**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **28/07/2020** | **Name:** | **Varshini MN** |
| **Course:** | **Coursera** | **USN:** | **4AL16EC089** |
| **Topic:** | **Network security and vulnerabilities** | **Semester & Section:** | **8th B** |
| **Github Repository:** | **varshinimn-test** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
|  |
| **REPORT**  **INTRODUCTION TO INJECTION FLAWS:**  Injection flaws, if we give a definition, they usually allow attackers relay malicious code through the vulnerable application to another system. Could be operating system, could be a database server, LDAP server, and just pretty much any component that accepts scripting as input. As you can see from this chart, they're fairly common. But what makes them special is that usually they're rated as high issues, top issues, and they're extremely dangerous. In the worst case scenario, they may allow full takeover of the vulnerable system. You may be familiar with OWASP Top 10 list, it's open web application security project.  It gives you a list of most common security vulnerabilities that afflict web applications. As you can see from year to year, from previous version in 2013 to the current version in 2017 of the list, injection vulnerabilities are at the very top of the list. They are considered as the most dangerous type of vulnerability out there. There's also SANS Top 25, a list that you may be familiar with. Here you can see it's the same picture. Positions one and two are taken by SQL injection and OS Command injection. There's agreement throughout the industry that these are the most dangerous types of vulnerabilities out there. Injection vulnerabilities, we hear about them in the news constantly.  They made possible some of the most dramatic hacks in recent history. The one that you probably heard of last year was Equifax hack, where hackers use this type of vulnerability to leak data of a 150 million US and Canadian citizens. That was truly massive, probably even the fact that some people on this call. Another example is the hack of TalkTalk, which is a British telecom company, records of 157,000 customers were exposed through SQL injection. If you read through news, you see these types of vulnerabilities come up very often and the end result of these types of leaks are lots of customer data being leaked, personal user information being leaked.  **SQL INJECTION:**  This is an interesting one because sometimes people say well, yes I run SQL queries, but no data is returned to the user and so what's the big deal in this case. There's actually many ways to do injection blindly and guess what data is just by the behavior of the system.   * There are two main types which Boolean-based which is you run a query and as you tweak the parameters, there could be one or two possible responses and because of how the system response, you could little by little guess what the data is, or there could be time-based. For example, you could sneak in delay into your query execution so that if data is of certain type, it does not delay and if there's something else in the content, it does delay * So it's probably easier to explain using an example. In red here, we have SQL expression that says that if password starts with the letter a, then sleep for ten seconds otherwise return for. So when the Docker executes this expression, if they guessed correctly, the responsible will come back 10 seconds or return 10 milliseconds I'm not quite sure, but we'll come back delayed * Whereas for all other characters they try, there will be no delay. So by being able to execute let's say 26 queries to account for all the letters in the English alphabet, they would be able to at the end of that, they'll be able to know the first letter in the password. So if there are no restrictions on the number of queries you could execute, you could write a script that tries letter in the password one-by-one and for each letter, go through all whole range of possible values and that way you could discover the password without ever getting data output to the screen from the database. So you have to be careful about that. Even if your application does not return data, attackers if they can sneak in SQL injection, they could get to the contents * There's also out of band SQL injection where they could trigger requests to other sites and exfiltrate data through some other ways not even through your application but through other channels.   **OTHER TYPES OF INJECTION**  A lot of applications now use NoSQL technology, and you may think that the likelihood of injection is reduced there and it is. However, even in NoSQL databases, there are places where you have expressions or little pieces of script that can be used. If you allow user input to reach that functionality unchecked, then injection can happen. So in this particular case, in MongoDB, there is a particular expression   * This user type equals three, which I've seen examples of something like this in real applications. Application authors specify this type of expression as a parameter, that's submitted from the UI. But that gives me as an attacker control over the expression. Instead of using a simple expression like this, I could inject something more dangerous. So at the bottom here, there's a piece of JavaScript that MongoDB does interpret JavaScript or a variant of JavaScript, where I could do a denial-of-service attack. So here, it doesn't loop for a very long time, delaying the execution of the query. So the user of your application could essentially hang your application if no checks are in place. So be careful there and not assume that if you're using NoSQL injection that you are safe. Please review all the functionality that you're using, and think about how it can be abused * XPath is a popular technology. XPath expressions operate on XML, on XML