Introduction

Open Web Application Security Project (OWASP) is an open project that concerns web vulnerabilities and provides best practices in order to avoid them.

Some principals are:

- Defense in depth: Security mechanisms must be layered (horizontal) and/or structured (vertical).
- Positive security model: Based on *while list*.
- Fail securely: errors and exceptions must be handled in a proper way in order to allow an attacker to exploit them.
- Least privileges: each account must have least privileges for performing their daily routine, ask for more privileges only if **strictly** necessary.
- No security by obscurity: If something is leaked, we are screwed.
- Keep security simple: restricting area to protect in order to focus the attention on small part of softwares.
- Detect intrusion: logging activities on a shared resource.
- Establish secure defaults: like password aging.

There are three main point where the security must be taken into account:

- On server side: data stealing/leaking, RCE, DOS.
- On client side: browsers' vulnerabilities, user info stealing.
- Protecting the network between these two entities: eavesdropping, active attacks (MITM), ISP vulnerabilities.

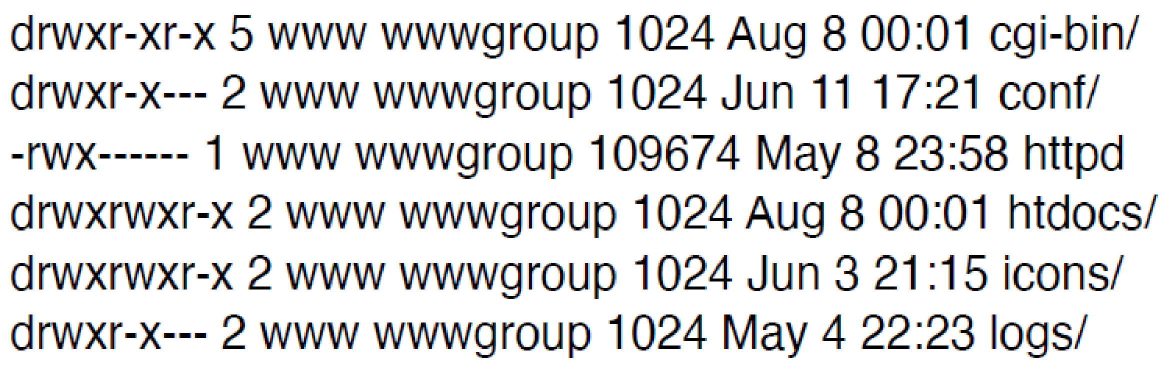
Security at server side

The vulnerabilities are usually related to the hosted applications/services. An example is a Web Server exposed over the internet.

On a server there are two important directories:

- document root: folder in Web server designated to contain Web pages (typical names: *htdocs, httpdocs, html, web, www*).
- server root: contains logs and configurations.

We must be aware of permissions given to directories and we must take into account what kind of content a folder is meant to contain. A good idea is to define correctly username and group.



```
drwxr-xr-x 5 www wwwgroup 1024 Aug 8 00:01 cgi-bin/
drwxr-x--- 2 www wwwgroup 1024 Jun 11 17:21 conf/
-rwx------ 1 www wwwgroup 109674 May 8 23:58 httpd
drwxrwxr-x 2 www wwwgroup 1024 Aug 8 00:01 htdocs/
drwxrwxr-x 2 www wwwgroup 1024 Jun 3 21:15 icons/
drwxr-x--- 2 www wwwgroup 1024 May 4 22:23 logs/
```

In this example we can notice that the root user is www and the related group is wwwgroup. Only www user has read, write and execution permissions on all the folders. The wwwgroup, instead, has read and execute on all the folders and write only content folders (htdocs and icons). The other users, i.e. the ones that will access the web server from the internet, have the most strict permissions, i.e. read and execute only on contents. This permissions setup must be recursive for sub folders.

Notice that many web servers allow selective access to portions of document root, based on browser's IP or authentication. But if we rely on the above setup for permissions, local user will be able to access all the pages since o+rx. A possible solution is to run server processes with id different from **nobody** (the one related to "other"), with minimal privileges and belonging to wwwgroup: g+rx and o+rwx for "restricted" sections.

Most web servers have additional capabilities that can increase the risk:

- Automatic directory listing.
- Symbolic link following.
- Server side include.
- User maintained directory.

Directory listing is a web server function that displays the directory contents when there is no index file in a specific website directory. This allows an attacker to know backup files left, temporary folders and files, symbolic links never removed, source code control logs, etc. Disabling the directory listing does not prevent from guessing file names!

Symbolic links extends the document tree on the server. They are really useful if used correctly. But sometimes symbolic links could bring some problems by the way they are used, i.e. reaching some specific folders that must be protected. Apache can be configured to follow a symbolic link only if the owner of the source folder and the destination one coincides. Also "aliases" could bring problems.

