Exam Report: 8.4.9 Practice Questions

Date: 1/23/2020 5:44:02 pm
Time Spent: 9:05

Candidate: Garsteck, Matthew
Login: mGarsteck

Overall Performance

Your Score: 60%

Passing Score: 80%

View results by: ○ Objective Analysis ● Individual Responses

Individual Responses

Question 1: Correct

You want to prevent your browser from running JavaScript commands that are potentially harmful. Which of the following would you restrict to accomplish this?

➡ ◉ Client-side scripts

○ Server-side scripts

○ ActiveX

○ CGI

Explanation

JavaScript is an example of client-side scripting, where the client system runs the scripts that are embedded in Web pages. When pages download, the scripts are executed.

ActiveX runs executable code within a browser, but ActiveX controls are not written using the JavaScript language. Server-side scripts execute on the server, and modify the Web pages served to clients based on the results of the scripts. The Common Gateway Interface (CGI) is scripting language that is often used to capture data from forms in a Web page and pass the data to an external program. CGI runs on the server to process Web form data.

References

LabSim for Security Pro, Section 8.4.
[All Questions SecPro2017_v6.exm WEB_APP_ATTACKS_01]

Question 2: Correct

A programmer that fails to check the length of input before processing leaves his code vulnerable to what form of common attack?

○ Backdoor

➡ ◉ Buffer overflow

○ Privilege escalation

○ Session hijacking

Explanation

Buffer overflow attacks are made possible by oversight of programmers. A simple check on the length (and sometimes format) of input data before processing eliminates buffer attacks.

A*backdoor* is a developer-planted or cracker-planted entry device that bypasses security to gain access to a system or software. A developer-planted backdoor is often a debugging tool that was mistakenly left in place when the software went to market. A cracker-planted device is often a remote access server that listens for inbound connections on a specific port. Either method can be used by an intruder to gain entry into a secured environment.

Session hijacking is the concept of being able to take over a communication session between a client and server. This usually involves taking over the identity of the client and fooling the server into communicating with the pseudo client. Privilege escalation is the act of a user to steal or obtain higher level privileges in a computer system.

References

LabSim for Security Pro, Section 8.4.
[All Questions SecPro2017_v6.exm WEB_APP_ATTACKS_03]

Question 3: Correct

Which of the following is an attack that injects malicious scripts into Web pages to redirect users to fake websites or gather personal information?

○ Drive-by download

➡ ● XSS

○ SQL injection

○ DLL injection

Explanation

Cross-site scripting (XSS) is an attack that injects scripts into Web pages. When the user views the Web page, the malicious scripts run allowing the attacker to capture information or perform other actions.

- XSS often relies on social engineering or phishing to entice users to click on links to Web pages that contain the malicious scripts.
- Some scripts redirect users to legitimate websites, but run the script in the background to capture information sent to the legitimate site.
- Scripts can be written to read (steal) cookies that contain identity information (such as session information).
- Scripts can also be designed to run under the security context of the current user. For example, scripts might execute with full privileges on the local system, or the scripts might run using the credentials used on a financial website.

A drive-by download is an attack where software or malware is downloaded and installed without explicit consent from the user. A *SQL injection* attack occurs when an attacker includes database commands within user data input fields on a form, and those commands subsequently execute on the server. A *DLL injection* attack occurs when a program is forced to load a dynamic-link library (DLL). This DLL then executes under the security context of the running application, and executes malicious code included with the injected DLL.

References

LabSim for Security Pro, Section 8.4.
[All Questions SecPro2017_v6.exm WEB_APP_ATTACKS_04]

Question 4: Incorrect

When you browse to a website, a pop-up window tells you that your computer has been infected with a virus. You click on the window to see what the problem is. Later, you find out that the window has installed spyware on your system.

What type of attack has occurred?

○ DLL injection

➡ ○ Drive-by download

● ~~SQL injection~~

○ Trojan horse

Explanation

A drive-by download is an attack where software or malware is downloaded and installed without explicit consent from the user. Drive-by downloads can occur in a few different ways:

- Through social engineering, the user is tricked into downloading the software. The user might not realize that clicking a link will install software, or the user might know that something is being installed, but not have a full understanding of what it is or what it does.
- By exploiting a browser or operating system bug, a site is able to install software without the user's knowledge or consent.

