STRIDE model analysis - Communication LTD

מגישים:

- הדר קימל 313559056
- אור שחר 313195216
- עדן בר 209037571
- ניר שלומן 316520956

➔ The **STRIDE** model is a widely recognized framework used in the field of cybersecurity to identify and categorize potential threats or risks to computer systems, software applications, and networks. It helps security professionals analyze various security issues and vulnerabilities.

➔ STRIDE is an acronym that stands for the following threat categories:

- **S**poofing Identity- This refers to an attacker impersonating someone else or replay a valid user's authentication to deceive users or systems. Spoofing attacks can lead to unauthorized access or data theft.

- **T**ampering with Data- This Involves the unauthorized modification of data or systems. Attackers may attempt to change data or tamper with configurations, leading to data integrity breaches or system malfunctions.

- **R**epudiation- Repudiation attacks involve an attacker denying their actions. This can undermine the accountability and integrity of the system.

- **I**nformation Disclosure- This refers to exposure of sensitive or confidential information by attackers. Information disclosure can lead to privacy breaches, identity theft or intellectual property theft.

- **D**enial of Service (DoS)- Denial of Service attacks aim to disrupt or disable the availability of a system, network or service. Attackers overload resources to make systems or services inaccessible to legitimate users.

- **E**levation of Privilege-Elevation of Privilege attacks involve an attacker gaining higher levels of access or privileges than originally authorized in order to gain control over sensitive resources, potentially compromising the security and integrity of the system.

➔ By considering each category, organizations can better understand and address the security risks associated with their systems and applications.

#	Weakness point description	Relevant threat from STRIDE
1	An attacker can fake a website like ours and perform a phishing attack in order to get a valid user's credentials.	• Spoofing Identity- if an attacker steals credentials via phishing he would be able to spoof his identity. • Elevation of Privilege- an attacker may not have any permissions but with the user's credentials he can elevate his privilege. • Information Disclosure- the information and attacker might be exposed to may be sensitive. • Repudiation- an attacker can deny his actions in the system and say it was the user.
2	Our system doesn't log or audit so an attacker can easily deny his actions in the system.	• Repudiation
3	If we had logs, an attacker can alter or even delete them in order to "cover his tracks".	• Repudiation- with changed or deleted logs an attacker can deny his actions. • Tampering with Data- changing or deleting logs is a way of interfering with data.
4	We don't have any limit on the amount of requests to our system.	• Dos- an attacker can send a large amount of requests and potentially crash our site.
5	Every user with access to our site (once registered) has basically all the privileges to almost every action in the system.	• Elevation of Privilege- a better approach would be to act according to the Principle of least privilege.
6	Our database has almost no protections so if a hacker potentially can get access to it, then he has manual access to all of the database and all of the tables in it. (there were no requirements on database protection)	• Elevation of Privilege- an attacker is not supposed to have any access to the database. • Tampering with Data- if an attacker gets manual access to the database, he can change or delete any data he wants. • Information Disclosure- with access to the database, an attacker can also be exposed to confidential or secret information.

7	We stored all the information (like the customers information) in the database as plaintext (except passwords which are encrypted)	● Information Disclosure- if an attacker gets access to this information he doesn't even have to decrypt it.
8	We have some character limit in the different fields on the forms (100 characters) but if we didn't have any character limit, a hacker could possibly perform a buffer overflow attack.	● DoS- if a hacker performs this and possibly crashes our site.
9	CSRF (Cross Site Request Forgery) is a type of security vulnerability that occurs when an attacker tricks a user's web browser into making an unintended or unauthorized request to a website or web application where the user is authenticated. Our site has protection against CSRF attacks. (built-in in HTML)	● Spoofing Identity ● Repudiation ● Information Disclosure ● Elevation of Privilege (Similar to #1)
10	An attacker can potentially perform an error analysis in the Register page to get some information and find out the system settings in order to try to exploit them.	● Information Disclosure- usually an attacker is not supposed to be exposed to this kind of information on the system. ● Elevation of Privilege- an attacker usually has no privileges in the system.
11	We use a configuration file for password requirements. If we didn't have one it would potentially mean that users passwords would be weak, or potentially another field vulnerable for injections. We also prevent the use of a dictionary to try and block brute-force attacks.	If an attacker gets access to our system it can lead to- ● Spoofing Identity ● Tampering with Data ● Repudiation ● Information Disclosure ● Elevation of Privilege (Similar to #1)
12	In the unsecured version of our code, it is vulnerable to SQL injection attacks in the Register, Login and Customers pages. In the secured version of our code we use Django's built-in features against SQL injection attacks.	● Tampering with Data- ● Information Disclosure, Elevation of Privilege- with a successful SQLi, an attacker can potentially be exposed to information he is not supposed to.

	Django converts the Python query to SQL query and then communicates with the database. The SQL queries are constructed using query parameterization. A query's SQL code is defined separately from the query's parameters since the parameters may be user-provided and therefore unsafe, they are escaped.	
13	In the unsecured version of our code, it is vulnerable to XSS stored attacks in the Customers page. In the secured version of our code we use Django's built-in features against XSS stored attacks. Django escapes specific characters which are particularly dangerous to HTML by converting the input and when it is rendered on the browser it doesn't execute because it's not a script anymore.	• Tampering with Data- remote code execution and injecting malicious scripts to the system. • Spoofing Identity, Repudiation- XSS has the potential to be used for identity theft. • Information Disclosure, Elevation of Privilege- XSS has the potential to be used for stealing secret information.