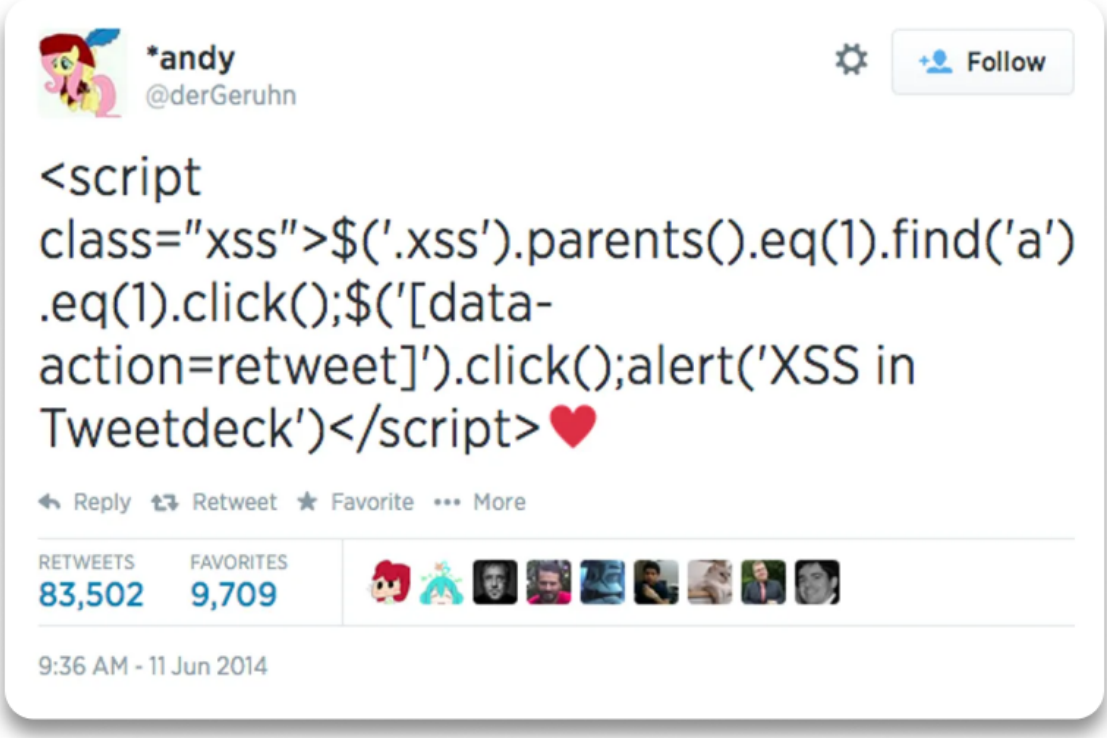
let's talk about

Cross-Site Scripting

XSS

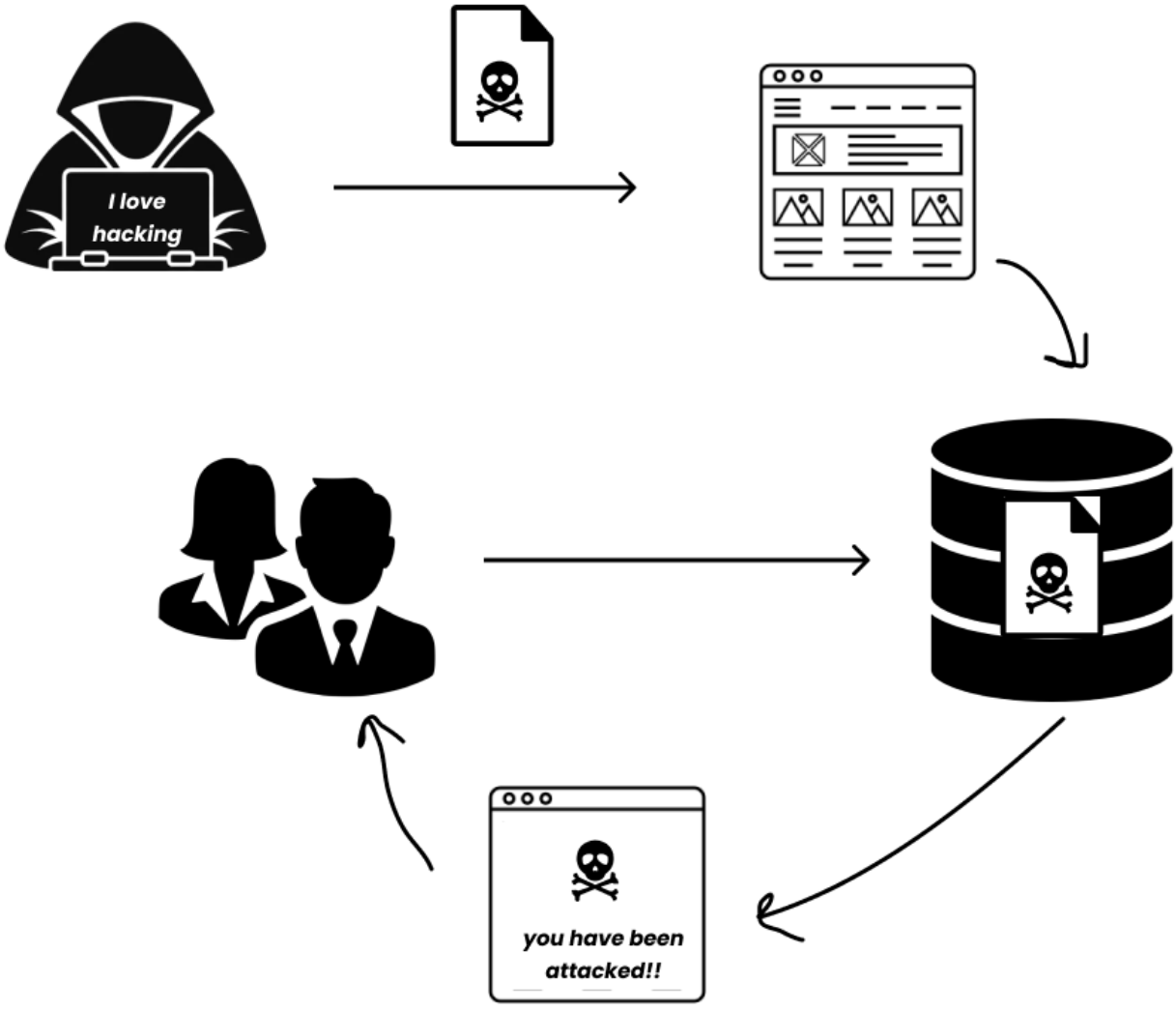
antra verma

On **June 11, 2014**, someone on twitter, exploited a **stored XSS** vulnerability in the Twitter Deck application and created a worm that **affected more than 80K twitter users** and forced them to retweet a particular message.



Let's take a look at what this attack is

Cross-site scripting (or XSS) is a type of attack in which the **attacker tries to inject or run bad code** on your website.



