Covers all Exam Objectives for CEHv6

Includes Real-World Scenarios, Hands-On Exercises, and Leading-Edge Exam Prep Software Featuring:

- Custom Test Engine
- Hundreds of Sample Questions
- Electronic Flashcards
- Entire Book in PDF

CEH™

Certified Ethical Hacker

STUDY GUIDE

Exam 312-50
Exam ECO-350

Kimberly Graves

SYBEX | SERIOUS SKILLS.

Table of Contents

Chapter 9

Attacking Applications: SQL Injection and Buffer Overflows

CEH EXAM OBJECTIVES COVERED IN THIS CHAPTER:

- ✓ What is SQL injection?
- ✓ Understand the steps to conduct SQL injection
- ✓ Understand SQL Server vulnerabilities
- ✓ Describe SQL injection countermeasures
- ✓ Overview of stack-based buffer overflows
- ✓ Identify the different types of buffer overflows and methods of detection
- ✓ Overview of buffer overflow mutation techniques

SQL injection and buffer overflows are hacking techniques used to exploit weaknesses in applications. When programs are written, some parameters used in the creation of the application code can leave weaknesses in the program. SQL injection and buffer overflows are covered in the same chapter because they both are methods used to attack application and are generally caused by programming flaws. Generally, the purpose of SQL injection is to convince the application to run SQL code that was not intended.

SQL injection is a hacking method used to attack SQL databases, whereas buffer overflows can exist in many different types of applications. SQL injection and buffer overflows are similar exploits in that they're both usually delivered via a user input field. The input field is where a user may enter a username and password on a website, add data to a URL, or perform a search for a keyword in another application. The SQL injection vulnerability is caused primarily by unverified or unsanitized user input via these fields.

Both SQL Server injection and buffer overflow vulnerabilities are caused by the same issue: invalid parameters that are not verified by the application. If programmers don't take the time to validate the variables a user can enter into a variable field, the results can be serious and unpredictable. Sophisticated hackers can exploit this vulnerability, causing an execution fault and shutdown of the system or application, or a command shell to be executed for the hacker.

SQL injection and buffer overflow countermeasures are designed to utilize secure programming methods. By changing the variables used by the application code, weaknesses in applications can be greatly minimized. This chapter will detail how to perform a SQL injection and a buffer overflow attack and explore the best countermeasures to prevent the attack.

SQL Injection

As a CEH, it's important for you to be able to define SQL injection and understand the steps a hacker takes to conduct a SQL injection attack. In addition, you should know SQL Server vulnerabilities, as well as countermeasures to SQL injection attacks.

SQL injection occurs when an application processes user-provided data to create a SQL statement without first validating the input. The user input is then submitted to a web application database server for execution. When successfully exploited, SQL injection can give an attacker access to database content or allow the hacker to remotely execute system commands. In the worst-case scenario, the hacker can take control of the server that is hosting the database. This exploit can give a hacker access to a remote shell into the server

file system. The impact of a SQL injection attacks depends on where the vulnerability is in the code, how easy it is to exploit the vulnerability, and what level of access the application has to the database. Theoretically, SQL injection can occur in any type of application, but it is most commonly associated with web applications because they are most often attacked. As previously discussed in Chapter 8, "Web Hacking: GOOGLE, Web Servers, Web Application Vulnerabilities, and Web-Based Password Cracking Techniques," web applications are easy targets because by their very nature they are open to being accessed from the Internet. You should have a basic understanding of how databases work and how SQL commands are used to access the information in the databases prior to attempting the CEH exam.

During a web application SQL injection attack, malicious code is inserted into a web form field or the website's code to make a system execute a command shell or other arbitrary commands. Just as a legitimate user enters queries and additions to the SQL database via a web form, the hacker can insert commands to the SQL Server through the same web form field. For example, an arbitrary command from a hacker might open a command prompt or display a table from the database. A database table may contain personal information such as credit card numbers, social security numbers, or passwords. SQL Servers are very common database servers and used by many organizations to store confidential data. This makes a SQL Server a high-value target and therefore a system that is very attractive to hackers.

