SQL Injection

SQL Injection is a common and well spread type of attack, it capitalizes on badly sanitized user inputs. The attacker is trying to exploit this vulnerability by inserting his own SQL commands into a search bar, login field or any other input field, accessing database tables to which he would normally not have permission. Aftermath of this vulnerability can result into severe damage to the company by the leak of sensitive information or lost data. How to simply avoid this is to use parametrized arguments in the SQL queries of your Data Access Layer. In our case we used Dapper which is a lightweight ORM that makes it really convenient to do fully parametrized queries without using any input concatenation.

Screenshot of parametrized dapper query

Cross Site Scripting

Cross Site Scripting also known as XSS is a client side code injection attack, where the attacker inserts scripts with malicious intent into the website. If successful, the malicious code could be inserted into the websites database and the scripts could then be execute on the original website without the users knowledge. Layouts of websites could be modified such as fake login pages, where the user would unknowingly hand in his login credentials to the attacker. The key to take care of this is proper input sanitation. In our case we validate all external input before displaying it on the page, if we expect the input to be for example a number within a certain range, we first validate it on the client side and after that we also validate it on the server side. Razor pages which we use, also offer a layer of protection. The Razor view engine provides default encoding and escaping for inputs and outputs.

Screenshot of Views file with form and @ HTML encoding

Cross Site Request Forgery

Cross Site Request Forgery also known as CSRF is a type of attack where the victim user needs to be logged in to a website and a secure session needs to be established at the time of the attack. The unsuspected user will then mostly thanks to social engineering be tricked into clicking on a phishing link with a forged request. This can result into unauthorized money transfers, data theft and changed login credentials. To prevent this there are some useful tools in the ASP.NET MVC which we utilized. Razor automatically generates anti forgery tokens when declaring method=”post” in the HTML form element. We also explicitly added an anti forgery token with HTML helper @Html.AntiForgeryToken. At last, we added the ValidateAntiForgeryToken in our Views controller for individual POST actions. Including this tag we get validation of the token which has been generated at the creation of the form, verifying the presence of the cookie.

Screenshot of Controller with ValidateAntiForgeryToken

Brute Force Attack

Brute Force also referred to as Brute forcing is a type of attack which is based on trial and error. The attacker tries a countless number of forceful attempts to break into accounts or crack encryptions. It is usually carried out through an automated software which executes the task. Depending on the strength of your passwords these attacks can take anywhere from seconds to many years. This is the reason why it is important to use strong passwords and encryption. Our minimal password requirements include at least six characters, upper case character, lower case character, numeric value and an non-alphanumeric value. Based on our password strength requirements it would take up to years for a brute force attacker to find the right combination. If we had more time we would have included a set number of allowed attempts to log into your account until eventually locking the account, another feature would have been 2 factor authentication. These precautions would make it almost impossible for any intruder to use a brute force attack to get into our system.

<https://www.kaspersky.com/resource-center/definitions/brute-force-attack>

Distributed Denial Of Service Attack

Distributed Denial of Service also wildly known as DDoS, is a type of attack where the perpetrator sends hundreds of thousands request to a web site, essentially flooding the network traffic and preventing normal users to enter. The website is not able to comprehend such an overwhelming bulk of requests and usually goes down. Dynamic IP Restriction is a way of preventing this kind of attack. It is monitoring IP addresses with suspicious activity based on patterns of behavior. If an IP address of such nature is detected it is blacklisted and denied access for the website.

<https://www.iis.net/downloads/microsoft/dynamic-ip-restrictions>

Man In The Middle Attack

Man In The Middle also known as MITM is a type of attack where the intruder intercepts communication between two parties (eg. user and website), this may lead to manipulation of the connection and theft of sensitive data. It usually happens on public WiFi hotspots where the internet connection is not private and the website visited does not use HTTPS (Hyper Text Transfer Protocol Secure). To prevent this, we use app.UseHttpsRedirection() in our startup file, this way only HTTPS is enforced meaning the secure connection is established and every communication is encrypted using a public key encryption.

Screenshot of startup file with app.UseHttpsRedirection

Over Posting attack

Over posting also known as mass assignment, is a type of cyberattack where the attacker is exploiting model binding to a request vulnerability. When a user is binding data to a model by an action like filling up a form, by design not all properties must be visible to him. If not properly secured the attacker is able to set values to properties, he normally would not be able to. Using MVC in our project we were naturally exposed to this vulnerability. Solutions to this would be adding DataAnnotations such as [Editable(false)] to the properties in the model class. This way the data binder would ignore any attempts of editing a specific property.

https://andrewlock.net/preventing-mass-assignment-or-over-posting-in-asp-net-core/