CROSS-SITE REQUEST FORGERY

All You Need to Know

Introduction to Cross-Site Forgery

- Continuously ranked in the OWASP Top 10 (until the 2017 release)
- CSRF is often confusing for software engineers
- High percentage of applications do not have proper CSRF protections in place
- If your web applications use cookie-based authentication, then you still have to address CSRF
- Modern frameworks have helped move CSRF from the OWASP Top 10
- Misconfigurations in frameworks such as Spring Boot, .NET MVC / Core, and NodeJS lead to most of the CSRF vulnerabilities identified in our assessments

WHAT IS CROSS-SITE REQUEST FORGERY

Cross-Site Request Forgery – What is it?

Cross-Site Request Forgery (CSRF) is an attack that abuses session management and tricks the browser into performing an attacker-controlled action

In order to show you how this works, you must understand session management in web applications. Let’s explore…

Session Management

- The HTTP protocol is a stateless protocol
  - ➢ The web application has no memory of previous requests or responses
- Many web applications rely on cookies to perform session management and appear stateful
- Logging into an application results in an authentication cookie being sent back to your browser, which identifies your user account
- As you navigate the website going forward, the browser sees that you are making a request to the application and sends the back the cookie
- This cookie tells the application the identity and account of the current user

Simply put, the cookie provides the application memory of your interaction with their site and achieves session management

How does this relate to CSRF?

Example: You are doing some online banking

You go to your bank, MyBank.com. You log in and MyBank.com sets a cookie in your browser. When you go to deposit a check, you scan a picture of the check and you send that image to your bank to deposit the check. Your browser, seeing that you are making a request to MyBank.com **helpfully** adds in your session cookie. This allows MyBank.com to know that this request came from your account.

Let's explore the example above and apply it to CSRF

CSRF Example

- Most users open one or more tabs in their browser window at a time.
- If you open a new tab and browse to a new website, Attacker.com, then you are opening yourself up to CSRF attacks.
- What you don’t know when you go to Attacker.com is that Attacker.com makes a request to MyBank.com to transfer $10,000 from your bank account to the attacker’s bank account (also known as a cross-post).
- The request came from your browser, but it originated from AttackerControlled.com.

Browser Actions from the Example

1) The browser sees a request from MyBank.com and "helpfully" inserts the session cookie

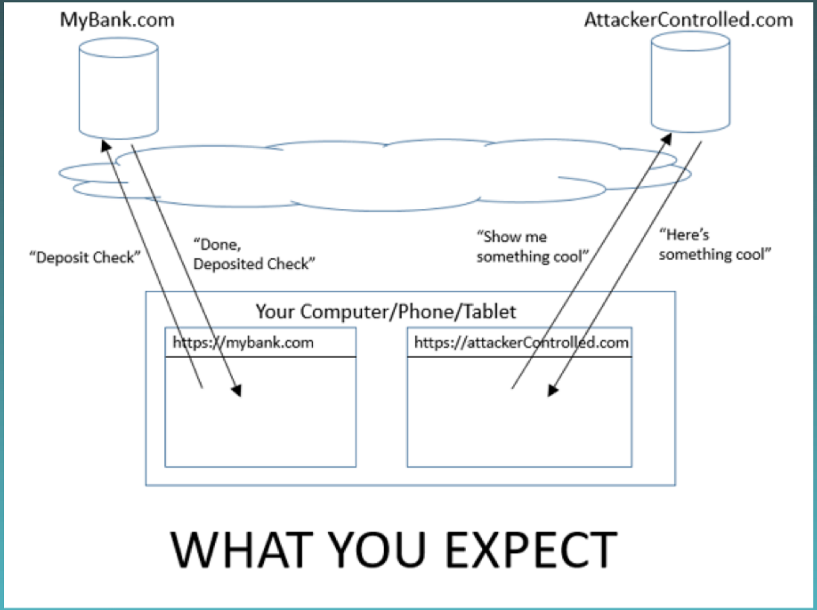
2) MyBank.com sees the request and transfers the $10,000