Types of XSS

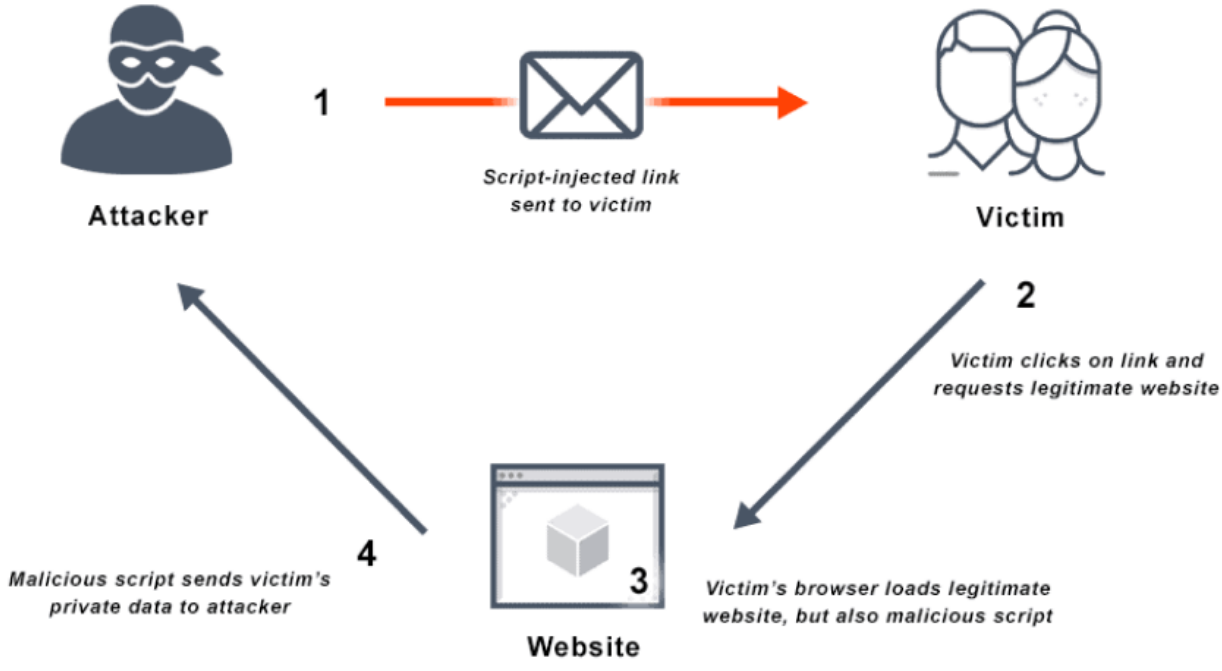
Reflected XSS

Stored XSS

DOM-based XSS

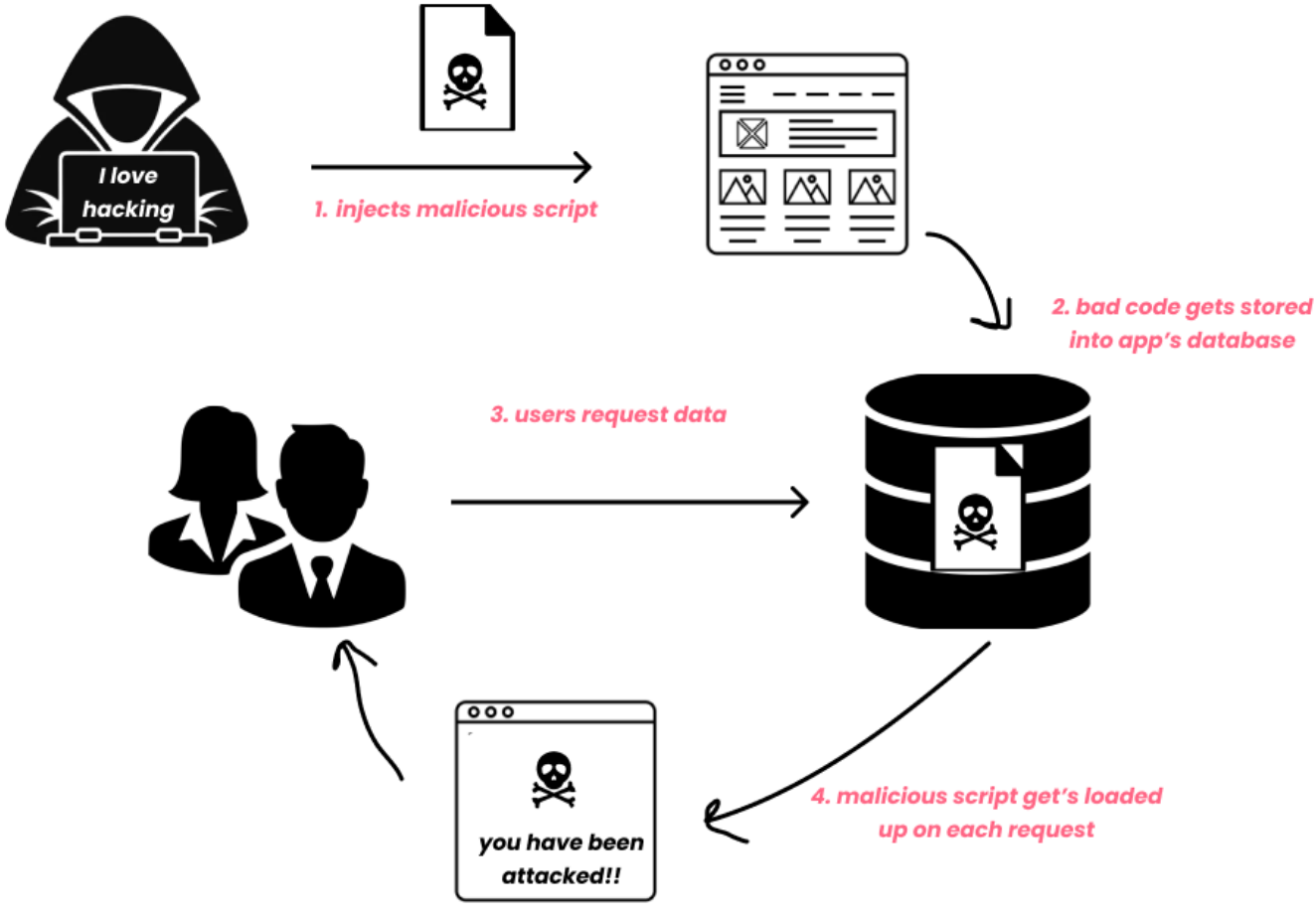
Reflected XSS

Reflected XSS is one of the most common types of XSS. To execute this type of invasion, **attackers craft malicious links, phishing emails**, or use various other techniques to trick victims into **sending malicious requests to the server**.



