

CMSC414 Computer and Network Security

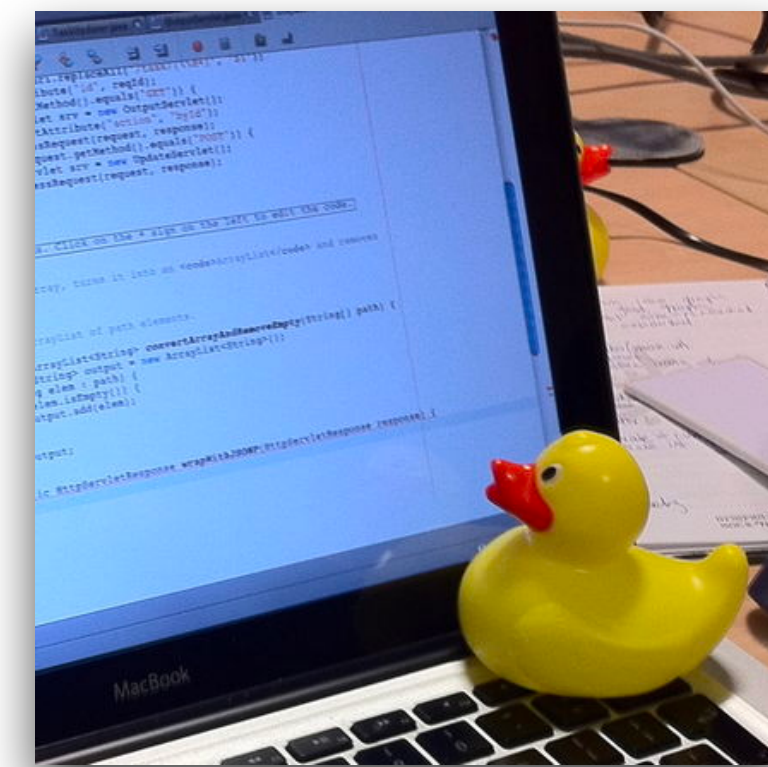
Cookies and CSRF

Yizheng Chen | University of Maryland
surrealyz.github.io

Feb 15, 2024

Project 1

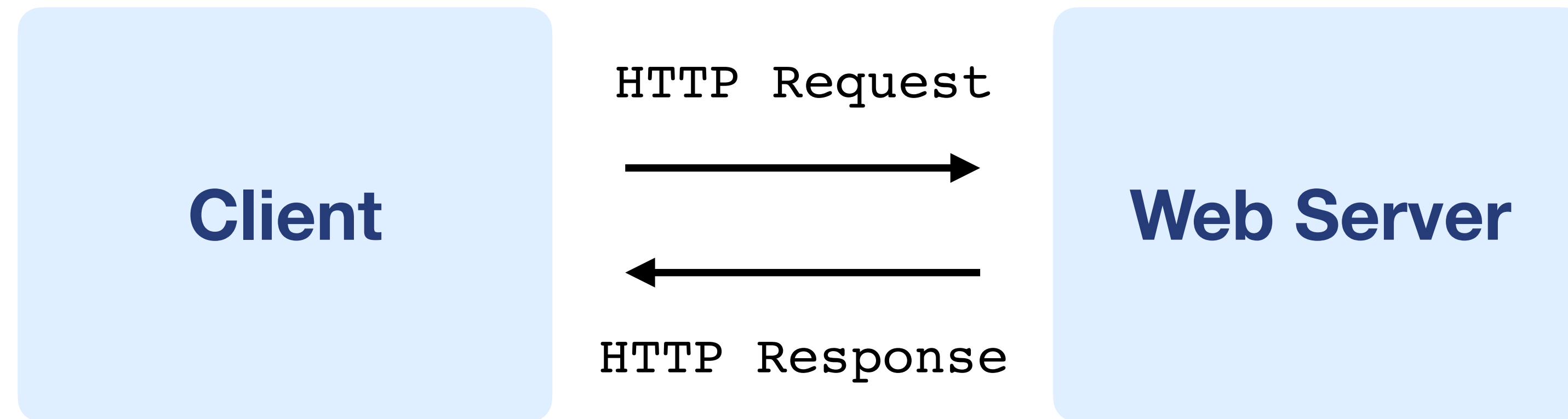
- Think through everything step by step, e.g.,
 - 1. Enumerate all variables**
 2. What kind of variables are they? Local? Static? Where do they belong?
 3. Check the size of all variables
 4. How are they used?
 5. Try rubber-duck debugging:
 - explain the code line by line