Server Side Includes (SSI) is a simple interpreted server side scripting language used almost exclusively for Web. It is mainly used for dynamically including the content of a file inside another, while the latter is being served by a Web server. An attacker could exploit some vulnerabilities of a Web server in order to dynamically load some server file that must be protected. There is also an exec directive that could allow an attacker to execute whatever he wants at server side, pay attention.

```
<!--#directive parameter=value parameter=value -->
```

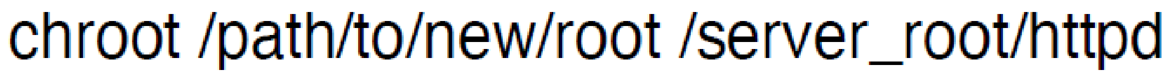
An SSI command is inserted in a .shtml file within an html comment. After interpreting it, SSI replace the comment with the output of the executed operation (include, exec, echo...).

A web server could offer the possibility to an user to create and maintain its own directory(ies) in its local filesystem. Indeed, an user can add to document root personal portions of content (~userid folder in apache). This is a potential breach in security due to unknown malicious operation that an user could perform. If this service is activated it is even more urgent to disable SSI and symbolic link following.

(Userid of server process) Typical scenario is when a Web server runs as root and, when a new connection arrives, it forks a new process running as nobody or similar. If the child process runs at root, well, the user could perform whatever malicious action he wants. Another scenario is when not even the parent is root! In this case it won't be able neither to open port 80 (only high port numbers).

In some cases it is really useful to share www and ftp areas. This approach is fine as long as remote users cannot upload a file via ftp that can be read or executed by the Web daemon. For example, the file could be not readable from nobody user, avoiding the possibility to upload a .cgi file.

A useful command is *chroot*:



```
chroot /path/to/new/root /server_root/httpd
```

It changes the root folder of the current running process and its children. We can create a tiny file system where the Web server can find all the resources it could need. A bare environment will make the approach effective, i.e. avoiding interpreters, shells, passwords or configurations files.

Another good practice is to isolate the server and put it outside the area protected by the firewall. If the server is attacked, the LAN is safe. Do not install www server and firewall in the same host!

- Dual Homed Host: it is an host having two network adapters, the first one connected to an untrusted network, the other one to a trusted local private network. Users of both networks can use applications installed on host. It can decide at application level if routing a packet from a network adapter to the other. In order to configure a "passage" a proxy must be installed on the firewall.
- Screened Host: it is a screening router that forwards all inbound traffic to a Bastion Host and only allows the Bastion Host to send packet outside. Moreover, few local hosts could be allowed to access directly the internet. The Bastion Host is the unique host reachable from the internet. It is MASSSIVELY protected. It uses a secure operating system and no unneeded software are installed on it like compilers or interpreters. It is characterized by a read-only file system and a lot of checker tools to check its integrity. In order to open a "passage", we must setup some rules allowing traffic on specific port/services.

In order to recognize a compromised server, we can use an host-based detection system, i.e. a software system controlling and analyzing the state of a computer system. It periodically checks log files (accesses and errors) for system commands like rm, login and /bin/sh, very long URLs, repeated and failed attempts to guess passwords.

Compromising client side

Attackers try to exploit vulnerabilities on clients for controlling the host, capturing data, deleting files and using host for attacking other hosts. This can be easily done by exploiting browsers' vulnerabilities: attackers can create a Web page that will install Trojan or spyware.

Trojan Horse

They are apparently useful application or tool, containing hidden features that can exploit user privileges. Often installed by unaware users or intruders. The success of an attack is based on the possibility that an user will run the Trojan by himself.

Spyware

Software that collects data of a computing system without owner's consent. Collected data are often sent to subjects interested in selling data for purpose like financial crimes, identity theft and spam. Spyware underlines the non consensual collection of data. Pay attention on software licenses that sometimes hides some tricky rule on user's data management. Spyware can be exploited through hijacking browsers' sessions, cookies, bots, false anti-spyware tools. The information that they are interested in are Internet activity, email contacts, clipboard contents, keystrokes, network traffic etc.

Vulnerabilities in browsers

Browsers' vulnerabilities are increasing rapidly due to increasingly complex Web Sites. Users should be able to evaluate risks related to software usage, but most of them doesn't want or can't understand how interact correctly and securely with softwares.

For example:

- Users don't really check links.
- Unexpected redirections.
- New functionalities in browsers at cost of lower security.
- Vulnerabilities discovered after publishing softwares.
- Web sites may require enabling specific functionalities.
- No capability to setup a browser in a secure way or users don't want to activate some security measures in order to make browsers more secure.

Notice that more functionalities browsers offers, wider is the "surface area" of interest for an attacker.

