**Web Application Security Testing**

Below are 13 web application security testing scenarios. All the inputs are taken as **command line arguments.** Sample input is attached to them accordingly.

1. **CORS request using cURL: (=> C**ors.py)

GET request is made using pycurl and headers are checked if CORS is enabled.

**INPUT :**

|  |
| --- |
| D:\assistanz\Scripts>cors.py  URL : https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS  CORS status :  Origin : \* |

1. **CSRF Token is verified:** (=> Csrf.py)

Inputs are taken in and csrftoken is obtained from session cookies and valid request is made.

**INPUT :**

|  |
| --- |
| D:\assistanz\Scripts>csrf.py  URL : https://www.djangosites.org/accounts/login/  Request method (GET, POST): get  Incase of Authentication enter username and password else skip:  Username : rpd99  Password : test123  Data (as dict) : {}  Request made successfully |

1. **SQL injection test:** (=> SqlInjection.py)

Inputs are taken in. Inputs are validated before sending request.

**INPUT :**

|  |
| --- |
| D:\assistanz\Scripts>sqlinjection.py  URL : https://www.hacksplaining.com/exercises/sql-injection#/hack-complete  Request method (GET, POST): get  Incase of Authentication enter username and password else skip:  Username :  Password :  Data (as dict) : {"username":"user@email.com","password":"' or 1=1--"}  Invalid characters in login data |

1. **WebSocket connection testing :** (=>WebSocket.py)

Inputs are taken and connection status is checked.

**INPUT:**

|  |
| --- |
| D:\assistanz\Scripts>websocket.py  URL : https://www.websocket.org/echo.html  Web socket connection closed |

1. **Sensitive Information in URL:** (=>SensitiveInfo.py)

Inputs are taken in and sensitive info in url is extracted.

**INPUT :**

|  |
| --- |
| D:\assistanz\Scripts>sensitiveinfo.py  URL : https://www.guvi.in  Sensitive Informations in https://www.guvi.in :  Schema : https  Host : www.guvi.in |

1. **SSL disable verification:** (=>sslDisable.py)

Input is taken in and ssl verification is disabled.

**INPUT :**

|  |
| --- |
| D:\assistanz\Scripts>ssldisable.py  URL : https://www.google.com  SSL disabled |

1. **Command Injection test:** (=>CommandInjection.py)

Inputs are taken in and inputs are validated.

**INPUT:**

|  |
| --- |
| D:\assistanz\Scripts>commandinjection.py URL : http://localhost:8080/command/ Request method (GET, POST): get Incase of Authentication enter username and password else skip: Username : Password : Data (as dict) : {"ip":"172.0.0.1&dir"} Invalid characters in login data |

1. **File Inclusion:**  (=>FileInclusion.py)

Inputs are taken in and they are validated.

**INPUT:**

|  |
| --- |
| D:\assistanz\Scripts>fileinclusion.py  URL : www.victim\_site.com/abc.php?test=http://www.attacker\_site.com/attack\_page  Request method (GET, POST): get  Incase of Authentication enter username and password else skip:  Username :  Password :  Data (as dict) : {}  Inclusion found |

1. **URL Manipulation:** (=>UrlManipulation.py)

Inputs are taken in and url is manipulated. Request is mde with the manipulated url and status code is checked.

**INPUT:**

|  |
| --- |
| D:\assistanz\Scripts>urlmanipulation.py URL : https://www.google.com/search?q=h Manipulated String : ?q=hello Request method (GET, POST): get Incase of Authentication enter username and password else skip: Username : Password : Data (as dict) : {} https://www.google.com/search?q=hello URL Manipulation attack successful |

1. **Cross-Site-Scripting:** (=>Xss.py)

Inputs are taken in and they are validated.

**INPUT:**

|  |
| --- |
| D:\assistanz\Scripts>xss.py  URL : https://public-firing-range.appspot.com/reflected/parameter/body?q=%3Cscript%3Ealert(%22Hacked%22)%3C/script%3E  Request method (GET, POST): get  Incase of Authentication enter username and password else skip:  Username :  Password :  Data (as dict) : {}  XSS found |

1. **Server Information in response:** (=>ServerInfo.py)

Inputs are taken in and server information in header is returned.

**INPUT:**

|  |
| --- |
| D:\assistanz\Scripts>serverinfo.py  URL : https://www.google.com  Request method (GET, POST): get  Incase of Authentication enter username and password else skip:  Username :  Password :  Data (as dict) : {}  Server Info :  {  "Server": "gws"  } |

1. **Third party URLs and sensitive informations:** (=>ThirdPart.py)

Inputs are taken in. Third party domain is checked first and then sensitive information in url is returned.

**INPUT:**

|  |
| --- |
| D:\assistanz\Scripts>thirdparty.py  URL : https://www.google.com  Domain : www.guvi.in  Third party  Sensitive Informations in https://www.google.com :  Schema : https  Host : www.google.com |

1. **Cookie encryption:** (=>CookieEncryption.py)

Inputs are taken in and cookie encryption is vertifed. Secure and httponly flag is matched to ensure encryption.

**INPUT:**

|  |
| --- |
| D:\assistanz\Scripts>cookieencryption.py  URL : https://www.google.com  Request method (GET, POST): get  Username :  Password :  Data (as dict) : {}  Cookies encrypted |