Agenda

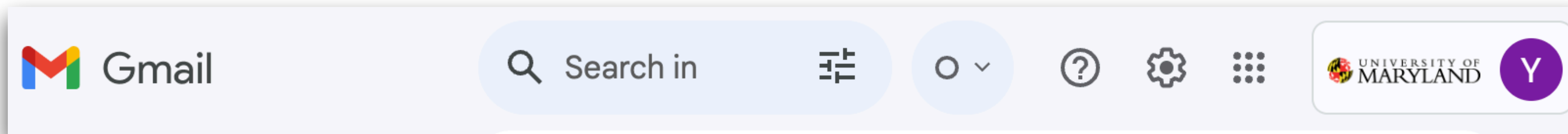
- Cookies
- Cross-Site Request Forgery (CSRF)

HTTP and HTTPS: Stateless Protocol



- Each request and response are independent of other requests and responses
- But, many features on the web requires some state...

Why do we need state?



- Shopping cart
- Account log in
- Website dark mode
- ...

HTTP Cookies

If we have something to represent the state:



Origin of the name

The term *cookie* was coined by web-browser programmer [Lou Montulli](#) . It was derived from the term *magic cookie* , which is a packet of data a program receives and sends back unchanged, used by [Unix](#) programmers. [\[6\]](#) [\[7\]](#)

https://en.wikipedia.org/wiki/HTTP_cookie

HTTP Cookies

First request, no state

Client

HTTP Request



Web Server

Client



HTTP Response



Web Server



Server stores state, indexes it with a



Send it back.
Client stores it.

HTTP Cookies

New requests
with



Client

HTTP Request



Web Server

Client

HTTP Response



Web Server



Use



to personalize
content

Cookies are key-value pairs

Set-Cookie: **key**=**value**; **options**;

