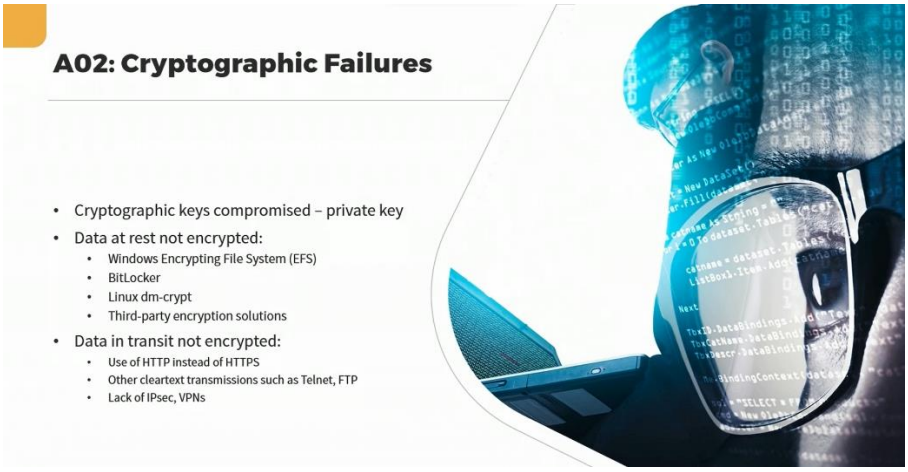
This PDF Lecture is made by
Abid Biswas, Ethical Hacker **(Purple Teaming)**
(CEH, NSE1, NSE2, NSE3,
$ISC^2(CC), Google\ Cyber\ Security\ Professional\ Cert$)
BSc in Information and Communications Engineering (ICE) at EWU

OWASP stands for Open Web Application Security Project. And the OWASP top 10 for 2021 is crucial in making sure that we take the proper steps to secure web applications from common attacks. The idea is that the OWASP top 10 identifies the 10 most common or serious web application security flaws. So, let's take a few minutes and let's go through them, starting with the most prevalent item, A01 Broken Access Control. This can mean numerous things; access control really relates to permissions to access a resource. One way that might be broken is with improperly assigned permissions. The violation of the principle of least privilege, such as providing too many permissions for a user or a group beyond what they need to do to perform a specific task. You could also include web application developers not properly mitigating against Cross-Site Request Forgery or CSRF attacks.

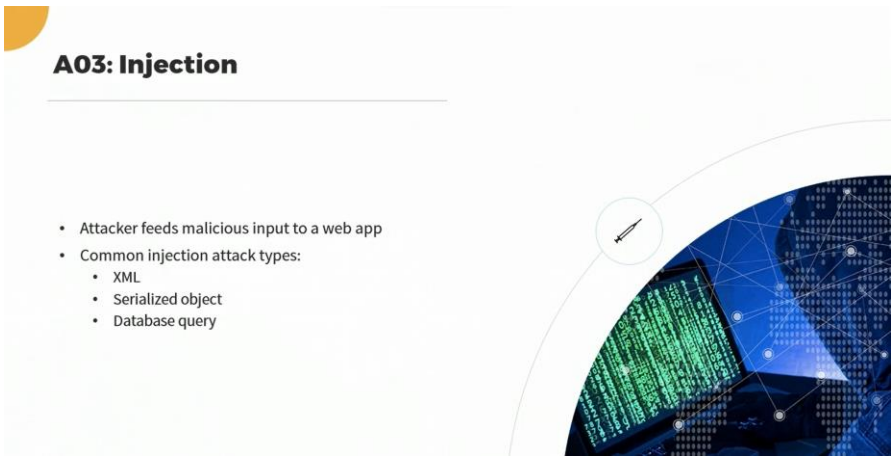
CSRF attacks take place when a user is already authenticated to a trusted website, but the attacker tricks the user, maybe through an email message or an instant message or something like that, perhaps to click a link which causes the user's web browser which is already authenticated to do something unknown to the user. And this can be mitigated using things like Secret Tokens that are also stored server-side to ensure that requests are authentic, basically making sure that certain types of HTTP requests can only be originated from your client-side code running in the client part of the app. Item A02 in the top 10 is entitled Cryptographic Failures. This can include things like cryptographic keys being compromised. So, if you're dealing with public and private key pairs, the public key is called a public key because it can be shared publicly, there's no security risk, but the related private key must be kept secret.



