Python for Penetration testing

XSS in Python

By

Aravind kumar

Professor:

Salem Osman

Description

- Cross-site scripting also known as XSS is a web security vulnerability that allows an attacker to compromise the interactions that users have with a vulnerable web application.
- The attacker aims to execute scripts in the victim's web browser by including malicious code in a normal web page.
- These flaws that allow these types of attacks are quite widespread in web applications that has user input.

Below are the codes explained in sections on what it does and how the actions are performed. This is just a sample for show how the XSS is done. To detect XSS for a specific website we may need to add more config to this code according to the needs.

Step 1

Firstly, we are going to install the request and bs4 libraries.

Command - pip3 install requests bs4

Step 2

Import the below libraries

Code

```
import requests

from pprint import pprint

from bs4 import BeautifulSoup as bs

from urllib.parse import urljoin
```

Step 3

- Since we are going to see about the XSS vulnerabilities, these are exploited in user inputs and forms.

- So, we are going to create a function to get all the forms from the HTML content of any web page

  Code

  ```
  def get_all_forms(url):
      """Given a `url`, it returns all forms from the HTML content"""
      soup = bs(requests.get(url).content, "html.parser")
      return soup.find_all("form")
  ```

- The above function returns a list of forms as **soup** objects, so we need to find a way to extract every form details and attributes like the action, method, and all the various input attributes.

  Code

  ```
  def get_form_details(form):
      """
      This function extracts all possible useful information about an HTML `form`
      """
      details = {}
      # get the form action (target url)
      action = form.attrs.get("action").lower()
      # get the form method (POST, GET, etc.)
      method = form.attrs.get("method", "get").lower()
      # get all the input details such as type and name
      inputs = []
      for input_tag in form.find_all("input"):
          input_type = input_tag.attrs.get("type", "text")
          input_name = input_tag.attrs.get("name")
          inputs.append({"type": input_type, "name": input_name})
      # put everything to the resulting dictionary
      details["action"] = action
      details["method"] = method
      details["inputs"] = inputs
      return details
  ```

- After we got the form details, we need another function to submit any given form:

Code

```
def submit_form(form_details, url, value):
    """
    Submits a form given in `form_details`
    Params:
        form_details (list): a dictionary that contain form information
        url (str): the original URL that contain that form
        value (str): this will be replaced to all text and search inputs
    Returns the HTTP Response after form submission
    """
    # construct the full URL (if the url provided in action is relative)
    target_url = urljoin(url, form_details["action"])
    # get the inputs
    inputs = form_details["inputs"]
    data = {}
    for input in inputs:
        # replace all text and search values with `value`
        if input["type"] == "text" or input["type"] == "search":
            input["value"] = value
        input_name = input.get("name")
        input_value = input.get("value")
        if input_name and input_value:
            # if input name and value are not None,
            # then add them to the data of form submission
            data[input_name] = input_value
    if form_details["method"] == "post":
        return requests.post(target_url, data=data)
    else:
        # GET request
        return requests.get(target_url, params=data)
```

- The above function takes form details which is the output of get_form_details() function we just wrote as an argument, that contains all form details.
- It also accepts the URL in which the original HTML form was put and the value that is set to every text or search input field.
- After we extract the form information, we just submit the form using request.get() or request.post() methods.
- Now that we have ready functions to extract all form details from a web page and submit them, it is easy now to scan for the XSS vulnerability now

Code

```
def scan_xss(url):
    """
    Given a `url`, it prints all XSS vulnerable forms and
    returns True if any is vulnerable, False otherwise
    """
    # get all the forms from the URL
    forms = get_all_forms(url)
    print(f"[+] Detected {len(forms)} forms on {url}.")
    js_script = "<Script>alert('hi')</scripT>"
    # returning value
    is_vulnerable = False
    # iterate over all forms
    for form in forms:
        form_details = get_form_details(form)
        content = submit_form(form_details, url, js_script).content.decode()
        if js_script in content:
            print(f"[+] XSS Detected on {url}")
            print(f"[*] Form details:")
            pprint(form_details)
            is_vulnerable = True
            # won't break because we want to print available vulnerable forms
    return is_vulnerable
```

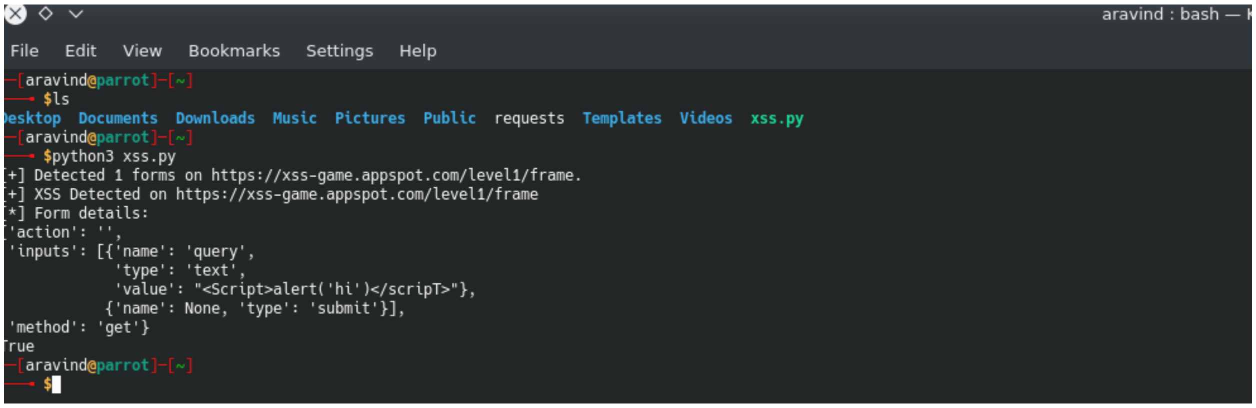
- Now let us see what the above function does
  - After a URL is given, it will check on how many forms are there and displays the number of forms detected.
  - It then iterates all over the forms and submit the forms with putting the value of all text and search input fields with a Javascript code.
  - If the Javscript code is injected and successfully executed, then this is a clear sign that the web page is XSS vulnerable.
- Now we us try with the following code to get input of and check whether there is any XSS vulnerability

  Code

  ```
  if __name__ == "__main__":
      url = "https://xss-game.appspot.com/level1/frame"
      print(scan_xss(url))
  ```

Step 3

- So, the output for the above input will be as below



```
─[aravind@parrot]─[~]
└──╼ $ls
esktop  Documents  Downloads  Music  Pictures  Public  requests  Templates  Videos  xss.py
─[aravind@parrot]─[~]
└──╼ $python3 xss.py
+] Detected 1 forms on https://xss-game.appspot.com/level1/frame.
+] XSS Detected on https://xss-game.appspot.com/level1/frame
*] Form details:
'action': '',
'inputs': [{'name': 'query',
            'type': 'text',
            'value': "<Script>alert('hi')</scripT>"},
           {'name': None, 'type': 'submit'}],
'method': 'get'}
rue
─[aravind@parrot]─[~]
└──╼ $
```

The goal is to make people aware of this kind of attacks and to basically lean how to detect XSS vulnerabilities.

Code

XSS.txt