Application Security & Secure Software Development

Web Browser Security

1. Implement good policies
2. Train users
3. User proxy & content filtering
4. Prevent malicious code

Cookies

1. Tracking cookies
2. Session cookies

Now, a tracking cookie is usually used by spyware to gather details on you. They're trying to learn what websites you go to, for how long, and what type of things you click on.

Now, session cookies, on the other hand, are used to keep track of users and their preferences and maybe even the things that they're putting into their shopping carts.

Now, many sites are realizing that cookies are not something that people like anymore, and so they're starting to migrate over to what's called server-side tracking instead.

LSOs

The second thing we want to cover in this lesson is locally shared objects, or LSOs. These are also known as Flash cookies, and they're stored in your Windows user profile under the Flash folder inside your roaming AppData folder. This is used by Adobe's Flash Player and it's less of an issue these days because Adobe Flash nis being phased out in favor of HTML5. LSOs can be disabled within your Flash Player settings if you're still using Flash, and this is also found inside the local settings manager in most of today's operating systems.

Software Development

SLDC

The software development life cycle is an organized process of developing a secure software application throughout its life cycle throughout the project.
The seven phases:

Planning and analysis

During this stage, the goals of the software project are determined, the stakeholder needs are assessed, and all of the high-level planning work is conducted. Essentially, this is where things go from a rough idea that someone had for a piece of software into a bit more formalized and well-developed concept that we can plan the rest of our development cycle against.

Software or systems design

It's during this stage that the application or system is defined, outlined, and diagrammed in detail. Essentially, this is where we focus on the overarching inputs and outputs of each function that are going to make up the final software that's going to be released to our customer.

Implementation

During implementation, programmers will begin to code all of the various functions that are needed for the final product. As each piece of the code is developed, the programmers will conduct some basic debugging and testing to ensure that its functionality is working properly. But, at this point, there's been no formal testing completed yet.

Testing

It's during this phase that we get the code and we check it through a myriad of different testing methodologies.

Integration

Whereas in phase four, we focused on testing the individual application or system, in phase five, the integration phase, we're focused on testing the end-to-end service to ensure that all of the pieces and all of the parts can communicate effectively and correctly.

Deployment

During deployment, your application or system will be moved into the production environment where your customers and your end users can now utilize it to perform their work.

Maintenance

This means that once your product is released into deployment, your work is still not finished. Instead, the programmers are now focused on bug fixes, patches, and updates to the version of the software that you're going to end up using.

Agile development

In a strict Waterfall scenario, you wouldn't even be able to add additional features until the initial product was already delivered. This means you're going to have to wait for the next version to release those and get those changes out to your end users. In response to this, a different way of software development has risen in popularity.

This is known as Agile development. Agile software development is performed in time-boxed or small increments to allow it to be more adaptive to changing requirements. In Agile, we still perform most of the phases that make up the Waterfall model, but the big difference is we do them much, much quicker.

DevOps

DevOps is a term created from the words development and operations. This is a way of conducting business where the software developers and the IT operations personnel work closely together to speed up the development and deployment of the applications and to get things out to the end user quicker. Because of the reduced timeline, it's a good idea to embed a security-minded person into the DevOps team as well to ensure that good cybersecurity is not sacrificed in an effort to get the product out quicker.

Testing Methods

The first type of testing is known as **System Testing**.
This comes in three varieties:

black-box testing, white-box testing, and gray-box testing.

Static Analysis is conducted by somebody who understands the language the program is written in and they can analyze the code for errors.

Dynamic Analysis, on the other hand, is performed on a program while it's being run. The most common type of dynamic analysis includes the use of fuzzing.

Software Vulns and Exploits

Backdoors

Backdoors consist of software code that's been placed in computer programs to bypass our normal authentication and other security mechanisms. These are often created by developers themselves in order to make it easier for them to update custom programs in the future. But, this is a horrible practice in terms of security.

Directory Traversal

A directory traversal, which is going to exploit insecurely-coded web applications and servers. A directory traversal is a method of accessing unauthorized directories by moving through the directory structure on a remote server.

Arbitrary code execution

Arbitrary code execution occurs when an attacker is able to execute or run commands on a victim computer. This might occur if someone walks by your desk at work, sees you're logged into the computer, but you're away from your desk. They start running a program on your computer.

RCE

A remote code execution occurs when the attacker is able to execute or run commands on a remote computer. Notice the key difference here between an arbitrary and a remote code execution. With a remote code execution, the attacker can run the commands remotely, such as through an interactive shell session or some other kind of attack. This is considered one of the worst type of exploits in the security world, and a vulnerability that allows this to occur is classified as critical under the Common Vulnerability Scoring System whenever there's a remote code execution that's possible.

Zero-day exploit

This is an attack against a vulnerability that is unknown to the original developer or manufacturer. Because of this, zero-day vulnerabilities have become a big business, with some companies paying thousands of dollars to penetration testers who can help to identify these vulnerabilities and report them under their bug bounty programs.

Buffer Overflow

A stack is a reserved area of memory where the program saves the return address when a function call instruction is received. Here is an example of a stack that's organized as first in, last out.

One of the mitigations against a buffer overflow attack is the use of address space layout randomization, also known as ASLR.
This is a programming technique that helps prevent an attacker's ability to guess where the return pointer for a non-malicious program has been set to call back by randomizing the memory addresses used by well-known programs, such as parts of the operating system.

XSS & XSRF

Cross-site scripting occurs when an attacker embeds malicious scripting commands into a trusted website. When this occurs, the attacker is trying to gain elevated privileges, steal information from the victims cookies, or gain other information stored by the victim's web browser. During a cross-site scripting attack, the victim is the user, not the web server. The web server has already been compromised, possibly.

There are three types of cross-site scripting attacks:

1. stored and persistent
2. reflected
3. DOM-based attacks

Whereas cross-site scripting focuses on exploiting the trust between a user's web browser and a website, cross-site request forgery instead exploits the trust that a website has in a user.
In a cross-site request forgery, the attacker forces the user to execute actions on a web server that they already have been authenticated to. For example, let's say that you've already logged into your bank's website and provided your username and your password. At this point, you're already authenticated and the website trusts you. If an attacker can send a command to the web server through your authenticating session, they are forging the request to make it look like it came from you.

SQL Injection

Well, SQL injection and code injections can be prevented very easily if you do proper input validation and use the concept of least privilege when you're accessing a database from a web application.

XML vulnerability

And so, when you're dealing with XML data, you want to make sure that it's submitted with encryption or input validation. If you submit XML data without encryption or without input validation, it's going to be vulnerable to spoofing, request forgery, and injection of arbitrary code. So, we want to make sure we prevent that.

- XML bomb: Now, this is where they take XML and they use this encoding to encode those entities that I just showed you and expand them to exponential sizes, consuming memory on the host and potentially crashing it.
- XXE: Now, this is an attack that embeds a request for a local resource.

Race Condition

Well, a race condition is a software vulnerability that occurs when the resulting outcome from execution processes is directly dependent on the order and timing of certain events. And those events failed to execute in the order and timing intended by the developer.

One of the most common ones was actually in 2016, and it's known as Dirty COW. Now, Dirty COW is a great example of a race condition that was used to exploit a computer vulnerability. Now, when I talk about COW, I'm not really talking about a cow like you see here on the screen. The COW actually stands for Copy On Write. Now, this exploit affected Linux operating systems in 2016, including some Android versions, because it's based on Linux. The exploit would cause a local privilege escalation bug that could be exploited through this race condition vulnerability because of the implementation of the programming for Copy On Write that was using the kernel's memory management system.