Lỗ hổng:
  - auth_token không thay đổi
  - auth_token không bị xóa khi người dùng đăng xuất
  - →Khi kẻ tấn công đánh cắp, auth_token có thể sử dụng nhiều lần cho tới khi mật khẩu của người dùng bị thay đổi

18

18

Tấn công Session fixation

1. Kẻ tấn công truy cập vào site.com và nhận được thẻ bài vô danh (anonymous token)
2. Nhúng thẻ bài vào địa chỉ URL trên một trang của evil.com
3. Người dùng đăng nhập vào site.com qua URL trên evil.com sẽ nhận được thẻ bài logged-in

   logged-in token = Generate(anonymous token)
4. Kẻ tấn công tính toán thẻ bài logged-in và thực thi các phiên giả mạo
   - Có thể lợi dụng lỗ hổng web server không đánh dấu thẻ bài hết hiệu lực khi người dùng đăng xuất

19

19

Phòng chống tấn công Session Hijacking

- Ngăn chặn đoán nhận cookie: sử dụng giá trị ngẫu nhiên có kích thước đủ lớn
- Ngăn chặn đánh cắp cookie
- Yêu cầu xác thực lại
- Sử dụng xác thực đa yếu tố
- Quản lý session token:
  - Sử dụng session token mới cho phiên mới
  - Sử dụng time-out cho session token. Làm mới session token khi time-out
  - Xóa session-token khi người dùng đăng xuất
- Kiểm tra dựa trên địa chỉ IP, thông tin thiết bị, phần mềm… chỉ là giải pháp hỗ trợ

20

20

3. TẤN CÔNG CSRF

Bùi Trọng Tùng,
Viện Công nghệ thông tin và Truyền thông,
Đại học Bách khoa Hà Nội

21

21

Cookie của bên thứ 3(third-party)

- Lợi dụng hiệu ứng bên (side effect) để tạo ra các HTTP Request trái phép.
  - Hiệu ứng bên (side effect): trong quá trình hiển thị (render) trang web, trình duyệt có thể tự động truy cập đến tài nguyên những liên kết khác mà không được kiểm soát.
- Giả sử trình duyệt (1st party) truy cập vào site A (2nd party).
- Nếu trên site A có địa chỉ URL của một tài nguyên nằm trên site B (3rd party), một thông điệp HTTP Request cho địa chỉ URL sẽ được phát đi với cookie của site B
- Lỗ hổng CSRF: Cú pháp của yêu cầu là đoán trước được

22

22

Kịch bản tấn công CSRF

23

23

Tấn công CSRF – Ví dụ

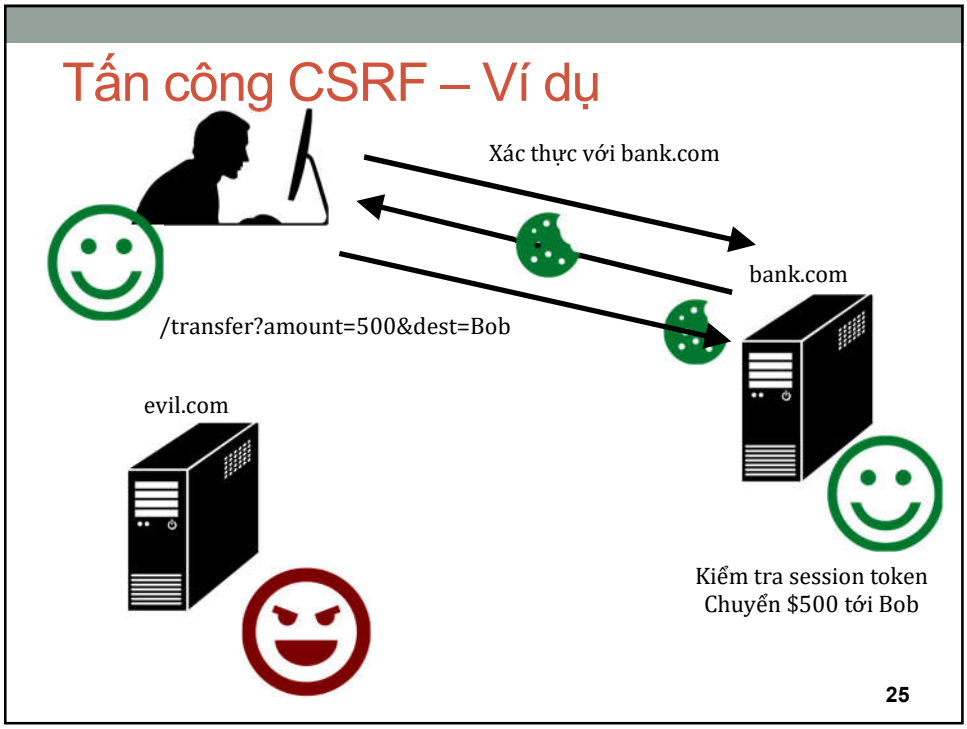
- Người dùng đăng nhập trên website bank.com và thực hiện một giao dịch chuyển tiền
  - Trình duyệt lưu cookie của phiên làm việc(gồm các thông tin người dùng đã xác thực với ngân hàng)
- Người dùng truy cập vào một trang web chứa đoạn mã độc(Ví dụ attacker.com):

```
<form  name=F  action=http://bank.com/BillPay.php>
<input  name=recipient   value=attacker> …
<script> document.F.submit(); </script>
```