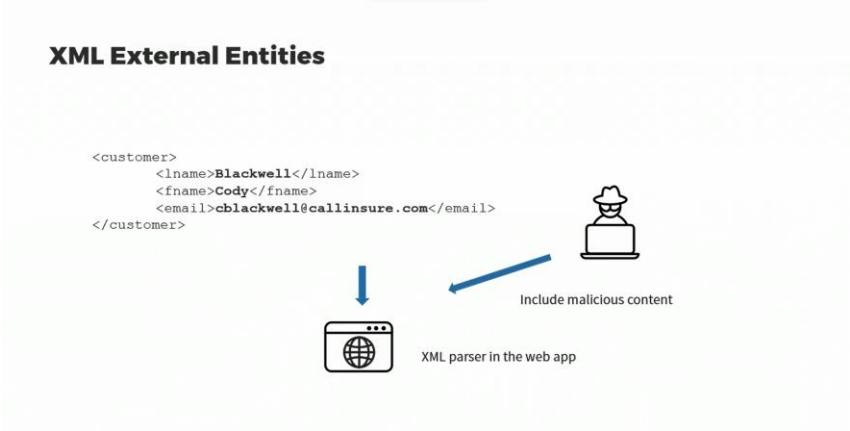
For example, with public and private key cryptography, when a message is encrypted, it's encrypted with the recipient's public key, and they then decrypt it with their related private key. Your private keys are used for things like decryption and for creating digital signatures. Another example of a cryptographic failure is the failure to not encrypt data at rest, using solutions like Windows Encrypting File System or EFS, Microsoft BitLocker, Linux dm-crypt, or any item from a wide array of third-party encryption solutions. Another failure would be to not encrypt data in transit data being transmitted over a network. Such as the use of HTTP instead of using the more secure HTTPS or using cleartext transmission protocols like Telnet or FTP, where the credentials are sent in plain text.

Anyone in that transmission path if they capture that traffic will easily see those credentials. Non encrypting data in transit could also include a lack of using things like IPsec to secure a local area network or not using a VPN to establish an encrypted tunnel from one end point to another over an untrusted network like the Internet. OWASP top 10 item A03 is Injection.

Injection attacks allow malicious users to feed malicious input to web app in some way. Common injection types would include things like XML injections, serialized object injections, database query injections like SQL injection attacks.

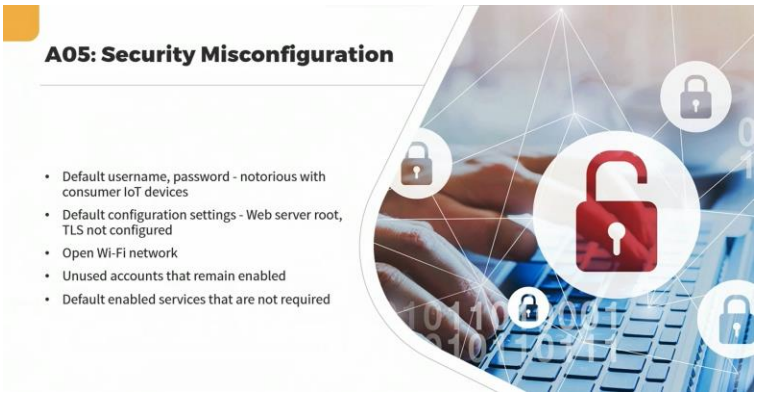


Pictured on the screen, we have an example of an XML external entity type of attack. **XML is a language that's used to describe data, whereas HTML is a language that's used to format data.**



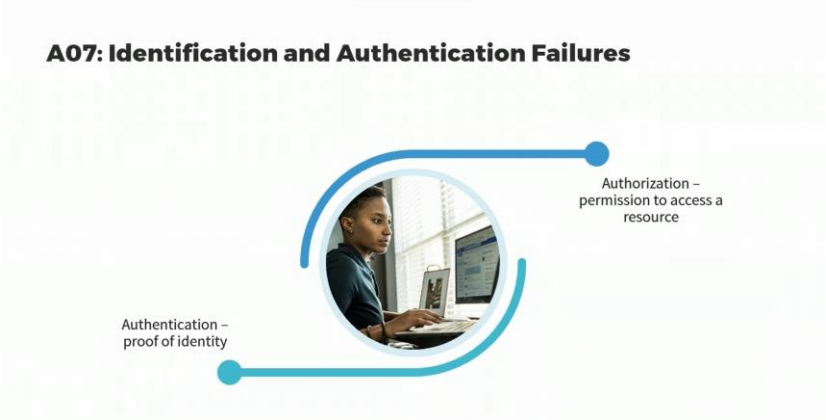
So, let's say we've got some XML representing a customer, and there are three fields, the last name, the first name, and the email address. Now if that XML is being fed into a web application component like an XML parser, which is part of the web app. If a malicious attacker can get to the point where they can inject malicious content into that XML before it's interpreted by the XML parser, they could wreak havoc. They could do things like request sensitive file information potentially from the server that would be then presented to the attacker. OWASP top 10 item in A04 is Insecure Design. Now the thing is securing a solution, a web application is difficult. It's time-consuming and it's expensive but not as expensive than if you have to address security problems after the software is developed. So, when you map out the costs over time, it's always more cost effective to think about security through all design phases right from the beginning.