Real World Scenario

Determining SQL Injection Vulnerabilities

While performing a black-hat penetration test on a corporate network, a security tester, Tom, found a custom application on one of the publicly accessible web servers. Since this was a black-hat test, Tom did not have access to the source code to see how the program had been created. But after performing some information gathering, he was able to determine that the server was running Microsoft Internet Information Server 6 along with ASP.NET, and this suggested that the database was Microsoft's SQL Server.

The login page of the web application had a username, a password field, and a forgotten password link, which ended up being the easiest way into the system. A forgotten password link works by looking in the user database for the user's email address and sending an email containing the password to that address.

So to determine if the forgotten password link was vulnerable to SQL injection, Tom entered a single quote as part of the data in the forgotten password field. The purpose was to see if the application would construct a SQL string literally without sanitizing the user input. When submitting the form with a quote in the email address, he received a 500 error (server failure), and this suggested that the user input was being parsed literally.

The underlying SQL code of the form probably looked something like this:

```
SELECT fieldlist
  FROM table
  WHERE field = '$EMAIL';
```

Tom typed his email address followed by a single quote in the forgotten email link field. The SQL parser of the web application found the extra quote mark and aborted with a syntax error. When Tom received this error message, he was able to determine that the user input was not being sanitized properly and that the application could be exploited. In this case, he did not need to continue and exploit the application since the error message was proof enough that the application was vulnerable to a SQL injection attack. As a result of this penetration test, the client was able to fix the SQL Server vulnerability.

Finding a SQL Injection Vulnerability

Before launching a SQL injection attack, the hacker determines whether the configuration of the database and related tables and variables is vulnerable. The steps to determine the SQL Server's vulnerability are as follows:

1. Using your web browser, search for a website that uses a login page or other database input or query fields (such as an "I forgot my password" form). Look for web pages that display the `POST` or `GET HTML` commands by checking the site's source code.
2. Test the SQL Server using single quotes (`''`). Doing so indicates whether the user input variable is sanitized or interpreted literally by the server. If the server responds with an error message that says *use 'a'='a'* (or something similar), then it's most likely susceptible to a SQL injection attack.
3. Use the `SELECT` command to retrieve data from the database or the `INSERT` command to add information to the database.

Here are some examples of variable field text you can use on a web form to test for SQL vulnerabilities:

- `Blah' or 1=1--`
- `Login:blah' or 1=1--`
- `Password::blah' or 1=1--`
- `http://search/index.asp?id=blah' or 1=1--`

These commands and similar variations may allow a user to bypass a login depending on the structure of the database. When entered in a form field, the commands may return many rows in a table or even an entire database table because the SQL Server is interpreting the terms literally. The double dashes near the end of the command tell SQL to ignore the rest of the command as a comment.

Here are some examples of how to use SQL commands to take control:
To get a directory listing, type the following in a form field:

```
Blah';exec master..xp_cmdshell "dir c:\*.* /s >c:\directory.txt"--
```

To create a file, type the following in a form field:

```
Blah';exec master..xp_cmdshell "echo hacker-was-here > c:\hacker.txt"--
```

To ping an IP address, type the following in a form field:

```
Blah';exec master..xp_cmdshell "ping 192.168.1.1"--
```

The Purpose of SQL Injection

SQL injection attacks are used by hackers to achieve certain results. Some SQL exploits will produce valuable user data stored in the database, and some are just precursors to other attacks. The following are the most common purposes of a SQL injection attack:

Identifying SQL Injection Vulnerability The purpose is to probe a web application to discover which parameters and user input fields are vulnerable to SQL injection.

Performing Database Finger-Printing The purpose is to discover the type and version of database that a web application is using and "fingerprint" the database. Knowing the type and version of the database used by a web application allows an attacker to craft database-specific attacks.

Determining Database Schema To correctly extract data from a database, the attacker often needs to know database schema information, such as table names, column names, and column data types. This information can be used in a follow-on attack.

Extracting Data These types of attacks employ techniques that will extract data values from the database. Depending on the type of web application, this information could be sensitive and highly desirable to the attacker.

Adding or Modifying Data The purpose is to add or change information in a database.

Performing Denial of Service These attacks are performed to shut down access to a web application, thus denying service to other users. Attacks involving locking or dropping database tables also fall under this category.