HTTP/1.1 200 OK

Date: Tue, 18 Feb 2014 08:20:34 GMT

Server: Apache

Set-Cookie: session-zdnet-production=6bhqca1i0cbciagu11sisac2p3; path=

Set-Cookie: zdregion=MTI5LjluMTI5LjE1Mzp1czp1czpjZDJmNWY5YTdkODU=

Set-Cookie: zdregion=MTI5LjluMTI5LjE1Mzp1czp1czpjZDJmNWY5YTdkODU=

Set-Cookie: **edition=us** **expires=Wed, 18-Feb-2015 08:20:34 GMT;** **path=/**

Set-Cookie: session-zdnet-production=59ob97fpinqe4bg6lde4dvvq11; pat

Set-Cookie: user_agent=desktop

Set-Cookie: zdnet_ad_session=f

Set-Cookie: firstpg=0

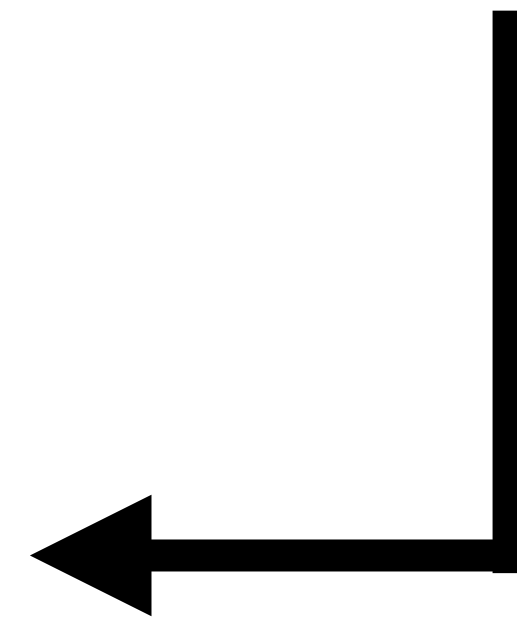
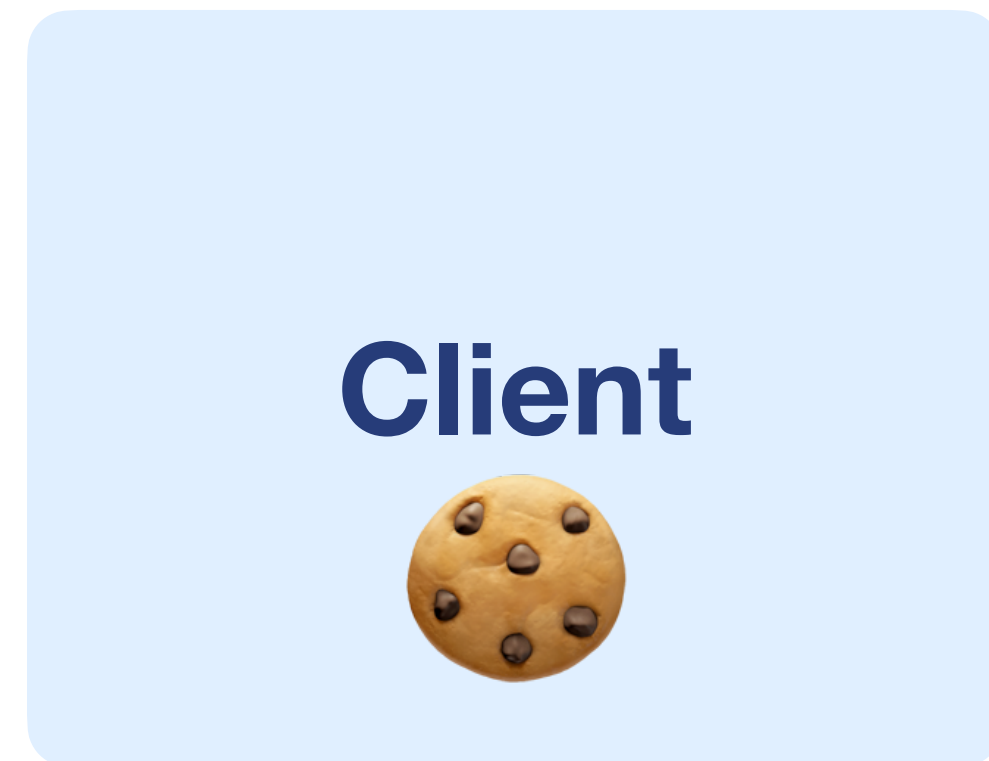
Expires: Thu, 19 Nov 1981 08:52:00 GMT

Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-ch

Server creates a cookie by including a **Set-Cookie** header in its response

Cookie Attributes

Set-Cookie: **edition=us** **expires**=Wed, 18-Feb-2015 08:20:34 GMT; **path**=/; **domain**=.zdnet.com



Semantics

- **Key value:** Store “us” under the key “edition”
- **Expires:** This value expires on Wed, Feb 18, 2015...
- **Path:** This should be available to any resource within a subdirectory of /
- **Domain:** This value should only be readable by any domain ending in .zdnet.com
- **Send the cookie to any future requests to <domain>/<path>**

Cookie Setting Policy

- The browser sends a cookie to a given URL if the cookie's **Domain** attribute is a domain-suffix of the URL domain, and the cookie's **Path** attribute is a prefix of the URL path
- For example, a cookie with Domain=`example.com` and Path=`/some/path` will be included on a request to `http://foo.example.com/some/path/index.html`
 - The URL domain ends in the cookie domain
 - The URL path begins with the cookie path.

Requests with cookies

Server creates a cookie by including a **Set-Cookie** header in its response

Response

```
HTTP/1.1 200 OK
Date: Tue, 18 Feb 2014 08:20:34 GMT
Server: Apache
Set-Cookie: session-zdnet-production=6bhqca1i0cbciagu11sisac2p3; path=/; domain=zdnet.com
Set-Cookie: zdregion=MTI5LjluMTI5LjE1Mzp1czp1czpjZDJmNWY5YTdkODU1N2Q2YzM5NGU3M2Y1ZTRmN0
Set-Cookie: zdregion=MTI5LjluMTI5LjE1Mzp1czp1czpjZDJmNWY5YTdkODU1N2Q2YzM5NGU3M2Y1ZTRmN0
Set-Cookie: edition=us; expires=Wed, 18-Feb-2015 08:20:34 GMT; path=/; domain=.zdnet.com
Set-Cookie: session-zdnet-production=59ob97fpinqe4bg6lde4dvvq11; path=/; domain=zdnet.com
```



