Nicholas Thomas / Evan Wilson

CST-235

September 23, 2018

Dr. Toure

CLC-Mini-Project-7

**Question 2**

Describe 3 potential security vulnerabilities of web services. Substantiate your answer with code examples and screenshots of execution. Seek instructor guidance as needed.

1. Buffer Overflow – the most common effects of this threat are DOS (denial of service), malicious code execution and the potential for data corruption. These attacks can be implemented into the XML portion of the code and can cause the memory overflow or just give an error message. In the event of a DOS attack, it will force the server to parse a extremely long file that will eventually cause the application to crash.

Example:

// A C program to demonstrate buffer overflow

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

int main(int argc, char \*argv[])

{

       // Reserve 5 byte of buffer plus the terminating NULL.

       // should allocate 8 bytes = 2 double words,

       // To overflow, need more than 8 bytes...

       char buffer[5];  // If more than 8 characters input

                        // by user, there will be access

                        // violation, segmentation fault

       // a prompt how to execute the program...

       if (argc < 2)

       {

              printf("strcpy() NOT executed....\n");

              printf("Syntax: %s <characters>\n", argv[0]);

              exit(0);

       }

       // copy the user input to mybuffer, without any

       // bound checking a secure version is srtcpy\_s()

       strcpy(buffer, argv[1]);

       printf("buffer content= %s\n", buffer);

       // you may want to try strcpy\_s()

       printf("strcpy() executed...\n");

       return 0;

}

Another attack that can cause a hacker to gain control of the application is to send a block of data that is stored and deployed through the insufficient buffer size, giving the hackers block of data control over the application.

1. XML Injections – the effects of this attack include data lost or theft, command execution and schema poisoning. If data is not validated it can become vulnerable to an SQL injection, which can be created and deployed using SOAP messages. And if the data is not properly validated with the web services, these SQL injections can slip right in and be executed.

Example:

----XML----

<!-- excerpt XML Schema for a SOAP message -->

<xs:element name="transaction">

<xs:complexType>

<xs:sequence>

<xs:element name="Stotal">

<xs:simpleType>

<xs:restriction base="xs:float">

<xs:minLength value="0"/>

<xs:maxLength value="10"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element name="credit\_card\_number">

<xs:simpleType>

<xs:restriction base="xs:integer">

<xs:minLength value="10"/>

<xs:maxLength value="10"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element name="expiration">

<xs:simpleType>

<xs:restriction base="xs:integer">

<xs:minLength value="8"/>

<xs:maxLength value="8"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

---- Unmodified transaction----

<!-- excerpt unmodified SOAP message -->

<transaction>

<total>4000.00<total>

<credit\_card\_number>

123456789

</credit\_card\_number>

<expiration>01012008</expiration>

</transaction>

----modified transaction----

<!-- excerpt modified SOAP message -->

<!-- excerpt unmodified SOAP message -->

<transaction>

<total>4000.00<total>

<credit\_card\_number>

<!-- Attack payload line below. If attack is executed succesful, the <total> tag with its value gets overwritten by 6.66 -->

123456789</credit\_card\_number><total>6.66</total><credit\_card\_number>123456789

</credit\_card\_number>

<expiration>01012008</expiration>

</transaction>

Schema Poisoning is the alteration of the pre-process instructions of the XML Schema, this will give the hacker the ability to replace original processing instructions with Malicious SOAP messages that will read and processed instead.

1. Session Hijacking – essentially identity theft, this is achieved by gaining access to the user’s valid session cookie for that application. The hijacker would try and intercept the SOAP messages in the say way and trying to hack into a web application except this form of hijacking is more dangerous and gives more control to the hijacker.

References:

Web Services Security Technology and Security Concerns - White Paper. (n.d.). Retrieved September 23, 2018, from <https://www.acunetix.com/websitesecurity/web-services-wp/>

Buffer Overflow Attack with Example. (2017, May 29). Retrieved from <https://www.geeksforgeeks.org/buffer-overflow-attack-with-example/>

XML Injection. (n.d.). Retrieved September 23, 2018, from <http://www.ws-attacks.org/XML_Injection>