QUICK REFERENCE

PROTECTING YOUR WEBSITE

Objective

Adequately protect company websites.

Since websites are one of the most exposed parts of a company's information system, it's very important to ensure that the websites are secure.

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

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

In both cases, the impact on the website owner obviously includes harm to their reputation and, in the case of a for-profit site, the attack can represent a financial loss.

The outsourcing of the hosting of a site does not transfer all the risks of intrusion to the host information system.

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

Target recipient: Website administrators

Last update: 2017-06

Preventive measures

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

Architecture

Use the defence-in-depth principle for a website's hardware and software architecture and its hosting infrastructure.

- Network filtering.
- Web Application Firewall.

Only use a bare minimum of application components.

- Delete unnecessary application components from content management systems.

- Make a list of the application components used.

Use secure protocols for site administration.

- Use secure protocols such as SSH. It's not recommended to administer the site via FTP protocol, as 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 for administration platform authentication.
- Optional: use IP address filtering.

Use the principle of least privilege.

- Associate the HTTP server to an unprivileged user account.
- Only HTTP clients can access essential files.
- A user associated with the web server should not have read (or write) access to files they don't need to access.
- Closely manage database access rights (use different DBMS user accounts depending on the components of the site; only give rights to tables when needed).

Limit the information provided on 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.

Filter networks.

- 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 done using HTTPS.

Applications

Main principles

- Don't manage on the client side all access to pages requiring a specific right of access.
- Don't delegate any verifications to clients.
- Process input data received from clients before interpreting (DBMS, browser, etc.).

Protect sessions

- Ensure that session identifiers are random and use at least 128-bit entropy.
- Use HTTPS protocol as soon as specific privileges are associated with a session.
- Associate HTTPOnly and secure attributes (in the case of an HTTPS site) with the session ID.
- Determine a session validity period based on business needs.
- Invalidate the session when logging off.
- For sensitive operations, force re-authentication for old sessions.

Protect authentication management functions.

- Prevent the listing of user names: generic error in case of authentication failure.
- Use methods such as links to reactivate passwords with limited-use random tokens.
- Use secret questions whose answers are not easy to find (e.g., social networks).

Manage access rights.

- Hold nominative accounts.
- Formalize account management (granting, withdrawal, change).
- Review rights on a regular basis.
- Don’t include permissions by default to all users.

Store passwords.

- Store passwords 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 is only accessible by authorized persons.

Include external content.

- Limit third-party content inclusion to a bare minimum.
- Conduct elementary checks to ensure content reliability.

Fix vulnerabilities

- Keep up to date and address vulnerabilities of the elements coming into play when setting up a website (content manager, extensions, languages used, libraries used, internet service provider software, site administration software, operating system)

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

Reactive measures

Make the point of contact associated with the site readily identifiable.

- Make sure the email address in the SOA record is valid so users can communicate with the person responsible for the site.

Monitor.

- Test the website regularly (scanning, penetration test) to identify anomalies.
- Regularly check the integrity or the directories that store the files associated with the website on the server in question.
- Make sure that an investigation is triggered when new files appear suddenly or when there is a change in server configuration.

Create logs.

- 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 case of an event

- 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. Only clean backups should be used, where applicable.

Additional information

- OWASP website: 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 (in French only)

Appendix 1: Protection measures for main online attacks

Attack	Protection measures
SQL injections	Use “prepared statements” • Additional implementation details in Java: http://docs.oracle.com/javase/7/docs/api/java/sql/PreparedStatement.html • In .NET, use LINQ to SQL: http://msdn.microsoft.com/en-us/library/bb425822.aspx
XSS	Ensure that all the data from external sources included on the webpage are protected, i.e., it has been processed to prevent interpretation in the context where 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 Generation of a UUID in Java Possibility of implementing an access control to validate access to the resource
Vulnerable components	Keep up to date and address vulnerabilities of the elements coming into play when 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