

***Software Security***

**“Cross-Site Scripting Attack**

**(XSS)”**

**Lecturer contact :**

**Dr. Drake Patrick Mirembe, Email:** [**drake.mirembe@mak.ac.ug**](mailto:drake.mirembe@mak.ac.ug)

**Website:** [**www.drakemirembe.org**](http://www.drakemirembe.org/)

**Course resources: https://muele.mak.ac.ug**

What we are going to cover



##### XSS

* + The Cross-Site Scripting attack
  + Reflected XSS
  + Persistent XSS
  + Damage done by XSS attacks
  + XSS attacks to befriend with others
  + XSS attacks to change other people’s profiles
  + Self-propagation
  + Countermeasures

# What are Cross-Site Attacks?



* In XSS, an attacker injects his/her malicious code to the victim’s browser via the target website.
* Cross-Site Attacks are a class of web security vulnerabilities that enable attackers to bypass the Same-Origin Policy (SOP) implemented in web browsers.
* The Same-Origin Policy is a security mechanism that prevents web pages from accessing resources from other domains.
* Cross-Site Attacks exploit this vulnerability by injecting malicious code into a web page that runs in the context of a different domain.

# Screen ClippingWhat are Cross-Site Attacks?



*Categories of Cross-Site Attacks:*



## Non-persistent (Reflected) XSS Attack

* Persistent (Stored) XSS Attack

### Non-persistent (Reflected) XSS Attack

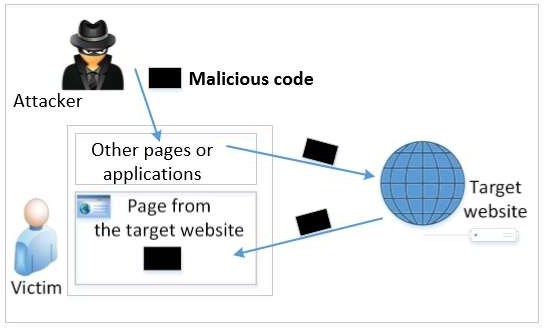


If a website with a reflective behavior takes user inputs, then

* Attackers can put JavaScript code in the input, so when the input is reflected back, the JavaScript code will be injected into the web page from the website.

### Non-persistent (Reflected) XSS Attack



**

##### Non-persistent (Reflected) XSS Attack



* Assume a vulnerable service on website : <http://www.example.com/search?input=word>, where word is provided by the users.
* Now the attacker sends the following URL to the victim and tricks him to click the link: [http://www.example.com/search?input=<script>alert(“attack”);</script](http://www.example.com/search?input=%3Cscript%3Ealert(“attack”)%3B%3C/script)>
* Once the victim clicks on this link, an HTTP GET request will be sent to the [www.example.com](http://www.example.com/) web server, which returns a page containing the search result, with the original input in the page. The input here is a JavaScript code which runs and gives a pop-up message on the victim’s browser.

## Persistent (Stored) XSS Attack



* Attackers directly send their data to a target website/server which stores the data in a persistent storage.
* If the website later sends the stored

data to other users, it creates a channel between the users and the attackers.

Example : User profile in a social network

is a channel as it is set by one user and viewed by another.

## Persistent (Stored) XSS Attack



* Attackers directly send their data to a target website/server which stores the data in a persistent storage.
* If the website later sends the stored

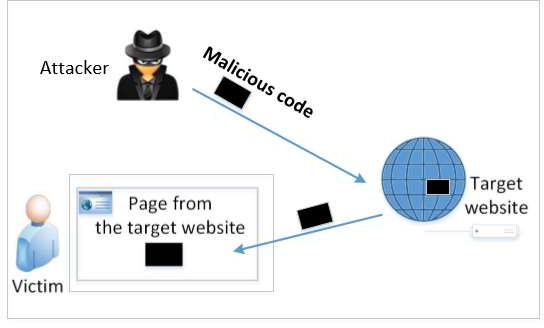
data to other users, it creates a channel between the users and the attackers.

Example : User profile in a social network

is a channel as it is set by one user and viewed by another.

## Persistent (Stored) XSS Attack





Persistent (Stored) XSS Attack



* These channels are supposed to be data channels.
* But data provided by users can contain HTML markups and JavaScript code.
* If the input is not sanitized properly by the website, it is sent to other users’ browsers through the channel and gets executed by the browsers.
* Browsers consider it like any other code coming from the website. Therefore, the code is given the same privileges as that from the website.