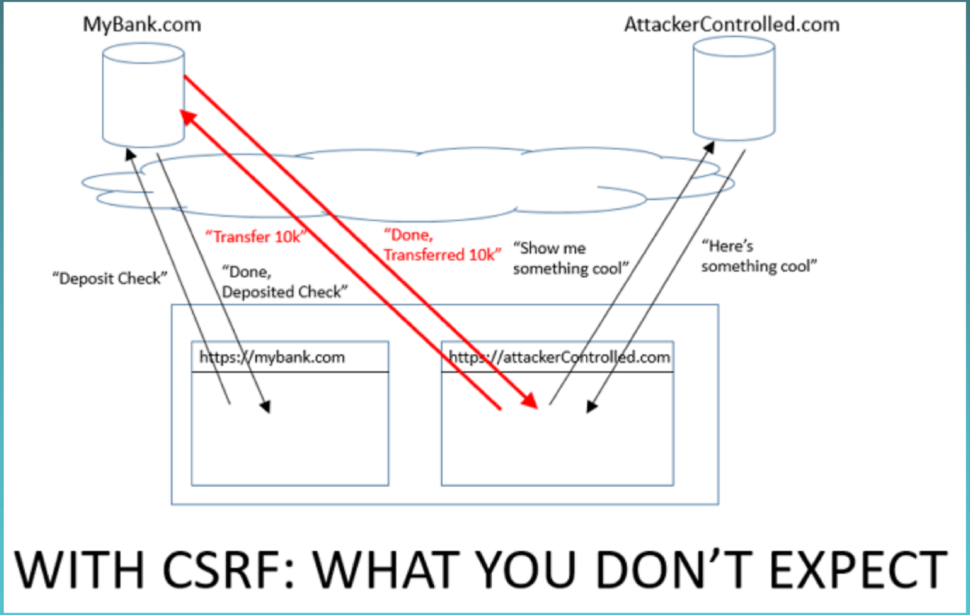
3) You were logged into MyBank.com when you visited Attacker.com

4) Evil Site, Attacker.com sent the forged request ("Request Forgery" in CSRF)

5) However, from the bank's perspective, this came from your account

Browser Actions from the example, cont.

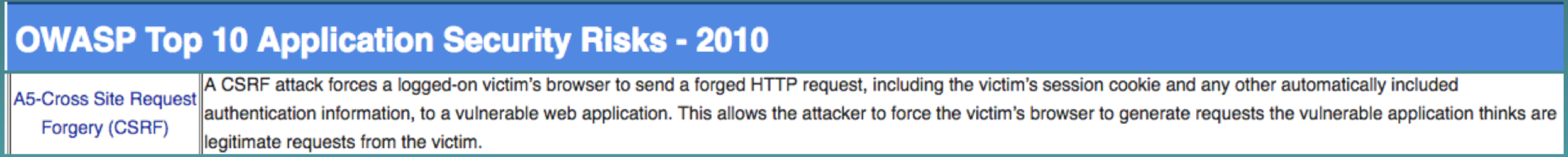




CSRF OWASP TOP 10 AND ATTACKS

Cross-Site Forgery Request and the OWASP Top 10

- CSRF vulnerabilities were very widespread 10 years ago
- OWASP first ranked CSRF #5 in the 2007 Top 10 list.
- In 2010, CSRF remained in the #5 position.



OWASP Top 10 Application Security Risks - 2010

A5-Cross Site Request Forgery (CSRF)	A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.

- In 2013, CSRF moved down to the #8 position before dropping from the list in the 2017 list.

Cross-Site Forgery Request and the OWASP Top 10

Why did it drop from the Top 10?
Why is this still relevant?
If you are relying on secure framework configurations, security teams MUST review and scan for CSRF misconfigurations

- Modern Single Page Applications (SPA) may not store authentication data in cookies. Instead, authentication data may be stored in the browser’s HTML5 session storage database.
- This eliminates CSRF because the browser will not attach the user's authentication cookie to forged requests.
- Secure frameworks (e.g., Spring Boot, .NET, and NodeJS) have built-in protection for CSRF.

