Learn to code group Utrecht security

- ***The Server***= This party is responsible for **serving** pages
- ***The Client***= This party *requests* pages from the **server** and displays them to the user. In most cases the cliënt is a **web browser**
- ***The User***= The user *uses* the **client** in order to surf the web, fill in forms, watch video's online etc.

Each sides programming refers to code which runs at the specific machine; the **server's** or the **client's**.

Server-side Programming

Server-side programming, is the general name for the kinds of programs which are run on the **Server**.

Uses

- •Process user input.
- •Display pages.
- •Structure web applications.
- •Interact with permanent storage (SQL, files).

Example Languages

- •PHP
- •Python
- •ASP.Net in C#, C++, or Visual Basic.
- •Nearly any language (C++, C#, Java). These were not designed specifically for the task, but are now often used for application-level web services.

Client-side programming

Much like the server-side, Client-side programming is the name for all of the programs which are run on the **Client**.

Uses

- •Make interactive webpages.
- •Make stuff happen dynamically on the web page.
- •Interact with temporary storage, and local storage (cookies,local storage).
- •Send requests to the server and retrieve data from it.
- •Provide a remote service for client-side applications, such as software registration, content delivery, or remote multi-player gaming.

Example languages

- •JavaScript (primarily)
- •HTML*
- •CSS*
- •Any language running on a client device that interacts with a remote service is a client-side language.

*HTML and CSS aren't really "programming languages" per-se. They are markup syntax by which the **Client** renders the page for the **User**.

JavaScript Security

Because of the wide-open nature of the Internet, security is an important issue. This is particularly true with the introduction of languages such as Java and JavaScript, because they allow executable content to be embedded in otherwise static web pages. Since loading a web page can cause arbitrary code to be executed on your computer, stringent security precautions are required to prevent malicious code from doing any damage.

Most common JavaScript Security Vulnerabilities

- ***Cross-Site Scripting (XSS)***= a type of computer security vulnerability typically found in web applications. XSS enables attackers to inject client-side scripts into webpages viewed by other users. A cross-site scripting vulnerability may be used by attackers to bypass access controls such as the same-origin policy. They can result in user data theft, account tampering, malware spreading, or remote control over a user's browser.

- ***Cross-Site Request Forgery (CSRF)*=** Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. CSRF attacks specifically target state-changing requests, not theft of data, since the attacker has no way to see the response to the forged request. With a little help of ***social engineering*** (such as sending a link via email or chat), an attacker may trick the users of a web application into executing actions of the attacker's choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth. If the victim is an administrative account, CSRF can compromise the entire web application.

A***n example of CSRF is*** ‘Clickjacking’

Clickjacking, also known as a "UI redress attack", is when an attacker uses multiple transparent or opaque layers to trick a user into clicking on a button or link on another page when they were intending to click on the the top level page. Thus, the attacker is "hijacking" clicks meant for their page and routing them to another page, most likely owned by another application, domain, or both.

Most common SQL Security Vulnerabilities

- ***Injection Vulnerabilities / Flaws***= Application sends untrusted data to an *interpreter*= A program that reads and executes code. This includes source code, pre-compiled code and scripts. Common interpreters include Perl, Python and Ruby interpreters which execute Perl, Python and Ruby code respectively.

  Injection flaws are easier to discover when examining / analyzing source code than via testing.

  Scanners and fuzzers can help detecting injection flaws.
- ***Code Injection***= Exploitation of a combuter bug by injecting code into a vulnerable program in order to change the course of execution. This can result in data loss or corruption, lack of accountability, denial of access / service (DOS, DDOS)
- ***Computer Worm***= Computer program (malware) that replicates itself in order to spread to other computers

Security Measures

- MySQL= Proprietair open source managementsystem for relational databases (RDBMS)
- Data **sanitazation=** the process of removing sensitive information from a document or other message, or sometimes encrypting it so that the document may be distributed to a broader audience
- Data **validation=** the process of ensuring that a program operates on clean, correct and useful data
  - Ensuring that emailaddresses contain a @ sign
  - That only digits are supplied when integer data is expected and that the length of a piece of data submitted is not longer than the maximum expected length

10 ways you can help prevent or mitigate SQL injection attack

1) Trust no-one
2) Don't use **dynamic SQL**= a programming methodology for generating and running SQL statements at run time, when it can be avoided
3) Update and patch!
4) Firewall: web application firewall software or appliance based (physical firewall)

5) Reduce attack surface
   - Get rid of any database functionality that you don't need
6) Use appropriate privileges
   - Don't connect to your database with admin privileges, using a limited account is safer and can limit what a hacker is able to do
7) Keep your secrets secret; assume that your app is not secure and act accordingly by encrypting or hasshing passwords and other confidential data including connection strings
8) Don't divulge more info than you need to; hackers can learn a great deal about database architecture from error messages so ensure that they display minimal info
9) Don't forget the basics; change passwords regurlaly
10) Buy better software; make codewriters responsible for checking the code and for fixing security flaws in custom apps before the software is delivered