A *SQL injection* attack occurs when an attacker includes database commands within user data input fields on a form, and those commands subsequently execute on the server. A *DLL injection* attack occurs when a program is forced to load a dynamic-link library (DLL). This DLL then executes under the security context of the running application, and executes malicious code included with the injected DLL. A Trojan horse is a program that masquerades as a legitimate program. In this scenario, you were not necessarily aware that a program was being installed, nor did the program present itself as a useful program for you to install.

References

LabSim for Security Pro, Section 8.4.
[All Questions SecPro2017_v6.exm WEB_APP_ATTACKS_05]

Question 5: Incorrect

Which of the following are subject to SQL injection attacks?

➡ ○ Database servers

● ~~Browsers that allow client-side scripts~~

○ Web servers serving static content

○ ActiveX controls

Explanation

A *SQL injection* attack occurs when an attacker includes database commands within user data input fields on a form, and those commands subsequently execute on the server. The injection attack succeeds if the server does not properly validate the input to restrict entry of characters that could end and begin a database command.

References

LabSim for Security Pro, Section 8.4.
[All Questions SecPro2017_v6.exm WEB_APP_ATTACKS_06]

Question 6: Correct

You have a website that accepts input from users for creating customer accounts. Input on the form is passed to a database server where the user account information is stored.

An attacker is able to insert database commands in the input fields and have those commands execute on the server.

Which type of attack has occurred?

○ Buffer overflow

○ DLL injection

○ Cross-site scripting

➡ ● SQL injection

Explanation

A *SQL injection* attack occurs when an attacker includes database commands within user data input fields on a form, and those commands subsequently execute on the server. The injection attack succeeds if the server does not properly validate the input to restrict entry of characters that could end and begin a database command. SQL injection attacks are prevented by proper programming methods that prevent commands from occurring within form data or that filter data to prevent such attacks.

A *buffer overflow* occurs when the operating system or an application does not properly enforce boundaries for how much and what type of data can be inputted. Hackers submit data beyond the size reserved for the data in the memory buffer, and the extra data overwrites adjacent memory locations. The extra data sent by the attacker could include executable code that might then be able to execute in privileged mode.

Cross-site scripting (XSS) is an attack that injects scripts into Web pages. When the user views the Web page, the malicious scripts run allowing the attacker to capture information or perform other actions. A *DLL injection* attack occurs when a program is forced to load a dynamic-link library (DLL). This DLL then executes under the security context of the running application, and executes malicious code included with the injected DLL.

References

LabSim for Security Pro, Section 8.4.
[All Questions SecPro2017_v6.exm WEB_APP_ATTACKS_07]

Question 7: Correct

Having poor software development practices and failing to program input validation checks during development of custom software can result in a system vulnerable to which type of attack?

○ Superzapping

➡ ● Buffer overflow

○ Dictionary

○ Denial of service

Explanation

Poor software development practices and failing to program input validation checks can leave a system vulnerable to buffer overflows. A buffer overflow occurs when software code receives more input than it was designed to handle and the programmer of that code failed to include input validation checks. When a buffer overflow occurs, the extra data is pushed into the execution stack and processed with the security context of the system itself. In other words, a buffer overflow attack often allows the attacker to perform any operation on a system.

Denial of service attacks exploit vulnerabilities in implementation and coding errors. Dictionary attacks are waged against logon prompts or stolen copies of a security accounts database. Superzapping attacks are specific attacks using a specialized utility named superzap to bypass the security of IBM mainframes to perform system alterations.

References

LabSim for Security Pro, Section 8.4.
[All Questions SecPro2017_v6.exm WEB_APP_ATTACKS_10]

Question 8: Correct

Which type of attack is the act of exploiting a software program's free acceptance of input in order to execute arbitrary code on a target?

○ Data diddling

○ TOC/TOU

➡ ● Buffer overflow

○ Covert channel exploitation

Explanation

The act of exploiting a software program's free acceptance of input in order to execute arbitrary code on a target is called a buffer overflow.

Data diddling is the changing or corrupting of data. TOC/TOU is a logon session replay attack. Covert channel exploitation is the use of timing or storage mechanisms to bypass security controls in order to leak information out of a secured environment.

References

LabSim for Security Pro, Section 8.4.
[All Questions SecPro2017_v6.exm WEB_APP_ATTACKS_11]

Question 9: Correct

As you browse the Internet, you notice that when you go to some sites, multiple additional windows are opened automatically. Many of these windows contain advertisements for products that are inappropriate for your family to view.

Which tool can you implement to prevent these windows from showing?