Stored XSS

In the case of **stored XSS**, the attacker injects the malicious script just once. After it's stored, the **application delivers the payload to all users accessing that part of the application**. Thus, it is also called persistent XSS.

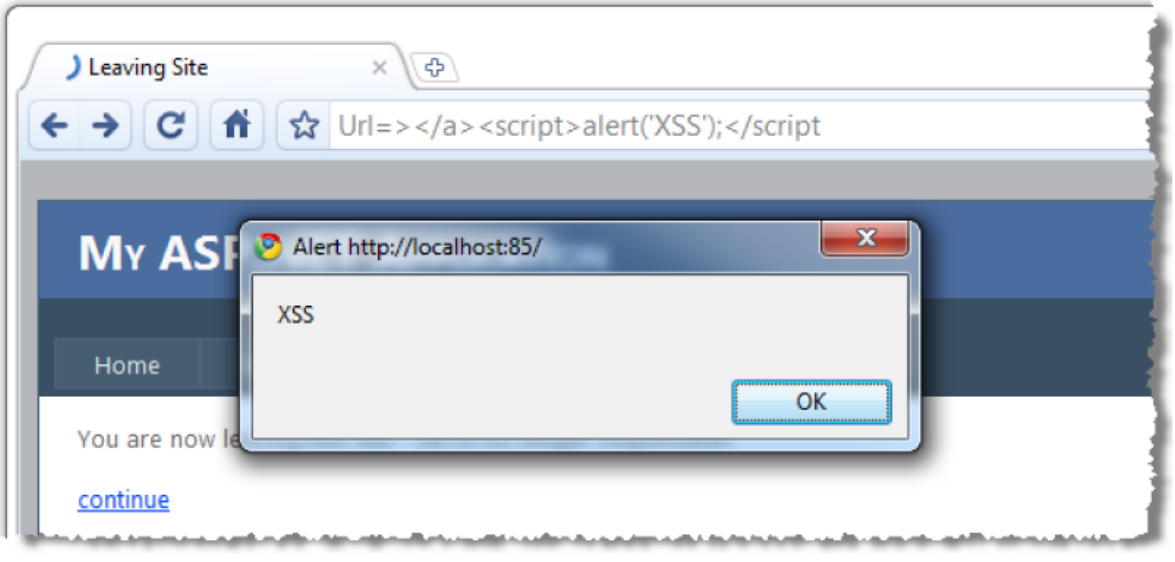


DOM-based XSS

DOM-based XSS attack is slightly different from previous two XSS techniques. It occurs when the **attack payload is executed by modifying the DOM**.

No involvement of the server in this case.

In order for the DOM-based XSS to happen, the JavaScript code of the web app needs to take input from a source that is controllable by the attacker, such as the URL in the browser's tab.



What can XSS be used for ?

- Carry out any action that the user is able to perform.
- Read any data that the user is able to access.
- Capture the user's login credentials.
- Perform virtual defacement of the web site.
- Inject trojan functionality into the web site.

Ways to Prevent XSS:

1 Implement a Content-Security Policy

- Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks including cross-site scripting (XSS) and data injection attacks.
- To enable CSP, you need to configure your web-server to return the Content-Security-Policy HTTP header. Alternatively, <meta> element can also be used to configure a policy, for example:

<meta http-equiv="Content-Security-Policy" content="default-src 'self'; img-src https://; child-src 'none'; ">*

2 Sanitize input

Whether for internal web pages or public websites, never trust the validity of user input data. Screen and validate any data fields, especially if it will be included as HTML output.

3 Use appropriate response headers:

To prevent XSS in HTTP responses that aren't intended to contain any HTML or JavaScript, you can use the Content-Type and X-Content-Type-Options headers to ensure that browsers interpret the responses in the way you intend.

Was it helpful?