Evading Detection This category refers to certain attack techniques that are employed to avoid auditing and detection.

Bypassing Authentication The purpose is to allow the attacker to bypass database and application authentication mechanisms. Bypassing such mechanisms could allow the attacker to assume the rights and privileges associated with another application user.

Executing Remote Commands These types of attacks attempt to execute arbitrary commands on the database. These commands can be stored procedures or functions available to database users.

Performing Privilege Escalation These attacks take advantage of implementation errors or logical flaws in the database in order to escalate the privileges of the attacker.

SQL Injection Using Dynamic Strings

Most SQL applications do a specific, predictable job. Many functions of a SQL database receive static user input where the only variable is the user input fields. Such statements do not change from execution to execution. They are commonly called static SQL statements.

However, some programs must build and process a variety of SQL statements at runtime. In many cases the full text of the statement is unknown until application execution. Such statements can, and probably will, change from execution to execution. So, they are called dynamic SQL statements.

Dynamic SQL is an enhanced form of SQL that, unlike standard SQL, facilitates the automatic generation and execution of program statements. Dynamic SQL is a term used to mean SQL code that is generated by the web application before it is executed. Dynamic SQL is a flexible and powerful tool for creating SQL strings. It can be helpful when you find it necessary to write code that can adjust to varying databases, conditions, or servers. Dynamic SQL also makes it easier to automate tasks that are repeated many times in a web application.

A hacker can attack a web-based authentication form using SQL injection through the use of dynamic strings. For example, the underlying code for a web authentication form on a web server may look like the following:

```
SQLCommand = "SELECT Username FROM Users WHERE Username = '"
SQLCommand = SQLComand & strUsername
SQLCommand = SQLComand & "' AND Password = '"
SQLCommand = SQLComand & strPassword
SQLCommand = SQLComand & "'"
strAuthCheck = GetQueryResult(SQLQuery)
```

A hacker can exploit the SQL injection vulnerability by entering a login and password in the web form that uses the following variables:

```
Username: kimberly
Password: graves' OR ''='
```

The SQL application would build a command string from this input as follows:

```
SELECT Username FROM Users
WHERE Username = 'kimberly'
AND Password = 'graves' OR ''=''
```

This is an example of SQL injection: this query will return all rows from the user's database, regardless of whether kimberly is a real username in the database or graves is a legitimate password. This is due to the `OR` statement appended to the `WHERE` clause. The comparison `''=''` will always return a `true` result, making the overall `WHERE` clause evaluate to `true` for all rows in the table. This will enable the hacker to log in with any username and password.

In Exercise 9.1, you will use HP Scramlr to test for SQL injection vulnerabilities.

EXERCISE 9.1

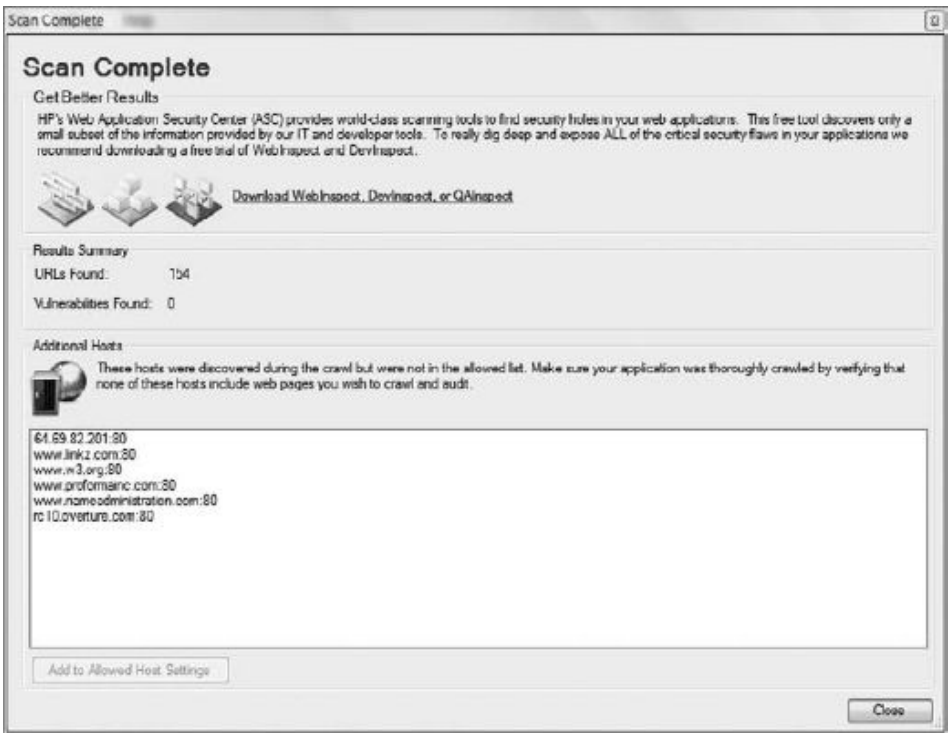
Using HP's Scrawlr to Test for SQL Injection Vulnerabilities