But insecure designs can result in error messages that disclose sensitive information or credentials that are stored in plain text. Whether it's being transmitted over the network, or whether being stored in a file somewhere, or even embedding credentials in code, may be embedding database usernames, passwords, and connection strings to connect to a database. When you're working in clouds like the Microsoft Azure cloud. Instead, you could use what's called a managed identity, which represents a location that code is running, like a virtual machine that you give permissions to instead of embedding permissions in the code. Item A05 is Security Misconfiguration. This is a big one, it could mean leaving a default username and password with the device, which is notorious with IoT devices. They're designed to be convenient.



People plug them in and forget about them and they still remain potentially visible to the Internet with the default username and password. Sticking with other default configuration settings like the web server root in the file system, not configuring TLS, which really means not using HTTPS connections. Having an open Wi-Fi network on which you've got components used by a web applications. Unused accounts that still remain enabled or default enabled services that aren't required yet they're still enabled. All of these can cause the overall security environment to be weakened, so we need to reduce the attack service. We need to harden our environment, even outside of the web application, even outside of the web application server, the entire ecosystem needs to be secure. Item A06 is Vulnerable and Outdated Components.

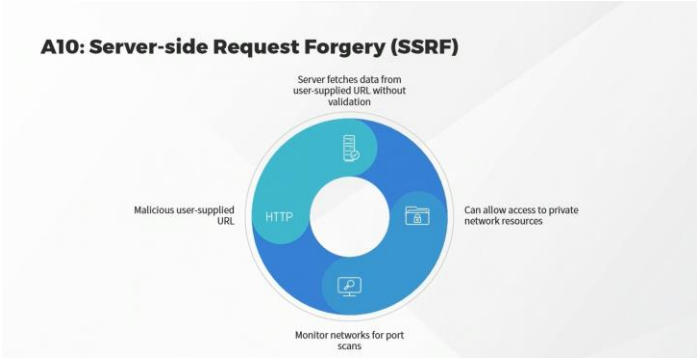
Sometimes software developers will use a third-party component in their web app without having a detailed knowledge of the component and its functionality, or perhaps not making sure that the latest updates have been applied to those third-party components. And this is especially difficult for a developer that inherits a system developed by another team or another developer where they might not be familiar with the fact that this component is being used because it might not have been properly documented. OWASP top 10 item A07 is Identification and Authentication Failures. Of course, authentication means the proving of one's identity. If all we're doing is requiring a username and a password to access a website and we're not even enforcing the use of complex passwords, that's an authentication failure because it might be easily guessed with password cracking tools. Authorization occurs afterwards and provides permission to a resource.



Well, this kind of failure could be related to things like having a web server running in Linux that runs under the root user account as opposed to another more restricted or limited user account. Item A08 is entitled Software and Data Integrity Failures. This can include things like a web app that makes calls to remote code running elsewhere and not verifying those calls for integrity. In other words, perhaps not using digital signatures when we send transmissions over a network that might involve one piece of code accessing another piece of code elsewhere or running a database query somewhere. It could also include software

developers that are downloading code or application containers or other components from unverified sources, and the deserialization of untrusted or unchecked data. Deserialization is related to object oriented programming where you've got a complex item like an object which has numerous methods and properties and it can be serialized so that it can be sent over a network, or it can be stored in a text file using some kind of format like JSON or XML. But deserializing means you're taking that format of that object and you're recreating an object to memory from it, so maybe from a file. And if that file isn't properly secured, if we're not using integrity checking, it might have been injected with malicious code that then become a part of the object during the deserialization process. Item is A09 is called Security Logging and Monitoring Failures. This can result from simply not having enough monitoring and logging of networks, devices on the networks, and even apps. Remember, web application security really is about the entire ecosystem hosting that web application.

It could be a lack of having centralized logging and prioritization of messages. For example, do we want security administrators wasting time looking at simple informational messages about the running of a web app. When instead we should have them look at or have alerts sent to them regarding only things like warnings of potential security incidents or critical performance failures. But then you have to think about the integrity of logging and monitoring data. Is it valid? Is it coming from a trusted source? Is it stored in a secured manner on a secured network? And finally, we have OWASP top 10 item A10 Server-side Request Forgeries or SSRFs. This means that we've got a malicious user supplied URL sent to a web application, sent to the web server. And what the web server will do is without properly checking or validating that user supplied URL, it will fetch data that is specified in that URL.



Potentially, from another internal server on the network where the actual web server resides. So, that means then it could potentially allow access to private network resources indirectly. That might even include attackers trying to run internal port scans to discover other items running internally, so you should be monitoring networks for port scans. Usually port scans are very loud, and they're easily detected on the network. Somebody should be made aware that port scans are running outside of normal acceptable times. If there ever is an acceptable time for ports scans to run on your particular network.