Subsequent visit

Client sends requests with the same cookies

```
HTTP Headers
http://zdnet.com/

GET / HTTP/1.1
Host: zdnet.com
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.11) Gecko/20101013 Ubuntu/9.04 (jaunty) Firefox/3.6.11
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
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Cookie: session-zdnet-production=59ob97fpinqe4bg6lde4dvvq11; zdregion=MTI5LjluMTI5LjE1Mzp1czp1czpjZDJmNWY5YTdkODU1N2Q2YzM5NGU3M2Y1ZTRmN0
```

Cookies Allow Personalized Content



- Shopping cart
- Account log in
- Website dark mode
- ...

Cookies Allow Behavior Tracking

- **Tracking users**
 - Advertisers want to know your behavior
 - Ideally build a profile *across different websites*
 - Read about iPad on CNN, then see ads on Amazon?!
 - How can an advertiser (A) know what you did on another site (S)?

S shows you an ad from A; A scrapes the referrer URL

Option 1: A maintains a DB, indexed by your IP address

Option 2: A maintains a DB indexed by a *cookie*

Problem: IP addrs change

- **“Third-party cookie”**
- **Commonly used by large ad networks (doubleclick)**

Example: Ad Network Tracks User Behavior

The image shows a screenshot of the Reddit homepage. At the top, there is a navigation bar with the Reddit logo and various subreddit categories like 'FRONT', 'ALL', 'RANDOM', etc. Below the navigation bar, there is a search bar and a login/sign-up section. The main content area displays a list of posts, each with a rank, score, title, and submission details. The first post is 'They should put a tiny message at the end of chapstick tubes congratulating you for not losing the damn thing.' with a score of 4615. The second post is 'Meet Bidy, The Traveling Hedgehog' with a score of 5533. The third post is 'Mt. Fuji overlooking Yokohama' with a score of 4808. The fourth post is 'RIP in peace' with a score of 3365. The fifth post is '[Image]Stop Letting People' with a score of 2344. The sixth post is 'Hacker Claims Feds Hit Him With 44 Felonies When He Refused to Be an FBI Spy' with a score of 3567. On the right side of the page, there are buttons for 'Submit a new link' and 'Submit a new text post'. At the bottom right, there is a large advertisement for a 'GIF TOURNAMENT BATTLE #3' featuring a silhouette of the Reddit mascot and a trophy. Below the ad, there is a link to 'discuss this ad on reddit'.

Ad provided by
an ad network

Snippet of reddit.com source

```
- <div class="side">
  + <div class="spacer">
  + <div class="spacer">
  + <div class="spacer">
  + <div class="spacer">
  + <div class="spacer">
  - <div class="spacer">
```

Our first time accessing adzerk.net

```
- <iframe id="ad_main" scrolling="no" frameborder="0" src="http://static.adzerk.net
  /reddit/ads.html?sr=-reddit.com,loggedout&bust2#http://www.reddit.com" name="ad_main">
  - <html>
    - <head>
      + <style>
      + <script type="text/javascript" async="" src="http://engine.adzerk.net
        /ados?t=1424367472275&request={"Placements":
        [{"A":5146,"S":24950,"D":"main","AT":5},
        {"A":5146,"S":24950,"D":"sponsorship","AT":8}], "Keywords": "-reddit.com%2Clogg
        %3A%2F%2Fwww.reddit.com%2F", "IsAsync":true, "WriteResults":true}">
      + <script src="//ajax.googleapis.com/ajax/libs/jquery/1.7.1
        /jquery.min.js" type="text/javascript">
      + <script src="//secure.adzerk.net/ados.js?q=43" type="text/javascript">
      + <script type="text/javascript">
      + <script type="text/javascript">
      + <script type="text/javascript" src="http://static.adzerk.net/Extensions
        /adFeedback.js">
      + <link rel="stylesheet" href="http://static.adzerk.net/Extensions
        /adFeedback.css">
    </head>
```