○ Phishing filter

○ Anti-adware

➡ ● Pop-up blocker

○ Anti-virus

○ Anti-spyware

Explanation

Use a pop-up blocker to prevent windows from automatically opening when you visit a Web site. Pop-up blockers typically do not block pop-ups that show when you click a button or a link, but will prevent the pop-up windows that open automatically as you navigate to other sites.

Use anti-virus software to scan attachments, downloads, or your system for malicious programs. Use anti-adware and anti-spyware software to prevent software that tracks your browsing history. While removing adware might prevent some pop-ups, it will not prevent all pop-ups unless the anti-adware software includes a pop-up blocker. Use a phishing filter to remove phishing e-mails or to prevent navigating to links that are disguised as legitimate links.

References

LabSim for Security Pro, Section 8.4.
[All Questions SecPro2017_v6.exm WEB_APP_ATTACKS_12]

Question 10: Incorrect

While using a Web-based order form, an attacker enters an unusually large value in the Quantity field.

The value she entered is so large that it exceeds the maximum value supported by the variable type used to store the quantity in the Web application. This causes the value of the quantity variable to wrap around to the minimum possible value, which is a negative number.

As a result, the Web application processes the order as a return instead of a purchase, and the attacker's account is refunded a large sum of money.

What type of attack has occurred in this scenario?

➡ ○ Integer overflow

● ~~URL hijacking~~

○ Watering hole

○ Buffer overflow

Explanation

An *integer overflow* occurs when a computational operation by a running process results in a numeric value that exceeds the maximum size of the integer type used to store it in memory. When this occurs, the value will wrap around and start again at its minimum value, in much the same way a mechanical odometer in a car rolls over to zero when the maximum number of miles it can record has been exceeded. This can allow an attacker to manipulate the value of variables, leading to unintended behavior by the system. In this scenario, the attacker has manipulated the quantity while purchasing items from an online store. This causes the value of the quantity variable to wrap around to the minimum possible value, which is a negative number. As a result, the web application processes the order form as a return instead of a purchase and the attacker's account is refunded a large sum of money.

A *buffer overflow* occurs when the operating system or an application does not properly enforce boundaries for how much and what type of data can be inputted. In a *watering hole* attack, the attacker uses reconnaissance to identify which Web sites the target person or organization frequently uses. The attacker then compromises one or more of those sites in some way, hoping that the target will access the site and be exposed to the exploit. *URL hijacking* occurs when an attacker registers domain names that correlate to common typographical errors made by users when trying to access a legitimate Web site.

References

LabSim for Security Pro, Section 8.4.
[All Questions SecPro2017_v6.exm WEB_APP_ATTACKS_13]

▼ **Question 11:** Incorrect

While using a Web-based game created using Adobe Flash, a Flash cookie is set on a user's computer. The game saves legitimate data in the Flash cookie, such as statistics and user preferences.

However, the game creator also programmed the game to track the Web sites that that user visits while the game is running and save them in the Flash cookie. This data is transferred to a server over an Internet connection without the user's permission.

What type of exploit has occurred in this scenario?

○ Buffer overflow

○ Zero-day

➡ ○ Locally shared object (LSO) exploit

● ~~Header manipulation~~

Explanation

A *locally shared object* (LSO) exploit has occurred in this scenario. LSOs are also referred to as *Flash cookies*. Adobe Flash uses LSOs to save data locally on a computer, such as information for a Flash game being played or user preferences. However, LSOs can also be used to collect information about the user's browsing habits without their permission. The Flash Player Settings Manager can be used to configure Flash to prevent LSOs from being saved on the local computer.

A *buffer overflow* occurs when the operating system or an application does not properly enforce boundaries for how much and what type of data can be inputted. *Header manipulation* is the process of including invalid data in an HTTP response header. A *zero-day* attack is an attack that exploits computer application vulnerabilities before they are known and patched by the application's developer.

References

LabSim for Security Pro, Section 8.4.
[All Questions SecPro2017_v6.exm WEB_APP_ATTACKS_14]

▼ **Question 12:** Correct

Recently, a Web site named www.vidshare.com has become extremely popular with users around the world. An attacker registers the following domain names:

- www.videoshare.com
- www.vidshar.com
- www.vidsshare.com

Each of these URLs points to a phishing Web site that tricks users into supplying their vidshare.com user names and passwords.

What type of attack has occurred in this scenario?

