TECHNICAL STUDIES

# Web application security

## Managing web application security risks

March 2010


Web application security working group

CLUB DE LA SECURITE DE L'INFORMATION FRANÇAIS

30 rue Pierre Sémard, 75009 PARIS

Tel.: +33 1 53 25 08 80 – Fax: +33 1 53 25 08 88 – e-mail: clusif@clusif.asso.fr

Web: http://www.clusif.asso.fr

# ACKNOWLEDGMENTS

CLUSIF would like to thank those people who made this document possible, particularly:

| | | |
|---|---|---|
| Michel | **Frenkiel** | *Mobilegov* |
| Sébastien | **Gioria** | *Groupe Y* |
| Philippe | **Larue** | *CBP* |
| Vincent | **Maret** | *Ernst & Young* |
| Michel | **Maximin** | *Crédit Logement* |
| Enrick | **Moulin** | *BT CyberNetworks* |
| Ludovic | **Petit** | *SFR* |
| Philippe | **Sarafoglou** | *PSA* |
| Herve | **Schauer** | *HSC* |
| Arnaud | **Treps** | *Accor* |
| Patrick | **Vanhessche** | *ADP* |

We also thank the CLUSIF members who contributed to the revision of the document.

# TABLE OF CONTENTS

# I - Introduction

This document is intended for directors, decision makers and IT managers responsible for the implementation and/or operational maintenance of an internet website or internal web application. The document presents the security risks of web applications and describes best practices for IT project management and risk management in order to control these risks during the development and life cycle of the application.

This document is not intended to describe in detail concepts, techniques or security tools. These will be examined by another CLUSIF working group, and result in an additional, complementary document.

# II - The indispensability – and new risks – of web technologies

## II.1 - Omnipresence

The activities of businesses and governments rely increasingly on web technologies and applications. The ease of implementation and use of these technologies has made them an omnipresent and essential component of online commercial sites, intranet and extranet applications, as well as the internet services offered and used by companies. Today, new applications are almost systematically developed with web technologies, while older applications are often made accessible via a web browser.

## II.2 - Vulnerability

Legally-sensitive and critically important company information is now managed using web applications in a context where theft of sensitive data, website intrusion and denial of service (DoS) attacks regularly make the headlines [1], [2], [3]. According to studies conducted by consulting firms, a large majority of websites are vulnerable to attacks. Two important trends have emerged on the security market in recent years:

- Attackers no longer act for reasons of personal prestige but for financial gain through fraud;
- Web applications have become a target of choice for hackers/hacking operations. The Gartner Group estimates that 75% of attacks now target these applications.

Surveys conducted in the U.S. suggest that over 60% of clients would cease to do business with a company if their personal data were at risk following a computer attack. Recent media coverage of a DoS attack on a commercial site demonstrated the impact that such an event can have. It is easy, then, to imagine the financial, legal and reputational damage that could be caused by a web application not initially designed, developed, installed or used in a secure manner.

## II.3 - Regulation and responsibility

In terms of computer security, organisations face a growing number of responsibilities, as legislators, regulatory bodies and clients demand an increasing degree of data protection and traceability from companies.

In light of the new severity of attacks on web applications, their growing frequency and the problems they pose, industrial regulation and standards have been reinforced. New solutions such as the PCI Data Security Standard (PCI-DSS) address the issue of web application security. Recent regulation stipulates that companies must ensure the development and maintenance of secure systems and applications, paying particular attention to web application vulnerabilities. In the United States for example, the state of California enacted the Notice of Security Breach Act (California Civil Code section 1798.82 1) (former Senate Bill 1386), which requires that verified or suspected breaches of client data privacy be disclosed to affected parties and made public.

For this reason it is no longer possible for either service providers, software publishers, subcontractors and developers, or user companies and clients to think of cyber security without also thinking about legal and regulatory frameworks. The way in which a web application is designed and developed can affect the accessibility, security and privacy of data. As a result, offering an application-based service can expose a company, a software publisher or application developer to civil [4] and/or criminal [5] lawsuits.

## II.4 - The role of management

Web application security can no longer be ignored. This is a management issue requiring special attention that goes beyond traditional solutions.