The user visited reddit.com

HTTP Get Request to Fetch an Ad

```
HTTP Headers
http://static.adzerk.net/reddit/ads.html?sr=-reddit.com,loggedout&bust2#http://www.reddit.com

GET /reddit/ads.html?sr=-reddit.com,loggedout&bust2 HTTP/1.1
Host: static.adzerk.net
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.11) Gecko/20101013 Ubuntu/9.04 (jaunty) Firefox/3.6.11
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
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Referer: http://www.reddit.com/

HTTP/1.1 200 OK
Date: Thu, 19 Feb 2015 17:37:51 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: keep-alive
Set-Cookie: __cfduid=dc3a93cd30ca47b76600d63cde283e9b81424367471; expires=Fri, 19-Feb-16 17:37:51 GMT; path=/; domain=.adzerk.net...
```

We are only sharing this cookie with *.adzerk.net; but we are telling them about where we just came from

Later, the user went to reddit.com/r/security

Another HTTP Get Request to Fetch an Ad

```
HTTP Headers
http://static.adzerk.net/reddit/ads.html?sr=security,loggedout&bust2#http://www.reddit.com

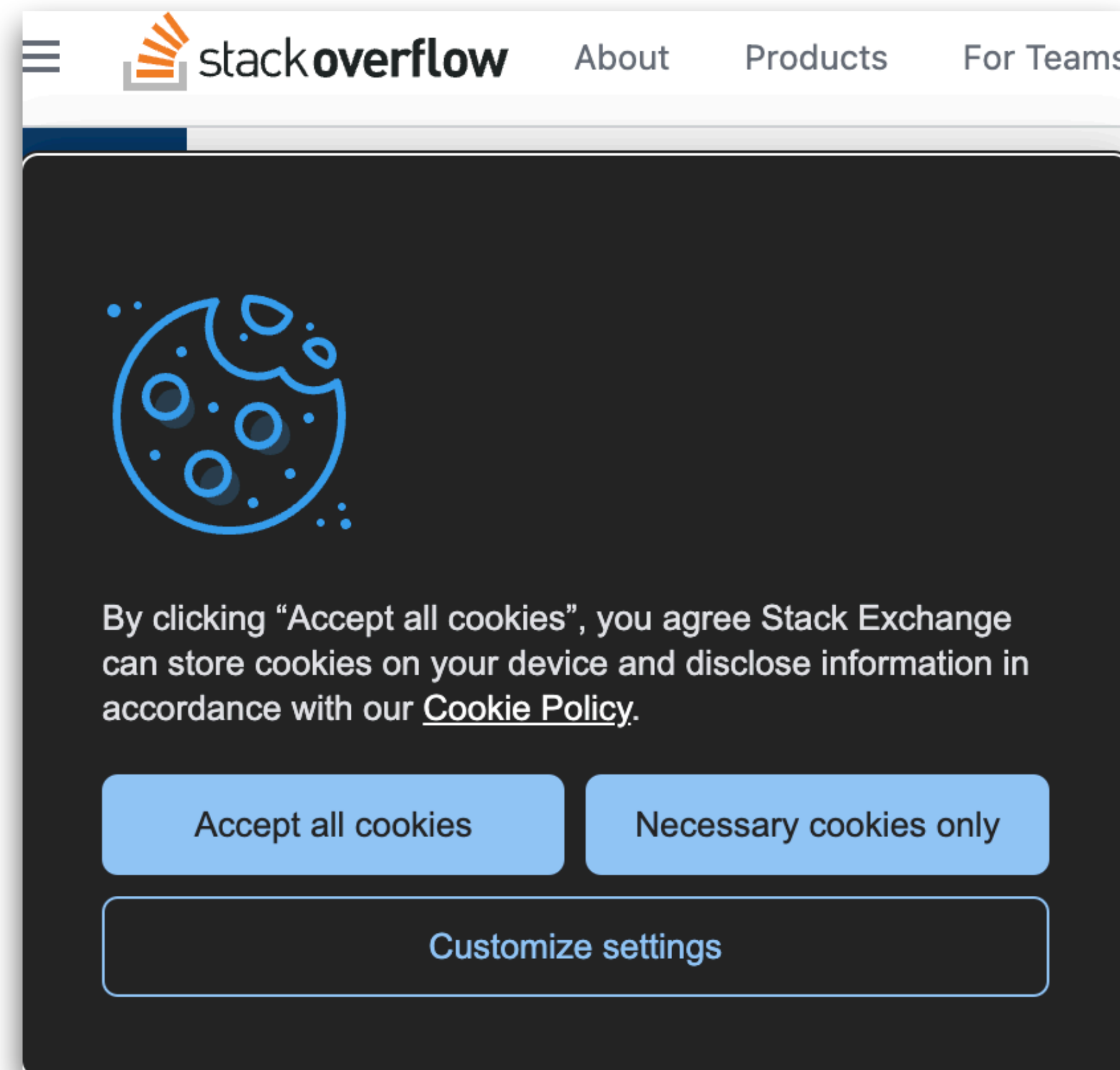
GET /reddit/ads.html?sr=security,loggedout&bust2 HTTP/1.1
Host: static.adzerk.net
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.11) Gecko/20101013 Ubuntu/9.04 (jaunty) Firefox/3.6.11
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Referer: http://www.reddit.com/r/security
Cookie: __cfduid=dc3a93cd30ca47b76600d63cde283e9b81424367471
```

