CS437 Assignment 1

In this assignment, you are asked to implement a code to demonstrate vulnerabilities and/or protection mechanisms picked by your group. Once the demonstration part is done, you are asked to evaluate your code with the static code analysis tools.

In this assignment, you will be working in a group environment. Please note that each responsibility can only be taken by 2 groups. If a responsibility is already taken by 2 groups, please pick another one. Each group must contain exactly 3 people. If you are unable to find one, you will be randomly assigned to a group.

Please take a look at Table 1 to find out your group members and pick a responsibility to work on with your group members. Vulnerability and protection mechanisms for each group can be found below.

Table 1: Code defects to pick and their explanations.

Responsibility ID	Explanation
1	(1) SQL injection Comment attack (CWE-89) We are looking for a Python Django web application that is vulnerable to SQL injection (comment attack). (A clear demonstration is needed) (2) Making Telnet and FTP requests to any IP We are looking for a desktop application that can grab the banner of any FTP and Telnet server. (A clear demonstration is needed) (3) Predictable salt problem (Hard-coded) We are looking for a web application that hashes passwords using hard-coded salt. Websites should allow users to sign up and login to the system. However, their password will be hashed using hard-coded salt. (Python Django application is preferred and a clear demonstration is needed)
2	(1) Shell/OS command injection vulnerability You asked to write a Python Django based web application that wishes to display the contents of a given file name to the user. This application must be vulnerable to OS command injection. Only logged in users should be able to see this vulnerable page. (A clear demonstration is needed) (2) Hard-coded credentials System must have root and admin accounts and these two account passwords must be hard-coded to the web interface. Your application will not query the database for these two accounts. Passwords could be strong. (Python Django application is preferred and a clear demonstration is needed) (3) Unsecure MongoDB database update You are asked to write a desktop application that queries an insecure MongoDB server (Weak password should be used when connecting to a database). Application should be able to create tables and put some data

	inside of them. Moreover, applications should be able to retrieve data for the database. MongoDB server must have a weak password to login. (A clear demonstration is needed)
3	XML External Entities (XXE) (1) XML External Entities (XXE) vulnerability demonstration You are asked to implement a website which is vulnerable to XXE. By using this vulnerability, students will be able to steal data from their website. (A clear demonstration is needed) (2) XML External Entities (XXE) vulnerability protection demonstration Once demonstration is completed, you are asked to implement another version of the same code which is not vulnerable to XXE. (Python Django application is preferred and a clear demonstration is needed)
4	Broken Authentication (1) Website authentication mechanism that allows authentication with weak passwords. No password checks are made. (Python Django application is preferred and a clear demonstration is needed) (2) A new website signup/login mechanism that prevent this and only accepts strong passwords (Python Django application is preferred and a clear demonstration is needed) (3) Another version of the website or page that applies rate limiting to the login page in order to prevent dictionary attacks (Python Django application is preferred and a clear demonstration is needed). (4) Another version of the website or page that applies CAPTCHA security mechanism after a few unsuccessful login attempts. (A clear demonstration is needed) All these should be a new page inside of the same website. CAPTCHA mechanisms can be selected by the groups.
5	Server-Side Request Forgery (SSRF) (1) You are asked to write a SSRF vulnerable website that allows attackers to port scan internal servers using SSRF vulnerability. (A clear demonstration is needed) (2) You are asked to write a SSRF vulnerable website that allows access to local files such as or internal services to gain sensitive information such as file:///etc/passwd and http://localhost:28017/. (A clear demonstration is needed) (3) You are asked to write a SSRF vulnerable website that allows attackers to abuse internal services to conduct further attacks such as Remote Code Execution (RCE) or Denial of Service (DoS). (Python Django application is preferred and a clear demonstration is needed)

6	Broken Access Control (1) You are asked to implement an application that uses unverified data in a SQL call that is accessing account information: *pstmt.setString(1, request.getParameter("acct"));* *ResultSet results = pstmt.executeQuery( );* An attacker simply modifies the browser's 'acct' parameter to send whatever account number they want. If not correctly verified, the attacker can access any user's account.(Python Django application is preferred) https://example.com/app/accountInfo?acct=notmyacct Only logged in users can do this. (A clear demonstration is needed) (2) You are asked to write another version of the code that simply protects the application for broken access control and SQL injection attacks.(Python Django application is preferred and a clear demonstration is needed) (3) Shell/OS command injection vulnerability(Desktop) You asked to write a Python desktop application that wishes to display the contents of a given file name to the user. This application must be vulnerable to OS command injection. This application must be specific to display the content of file names entered by the users.

