BÀI 8.

AN TOÀN DỊCH VỤ WEB
QUẢN LÝ PHIÊN

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

1. COOKIE

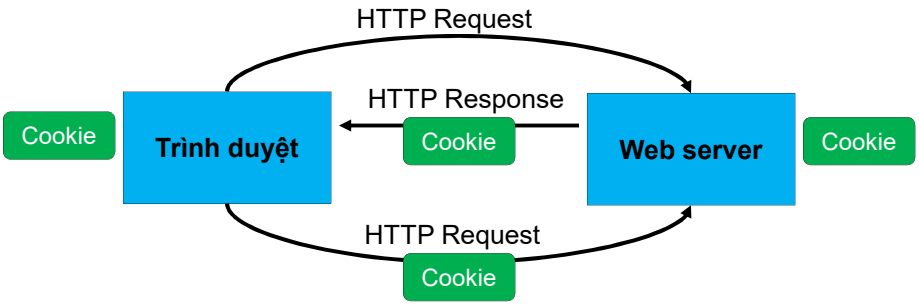
Bùi Trọng Tùng,
Viện Công nghệ thông tin và Truyền thông,
Đại học Bách khoa Hà Nội

HTTP là giao thức stateless

- Một phiên hoạt động của HTTP:
  - Trình duyệt kết nối với Web server
  - Trình duyệt gửi thông điệp yêu cầu HTTP Request
  - Web server đáp ứng với một thông điệp HTTP Response
  - …lặp lại…
  - Trình duyệt ngắt kết nối
- Các thông điệp HTTP Request được xử lý độc lập
- Web server không ghi nhớ trạng thái của phiên HTTP

3

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

4

HTTP Cookie - Ví dụ

HTTP Response

```
HTTP/1.1 200 OK
Server: nginx/1.10.1
Date: Mon, 14 Nov 2016 09:19:19 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
X-Powered-By: PHP/5.4.45
Set-Cookie: vflastvisit=1479115159; expires=Tue, 14-Nov-2017 09:19:19 GMT; path=/; domain=vozforums.com; secure
Set-Cookie: vflastactivity=0; expires=Tue, 14-Nov-2017 09:19:19 GMT; path=/; domain=vozforums.com; secure
Expires: 0
Cache-Control: private, post-check=0, pre-check=0, max-age=0
Pragma: no-cache
Content-Encoding: gzip
```

5

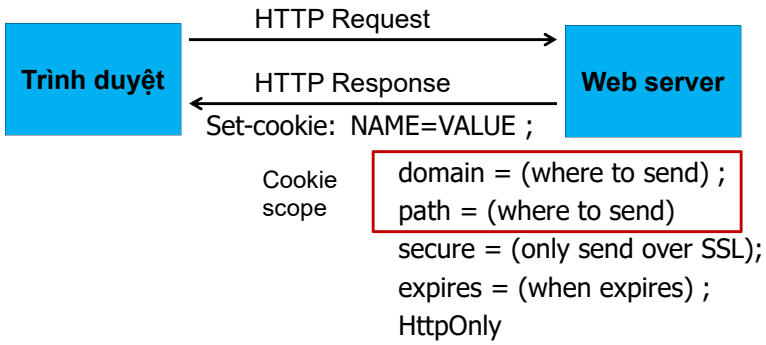
HTTP Cookie - Ví dụ

- HTTP Request

```
GET /clientscript/vbulletin_important.css?v=380 HTTP/1.1
Host: vozforums.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:49.0) Gecko/20100101 Firefox/49.0
Accept: text/css,*/*;q=0.1
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://vozforums.com/
Cookie: vflastvisit=1479115159; vflastactivity=0
Connection: keep-alive
```

6

HTTP Cookie



- Cookie scope: chỉ định các trang web sẽ gửi cookie tới
- HttpOnly: không thể đọc cookie bằng Javascript tại client

7

Chính sách SOP cho cookie

- Địa chỉ URL: scheme://domain:port/path?params
- Nguồn(origin) của cookie được xác định bởi: domain, path và scheme(không bắt buộc)
- Thiết lập cookie: một trang web có thể thiết lập cookie cho các trang có cùng tên miền, hoặc mang tên miền cấp trên(trừ tên miền cấp 1)
- Ví dụ: trang Web có domain là login.site.com:
  - Thiết lập được cookie với domain = login.site.com, site.com
  - Không thiết lập được với domain = othersite.com, other.site.com, .com
  - path: bất kỳ giá trị nào

8

Chính sách SOP cho cookie

- Đọc cookie: Server có thể đọc được tất cả cookie trong scope của nó

→Trình duyệt gửi tất cả cookie trong scope(domain và path) tới server:

  - Nếu giá trị secure được thiết lập thì cookie chỉ được gửi nếu giao thức là HTTPS

- Ví dụ: cookie với domain = example.com và path = /some/path/ sẽ được đính kèm vào thông điệp HTTP Request tới địa chỉ

http://foo.example.com/some/path/subdirectory/hello.html

9

SOP cho cookie – Ví dụ khác

- Hai cookie được thiết lập bởi login.site.com

cookie 1	cookie 2
userid = u1	userid = u2
domain = login.site.com	domain = .site.com
path = /	path = /
secure	

- Cookie được đặt trong HTTP Request như sau:

http://checkout.site.com/	cookie: userid=u2
http://login.site.com/	cookie: userid=u2
https://login.site.com/	cookie: userid=u1; userid=u2

10

Cookie của bên thứ 3(third-party)

- 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 (nếu có)

→ cơ sở để tấn công CSRF

- Phòng chống: sử dụng thuộc tính SameSite = lax | strict cho cookie
  - 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)
  - Hỗ trợ trên Chrome 51 và Opera 39 trở đi

11

SameSite cookie – Ví dụ

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/

12

Đọc ghi cookie tại trình duyệt

- Truy cập qua đối tượng DOM: document.cookie
- Thiết lập giá trị:

  document.cookie = "name=value; expires=…; "
- Hiển trị cookie: alert(document.cookie)
  - Hiển thị dưới dạng 1 chuỗi gồm giá trị trong các thuộc tính của tất cả cookie đã lưu cho tài nguyên này
- Xóa cookie:

  document.cookie = "name=; expires= *[Ngày trong quá khứ]* "

13

CÁC LỖ HỔNG CỦA COOKIE

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

14

Các lỗ hổng khi sử dụng 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 phát lại

15

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?