Given the increasing complexity and frequency of web application risks, decision makers must take action to sufficiently protect the web applications developed and/or used by their companies. As is done for quality, management needs to define security objectives, delegate related tasks, and ensure these objectives are met. Neglecting to do so would expose them to considerable risk on today's market.

Web application security must be factored in as early as the design stage, during development and integration, and throughout the life cycle of the application. It must be an integrated component of the application and not be added on at the end of the development cycle. Beyond considerations of design and implementation, web application security must also be tested and checked before production and as it evolves.

While technical aspects, tools and methodology are necessary to secure a web application, also important is the human factor. Management teams must therefore be sure that stakeholders, general contractors, project managers and developers understand what is at stake and know and use best practices.

# III - Web application security: Myth and reality

Even if a management team is generally aware of the security risks associated with web applications, some ideas remain a stumbling block to fully understanding the problems and making the right decisions to protect a web environment. The stance taken by certain security solution vendors, the desire on the part of web professionals to reassure clients, and efforts to explain web security to the general public have helped feed several dangerous myths.

## *III.1 - Myth #1 – "The developer will provide me with a secure solution without me asking"*

Web application security is not a given. To the contrary, if specific actions are not taken, the solution used most likely contains vulnerabilities which may affect the confidentiality, integrity and accessibility of the application and data involved. What applies to any application does even more so to a form of technology originally designed to allow completely free access to information, wherein authentication mechanisms, session management tools and controlled access are not built in.

Product clients must therefore ensure that security best practices are understood and applied by the general contractor, and be able to check how secure an application is (see the chapter below on security verification).

## *III.2 - Myth #2 – "Only hackers know how to exploit web application weaknesses"*

To the contrary, security weaknesses in web applications are often easy to exploit. A hacker often only needs a basic web browser to identify and exploit security weaknesses. Additional tools such as local proxy programs, capable of intercepting requests between a browser and web server, are available for free online.

Furthermore, web technologies are based on a series of protocols, languages and open architecture, the specifications of which are freely accessible, making it possible for anyone to study these references and the exchanges between a web application and clients in order to identify implementation or design flaws.

## III.3 - Myth #3 – "My website is secure – it's protected by SSL"

SSL technology was created to prevent sensitive data such as payment details from being transmitted in plain text across the Internet. All commercial sites now display SSL certificate logos. When e-commerce took off, the media told users to only trust a site if this famous padlock appeared in their web browser status bar. It is easy to see the source of the (misguided) belief that a SSL-certified site is necessarily protected.

While SSL as a security mechanism is needed to protect the confidentiality and integrity of data exchanged between client and server, it does not protect web applications 100%, particularly from attacks contained in network communications sent by the client to a server through the SSL protocol.

## III.4 - Myth #4 – "My firewall protects me from attacks"

When companies originally began using Internet, they installed security features at the network level – firewalls, for example. If correctly configured, these devices remain effective in most cases for blocking non-authorised access to infrastructure on which web applications are hosted, such as network services of operation systems or databases. Firewalls are necessary. Unfortunately, however, they cannot be used to secure a web application because application flaws are exploited via the normal communication channels needed for the application to function properly, and which are therefore authorized by the firewall.

## III.5 - Myth #5 – "Flaws in internal applications are not important"

An internal application not published on the Internet may be less vulnerable than an internet site which is accessible 24 hours a day to hundreds of millions of users. If it contains flaws, however, it is vulnerable to attacks by anyone with access to the internal network and a simple web browser.

What's more, the use of web technology for internal applications makes the web browser a tool of choice to access both a company's external and internal web resources. Situations may then arise in which an internal user uses the same web browser to simultaneously access a sensitive internal application and an external website possibly controlled by a hacker. Via web technology, a 'malicious' website can then use the web browser of someone accessing this site to transmit requests to another site – including internal ones – without the user knowing. Here, then, is one way an outside hacker may exploit the flaws of an intranet web application via Internet – despite the presence of a firewall – using the web browser as a nexus.

# IV - Primary security flaws

The major security flaws observable in web applications can be divided into three main categories:

## *IV.1 - Incorrect handling of authentication, and access control*

Authentication and access control parameters (submitted by a user or from application to application) very often are incorrectly used, managed and analysed. This can create a risk of identity theft and allow access to illegitimate functions or data, also creating a risk for data confidentiality and integrity breaches.