○ Command injection

○ Watering hole

○ Buffer overflow

➡ ● Typosquatting

Explanation

Typosquatting (also called *URL hijacking*) occurs when an attacker registers domain names that correlate to common typographical errors made by users when trying to access a legitimate Web site. The typosquatter's intentions may be benign or malicious in nature. They may be simply trying to coerce the legitimate site owner to buy the domain name from them. Alternatively, they may be attempting to compromise unsuspecting users by redirecting them to a phishing site that looks like the legitimate Web site. They may even use this exploit to install drive-by malware.

A *buffer overflow* occurs when the operating system or an application does not properly enforce boundaries for how much and what type of data can be inputted. In a *watering hole* attack, the attacker uses reconnaissance to identify which Web sites the target person or organization frequently uses. The attacker then compromises one or more of those sites in some way, hoping that the target will access the site and be exposed to the exploit. In a *command injection* attack, the attacker injects and executes unwanted commands on the application.

References

LabSim for Security Pro, Section 8.4.
[All Questions SecPro2017_v6.exm WEB_APP_ATTACKS_15]

Question 13: Incorrect

Match the exploit on the right with the appropriate description on the left.

Watering hole attack

~~A vulnerability in a running process allows an attacker to inject malicious instructions and run them.~~

An attacker compromises a Web site, hoping that a target individual will access the site and be exposed to the exploit.

Arbitrary code execution exploit

~~An attacker compromises a Web site, hoping that a target individual will access the site and be exposed to the exploit.~~

A vulnerability in a running process allows an attacker to inject malicious instructions and run them.

LSO exploit

✓ A Flash cookie is used to collect information about the user's browsing habits without their permission.

Zero-day attack

✓ An attacker exploits computer application vulnerabilities before they are known and patched by the application's developer.

Explanation

In *a watering hole* exploit, an attacker compromises a Web site, hoping that a target individual will access the site and be exposed to the exploit. In an *arbitrary code execution* exploit, a vulnerability in a running process allows an attacker to inject malicious code and run it. In a *zero-day* exploit, an attacker exploits computer application vulnerabilities before they are known and patched by the application's developer. In a *locally-shared object* (LSO) exploit, a Flash cookie is used to collect information about the user's browsing habits without their permission.

References

LabSim for Security Pro, Section 8.4.
[All Questions SecPro2017_v6.exm WEB_APP_ATTACKS_16]

Question 14: Incorrect

An attacker inserts SQL database commands into a data input field of an order form used by a Web-based application. When submitted, these commands are executed on the remote database server, causing customer contact information from the database to be sent to the malicious user's Web browser.

Which practice would have prevented this exploit?

○ Using the latest browser version and patch level.

● ~~Implementing a script blocker.~~

➡ ○ Implementing client-side validation.

○ Installing antivirus, anti-spyware, pop-up blockers, and firewall software.

Explanation

Client-side validation should have been used on the local system to identify input errors in the order form before the data was ever sent to the server. In this example, if the user entered SQL commands in an order form field, the error would have been immediately detected and blocked before the data was submitted to the server.

Using the latest browser version and patch level, installing anti-malware software, and using a script blocker are valuable security measures, but would not have prevented the exploit in this scenario.

References

LabSim for Security Pro, Section 8.4.
[All Questions SecPro2017_v6.exm WEB_APP_ATTACKS_17]

▼ **Question 15:** Correct

While using a Web-based order form, an attacker enters an unusually large value in the Quantity field.

The value she entered is so large that it exceeds the maximum value supported by the variable type used to store the quantity in the Web application. This causes the value of the quantity variable to wrap around to the minimum possible value, which is a negative number.

As a result, the Web application processes the order as a return instead of a purchase, and the attacker's account is credited with a large sum of money.

Which practices would have prevented this exploit? (Select two.)

□ Using the latest browser version and patch level.

□ Installing the latest operating system updates.

➡ ☑ Implementing server-side validation.

□ Installing antivirus, anti-spyware, pop-up blockers, and firewall software.

➡ ☑ Implementing client-side validation.

Explanation

Client-side validation and *server-side validation* should have been used to identify input errors in the order form. In this example, if the user entered an invalid quantity in an order form field, client-side validation would have detected and blocked the error before the data was submitted to the server. Server-side validation should have also been used after the data was sent to the server to detect errors. Experienced attackers can circumvent client-side validation techniques by sending data to the server from outside the application's standard user interface, bypassing any input validation measures that may have been implemented on the client.

Using the latest browser version and patch level, installing the latest operating system updates, and using a script blocker are valuable security measures, but would not have prevented the exploit in this scenario.

References

LabSim for Security Pro, Section 8.4.
[All Questions SecPro2017_v6.exm WEB_APP_ATTACKS_18]