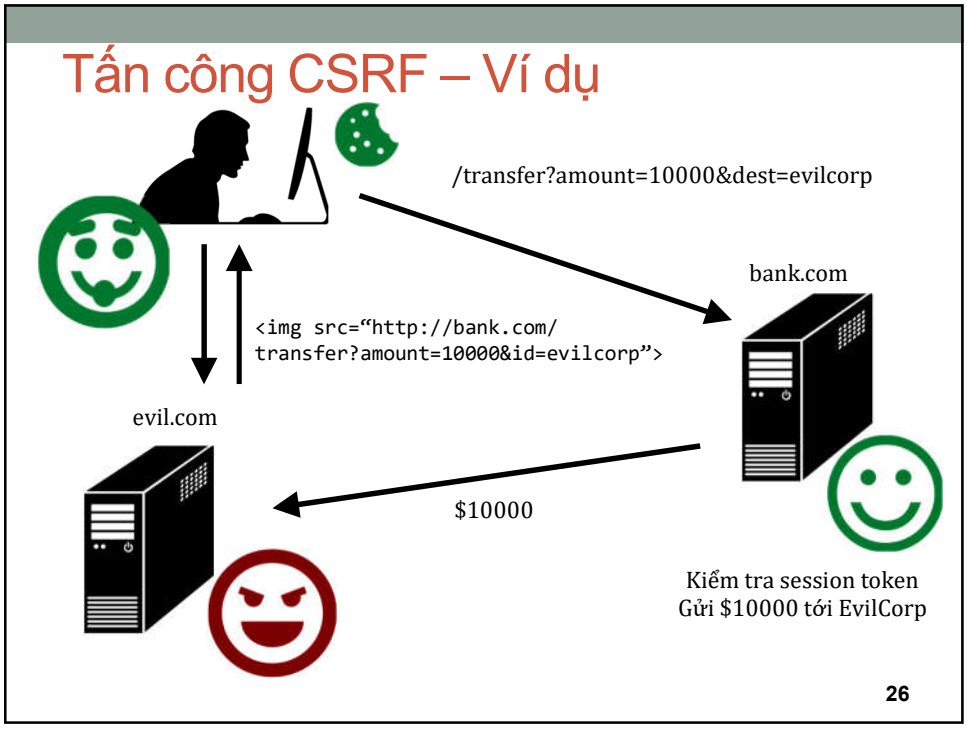
- Khi hiển thị trang web, trình duyệt tạo một HTTP Request với cookie ở trên để thực thi giao dịch chuyển tiền cho attacker

24

24


Tấn công CSRF – Ví dụ
Xác thực với bank.com
bank.com
/transfer?amount=500&dest=Bob
evil.com
Kiểm tra session token
Chuyển $500 tới Bob
25

25


Tấn công CSRF – Ví dụ
/transfer?amount=10000&dest=evilcorp
bank.com
<img src="http://bank.com/
transfer?amount=10000&id=evilcorp">
evil.com
$10000
Kiểm tra session token
Gửi $10000 tới EvilCorp
26