1. Download Scrawlr from `www.HP.com`.
2. Install Scrawlr on your Windows lab PC.
3. Open the Scrawlr program.
4. Type a target web address in the URL Of Site To Scan field:

5. Click the Start button to start the audit of the website for SQL injection vulnerabilities.

EXERCISE 9.1 ***(continued)***

6. Once the SQL injection vulnerability scan is complete, Scrawlr will display additional hosts linked from the scanned site. It is a best practice to scan the linked sites as well as the main site to ensure no SQL injection vulnerabilities exist.



SQL Injection Countermeasures

The cause of SQL injection vulnerabilities is relatively simple and well understood: insufficient validation of user input. To address this problem, defensive coding practices, such as encoding user input and validation, can be used when programming applications. It is a laborious and time-consuming process to check all applications for SQL injection vulnerabilities.

When implementing SQL injection countermeasures, review source code for the following programming weaknesses:

- Single quotes
- Lack of input validation

The first countermeasures for preventing a SQL injection attack are minimizing the privileges of a user's connection to the database and enforcing strong passwords for SA and Administrator accounts. You should also disable verbose or explanatory error messages so no more information than necessary is sent to the hacker; such information could help them determine whether the SQL Server is vulnerable. Remember that one of the purposes of SQL injection is to gain additional information as to which parameters are susceptible to attack.

Another countermeasure for preventing SQL injection is checking user data input and validating the data prior to sending the input to the application for processing.

Some countermeasures to SQL injection are

- Rejecting known bad input
- Sanitizing and validating the input field

Buffer Overflows

As a CEH, you must be able to identify different types of buffer overflows. You should also know how to detect a buffer overflow vulnerability and understand the steps a hacker may use to perform a stack-based overflow attack. We'll look at these topics, as well as provide an overview of buffer-overflow mutation techniques, in the following sections.

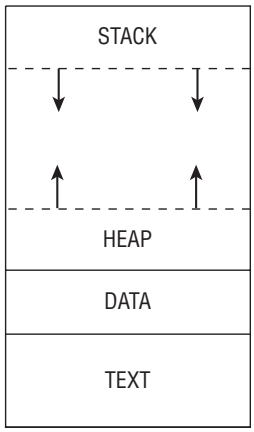
Types of Buffer Overflows and Methods of Detection

Buffer overflows are exploits that hackers use against an operating system or application; like SQL injection attacks, they're usually targeted at user input fields. A buffer overflow exploit causes a system to fail by overloading memory or executing a command shell or arbitrary code on the target system. A buffer overflow vulnerability is caused by a lack of bounds checking or a lack of input-validation sanitization in a variable field (such as on a web form). If the application doesn't check or validate the size or format of a variable before sending it to be stored in memory, an overflow vulnerability exits.

The two types of buffer overflows are stack based and heap based.