That risk is exacerbated by the fact that a web server is configured to allow access to published content by default. To restrict specific content to certain users, the developer must expressly install authentication and access control mechanisms for each item (e.g. web pages) requiring protection.

## *IV.2 - Injection vulnerabilities*

Injection consists of inserting (or injecting) specially formulated data into a function, program or script in order to affect its normal execution (for example, to modify a database or read sensitive data). Injection vulnerabilities can affect database access (SQL injection), directory services (LDAP injection), operating systems (command injection) and dynamic web content (cross-site scripting, or XSS).

By injecting invalid data, an attacker can alter the behaviour of the application and even compromise an entire application environment. As a result, availability, integrity and confidentiality of data access could be affected.

Hackers are able to carry out injection-type attacks by using what is called a 'local proxy'. These are freely available on the Internet. They allow someone to inject specifically formulated input into forms and HTTP headers or cookies in order to exploit vulnerabilities, regardless of any security measures built into the web browser.

## *IV.3 - Information leakage*

If the internal functions and components of an application are not sufficiently layered, or if the application displays non-secure data through error messages and comments, sensitive data may be leaked, such as client usernames and accounts, application component types and versions, SQL queries, session information, input data, cookies and even application information. This may lead to a risk of loss of confidentiality and can also provide an attacker with information that would facilitate later attacks (SQL injection, for example).

For further information on web application vulnerabilities, consult the OWASP Top Ten Project [6].

# V - Best practices for creating secure web applications

In light of the security risks a web application presents, it is essential that organisations apply best practices which allow them to develop applications offering a level of security in line with activity-specific risks.

These best practices must be used in a complementary and coherent way by everyone involved in the project. Security must be addressed on a proactive, not reactive, basis, throughout the life cycle of a project – not added on and tested at the end of the development cycle. Taking security into account as early as possible can also optimise costs and development time. The later security is factored into the development process, the more expensive it will be to correct vulnerabilities.

Security best practices must be implemented at every stage of the development cycle:

## *V.1 - Training and building awareness*

Web application vulnerabilities result from best practices not being followed by developers at a given stage of the development cycle, be it design, implementation or integration. Every team member working on a project must be made aware of what is at stake and of security risks, and trained in basic security measures. Each participant is important: general contractors must be able to identify security issues to then identify needs. Contractors and developers must be trained to use appropriate and efficient security mechanisms. Awareness building and training must cover security risks and mechanisms specific to web applications and technologies.

Security mechanisms are constantly evolving, and new vulnerabilities are uncovered each year. It is therefore important that stakeholders receive training on a regular basis. They should also monitor security developments to stay informed of new intrusion and hacking techniques.

## *V.2 - Business requirements and risk assessment*

Assessing security needs is vital. It is at this stage that developers identify functional needs which may have an impact on security, along with security needs specific to the professional activity involved (presence on the internet, sensitivity of user data, 24-hour accessibility, traceability, legal responsibilities, user populations and uses). At this stage, the help of a web application security expert can be useful in identifying security risks and needs.

To ensure coherence with the contractor's goals, a cost assessment can be carried out at this stage using a method such as OpenSAMM, which provides an estimate of costs of the different stages of the development cycle [7].

Risks can then be evaluated in order to model and anticipate threats related to usage context and the functionality of the web application. Doing this ensures that all risks have been taken into account and makes it possible to find appropriate security solutions and measures based on the likelihood and impact of identified potential risks. Threat risk modelling methods and tools are available to facilitate this process. [8]

## V.3 - Design and implementation

Once security needs and risks have been identified, web application security can be designed by defining precisely what security mechanisms will ensure security goals are met and respond to threats (authentication, access control, protection against injection attacks, etc.). A concept document must describe these mechanisms in detail and provide long-term visibility on implementing security measures and managing threats. Security mechanisms must also be designed according to the 'defence in depth' principle, wherein a first line of defence that fails or is attacked is backed up by a second or even third line to protect system resources.