Cookies Allow Behavior Tracking

- The “Referer”¹ field allows the Ad Network to track users, indexed by the cookie
 - Specifically, “third-party cookie”

¹: the “Referer” field represents a roughly three decade old misspelling of referrer

GDPR Cookie Compliance



General Data Protection Regulation

Session Cookies and Web Authentication

- An *extremely common* use of cookies is to track users who have already authenticated
- If the user already visited <http://website.com/login.html?user=alice&pass=secret> with the correct password, then the server associates a “*session cookie*” with the logged-in user’s info

Session Cookies and Web Authentication

- An *extremely common* use of cookies is to track users who have already authenticated
- If the user already visited <http://website.com/login.html?user=alice&pass=secret> with the correct password, then the server associates a “*session cookie*” with the logged-in user’s info
- Subsequent requests (GET and POST) include the cookie in the request *headers* and/or as one of the *fields*:
<http://website.com/doStuff.html?sid=81asf98as8eak>
- The idea is for the server to be able to say “I am talking to the same browser that authenticated Alice earlier.”

Session Cookies and Web Authentication

- **Session cookies (session tokens)** are a special type of cookie that keep users logged in over many requests and responses
- If an attacker steals your session token, they can log in as you!

Agenda

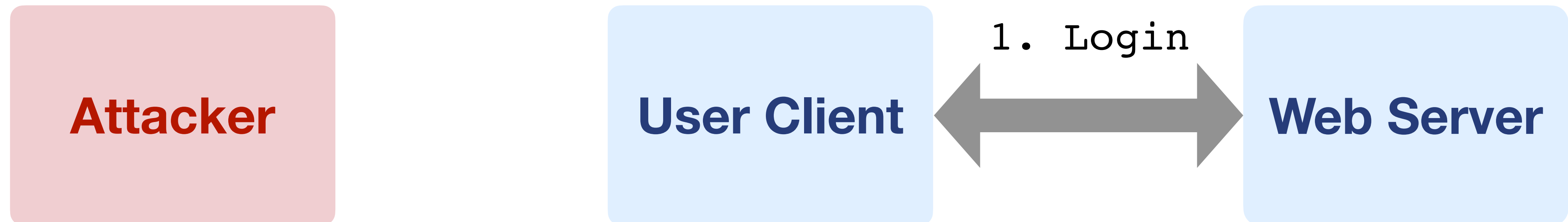
- Cookies
- Cross-Site Request Forgery (CSRF)

Cross-Site Request Forgery (CSRF)

- Idea: What if the attacker tricks the victim into making an unintended request?
 - The victim's browser will automatically attach relevant cookies
 - **The server will think the request came from the victim!**
- **Cross-site request forgery (CSRF or XSRF):** An attack that exploits cookie-based authentication to perform an action as the victim