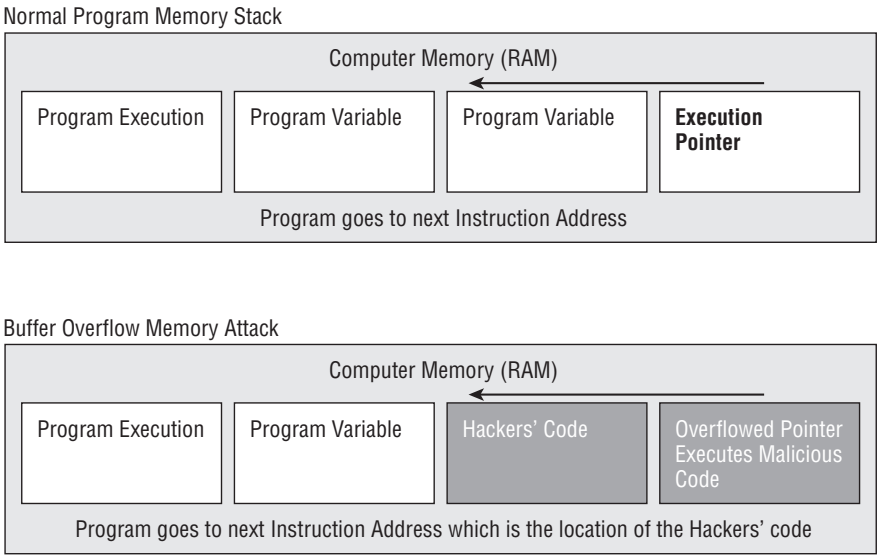
The *stack* and the *heap* are storage locations for user-supplied variables within a running program. Variables are stored in the stack or heap until the program needs them. Stacks are static locations of memory address space, whereas heaps are dynamic memory address spaces that occur while a program is running. A heap-based buffer overflow occurs in the lower part of the memory and overwrites other dynamic variables. See Figure 9.1.

FIGURE 9.1 Stack versus Heap Memory



A call stack, or *stack*, is used to keep track of where in the programming code the execution pointer should return after each portion of the code is executed. A stack-based buffer overflow attack (Figure 9.2) occurs when the memory assigned to each execution routine is overflowed. As a consequence of both types of buffer overflows, a program can open a shell or command prompt or stop the execution of a program. The next section describes stack-based buffer overflow attacks.

FIGURE 9.2 A stack-based buffer overflow attack



To detect program buffer overflow vulnerabilities that result from poorly written source code, a hacker sends large amounts of data to the application via a form field and sees what the program does as a result.

The following are the steps a hacker uses to execute a stack-based buffer overflow:

1. Enter a variable into the buffer to exhaust the amount of memory in the stack.
2. Enter more data than the buffer has allocated in memory for that variable, which causes the memory to overflow or run into the memory space for the next process. Then, add another variable, and overwrite the return pointer that tells the program where to return to after executing the variable.
3. A program executes this malicious code variable and then uses the return pointer to get back to the next line of executable code. If the hacker successfully overwrites the pointer, the program executes the hacker's code instead of the program code.

Most hackers don't need to be this familiar with the details of buffer overflows. Prewritten exploits can be found on the Internet and are exchanged between hacker groups. Exercise 9.2 walks through using Metasploit to perform a Buffer Overflow attack.

The memory register that gets overwritten with the return address of the exploit code is known as the EIP.

EXERCISE 9.2

Performing a Buffer Overflow Attack Using Metasploit

1. Open the Metasploit Framework.
2. Start the test machine running Windows Server with IIS.
3. From Metasploit, run the IIS Buffer Overflow attack against the test machine running IIS.
4. Choose a payload to deliver to the IIS target system via the buffer overflow exploit.

Buffer Overflow Countermeasures

As you can see, hackers can graduate from standard buffer overflows to redirecting the return pointer to the code of their choosing. A hacker must know the exact memory address and the size of the stack in order to make the return pointer execute their code. A hacker can use a No Operation (NOP) instruction, which is just padding to move the instruction pointer and does not execute any code. The NOP instruction is added to a string before the malicious code to be executed.

If an intrusion detection system (IDS) is present on the network, it can thwart a hacker who sends a series of NOP instructions to forward to the instruction pointer. To bypass the

IDS, the hacker can randomly replace some of the NOP instructions with equivalent pieces of code, such as `x++,x-;?NOPNOP`. This example of a mutated buffer overflow attack can bypass detection by an IDS.

Programmers should not use the built-in `strcpy()`, `strcat()`, and `streadd()` C/C++ functions because they are susceptible to buffer overflows. Alternatively, Java can be used as the programming language since Java is not susceptible to buffer overflows.

