Computer Security

A series of notes on the "Computer Security" course as taught by Stefano Zanero during the second semester of the academic year 2018-2019 at Politecnico di Milano.

Ass-Savers

- ***Buffer-overflow***
  - Ogni volta che pushi nella stack vai a occupare indirizzi più bassi (zone alte nel disegno)
  - Quando hai una struct pushi i datatype dal basso verso l'alto, e.g. se ho una struct che dichiara un *char*, un *int* e un *array* $\rightarrow$ pusho prima l'array, poi l'int e alla fine il $char$
  - All'interno di una funzione pusho le variabile in ordine di "spazio occupato", e.g. se all'interno di una funzione dichiaro prima un int $(4$ byte), poi un $char$ $(1$ byte), un array di 5 $char$ $(5$ byte) e un array di 2 int $(8$ byte) $\rightarrow$ pusho prima il $char$, poi l'int, poi l'array di $char$ e poi l'array di int
- ***Firewalls***

  Quando penso a una regola mi devo chiedere sempre : "Chi vuole iniziare la comunicazione??" chiamiamolo X

  la regola sarà nella direzione X $\rightarrow${destinatario}

Definitions

- ***Cross-Site-Scripting***
  - ***Reflected***

    User input is directly returned by the web application in a response (e.g. error message,search result) which includes some or all of the input provided by the user un the request, *without being stored* and made safe to render in the web browser.

    ```
    <?php
    $var = $HTTP['varible_name']; // retrieve content from request
    echo $var //print variable in the response
    ?>

    malicious crafted URL : http://example.com/?variable_name=
    <script>alert('XSS')</script>
    ```

  - ***Stored***

    The attacker(malicious) input is stored on the target server in a database (e.g. comment field) and then is retrieved by a victim from the web application.

  - ***DOM based***

    User provided input never leaves the victim's browser: malicious payload is executed by client side script to modify/update the DOM "environment" (e.g. dynamic pages, forms)

```
...
<script>
    document.write("<b>Current URL</b> : " + document.baseURI);
</script>
...
malicious crafted URL : http://example.com/test.html#<script>alert('XSS')</script>
```

- ***Cross-Site Request Forgery (CSRF)*** DUBBIO! 5/2/2018

  Forces an user to execute a wanted action (state-changing action) on a web application in which it is authenticated (e.g. with cookies)

  Key concept: malicious requests (e.g. crafted links) are routed to the vulnerable web application through the victim's browser: web sites can not distinguish if the requests coming from authenticated users have been originated by an explicit user interaction or not.

- ***What is a prepared statement?***

  A prepared statement is a feature used to execute the same (or similar) SQL statements repeatedly with high efficiency.

  Basically it's just template.

  Prepared statements basically work like this:

  - Prepare : An SQL statement template is created and sent to the database. Certain values are left unspecified, called parameters (labeled "?"). Example: INSERT INTO MyGuests VALUES(?, ?, ?)
  - The database parses, compiles, and performs query optimization on the SQL statement template, and stores the result without executing it
  - Execute : At a later time, the application binds the values to the parameters, and the database executes the statement. The application may execute the statement as many times as it wants with different values

- ***ARP spoofing / poisoning***

  https://www.ilsoftware.it/articoli.asp?tag=ARP-cos-e-e-cosa-sono-gli-attacchi-poisoning_18690

- ***ICMP (Internet Control Message Protocol)***

  It is a protocol able to notify errors and failures without executing any correction

Dubbi

- 5/2/18 Es. 2 (SQL injection) Es. 3 (Firewalls)
-