**CROSS-SITE REQUEST FORGERY POC GENERATOR OF COMMON DATA FORMS (JSON, XML) AND TOKEN REDUNDENCY CHECKER**

**Submitted By:**

**Yuvaraj Subramani**

**Student No: 18202158**

**Abstract:**

Web applications in this current technology days are widely used, and it is part of every user’s day to day activities, but at the same time they are usually subjected to several types of vulnerabilities. One among them is Cross-Site Request Forgery is present in the normal request response pattern of HTTP protocol.

The aim of the project is to do a Proof-of-Concept generator tool for Cross-Site Request Forgery (CSRF) attack for any given HTTP request of commonly used data forms (Json, xml, html and other multipart form data) and its and automated test scripting tool which writes a code in order to overcome the difficulty of doing it manually. This approach is based on the already existing extension of CSRF POC in Burp suites commercial version and on other standalone applications which are all based on HTML file creation data types. This enhanced model will be helpful for people who are new entrant for testing out the CSRF in different data forms where they don’t need to write the each time manually for different data forms and this will be effective as well by monitoring the resulting request(s) that are made through the Proxy. While much effort has been spent on countermeasures and detection of XSS and other vulnerabilities, to date, the detection of CSRF vulnerabilities is still performed predominantly manually.

**Introduction:**

Web application are being widely used in every aspects of fields and it has been around since the world wide web gained the mainstream popularity. The applications were initially simple as easy as just to display message board. But nowadays, the web applications were drastically changed to do all sorts of works starting from performing money transactions through online, filing income taxes, online shopping applications till staying in touch with friends through social platforms. It has evolved in every place such as banking, e-commerce and communications.

Based on a report conducted by ‘Positive Technologies’, nearly 44 percent of web applications are exploitable or vulnerable to data leakage and security problems and currently there are 4.1 billion active internet users in one way or other.

This Project is focussed on implementing a scripting tool which allows to create a POC generator for CSRF which tricks a browser to do unwanted actions in a web application whenever a user is logged in. This CSRF was listed in the OWASP’s (open source web application security project) top ten Web Application attacks list. An effective CSRF attack will be upsetting to both to client level and to end user level. It can result in damaged client relationships, unauthorized fund transfers, changed passwords and data theft - including stolen session cookies.

The attack is conducted using malicious an email or a link which makes the end user or the victim by sending a forged request to a server. Since the single token is used to authenticate their logged in application, it will be difficult to find a forged one.

**CSRF Attack Example:**

User -> Browser opened with multiple sites -> Browser sending request to server without checking origin domain -> Server (Authenticate user based on session token and accept request).

CSRF usually takes the advantage of HTTP protocols functionality to send session cookie for each request to server once user authenticate successfully, which helps server to confirm that the request is coming from authenticated user. CSRF attacker initially checks the request pattern i.e. type of request (GET request or POST request), parameters names, type of parameters values etc. CSRF attackers embed the request they want to execute in HTML tags due to which attack become invisible and while loading page.

A successful exploit will allow the attacker to make use of compromised user data and operations to do anything. Example to alter passwords, or to transfer amount, user account takeover etc.

CSRF PoC is achieved when CSRF attackers embed the request they want to execute in HTML tags due to which attack become invisible and while loading page, the request gets executed. Also, in the form submission either GET or POST when the attacker gets all form fields, it can be easily done by embedding these fields into his web page.

**Approach:**

All the currently available CSRF POC generator including the Burp suite’s one is doing it in html data forms. The other data forms are untouched for example like JSON or xml. The payload must be written manually each time to do a POC on the same and the data parsing will be difficult. I believe that this python written tool will help us to build strong and robust CSRF protection mechanism by testing out the vulnerability.

**Literature Survey:**

I have highlighted the relevant literature that uses various techniques to detect CSRF. The goal of this survey is to find the related PoC of already existing data forms and to find a relevant usage to for the commonly used other data forms.

Most of the available Commercial or standalone CSRF PoC generator are using the html data forms to create a csrf.html and these tools can be used to create CSRF PoCs for the following HTTP-Request types:

* Simple GET Requests
* Simple POST Requests

Hossain Shahriar and Mohammad Zulkernine, 2010 describes the proof of identity of CSRF attacks with the reason of visibility and content checking of suspected requests. The working of the system is to intercept a suspected request containing parameters and inputs in the value field. In an open window it relates them with one of the visible forms.

Birhanu Eshete, Adolfo Villafiorita, Komminist Weldemariam, 2011 performed assessment of security misconfiguration vulnerabilities in web server environments. Developed a usable tool to perform automated web security configuration vulnerability auditing, fixing and safety rating for Apache, MySQL and PHP. Also conducted a detailed evaluation of the tool on eleven real-life online application development and deployment environments.

These papers provide a different view on how the CSRF attack can be applied and on checking how the request is intercepted from the server.

**Methodology:**

The common request has the body type as html so it will be easy to do csrf PoC. But few web applications will have the bodies as JSON or XML, in those requests where they have bodies (e.g. XML or JSON) that can only be generated using either a form with plain text encoding, or a cross-domain XHR and with and well-formatted JSON request. If the server side code on JSON Library isn’t validating the Content-Type then it will process the request and allows a successful CSRF vulnerability for the particular we application.

**References:**

1. OWASP. <https://www.owasp.org/index.php/CSRF> , Cross-Site Request Forgery, Testing for CSRF (OWASP-SM-005)
2. Grossman, “Cross Site Request Forgery ‘The Sleeping Giant of Website Vulnerabilities’”, in RSA Conference, San Francisco, April 2008
3. Owasp Zap Standalone tool, <https://resources.infosecinstitute.com/csrf-proof-of-concept-with-owasp-zap/#gref>
4. CSRF PoC generator of HTTP request with HTML file <https://github.com/merttasci/csrf-poc-generator>
5. CSRF Vulnerabilities and Defensive Techniques, <https://search-proquest-com.ucd.idm.oclc.org/docview/1622632210?accountid=14507&pq-origsite=summon>