Steps of a CSRF Attack

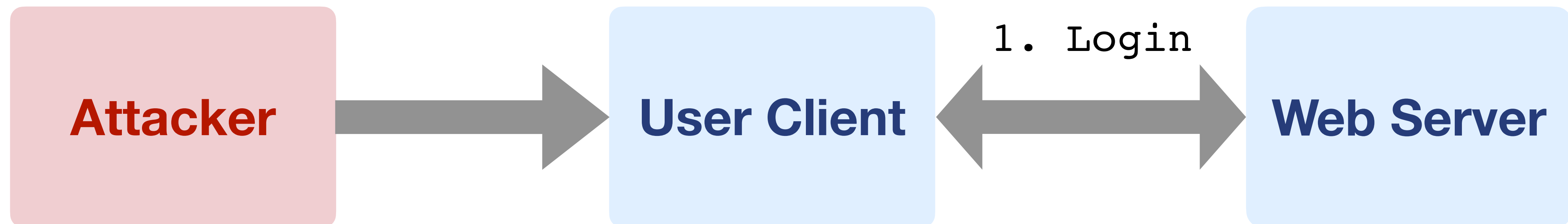
1. User authenticates to the server, receives a **cookie** with a valid **session token**



Steps of a CSRF Attack

1. User authenticates to the server, receives a **cookie** with a valid **session token**
2. Attacker **tricks** the victim into making a malicious request to the server

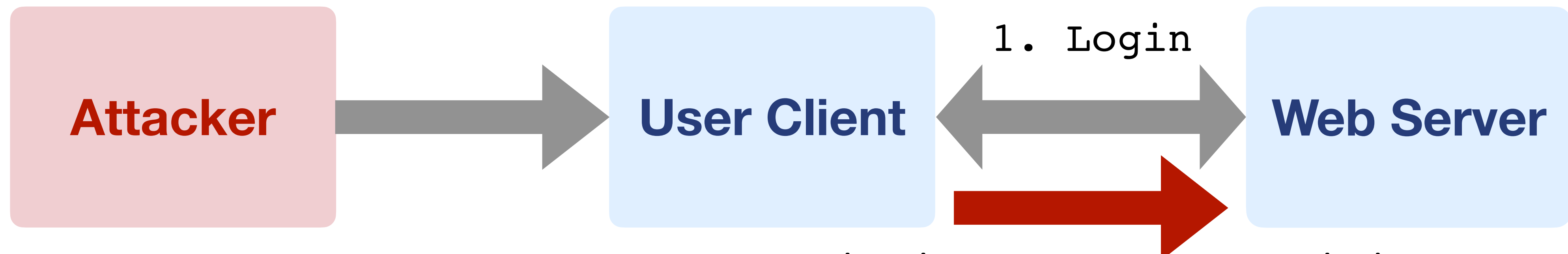
2. Tricks the victim to
make some malicious request



Steps of a CSRF Attack

1. User authenticates to the server, receives a **cookie** with a valid **session token**
2. Attacker **tricks** the victim into making a malicious request to the server
3. The victim **makes the malicious request**, attaching the cookie, server accepts it

2. Tricks the victim to make some malicious request



3. The victim makes the malicious request with session cookie

Steps of a CSRF Attack

1. User authenticates to the server, receives a cookie with a valid session token
- 2. Attacker tricks the victim into making a malicious request to the server**
3. The victim makes the malicious request, attaching the cookie, server accepts it

How to trick the victim into making such a request?

Executing a CSRF Attack

- Trick the victim into “clicking” a link (HTTP GET)
 - `https://bank.com/transfer?amount=100&recipient=mallory`
 - Transfer \$100 to Mallory
- Strategy #1: Trick the victim to open an attacker’s website, which contains some JavaScript that makes the actual malicious request

Executing a CSRF Attack

- Trick the victim into “clicking” a link (HTTP GET)
 - `https://bank.com/transfer?amount=100&recipient=mallory`
 - Transfer \$100 to Mallory
- Strategy #2: Include this in an email, or some website the victim visits
 - ``

Executing a CSRF Attack

- Trick the victim into making a HTTP POST request
- Strategy #1: Example POST request: trick the victim to submit a form

```
<form name=evilform action=https://bank.com/transfer>
```

```
<input name=amount value=100>
```

```
<input name=recipient value=mallory>
```

```
</form>
```

```
<script>document.evilmform.submit();</script>
```

Executing a CSRF Attack

- Trick the victim into making a HTTP POST request
- Strategy #2: Trick the victim to open an attacker's website, which contains some JavaScript that makes the actual HTTP POST request
- Strategy #3: Put JavaScript in the Ad of a website that the victim visits

CSRF Example

News > Privacy


Researchers find security holes in NYT, YouTube, ING, MetaFilter sites

Attackers could have used vulnerabilities on several Web sites to compromise people's accounts, allowing them to steal money, harvest e-mail addresses, or pose as others online.



