jQuery Injection, SQL Injection

jQuery injection

- ❑ jQuery simplifies client-side web dev.

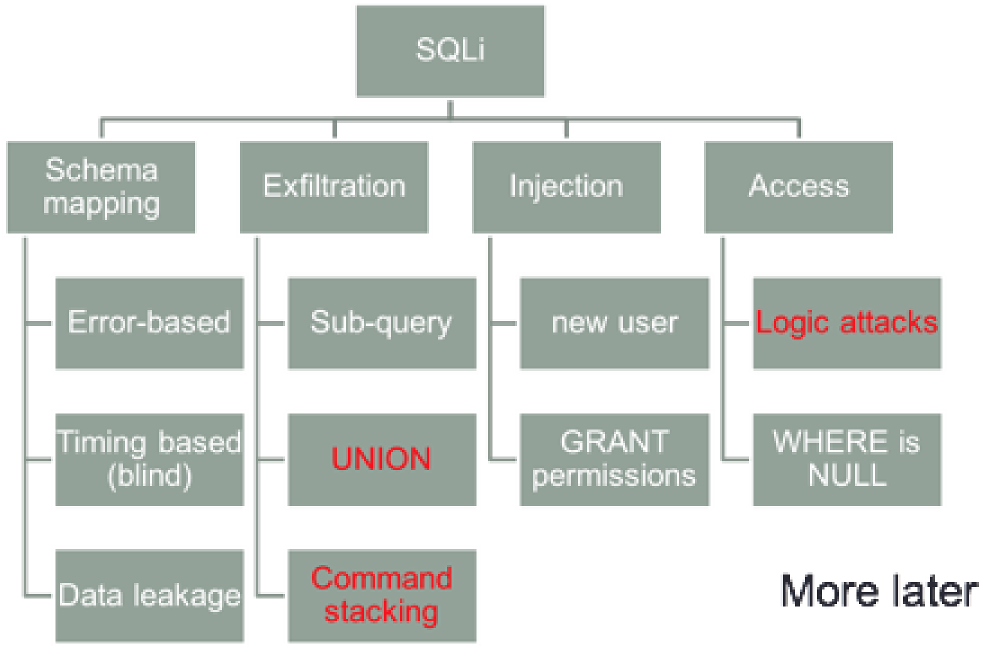
- ❑ Code is very compact, very powerful
  - Easy to hide malicious includes and function calls.
  - Can reveal hidden page contents (passwords, hidden fields, secrets)

- ❑ Solutions:
  - code inspection, sanitise
  - avoid eval(), html(), other unsafe functions

SQL injection

- ❑ Input from web form pasted directly into an SQL command string
- ❑ String sent to database server and executed.

Types:



Passthrough functions

- php and MySQL have functions which let you pass values (e.g. filenames) through to the operating system.
- If not carefully filtered, these allow a user to read, write, upload and download files.
- Defense: Restrict rights of webserver files to read or execute only. Web server must not run as root.
- Validate/filter input.

Best Practices: Web Security

- ❑ Filter
- ❑ Sanitize
- ❑ Validate
- ❑ Both client side **and** server side (Defence in depth)
- ❑ Client side – use well-proven scripts
- ❑ Server side – look-up tables (for validation), filter/sanitise for SQL, BASH, Perl, DOS scripts
- ❑ Restrict Database privileges (later lecture)

XML / SOAP

- ❑ SOAP is an XML protocol for transmitting data and executing functions on remote objects through port 80.
- ❑ It allows application developers allow *remote code execution* despite the port blocking that sys. admins. use to prevent hacking.
- ❑ Shell scripts (OS commands) can be wrapped in XML, passed though port 80 as web traffic, and executed on the server.

Best Practices: Web Security

- ❑ Scripting pages and connection strings must be stored in a web-inaccessible directory. Only the scripts should be able to access the source code of other scripts. Web users must only be able to execute scripts and read html pages.

- ❑ Safe quotes" and other forms of executable script filtering must be enabled.
  The following characters must be filtered or escaped:
  " ' ; - > < ? @ & = (more)

Best Practices: Web Security

- ❑ Only one connection string must be used. It should be stored in a single file and be included into php/asp/jsp/cgi pages as needed. It must not be readable by web users or be stored in a script file. Ideally it should be in a directory which is not accessible to web users.

- ❑ Only the web server user shall be able to access the database. The web server account must be limited. The root password for the database must be set to a non-default value.

Best Practices: Web Security

- ❑ Uploadable content (from forums, chat or discussion boards) must be stored on a server with a different domain name to the one used to serve pages (see youtube for an example). This prevents uploaded user's javascript from accessing the javascript functions from the main web domain.

  –-Enforces javascript *same-origin policy*.

Best Practices: Web Security

- ❑ Transaction systems must not be hosted on a shared host - the same box as other services or other company's web sites/transaction systems/databases.

- ❑ Directory browsing must be disabled.

- ❑ Text input by the user must NEVER be echoed back to the user without filtering (XSS). This includes URL requests, file names, user names, passwords, and all other forms of text or numbers.

Best Practices: Web Security

- ❑ Passwords should not be sent in plain text – use a client-side hashing script or pre-shared encryption key to obfuscate passwords.
- ❑ Or use TLS/HTTPS

- ❑ Passwords should not be stored in plain text.
  - Store password hashes.
- ❑ SQL strings should not be sent.
  - Upload parameters and call stored procedures (pl/SQL) and pass the parameters to them.

Best Practices: Web Security

- ❑ Mixed content including IFrames, mashups, InnerHTML and Frames should not be used.
- ❑ Debug mode and echoing diagnostic messages to the public internet must be disabled on production servers. Such techniques must only be used when the servers in question are not connected to the public internet.
- ❑ No information about the server technology should be revealed to the user. No "row inserted" messages, references to "production server" and so on.
  - No stack traces!

Best Practices: Web Security

- ❑ Don't use a server as a normal desktop - i.e. don't run client-side programs, check e-mail or surf the web from a server. Don't install consumer-grade software such as MS Office.
- ❑ Data passed to script pages (php, asp) should be sent using the POST method. GET should only be used for navigation. On the server side, only information from the Request.Form() collection should be used.

Best Practices: Web Security

- ❑ Login state should be maintained by session variables. These use cookies but are volatile.
  - $_SESSION['user'] = $user;
  - $_SESSION['loggedin'] = '1';
  - Set secure bit on cookie (https only)
- ❑ The use of cookies to record user data should be discouraged. Unexpected input (e.g. strange URL parameters) should cause the session to be dropped; *i.e.* logout and redirect to the home page.
- ❑ Certificates (for https) should be up to date. Expired certificates train the user to ignore security warnings.