Cross-Site Scripting (XSS)

- follow:me github
  - By:Mahmoud ibrahim
    follow me :
    Linkedin :link: https://www.linkedin.com/in/mahmoud-ibrahim-9b5364244
    github:https://github.com/Az0x7
- What is JavaScript?
  - A dynamic web-application stands up over three pillars i.e. HTML – which determines up the complete structure, CSS – describes its overall look and feel, and the JavaScript – which simply adds powerful interactions to the application such as alert-boxes, rollover effects, dropdown menus and other things.
- JavaScript Event Handlers
  - When a JavaScript code is embedded over into HTML page, then this JavaScript "react" on some specific events like:
    When the page loads up, it is an event. When the user clicks a button, that clicks is too an event. Other examples such as – pressing any key, closing a window, resizing a window, etc. Therefore such events are thus managed by some event-handlers.
  - Onload
    - Javascript uses the onload function to load an object over on a web page
      - <body onload=alert('Welcome to Hacking Articles')>
  - Onmouseover
    - With the Onmouseover event handler, when a user moves his cursor over a specific text, the embedded javascript code will get executed.
      - surprise
  - onmouseout
    - Use this to invoke JavaScript if the mouse goes pass some link
  - onunload
    - Use this to invoke JavaScript right after someone leaves this page.
  - onclick:
    - Use this to invoke JavaScript upon clicking (a link, or form boxes)
- Introduction to Cross-Site Scripting (XSS)
  - Cross-Site Scripting often abbreviated as "XSS" is a client-side code injection attack where malicious scripts are injected into trusted websites. XSS occurs over in those web-applications where the input-parameters are not properly sanitized or validated which thus allows an attacker to send malicious Javascript codes over at a different end-user. The end user's browser has no way to know that the script should not be trusted, and will thus execute up the script.
- Impact of Cross-Site Scripting
  - the attacker could:
    - Capture and access the user's authenticated session cookies.
    - Uploads a phishing page to lure the users into unintentional actions.
    - Redirects the visitors to some other malicious sections.
    - Expose the user's sensitive data.
    - Manipulates the structure of the web-application or even defaces it.

    However, XSS has been reported with a "CVSS Score" of "6.1" as on "Medium" Severity under
    - CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
    - CWE-80: Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)
- Types of XSS
  - Stored XSS
    - "Stored XSS" often termed as "Persistent XSS" or "Type I", as over through this vulnerability the injected malicious script gets permanently stored inside the web application's database server and the server further drops it out back, when the user visits the respective website.
  - Reflected XSS
    - The Reflected XSS also termed as "Non-Persistence XSS" or "Type II", occurs when the web application responds immediately on user's input without validating what the user entered, this can lead an attacker to inject browser executable code inside the single HTML response. It is termed "non-persistent" as the malicious script does not get stored inside the web-server's database, thus the attacker needs to send the malicious link through phishing in order to trap the user.
    - Reflected XSS is the most common and thus can be easily found over at the "website's search fields" where the attacker includes some arbitrary Javascript codes in the search textbox and, if the website is vulnerable, the web-page return up the event as was described into the script.
      Reflect XSS is a major with two types:
      § Reflected XSS GET
      § Reflected XSS POST
  - DOM-Based XSS
    - The DOM-Based Cross-Site Scripting is the vulnerability which appears up in a Document Object Model rather than in the HTML pages.
    - DOM-based XSS vulnerabilities usually arise when JavaScript takes data from an attacker-controllable source, such as the URL, and passes it to a sink (a dangerous JavaScript function or DOM object as eval()) that supports dynamic code execution.
- Mitigation Steps
  - Developers should implement a whitelist of allowable inputs, and if not possible then there should be some input validations and the data entered by the user must be filtered as much as possible.
    · Output encoding is the most reliable solution to combat XSS i.e. it takes up the script code and thus converts it into the plain text.
    · A WAF or a Web Application Firewall should be implemented as it somewhere protects the application from XSS attacks.
    · Use of HTTPOnly Flags on the Cookies.
    · The developers can use Content Security Policy (CSP) to reduce the severity of any XSS vulnerabilities
- tool
  - XSSer
    - python3 xsser --url "http://192.168.0.9/bWAPP/xss_get.php?firstname=XSS&lastname=test1&form=submit" --cookie "PHPSESSID=q6tlk21lah0ois25m0b4egps85; security_level=1" --auto
- Cross-Site Scripting Exploitation
  - XSS through File Upload
    - Web-applications somewhere or the other allow its users to upload a file, whether its an image, a resume, a song, or anything specific. And with every upload, the name reflects back on the screen as it was called from the HTML code.
  - Reverse Shell with XSS
    - Generating a pop-up or redirecting a user to some different application with the XSS vulnerability is somewhere or the other seems to be harmless. But what, if the attacker is able to capture up a reverse shell, will It still be harmless? Let's see how we could do this.
  - System Exploitation Over XSS
  - CSRF with XSS
  - NTLM Hash Capture with XSS
  - Session Hijacking with Burp Collaborator Client
  - Credential Capturing with Burp Collaborator
  - XSS to SQL Injection