Elinor Mills 

Oct. 2, 2008 2:31 p.m. PT

2 min read 

- By forcing the victim to make a request, the attacker could:
- Add any videos to the victim's "Favorites"
- Add any user to the victim's "Friend" or "Family" list
- Send arbitrary messages as the victim
- Make the victim flag any videos as inappropriate
- Make the victim share a video with their contacts
- Make the victim subscribe to any channel
- Add any videos to the user's watchlist

2023 CWE Top 25 Most Dangerous Software Weaknesses

[Top 25 Home](#)

Share via: [Twitter](#)

[View in table format](#)

[Key Insights](#)

[Methodology](#)

1

Out-of-bounds Write

[CWE-787](#) | CVEs in KEV: 70 | Rank Last Year: 1

2

Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

[CWE-79](#) | CVEs in KEV: 4 | Rank Last Year: 2

3

Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

[CWE-89](#) | CVEs in KEV: 6 | Rank Last Year: 3

4

Use After Free

[CWE-416](#) | CVEs in KEV: 44 | Rank Last Year: 7 (up 3) ▲

5

Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')

[CWE-78](#) | CVEs in KEV: 23 | Rank Last Year: 6 (up 1) ▲

6

Improper Input Validation

[CWE-20](#) | CVEs in KEV: 35 | Rank Last Year: 4 (down 2) ▼

7

Out-of-bounds Read

[CWE-125](#) | CVEs in KEV: 2 | Rank Last Year: 5 (down 2) ▼

8

Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')

[CWE-22](#) | CVEs in KEV: 16 | Rank Last Year: 8

9

Cross-Site Request Forgery (CSRF)

[CWE-352](#) | CVEs in KEV: 0 | Rank Last Year: 9

10

Unrestricted Upload of File with Dangerous Type

[CWE-434](#) | CVEs in KEV: 5 | Rank Last Year: 10

CSRF Defenses

- CSRF defenses are implemented by the server (not the browser)
- Defense: CSRF tokens
- Defense: Referer validation

CSRF Tokens

- Recall the attack:
 - Attacker structures the HTTP request in attacker's website, Ad, form, etc.
 - Tricks the victim client into making the request
- Idea: Server does not accept this request, if it doesn't contain some secret; Only a legitimate request from a benign webpage can fetch the secret.
 - Secret: CSRF Tokens

CSRF Tokens

- **CSRF token:** A secret value provided by the server to the user. The user must attach the same value in the request for the server to accept the request.
 - CSRF tokens cannot be sent to the server in a cookie!
 - The token must be sent somewhere else (e.g. a header, GET parameter, or POST content)
 - CSRF tokens are usually valid for only one or two requests

CSRF Tokens

- **CSRF token:**
 - The server needs to generate a new CSRF token every time a user requests the content.
 - CSRF tokens should be **random** and unpredictable so an attacker cannot guess the CSRF token.
 - The server also needs to maintain a mapping of CSRF tokens to session tokens, so it can validate that a request with a session token has the correct corresponding CSRF token.

Session Cookie vs CSRF Tokens

- Session cookie: keeps logged in state
- CSRF token: server checks the validity of individual requests from the client

Referer Validation

- Recall the attack:
 - Attacker structures the HTTP request in attacker's website, Ad, form, etc.
 - Tricks the victim client into making the request
- Idea: the malicious requests do not come from the legitimate website, so can we track where the requests come from?

Referer Validation

- Malicious Request Referer is an untrusted website (e.g., evil.com)
- Reject any requests with untrusted or suspicious Referer headers
- Problem: some browsers, OSes, network monitoring systems remove Referer content for privacy reasons