26

Tấn công CSRF – Ví dụ

- "Drive-By Pharming", Sid Stamm, Zulfikar Ramzan, Markus Jakobsson, ICICS'07
- Các thiết bị router phục vụ cho gia đình/văn phòng nhỏ cho phép cấu hình qua giao diện Web
  - Người dùng cần đăng nhập. Tuy nhiên hầu hết dùng tài khoản mặc định của nhà sản xuất
- Tấn công Drive-by Pharming:
  - Người dùng truy cập vào website chứa mã độc
  - Một Javascript quét và tìm địa chỉ router
  - Sử dụng Javascript để phát hiện hãng sản xuất và tên sản phẩm (thường thể hiện qua logo)
  - Login và sửa thông số cấu hình qua phương thức GET trên URL

27

27

Phòng chống tấn công CSRF

- Kiểm tra trường Referer


```
Referer: http://www.facebook.com/home.php
```

- Kiểm tra giá trị ẩn:


```
<input type=hidden value=23a3af01b>
```

- Sử dụng xác thực đa yếu tố, CAPTCHA
- Sử dụng thuộc tính SameSite

28

28

Kiểm tra trường Referer

- Trường Referer ghi nhận địa chỉ phát sinh thông điệp HTTP Request. Ví dụ:

```
Referer: http://bank.com/
Referer: http://evil.com/index.html
```


- Hạn chế:
  - Lộ thông tin cá nhân, lịch sử duyệt Web của người dùng
  - Trường Referer có thể bị loại bỏ theo nhiều cách(chính sách của firewall, trình duyệt) → không thể quyết định chặn hoặc không chặn
  - Không giữ địa chỉ gốc khi chuyển hướng (redirect)
- Giải pháp: sử dụng trường Origin

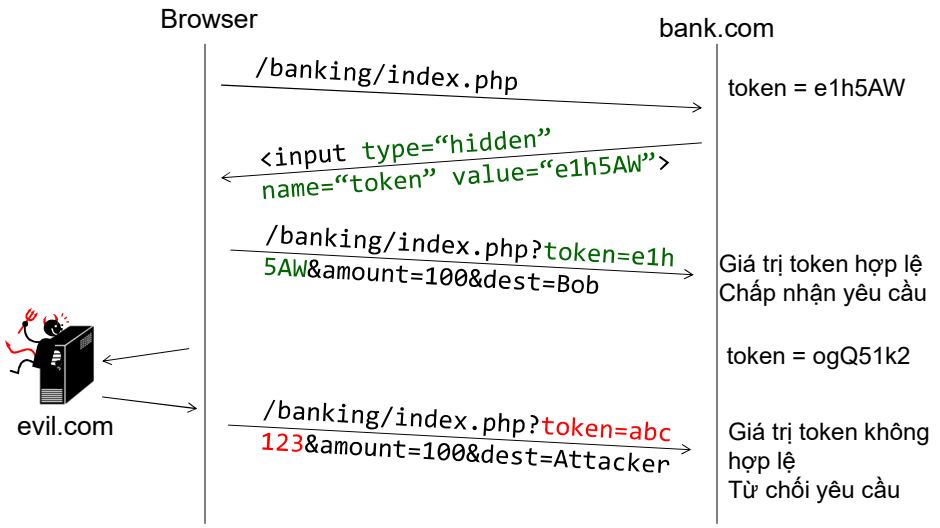
Thuộc tính SameSite

- Thuộc tính của cookie: SameSite = lax | strict
  - strict: không gửi kèm cookie cùng bất kỳ HTTP Request nào
  - lax: chỉ gửi kèm cookie với các thông điệp HTTP Requets có phương thức GET và phát sinh do việc chuyển hướng truy cập(thay đổi địa chỉ trên thanh địa chỉ của trình duyệt)

request type	example code	cookies sent
link	<a href="…">	normal, lax
prerender	<link rel="prerender" href="…">	normal, lax
form get	<form method="get" action="…">	normal, lax
form post	<form method="post" action="…">	normal
iframe	<iframe src="…">	normal
ajax	$.get('…')	normal
image	<img src="…">	normal

https://www.sjoerdlangkemper.nl/2016/04/14/preventing-csrf-with-samesite-cookie-attribute/

Kiểm tra giá trị ẩn



31

31

Phòng chống tấn công CSRF phía server

- https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet
- https://www.owasp.org/index.php/Reviewing_code_for_Cross-Site_Request_Forgery_issues

32

32

Bài giảng sử dụng một số hình vẽ và ví dụ từ các bài giảng:

- Computer and Network Security, Stanford University
- Computer Security, Berkeley University

33

33