### Damage that can be caused by XSS



* Web defacing: JavaScript code can use DOM APIs to access the DOM nodes inside the hosting page. Therefore, the injected JavaScript code can make arbitrary changes to the page. Example: JavaScript code can change a news article page to something fake or change some pictures on the page.
* Spoofing requests: The injected JavaScript code can send HTTP requests to the server on behalf of the user. (Discussed in later slides)
* Stealing information: The injected JavaScript code can also steal victim’s private data including the session cookies, personal data displayed on the web page, data stored locally by the web application.

#### Self-Propagation XSS Worm - Samy’s worm



* Sa Samy’s worm my is a cross-site scripting worm (XSS worm) that was designed to propagate across the social networking site **MySpace** by Samy Kamkar.
* Using Samy’s worm, not only will the visitors of Samy’s profile be modified, their profiles can also be made to carry a copy of Samy’s JavaScript code. So, when an infected profile was viewed by others, the code can further spread.
* **Challenges: How can JavaScript code produce a copy of itself? Two typical approaches:**
  + DOM approach: JavaScript code can get a copy of itself

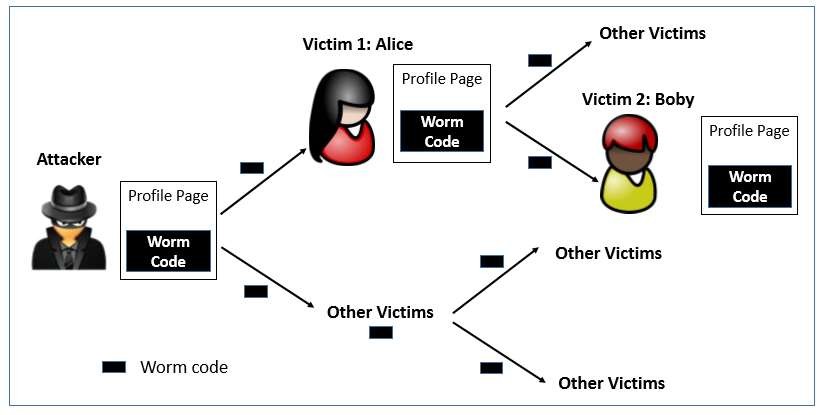
directly from DOM via DOM APIs

* + Link approach: JavaScript code can be included in a web

page via a link using the src attribute of the script tag.

#### Self-Propagation XSS Worm - Samy’s worm



**

*Self-Propagation XSS Worm*



**Document Object Model (DOM) Approach :**

* + DOM organizes the contents of the page into a tree of objects (DOM nodes).
  + Using DOM APIs, we can access each node on the tree.
  + If a page contains JavaScript code, it will be stored as an object in the tree.
  + So, if we know the DOM node that contains the code, we can use DOM APIs to get the code from the node.
  + Every JavaScript node can be given a name and then use the document.getElementByID() API to find the node.

###### How to prevent *XSS*



* **Input Validation:** Implement input validation on the server- side to ensure that user input is safe to use. This can help prevent injection attacks such as Cross-Site Scripting (XSS).
* **Output Encoding:** Encode all output data to prevent Cross- Site Scripting attacks. This can help prevent malicious code from being injected into web pages and executed in a user's browser.
* **CSRF Tokens:** Use CSRF tokens to prevent Cross-Site Request Forgery attacks. A CSRF token is a unique token that is generated by the server and included in a form or request. The token is then validated by the server to ensure that the request is legitimate.

###### How to prevent *XSS*



* **Content Security Policy (CSP):** Implement a Content Security Policy (CSP) to restrict the sources from which a web page can load resources. This can help prevent Cross-Site Scripting (XSS) attacks, as well as other types of attacks such as Clickjacking.
* **Secure Coding Practices:** Ensure that your developers are following secure coding practices such as input validation, output encoding, and secure authentication methods. Regular code reviews and security testing can also help identify and mitigate vulnerabilities.

###### How to prevent *XSS*



* **Keep Software Updated:** Keep all software, including web servers, frameworks, and libraries, up to date with the latest security patches and updates. Many vulnerabilities are discovered and patched regularly, so staying current with updates is critical to maintaining a secure environment.
* **Use HTTPS:** Implement HTTPS (SSL/TLS) on your website to ensure that all communication between the server and the client is encrypted. This can help prevent attackers from intercepting and modifying data in transit, such as session cookies or form data.

###### Summary



* In conclusion, Cross-Site Attacks are a serious security threat to web applications.
* They can result in the theft of sensitive information, unauthorized access to user accounts, and more.
* To protect your web applications against Cross-Site Attacks, it is important to understand the vulnerabilities that can be exploited and implement appropriate security measures to mitigate them.

THANK YOU!



