Web Applications Security and Vulnerability Analysis Financial Web Applications Security Audit – A Case Study

Tiago Vieira, Carlos Serrão
School of Technology and Architecture ISCTE-IUL/ISTAR-IUL
Lisbon, Portugal

Abstract

Information security can no longer be neglected in any area. It is a concern to everyone and every organization. This is particularly important in the finance sector, not only because the financial amounts involved but also clients and organization's private and sensitive information. As a way to test security in infrastructures, networks, deployed web applications and many other assets, organizations have been performing penetration testing which simulates an attacker's behavior in a controlled environment in order to identify its vulnerabilities. This article focus on the analysis of the results of security audits conducted on several financial web applications from one institution with aid of automatic tools in order to assess their web applications security level. To help in security matters, many organizations build security frameworks for vulnerability assessment, security assessment, threat modeling, penetration testing, risk management and many more. As for penetration testing, organizations such as OWASP provide vulnerability and security information, a testing methodology, risk analysis and penetration testing tools.

1. Introduction

The finance sector is one with the most valuable assets in information technology. Banking account information, client's sensitive data and transactions are a few examples. They communicate with clients though web platforms and need to insure security and confidentiality. Financial entities are investing in pen(etration) testing, a line of defense in information technology to assert security in applications, systems and networks [1].

A pen test simulates an attacker's behavior (commonly known as hacker) but in a controlled environment to identify and mitigate possible vulnerabilities [3]. A great number of organizations provide frameworks and services to assess security such as pen testing, risk assessment, threat modeling and even teach ethical hacking [4][5][6]. An ethical hacker is a security professional who uses hacking tools and techniques in a legitimate way and with consent from an organization to test and find vulnerabilities in a system [7]. Pen test is used mainly in the end of the software development process. Whether other security software development processes are adopted or not in this life cycle, pen testing will ultimately check software security [8].

This article presents the web applications testing results, its conclusions and evaluates the tested institution in terms of security maturity. The results gathered reflect the audit of real systems developed with security considerations, domain driven development patterns in .Net technologies and with a limited budget and limited development time with inherent project management and SDLC constraints. Based on this security audit, a superficial understanding of other financial institutions can be made and what vulnerabilities emerge in the finance sector web applications that are based on the same development technologies. On top of that, link the vulnerabilities found with OWASP Top 10.

2. Penetration Testing

During the penetration testing, information security specialists access tools and techniques capable of compromising systems, networks or applications in their confidentiality, integrity and availability [3]. The first step before starting a pen test is make sure the rules of engagement are set and the organization formally authorizes the pen test and its conditions [1]. Pen tests can adopt a black box, grey box or white box approach. In a black box approach, commonly the hacker's environment, is where the ethical hacker has no knowledge of the system he is testing. In the white box approach the systems are well known and there might even be access to the source code [9]. Pen testing is more than just finding vulnerabilities, is also the process of verifying if they can be exploited and suggest possible mitigations [10].

A penetration test is a complex and time consuming endeavor, not only that, the time windows to perform the tests is short. Before the test begin, they have to be coordinated and accepted with organizations management, the it administrators have to be on alert for any situation and continuing applications development can't be halted. Typically, web applications are also extensive and for that matter, the tests should be organized and methodical so that the best approach is reached.

There are many vulnerabilities in web applications. OWASP [13] keeps track of the top 10 most critical ones (OWASP Top 10) ordered by risk and probability [11]. This list is updated based on data from several security specialized organizations and individuals. Alongside OWASP, Web Application Security Consortium (WASC) is another institution devoted to the development of security standards [12] that also ranks web application security risks.

There are at a security professional disposal a set of penetration testing methodologies and their use is most important but it's the security team responsible for choosing the one who better suits their needs. A pen testing methodology organizes a testing program and helps organizations prepare an auditing, if applicable [4][5][6].