16

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:

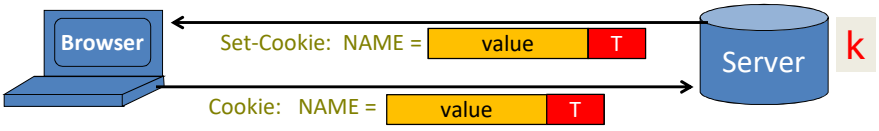
    Set-Cookie: SSID=attacker; secure
  - →HTTPS cookie vẫn có thể bị ghi đè
  - →HTTPS không đảm bảo tính toàn vẹn cho cookie

17

Giải pháp

- 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,)

- Để chống tấn công phát lại: sử dụng session-id
- Chống tráo đổi với cookie của phiên làm việc khác: sử dụng địa chỉ IP

18

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);

19

2. QUẢN LÝ PHIÊN

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

20

Phiên(session) là gì?

- Một chuỗi các thông điệp HTTP Request và HTTP Response được trao đổi giữa một trình duyệt và một hay nhiều 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

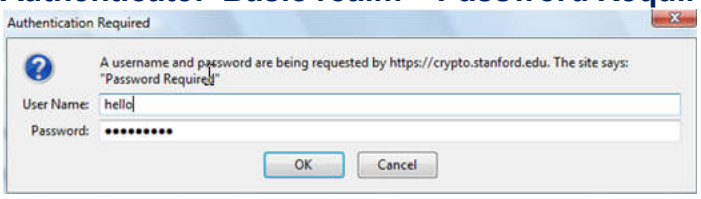
  → ứng dụng điển hình của cookie

21

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=

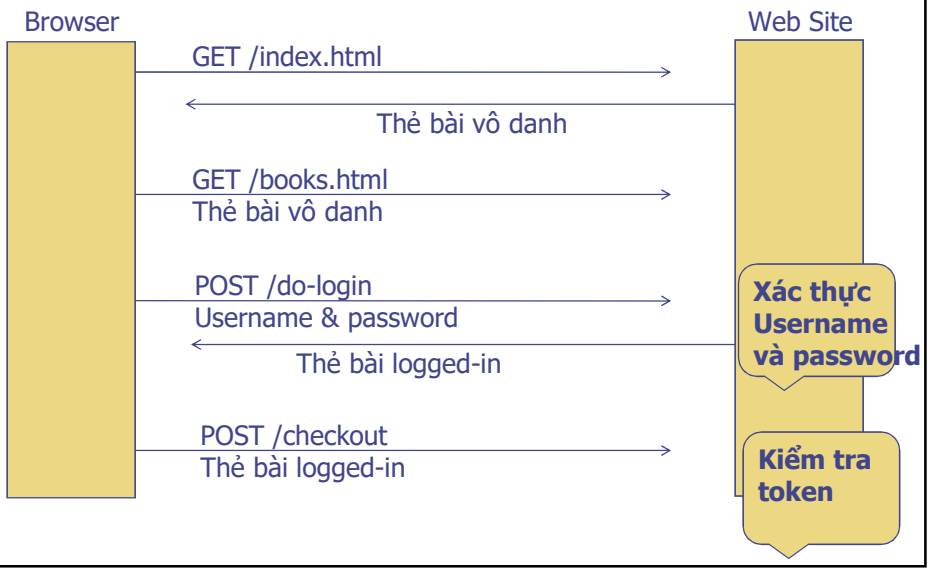
22

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

23

Sử dụng thẻ bài (session token)



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?

25

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

26

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>

27

Xử lý đăng xuất

- Ứng dụng phải cung cấp chức năng đăng xuất:
  - Kết thúc phiên hiện tại
  - Cho phép người dùng đăng nhập với tài khoản khác
  - Ngăn cản người dùng khác sử dụng phiên trái phép
- Xử lý khi đăng xuất:
  1. Xóa Session Token tại client(sử dụng Expire cho cookie)
  2. Xóa/đánh dấu Session Token đã hết hạn tại server

  →Nhiều website không thực hiện (2)

  →Nguy cơ?

28

Session Hijacking

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

→gửi yêu cầu mạo danh người dùng

- Ví dụ: FireSheep
  - Add-on trên Firefox cho phép đánh cắp Session Token trên Facebook qua mạng WiFi
  - Giải pháp: sử dụng HTTPS
- Các kỹ thuật khác:
  - XSS
  - Lợi dụng giá trị thẻ bài không được sinh ngẫu nhiên

29

Phòng chống

- Sinh thẻ bài ngẫu nhiên: sử dụng API được cung cấp bởi framework

Rails: token = MD5(current time, random nonce)

- Sử dụng địa chỉ IP để sinh thẻ bài
- Sử dụng thông tin khác của client: trình duyệt, thiết bị…
- Sử dụng SSL session ID

30

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
4. Kẻ tấn công ăn cắp thẻ bài logged-in (thường dùng tấn công XSS) 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

- Phòng chống:
  - Xác thực đa yếu tố
  - Sử dụng session token mới cho mỗi yêu cầu

31

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

32