Summary

SQL injection and buffer overflows are hacking methods used to exploit applications. Web applications are especially vulnerable to attack as they have easy access for hackers in the form of user input fields, such as username, password, forgotten password, and price fields. The strict interpretation and unsanitized input is able to directly interact with the database and can cause the database to reveal confidential information. Buffer overflows exist in two types, stack based and heap based, which attack different areas of the memory allocation space. SQL injection and buffer overflow attacks can be prevented by validating user input and limiting the length of a user input field. These two countermeasures can fix most application vulnerabilities and protect applications from SQL injection and buffer overflow attacks.

Exam Essentials

Know how SQL injection and buffer overflow attacks are similar. SQL injection and buffer overflows are similar in that both attacks are delivered via a web form field.

Understand the purposes of SQL injection. The purposes of SQL injection attacks can be to obtain user data from a database or to perform information gathering on the database and application vulnerabilities.

Understand SQL injection countermeasures. Utilizing correct programming code without single quotes and performing bounds-checking and input validation are SQL injection countermeasures.

Know the difference between a stack-based and a heap-based buffer overflow. Stacks are static locations of memory address space, whereas heaps are dynamic memory address spaces.

Understand how to bypass an IDS using a buffer overflow attack. An IDS looks for a series of NOP instructions. By replacing the NOP instruction with other code segments, a hacker can effectively bypass an IDS.

Understand buffer overflow and SQL injection countermeasures. Bounds-checking and sanitizing the input from a web form can prevent a buffer overflow and SQL injection vulnerability.

Review Questions

1. Entering **`Password::blah'`** or **`1=1-`** into a web form in order to get a password is an example of what type of attack?
   - **A.** Buffer overflow
   - **B.** Heap-based overflow
   - **C.** Stack-based overflow
   - **D.** SQL injection
2. Replacing NOP instructions with other code in a buffer overflow mutation serves what purpose?
   - **A.** Bypassing an IDS
   - **B.** Overwriting the return pointer
   - **C.** Advancing the return pointer
   - **D.** Bypassing a firewall
3. Which of the following is used to store dynamically allocated variables?
   - **A.** Heap overflow
   - **B.** Stack overflow
   - **C.** Heap
   - **D.** Stack
4. What is the first step in a SQL injection attack?
   - **A.** Enter arbitrary commands at a user prompt.
   - **B.** Locate a user input field on a web page.
   - **C.** Locate the return pointer.
   - **D.** Enter a series of NOP instructions.
5. What command is used to retrieve information from a SQL database?
   - **A.** `INSERT`
   - **B.** `GET`
   - **C.** `SET`
   - **D.** `SELECT`
6. Which of the following is a countermeasure for buffer overflows?
   - **A.** Not using single quotes
   - **B.** Securing all login pages with SSL
   - **C.** Bounds checking
   - **D.** User validation

7. What does NOP stand for?
   - **A.** No Operation
   - **B.** Network Operation Protocol
   - **C.** No Once Prompt
   - **D.** Network Operation

8. What information does a hacker need to launch a buffer overflow attack?
   - **A.** A hacker needs to be familiar with the memory address space and techniques of buffer overflows in order to launch a buffer overflow attack.
   - **B.** A hacker needs to understand the differences between heaps and stacks.
   - **C.** A hacker must be able to identify a target vulnerable to a buffer overflow attack.
   - **D.** A hacker must be able to perform a port scan looking for vulnerable memory stacks.

9. Why are many programs vulnerable to SQL injection and buffer overflow attacks?
   - **A.** The programs are written quickly and use poor programming techniques.
   - **B.** These are inherent flaws in any program.
   - **C.** The users have not applied the correct service packs.
   - **D.** The programmers are using the wrong programming language.