Following the design of an application and its security features comes the implementation phase, in which developers generate source code and assemble components. It is imperative that those involved in the implementation phase be specifically trained in secure web application development. In addition, teams must be given certain tools:

- a reference document containing best practices for secure application development;
- a self-review checklist to ensure nothing has been forgotten;
- an application programming interface (API), or 'security' framework, to avoid having to recreate basic security mechanisms (authentication, filtering data, etc.). Special attention must be paid to the use of APIs and internal/external frameworks, which must be validated and audited to ensure they do not contain vulnerabilities or malevolent code.

Also during this phase, teams can consult a security expert to validate the security mechanisms they have developed. They can also refer to the OWASP Guide to building secure web applications and web services [9].

## V.4 - Acceptance for security

At the end of implementation, while the user's acceptance is performed, the security of the application must be verified to provide practical evidence that it will execute securely during an attack (see the chapter on security verification, below).

# VI - Verifying web application security

## *VI.1 - Why verify?*

Most often, a web application is made up of a complex set of software modules (web server, application server, script engine, database, firewall, reverse proxy) and specially developed source code which manages the user interface, application processing and access to data. Even if security was taken into consideration at the beginning of a project, design, implementation and integration errors are often present which ultimately enable data and processing security breaches. Verifying the security of web applications is therefore essential, and monitoring must be performed throughout the application development life cycle.

## *VI.2 - How to verify*

Several techniques can be used to check whether a web application is secure. Each presents advantages and disadvantages:

### VI.2.1 - Specification auditing

This technique involves looking at threat scenarios and evaluating how the technical architecture and security mechanisms of specifications are able to protect the application, data and processing. It can be carried out at the design stage and before implementation, irrespective of the technologies used. Specification auditing cannot prevent problems from arising during implementation, however.

### VI.2.2 - Code review

Code review consists of analyzing an application's source code (Java code, JSP, PHP, .Net, C/C++, etc.) to ensure that:

- security mechanisms to protect the application are in fact present;
- internal programming conventions have been respected;
- the source code contains no bugs which would allow an attacker to evade security mechanisms.

Code review presents a number of advantages. By analysing source code, a developer:

- can verify compliance with best practices and the defence in depth principle;
- can be reasonably sure that no vulnerabilities are present;
- can detect vulnerabilities that an penetration test cannot;
- can easily identify the necessary steps to eliminate the vulnerability;

Code review can be carried out after implementation, even of only a part of the application.

However:

- All source code must be available in electronic form, which is not always possible (particularly in the case of compiled libraries);
- Sometimes the produced code is different from the audited code;
- It is difficult for the analyst to have a global view of the application and how secure it is based on the source code alone. As a result, this technique is often combined with that of penetration testing.

The OWASP has published a web application code review guide [10].

### VI.2.3 - Penetration testing

Penetration testing is a technique already used in environments unrelated to web applications. It involves exposing the web application to a real attack situation by simulating the actions of a hacker. Tests can be performed:

- without prior knowledge or initial clearances, to stimulate an external attack (known as black-box testing);
- with normal user knowledge and clearance, to simulate an attack by a legitimate, but malicious, user (known as gray-box testing);
- with developer-level knowledge (by making the source code available to the security expert) (known as white-box testing).

A penetration test is designed to:

- detect not only vulnerabilities in the web application itself, but also vulnerabilities (e.g. missing patches, configuration problems) on the platform (OS, web server, database, etc.) hosting the application;
- provide a practical way of verifying the security situation of a given environment from A to Z.

Penetration tests also present some disadvantages:

- They can create denial of service risks in the application, which is problematic when testing a production environment;
- They can only be carried out at the end of the application development cycle. A design flaw detected at this stage can therefore be very expensive to remedy;
- They can not be used to evaluate security functionalities and mechanisms which are not accessible. They cannot be used to test defense in depth features;
- It is sometimes difficult, or even impossible, to cover absolutely all forms of attacks (different types of injection attacks, for example).

For further information, refer to the OWASP Testing Guide [11]. CLUSIF has similarly published a document on non-web application related intrusion tests [12].