Well-known CSRF Attacks

There have been several major security breaches that occurred due to the exploitation of CSRF vulnerabilities:

1. Attack in Brazil against TP-link modems [1]
   Modem admin screen allows attackers to change the DNS server address of a victim's TP-link router. Attackers then control DNS queries sent to the router leaving victims vulnerable to malware distribution and man-in-the-middle attacks
2. GoDaddy [2]
   DNS management interface allows attacks to edit DNS nameserves, zone, and renewal settings
3. PayPal [3]
   User profile interface allows attackers to change a user's pro image

PROTECTION AGAINST CSRF ATTACKS

Protecting a Site from Cross-Site Request Forgery Attacks

- The most common CSRF defense uses synchronization tokens, aka, anti-CSRF tokens to make requests unique.
- These tokens contain a large random string that is placed into a hidden form field.
- Requests to the application will contain the large random string.
- The server is able to verify the token value before processing the request

Attackers will not know the random token

⇩

Which allows the server to identify forged requests and deny the requested action

Protecting a Site from Cross-Site Request Forgery Attacks

Alternatively, some applications do not use cookies for authentication

Storing authentication tokens, such as JSON Web Tokens (JWT), in the HTML5 session store prevents the browser from accidentally sending the token during a forged request.

This configuration is not vulnerable to CSRF, but does elevate the risk of other attacks like Cross-Site Scripting.

Protecting a Site from Cross-Site Request Forgery Attacks

Using a CSRF Token

- CSRF Tokens must be **unique** per request or per session
- Must be placed into the browser's **hidden form field**
- **Cannot be stored solely in a cookie** (if they are placed into a cookie, the authentication cookies and token will automatically be sent by the browser and the protection is defeated)
- The CSRF protection provided by .NET MVC places the token into a hidden form field **and** cookie. This **allows the server to compare the two values on the server-side and avoid tracking CSRF tokens in session variables**

Must be protected from the attacker to be effective. Applications vulnerable to Cross-Site Scripting can allow attackers to steal CSRF tokens and submit a valid forged request.

SUMMARY

In Summary

- Despite being removed from the 2017 OWASP Top 10, CSRF still requires the attention of application and security teams
- There are simply too many legacy applications and misconfigured framework options to ignore the issue
- Our security assessments still identify CSRF vulnerabilities in over 50% of web applications

If your team needs help understanding and reviewing your applications for CSRF, contact us to find out more about our application security assessments.

About Cypress Data Defense:

Our goal is to help organizations secure their IT development and operations using a pragmatic, risk-based approach. The diverse background of our founders allows us to apply security controls to governance, networks, and applications across the enterprise.

Contact us to learn more!

https://www.cypressdatadefense.com/contact/

Email: info@cypressdatadefense.com

Phone Number: 720.588.8133

CYPRESS
DATA DEFENSE

References:

[1] "Attack hijacks DNS settings on home routers in Brazil"
Retrieved from: https://www.pcworld.com/article/2602040/attack-hijacks-dns-settings-on-home-routers-in-brazil.html

[2] "CSRF Flaw Allowed Attackers to Hijack GoDaddy Domains"
Retrieved from: https://www.securityweek.com/csrf-flaw-allowed-attackers-hijack-godaddy-domains

[3] "Paypal Fixes CSRF Vulnerability in Paypal.me"
Retrieved from: https://threatpost.com/paypal-fixes-csrf-vulnerability-in-paypal-me/119435/

"CSRF Attacks"
Retrieved from: https://docs.spring.io/spring-security/site/docs/current/reference/html/csrf.html

"Prevent Cross-Site Request Forgery attacks in ASP.NET Core
Retrieved from: https://docs.microsoft.com/en-us/aspnet/core/security/anti-request-forgery

"Expressjs /csurf"
Retrieved from: https://github.com/expressjs/csurf

CYPRESS
DATA DEFENSE