trees. Sometimes examples of an XPath using grease to search for login credentials in XML documents. If it's not done carefully, then it could expose you to injection. So in this case, we have search statement looking for a particular username and particular password. If the input is like that, it's fine, there is no issue. However, if attacker can inject another pattern that you are already familiar with from SQL injection, they could cause this query to find any user with any password and basically log you in. So be careful of that and sanitize your input in order to let something like this happen. LDAP, we use LDAP a lot in many of our products, and LDAP also has a syntax. So I think you can see the pattern developing, pretty much anything with the syntax where you can specify an expression or a little script could be abused. So in this particular case, we have an LDAP expression that finds user and password * The ampersand says that it has to be both that user and that password in the LDAP directory. Again, if the input is benign, it'll work just fine. However, if an attacker sneaks in a malicious syntax that matches any user and plays with the expression so that the password doesn't really matter. So as you can see in red here, then if you rely on this expression to log in your users, and you don't look at your input to not protect it, you could let someone login without username and password, just by using this * There are other types of injection, that applications use all kinds of templating engines. Those are sometimes vulnerable and there are many other technologies. Recommendations for avoiding injection are pretty much similar across the board. So overall just to recap, try to use functionality with reduced scope. Only use the best tool for the job. Execute with least privilege. You say functionality prevents injection. Do not let user input reach the critical resource unchanged as much as possible. Sanitize your input with whitelists, not blacklists. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date:** | **28/07/2020** | **Name:** | **Varshini MN** | |
| **Course:** | **Salesforce** | **USN:** | **4AL16EC089** | |
| **Topic:** | **Salesforce platform basics** | **Semester& Section:** | **8th B** | |
| **AFTERNOON SESSION DETAILS**    **s2.PNG** | | | |
| **REPORT**  **Power Up with AppExchange**  **Learning Objectives**   * After completing this unit, you’ll be able to: * Develop your own AppExchange strategy. * Install an app from AppExchange.The Power of Voice   To help set the stage for this module—let’s start by talking about the power of voice. At Amazon, we believe voice represents the next major disruption in computing.Voice interfaces are the next progression in a series of ever-adapting user interfaces that we use every day. In the early days of computing, there was the venerable punch card, which was a limited character interface. The next step up was to text-only graphical user interfaces (GUI). Following the introduction of the mouse, we then had a progression of GUI that used more and more advanced layouts with modern operating systems. In the 90s, the explosion of the Internet and web pages drove web design as the new frontier. Enter the smartphone in the early 2000s with a new touch-based interface. And now, with advancements in neural nets, natural language processing, and speech recognition, we have voice user interfaces (VUI).  VUI have also evolved over time. The days of “Press 1 for the front desk, press 2 for housekeeping, press 3 for reservations” are slowly shifting to a more conversational flow that is more natural for users and allows them to be more succinct and accurate in their request. This evolution is referred to as conversational user interface.  **User Makes a Request**  When you say the wake word (in this case we are using Alexa ), the light ring around the Echo begins to glow blue to indicate Alexa is now listening and streaming that data to the cloud. The captured audio is called an utterance . Note : You can also change the wake word to a couple of other words: Echo, Computer, and Amazon.  **Amazon Alexa Cloud**  Once the utterance has been received in the cloud, a series of speech models are applied to it using automatic speech recognition (ASR) and natural language understanding (NLU) to figure out what you wanted and where to route that. In the previous example, Alexa figured out that this was an intent to check the weather. Intents are registered by a skill that can handle the intents, and the skill provides a number of sample utterances to help Alexa map out where requests go.  **Using Voice Interfaces**  Let’s examine a common interaction with Alexa. If you don’t know who Alexa is:  Alexa is the brain behind the Amazon Echo family of devices and other Alexa-enabled devices. Using Alexa is as simple as asking a question—just ask, and Alexa responds instantly. Alexa lives in the cloud and is always getting smarter.  Getting back to that conversation, it can look something like this:  A typical user : “Alexa, do I need an umbrella today?”  Alexa : “It might rain in Seattle today. There’s a 55% chance. You can expect about 0.14 inches.”  A simple question, but many different things needed to happen to get that response. And yes, it does rain frequently in Seattle.  **What Is AppExchange?**  You’re probably comfortable with the idea of app stores. Whether you’re downloading apps on your phone, tablet, computer, or other device, you have to download and install apps to make the most of your technology.  Salesforce is the same way. Earlier, we mentioned the enterprise ecosystem. Salesforce has a community of partners that use the flexibility of the Salesforce platform to build amazing apps and other solutions that anyone can use. These offerings are available (some for free, some at a cost) for installation on AppExchange | | | |