Frequently used technologies:

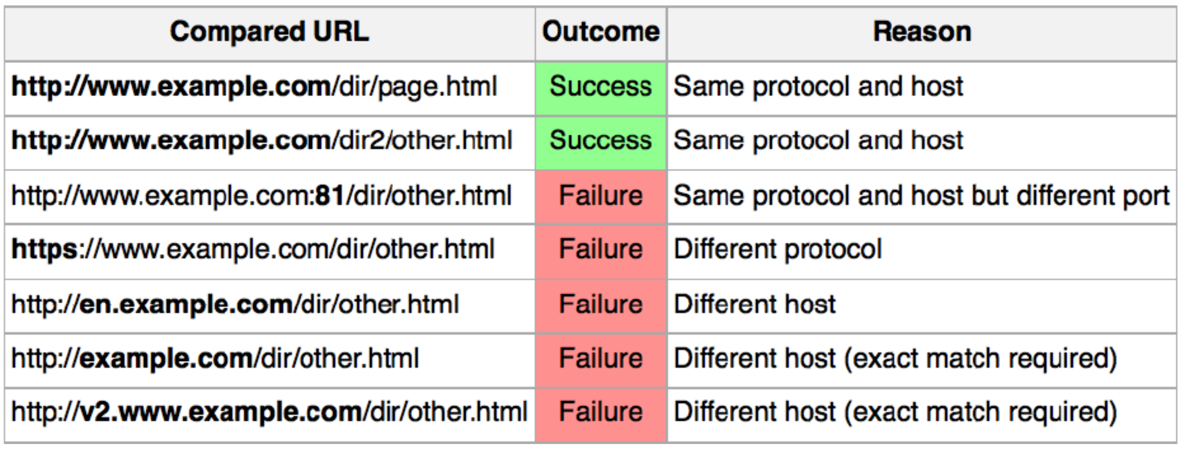
- ActiveX
- Java
- Plug-ins
- Cookies
- JavaScript

ActiveX technology allows the browser to use applications or parts of them. It is not a programming language but an extension of a application. ActiveX controls are files containing commands and functions which can be imported in a browser in order to exploit them. They can access the whole Windows system.

Java Applets are provided by Web sites and executed by a Java VM instanced by the browser. In principle, they are less dangerous than ActiveX because running inside a sandbox. The problem is how securely was implemented the sandbox itself.

Plug-ins are applications that have been thought for being used in browsers. E.g. Adobe Flash. They are executed only inside the browsers. They are subjected to attack techniques like buffer overflow and can perform cross-domain violations when the "same-origin" policy is not complied.

Same Origin Policy: the "origin" is defined by the domain name, application layer protocol and port number of the HTML document running the script. If these value are the same, two resources are considered to be part of the same origin. These are some examples where it is shown how the policy acts.



Compared URL	Outcome	Reason
http://www.example.com/dir/page.html	Success	Same protocol and host
http://www.example.com/dir2/other.html	Success	Same protocol and host
http://www.example.com:**81**/dir/other.html	Failure	Same protocol and host but different port
https://www.example.com/dir/other.html	Failure	Different protocol
http://**en.example.com**/dir/other.html	Failure	Different host
http://**example.com**/dir/other.html	Failure	Different host (exact match required)
http://**v2.www.example.com**/dir/other.html	Failure	Different host (exact match required)

A cookie is a file that is stored on client side for storing data useful to specific Web Sites. The information which is saved is decided by the Web Site itself. They can include visited pages, personal information and whatever. The cookie is only readable by the Web Site that created it. There are two kind of cookies:

- Session cookie: it lasts for a session, i.e. when the tab is closed or the session expires, it is deleted by the browser.
- Persistent cookie: its duration is related to an expiration date set by the Web Server.

They were originally introduced to implement shipping carts. They are now used for mane many things like tracking users and their activities on the Web Server, for collecting data, to implement the "Remind me" option.

Web pages can host contents coming from other Web servers. The cookies coming from those servers are called *third party cookies*. Some organizations (like Google) collects a lot of data about users and provides capabilities to other web sites to request these "anonymous" data in order to show personalized advertisements. These information are shared through third party cookies sent by these organizations based on a signature of our browser. User profiling is considered as a potential threat to user privacy. In fact, nowadays users are asked to accept cookie policies in order to visualize correctly the content of whatever Web Site. Cookies usage for technical reason is allowed by the law, but an usage for tracking purpose must be accepted by the users theirself. There are a lot of problem with cookies usage, we must take care of that.

An alternative to cookies is DOM Storage considered more secure. It stores pairs (key, value) in a secure setting for future use. They could store up to 5MB, instead a cookie can store up to 4KB. They could be persistent without an expiration date. DOM data are not transmitted each time there is a connection with the Web server, avoiding so to send useless data.