REFERENCE SHEET

PROTECTING YOUR WEBSITE

Purpose of the document

Protect your company's website.

Since a company's website is one of the most exposed parts of its information system, it’s very important to make sure it's secure.

The most common threats to websites are defacements and Denial of Service (DoS) attacks:

- Defacement is an attack where an unauthorized person modifies a site by replacing hosted content with their own to send a political message, denigrate the website owner, or simply claim responsibility for the attack.
- DoS attacks are aimed at making sites unavailable to legitimate users.

In either case, the impact on the website owner obviously includes harm to their reputation and, in the case of for-profit sites, an attack can mean financial loss.

Outsourcing site hosting does not transfer all risk of intrusion to the host information system.

Protecting against these threats requires preventive measures and mechanisms to detect attack attempts.

Below is a non-exhaustive list of best practices to enhance website protection against attacks. Be sure to adapt these recommendations to the context.

Architecture

Use the defence-in-depth principle for the website’s hardware and software architecture and hosting infrastructure

- Network filtering
- Web Application Firewall

Application components should be limited to the strict minimum

- Delete unnecessary application components from content management systems.
- Inventory the application components used.

Use secure protocols for site administration

- Use secure protocols such as SSH. Administering the site via FTP protocol is not recommended because it does not protect passwords or the content transmitted.
- Make sure your site is administered via a web interface that uses HTTPS.
- Use complex passwords to authenticate the administration platform.
- Optional: use IP address filtering.

Apply the principle of least privilege

- Associate the HTTP server to an unprivileged user account.
- Only HTTP clients can access essential files.
- Users associated with the web server should not have read (or write) access to files they don't need to access.
- Database access rights should be managed closely. (Use different DBMS user accounts depending on the site components, and only give rights to tables when necessary).

Limit information provided concerning the technical functioning of the site

- Delete any visible items on the page that could indicate the tools used.
- Limit debugging information (e.g. SQL query).
- Use personalized error pages.
- Reject TRACE HTTP requests.

Network filtering

- Establish an inflow and outflow matrix (ports and destinations).
- Use HTTPS protocol for communications between the client and server.
- Use the HSTS mechanism to indicate to browsers that requests to the site in question must always be sent using HTTPS.

Application

General principles

- All access to pages requiring specific access rights must not be managed on the client's side.
- Verification must not be delegated to clients.
- Data received from clients must be processed before being interpreting (DBMS, browser, etc.).

Protect sessions

- Make sure session identifiers are random and use at least 128-bit entropy.
- Use HTTPS protocol as soon as specific privileges are associated with a session.
- HTTPOnly and Secure attributes (for HTTPS sites) must be associated with the session ID.
- Establish a session validity period based on business needs.
- Invalidate the session when logging off.
- When sensitive operations are being performed, force re-authentication for old sessions.

Protect authentication management functions

- Prevent usernames from being listed: generic error in case of authentication failure.
- Use methods such as links to reactivate passwords with limited-use random tokens.
- Use secret questions with answers that are not easy to guess (e.g. social networks).

Rights management

- Have nominative accounts.
- Formalize account management (grant, withdraw, change)
- Review rights on a regular basis.
- Don't include permissions by default to all users.

Password storage

- Passwords must be stored in a form transformed by a non-reversible cryptographic function.
- Use a random value (salt) for password transformation
- Store passwords in a keystore or in a Properties file that can only be accessed by authorized persons.

Inclusion of external content

- Limit inclusion of third-party content to a strict minimum.
- Perform basic checks to ensure content reliability.

For more information on protection measures, see **Appendix 1.**

Reactive measures

A point of contact associated with the site must be readily identifiable

- Valid email address in the SOA record so users can communicate with the person responsible for the site.

Monitoring

- The website must be tested regularly (scanning, penetration test) to identify anomalies.
- Regularly check the integrity of the directories that store the files associated with the website on the server in question.
- An investigation should be triggered when new files appear suddenly or when there's a change in server configuration.

Logging

- Establish a logging policy.
- Log the web server and include the following information: IP, date and time of requests, URL requested, error code, HTTP request referrer fields and user-agent field.
- Log the other systems (local firewall, system reboot, administration interface connection).

In the event of an incident

- Gather and preserve all information available to conduct investigations.
- Search for other intrusions (phishing pages, malware insertion, changes to site configuration, installation of backdoors).
- Make sure that the restored website does not contain any malicious items. If a backup is used, it must be clean.

Additional information

OWASP site: www.owasp.org (OWASP secure coding practices quick reference guide)

ANSSI document: https://www.ssi.gouv.fr/uploads/IMG/pdf/NP_Securite_Web_NoteTech.pdf

Appendix 1: Protection measures for the main online attacks

Attacks	Protection measures
SQL injections	Use “PreparedStatement” • Additional implementation details in Java: http://docs.oracle.com/javase/7/docs/api/java/sql/PreparedStatement.html • In .NET, use LINQ to SQL (*Language-Integrated Query*): http://msdn.microsoft.com/en-us/library/bb425822.aspx

XSS	Make sure all the data from external sources included on the webpage are protected, i.e. processed to prevent interpretation in the context it is used.
	
	In Java:
	• Encoding:
	  ○ **OWASP Java Encoder**
	  ○ **TagTags JavaServer Pages Standard Tag Library (JSTL)**
	  ○ **StringEscapeUtils - Apache Commons**
	  ○ **HtmlUtils.htmlEscape() – Spring**
	  ○ **HtmlEscapers.htmlEscapers().escape() – Google Guava**
	• Validation:
	  ○ **BeanValidation**
	In .net:
	• Validation:
	  ○ **RegularExpressionValidator**
	  ○ **RangeValidator**
	  ○ **CustomValidator**

Insecure direct object reference	Randomly generate a non-guessable GUID identifier Generate a UUID (universally unique identifier) in Java Gives you the option of implementing an access control to validate access to the resource
Vulnerable components	Keep up to date and address the vulnerabilities of elements involved in setting up a website (content manager, extensions, languages used, libraries used, Internet service provider software, site administration software, operating system)
CSRF	Add a **token-type** secret • Put the token in a POST or an HTTP header in a hidden field