Jing Li 21902204

Contents

[Task 1: Open Web Application Security Project (OWASP)-A1 –Injection 1](#_Toc18093980)

[Task 2: SCS0001 - Command Injection 2](#_Toc18093981)

[Task 3: 3](#_Toc18093982)

# Task 1: Open Web Application Security Project (OWASP)-A1 –Injection

1. Explain why you chose it
2. What impact it can have on your applciation
3. Whether your application is vulnerable to this risk
4. How to prevent the risk
5. provide two examples of attack scenarios.

1.As we all know, User Data is extremely important. Injection vulnerabilities are often found in SQL.

2.SQL injection refers to the attacker inserting the SQL command into the string submitted by the web form to achieve the purpose of causing the server to execute a malicious SQL command.

3.This vulnerability is the most effective, the attacker's favorite active attack method.

The cause of the vulnerability is the function of SQl. SQL is an interpreted language. As an interpreted language, it is run by a runtime component. The statement that interprets the code and executes the instructions contained in it.

For example, if you are a machine in a warehouse, your job is to find an item and then put it on the conveyor. Then someone needs to give instructions to it, and then wait for the user to fill it out.

You can compare the form before execution to a fill-in-the-blank question:

such as removing the box **1** from the **a** area of ​​the a shelf, and putting it on the conveyor belt.

(The bold font is the empty space for the fill-in-the-blank question, which is provided by the person who posted the character. )

So you executed as it was, and one command was completed.

But , if there is no rule limit for the content to be filled.

If it is \*\*: from the **a** area of ​​the a shelf, remove the box **1** and throw it to me \*\* and then ignore the rest of the instructions.

His data is composed of the code written by the programmer and the data submitted by the user. During the development process, no special characters are filtered, and the variables are bound, so that the attacker can take various postures and various ideas on your server ,get useful information through an error.

4.This will test how the back-end filtering is done. Otherwise, the attacker will manipulate the database of the site at will.

In addition to assisting with security products, we must also consider the problem from the root, that is, to make judgments on user input data and filter illegal input.

5.

which is suitable for application managers or anyone responsible for the lifecycle of the application.

JavaScript is now the primary language of the web with node.js running server side and modern web frameworks such as Bootstrap, Electron, Angular, and React running on the client.

# Task 2: SCS0001 - Command Injection

1. introduction
2. Technical details
3. code examples;
4. detection method
5. conclusion

SCS0001 - Command Injection

1.The dynamic value passed to the command execution should be validated.

**2.Risk:**If a malicious user controls either the FileName or Arguments, he might be able to execute unwanted commands or add unwanted argument. This behavior would not be possible if input parameter are validate against a white-list of characters.

3. **Vulnerable Code**

var p = new Process();

p.StartInfo.FileName = "exportLegacy.exe";

p.StartInfo.Arguments = " -user " + input + " -role user";

p.Start();

#### Solution

Regex rgx = new Regex(@"^[a-zA-Z0-9]+$");

if(rgx.IsMatch(input))

{

var p = new Process();

p.StartInfo.FileName = "exportLegacy.exe";

p.StartInfo.Arguments = " -user " + input + " -role user";

p.Start();

}

SCS0005 - Weak Random Number Generator

**Risk:**The use of a predictable random value can lead to vulnerabilities when used in certain security critical contexts.

Vulnerable Code

var rnd = new Random();

byte[] buffer = new byte[16];

rnd.GetBytes(buffer);

return BitConverter.ToString(buffer);

#### Solution

using System.Security.Cryptography;

var rnd = RandomNumberGenerator.Create();

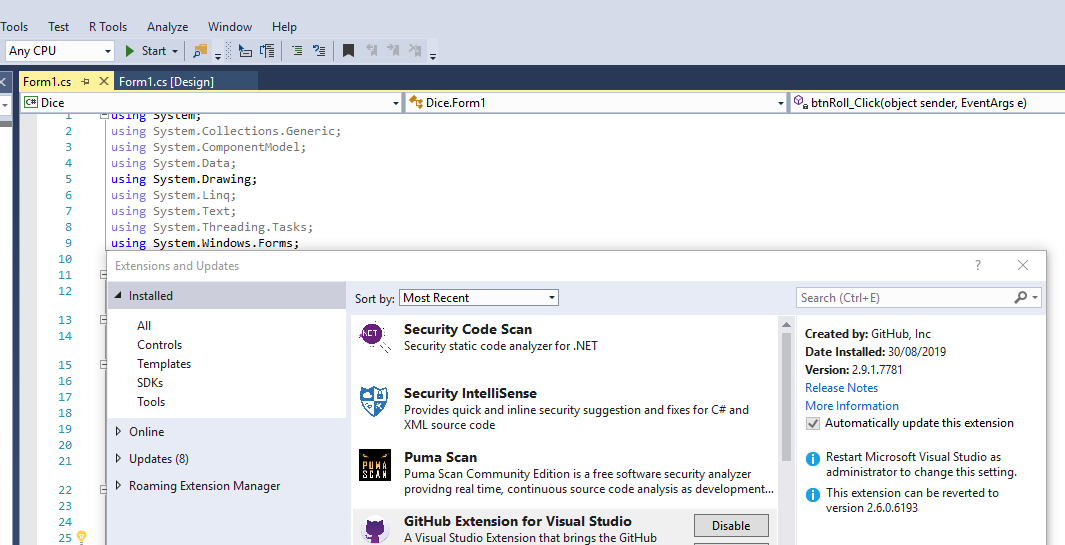
Random seed = new Random();

//generate a random number as seed so that two numbers will be different

int Dice1 = seed.Next(6) + 1;

//Since C# counts number from 0, it needs to plus 1 in order to start from 1.

detection method: I use Puma Scan and Security IntelliSense online tools to be installed in MS Visual Studio to detect.



# Task 3:

1. What were the vulnerabilities exposed in the above two tasks?

2. How were the vulnerabilities exploited?

3. Can this exploit be detected? If so, how?

4. What are the defences that can be used for these types of vulnerability? Include one to three paragraphs on secure coding issues in your answer.

5. If you were submitting a security policy for consideration, what policy would you propose to cover the security risk exposed in the above two tasks?