7	Broken Access Control
	(1) You are asked to implement a website where an attacker simply forces browsers to target URLs. Admin rights are required for access to the admin page (`admin_getappInfo`).
	
	`https://example.com/app/getappInfo`
	`https://example.com/app/admin_getappInfo`
	
	If an unauthenticated user can access either page, it's a flaw.
	
	If a non-admin (any user) can access the admin page, this is a flaw.(Python Django application is preferred)
	
	(Two different demonstration here for both unauthenticated user and non-admin user)
	
	(2) You are asked to write another version of the code that simply protects the application for broken access control.(Python Django application is preferred and a clear demonstration is needed)
	
	(3) Set a Referrer policy
	Browsers use the Referer header as a way to send information to a site about how users got there. By setting a Referrer Policy you can help to protect the privacy of your users, restricting under which circumstances the Referer header is set.
	Not allow any request without Referrer

8	Injection XSS (1) You are asked to write a website that is vulnerable to cross-site scripting (XSS) attacks.(Python Django application is preferred and a clear demonstration is needed)

	(2) You are asked to write another version of the code that prevents this.(Python Django application is preferred and a clear demonstration is needed)
9	Vulnerable and Outdated Components & Insecure design (1) Design and implement a website that uses two vulnerable and outdated components that introduce vulnerabilities that can leak information or lead to compromise of the server or the website. This vulnerability can be easily patched by updating the component. Websites should be implemented in Python. Components will be selected by the students.(A clear demonstration is needed) (2) A credential recovery workflow might include “questions and answers”, which is prohibited by NIST 800-63b. Introduce questions and answers component to the website where you allow users to change their passwords using this mechanism. No anti-bot measures must be present. (A clear demonstration is needed) 3) You are asked to create a new page with another credential recovery workflow method that uses users email to recover passwords. (A clear demonstration is needed)
10	(1) CWE-489: Leftover Debug Code Design and implement a website where you leave debug functionality that could allow attackers to use this mistake as a backdoor. These backdoor entry points create security risks and provide access to the system. (2) Security Misconfiguration Moreover, the same website's configuration allows detailed error messages. This potentially exposes sensitive information or underlying flaws such as component versions that are known to be vulnerable. (3) Using weak hashing algorithms and no password for database update You are asked to write a desktop application that allows users to authenticate a random application. This application should be able log users in using passwords inside of the database. However, the database must have a weak password. Moreover, sensitive information must be stored using weak hash without salt.

Part 1: Implementation and Demonstration

1) In this part, you are asked to implement Python code/Python website which would satisfy the responsibility chosen by your group.

    1.1) Once the implementations are done you are going to be asked to demonstrate the vulnerabilities or protection mechanisms via Zoom. First, you should introduce your code and then demonstrate the vulnerability as follows:

    - Feed the code with malicious inputs or bypass mechanisms to demonstrate the vulnerability or protection mechanism.
    - If a database is created, also explain how this database has been created and populated!!
    - Each protection mechanism and vulnerability must be explained in great detail. During the demonstration, code must be explained!

    What needs to be provided for this part:

    - Code(s) associated with the responsibilities.
    - Video of the demonstration
    - A report (Remark: Report should contain the link to the video.)
    - ***Sources (Weblink, book or magazine) used to write the code.***
    - Lastly, include an explanation of the code. Screenshots must be represented.

Part 2: Testing with Static Code Analysers

1) Please take a look at the tools mentioned in Table 2. Also find another one yourself. Then, start installing them one by one. Once installation is done, you are asked to use these tools on your source code.
   You can use Windows/Mac/Linux machines. Or you can download a new virtual machine to use. This is up to you!
   Please Mention which one of these operating systems you use.

Table 2 : Tools

Tools
-Bandit -PYT (Python Taint) -Rough-Auditing-Tool-for-Security -Prospector
Another static code analyser selected by you (One of them must be find by you)

1.1) Effectiveness of the tools

Now you are asked to test these tools using code implemented to satisfy selected responsibility.
For each tool, explain what has been detected and whether the tool managed to detect the vulnerability given to you in the responsibility part.

Please provide screenshots to prove that you have used this tool.

Your report should include a table that specifies shared responsibilities among team members if there is any. Basically explaining who did what part.

Moreover, after the deadline you will have another demonstration with the teaching assistant. We will provide more information about this later on.