### VI.2.4 - Reviewing hosting infrastructure

The security of a web application can depend on how the infrastructure hosting it is configured. An incorrectly configured web server, for example, may allow unauthorized access to it, thus threatening the security of the web application. In addition to a code review, it may be wise to review the parameters of the web server, application servers, databases and firewalls used to execute the application to make sure best practices are being used.

## VI.3 - Automation

Commercial and open-source tools can be used to carry out code review and penetration tests. This kind of software, based on signature bases and vulnerability models, automates tasks and processes significant volume in a relatively short time.

While they demonstrate a certain degree of intelligence, these tools sometimes fail to identify very specific vulnerabilities, particularly at the business logic level. They are also capable of producing false positive results. As a result, these tools must be used by an experienced auditor who can configure them correctly, analyse produced results, and provide additional information if necessary.

## VI.4 - When should web application security be verified?

Too often, web application security checks are still performed just before production, and in some cases, after. Taking this approach can result in considerable costs: if the check uncovers any security flaws that need to be corrected, the cost of these corrections will increase the later they are discovered.

It is therefore better to schedule web application security checks throughout the whole development cycle. The following approach could be used:

- Paper, or hard copy review as early as the design phase;
- Code review during implementation;
- Penetration tests parallel to user tests;
- Infrastructure review prior to production.

A web application must also be tested to see how secure it is throughout its whole life cycle, including when it is modified or new functionalities are added to it.

# VII - Development – the externalisation factor

The development of a web application may be externalized, in whole or in part, for several reasons: to access outside specialized knowledge, save time, reduce costs, etc.

It is important, however, that web application security be dealt with upstream. Clients must describe their security needs in the requirements document and verify that these needs are taken into account. These issues are extremely important when a project is subcontracted. In this case, the following may be demanded of the service provider:

- a commitment to follow a secure development guide and to train their developers in web application security;
- documentation related to the security architecture and mechanisms put in place;
- a commitment to cover the cost of correcting any security flaws that are discovered during the application life cycle in either the written code or in any of the components used (open source or not);

Upon delivery, the client or purchaser should perform a security review on the web application using one of the techniques described in the “verifying web application security” chapter above. This security review is even more important in a subcontracted project, as price pressures can cause a service provider to neglect security. Security must be a key feature of what a subcontractor’s client commissions and validates, before officially accepting and paying for work.

# VIII - References


[1] http://www.lemonde.fr/technologies/article/2009/08/20/130-millions-de-cartes-bancaires-piratees-aux-etats-unis_1230199_651865.html

[2] http://www.zdnet.fr/actualites/internet/0,39020774,39703886,00.htm

[3] http://pro.01net.com/editorial/379143/une-attaque-massive-transforme-des-sites-web-en-nids-a-virus/)

[4] https://www.owasp.org/index.php/OWASP_Secure_Software_Contract_Annex

OWASP Secure Software Contract Annex: This contract Annex is intended to help software developers and their clients negotiate and capture important contractual terms and conditions related to the security of the software to be developed or delivered. The reason for this project is that most contracts are silent on these issues, and the parties frequently have dramatically different views on what has actually been agreed to. We believe that clearly articulating these terms is the best way to ensure that both parties can make informed decisions about how to proceed.

[5] CNIL – Law No.2004-801 of 6 August 2004 on the automatic handling of personal data, modifying Law No.78-17 of 6 January 1978 on computers, files and liberties; Article 35 of the Criminal Code; Articles 226-16 and 226-17

[6] http://www.owasp.org/index.php/OWASP_Top_Ten_Project

[7] http://www.opensamm.org/

[8] http://www.owasp.org/index.php/Threat_Risk_Modeling

[9] http://www.owasp.org/index.php/Category:OWASP_Guide_Project

[10] http://www.owasp.org/index.php/Category:OWASP_Code_Review_Project

[11] http://www.owasp.org/index.php/Category:OWASP_Testing_Project

[12] https://www.clusif.asso.fr/fr/production/ouvrages/pdf/TestIntrusion.pdf


# CLUB DE LA SÉCURITÉ DE L'INFORMATION FRANÇAIS

30, rue Pierre Sémard
75009 Paris
✆ 01 53 25 08 80
clusif@clusif.asso.fr

*Download CLUSIF productions at:*

**www.clusif.asso.fr**