10. Which command would a hacker enter in a web form field to obtain a directory listing?
    - **A.** `Blah';exec master..xp_cmdshell "dir *.*"--`
    - **B.** `Blah';exec_cmdshell "dir c:\*.* /s >c:\directory.txt"--`
    - **C.** `Blah';exec master..xp_cmdshell "dir c:\*.* /s >c:\directory.txt"--`
    - **D.** `Blah';exec cmdshell "dir c:\*.* "--`

11. What are two types of buffer overflow attacks?
    - **A.** Heap and stack
    - **B.** Heap and overflow
    - **C.** Stack and memory allocation
    - **D.** Injection and heap

12. Variables that are gathered from a user input field in a web application for later execution by the web application are known as _____.
    - **A.** Delayed execution
    - **B.** Dynamic strings
    - **C.** Static variables
    - **D.** Automatic functions

13. What is one purpose of SQL injection attacks?
    A. To create heap-based buffer overflows
    B. To create stack-based buffer overflows
    C. To perform NOP execution
    D. To identify vulnerable parameters

14. Which application will help identify whether a website is vulnerable to SQL injection attacks?
    A. BlackWidow
    B. Metasploit
    C. Scrawlr
    D. SQL Block

15. A countermeasure to buffer overflows is to use the _____ programming language because it is not susceptible to buffer overflow attacks.
    A. Java
    B. Netscape
    C. Oracle
    D. ASP

16. You are a programmer analyzing the code of an application running on your organization's servers. There are an excessive number of `fgets ()` commands. These are C++ functions that do not perform bounds checking. What kind of attack is this program susceptible to?
    A. Buffer overflow
    B. Denial of service
    C. SQL injection
    D. Password cracking

17. Which of the following are countermeasures to SQL injection attacks? (Choose two.)
    A. Rejecting known bad input
    B. Sanitizing and validating input field
    C. Performing user validation
    D. Ensuring all user input is a variable

18. An ethical hacker is performing a penetration test on a web application. The hacker finds a user input field on a web form and enters a single quotation mark. The website responds with a server error. What does the error indicate?
    A. The web application is susceptible to SQL injection attacks.
    B. The web application is not susceptible to SQL injection attacks.
    C. The server is experiencing a denial of service.
    D. The web application has crashed.

19. SQL statements that vary from execution to execution are known as _____ strings.

A. Variable

B. Dynamic

C. Application-based

D. Static

20. When is a No Operation (NOP) instruction added to a string?

A. After the malicious code is executed

B. Before the malicious code is executed

C. At exactly the same time the malicious code is executed

D. During the time the malicious code is executed

Answers to Review Questions

1. D. Use of a single quote indicates a SQL injection attack.
2. A. The purpose of mutating a buffer overflow by replacing NOP instructions is to bypass an IDS.
3. C. A heap is using to store dynamic variables.
4. B. The first step in a SQL injection attack is to locate a user input field on a web page using a web browser.
5. D. The command to retrieve information from a SQL database is `SELECT`.
6. C. Performing bounds checking is a countermeasure for buffer overflow attacks.
7. A. NOP is an acronym for No Operation.
8. C. All a hacker needs to be able to do to launch a buffer overflow attack is to identify a target system. A hacker can run a prewritten exploit to launch a buffer overflow.
9. A. Programs can be exploited because they're written quickly and poorly.
10. C. The command `Blah';exec master..xp_cmdshell "dir c:\*.* /s >c:\directory.txt"--` obtains a directory listing utilizing SQL injection.
11. A. Heap and stack are the two types of buffer overflows.
12. B. Dynamic strings are user input fields stored for later execution by the application.
13. D. One purpose of attacking a SQL database–based application is to identify user input parameters susceptible to SQL injection attacks.
14. C. HP's Scrawlr will scan a web URL to determine if the site is vulnerable to SQL injection attacks.
15. A. A recommended countermeasure to buffer overflow attacks is to use Java-based applications, which are not susceptible to buffer overflow attacks.
16. A. Applications that do not perform bounds checking on user input fields are susceptible to buffer overflow attacks.
17. A, B. Rejecting known bad input and sanitizing and validating user input prior to sending the command to the SQL database is a countermeasure to SQL injection attacks.
18. A. A server error in response to a single quotation mark in a web application user input field indicates the application is not sanitizing the user data and is therefore susceptible to SQL injection attacks.
19. B. Dynamic strings are built on the fly from user input and will vary each time the command is executed.
20. B. A NOP instruction is added to a string just before the malicious code is to be executed.