In this case, the chosen approach was the OWASP Testing Guide. This is the forth release of this open source web testing framework created and maintained by OWASP. OWASP is a nonprofit organization that promotes web security with a vast number of resources produced all connected proving to be one of the best choices in vulnerability testing and risk management. Some examples are the OWASP Code Review Project, the Developers Guide and web scanner OWASP Zed Proxy [4]. The OWASP Testing Guide is a framework exclusive to web security.

3. Web Scanners

A web scanner is a tool built to simplify the pen tester task. They are able to perform automatic attacks to web applications with little or none human intervention [14]. A good web scanner provides similar behavior to a web browser. The functionalities that make a complete web scanner according to WASC defined in the WASSEC are:

1. Protocol Support - Like a web browser, a web scanner must be able to communicate though HTTP and support its protocols, simple HTTP or HTTP over SSL/TLS. There are many browsers and versions, a web application provider cannot guarantee that the client uses the most updated and secure browsers, for that matter, a web scanner performs better if it simulates different browsers and versions.
2. Authentication - Is the way the user confirm he is who he says and it has access to the request he makes through the browser. Most web applications, specially applications with different levels of clearance, have different authentication methods who will make the web scanner useless if it cannot support, for example, HTTP Negotiate or Federated authentication methods.
3. Session Management - During the scan, a "living" session must be maintained with the application at all time. Without it, the scanner cannot perform the crawling or attacks to levels where "in-session" is required.
4. Crawling - One of the main functions of a web scanner is crawling, the ability to discover which pages exist in the web application so that a full test can be made [15]. Web scanners allow fast testing and fuzzing, multiple attack modes and ease the pen tester task. In a black or grey box pen testing where the pen tester has no access to the application source code or when he does not have knowledge of the programming language web scanners are ideal [4]. Despite that, the amount of requests a web scanner can perform automatically is substantially bigger than manual testing. The test time reduction and coverage are two of the most important advantages a web scanner can provide [6].
5. Parsing - Parsing is the ability to read, interpret and comprehend the contents present in web applications. Contents such as Javascript, HTML and Flash. Web applications can have multiple technologies with many implementations. Parsing is one way of identifying vulnerabilities that may exist in the code.
6. Testing - These are the attack components of a web scanner. The greater vulnerabilities type coverage, the better. This is the module responsible for attacking configuration and vulnerability exploitation. The more attacks and procedures a web scanner knows, the better results can be obtained but a low false positives percentage is also a good trey in a web scanner. If the web scanner classifies many findings as vulnerabilities but in fact are false positives, it will require the pen tester to invest much time in validate the findings.
7. Command and control - A web scanner is also a complex tool with many features, a good user interface and significant usability is required for smooth testing. Functionalities like "pause and resume" scans, support multiple users and real time analysis. The usage of the web scanner should be simple so that all real work is in the pen test and not in the learning of usage of the tools.
8. Reporting - A web scanner should allow to create documents to view results outside the tool scope in formats such as ".doc", ".xml" or ".pdf". The scan crucial information should be gathered and classified in a standard report format. If the scanner also provides information about the vulnerability or links to references, it simplifies the communication with it departments for critical vulnerabilities.

Web scanner functions are based on the most common vulnerabilities through techniques such as fuzzing and input testing therefor, even one tool may not be enough. A web scanner can be most effective in certain circumstances and poor in others. Using more than one tool, can provide better confidence in the test results [6].

The most negative aspect of the automated web scanners is the heavy generation of false positives.

These are situations incorrectly classified as vulnerabilities by the web scanner, which require the pen tester to spend much time confirming them [16]. Another crucial point of the web scanner are its configuration options - for instance, if a web scanner can't perform authentication, the web scanner will not be able to pass the login page and therefore complete the web application test [16].

4. Test Environment

This case study includes results from 4 applications with several modules each based on .Net technologies from versions 2.0 to 4.5. The applications were developed by different teams and have different components. Although the applications are accessible outside the tested institution, they are accessible though dedicated secure connections. The performed tests were accomplished in an internal network.

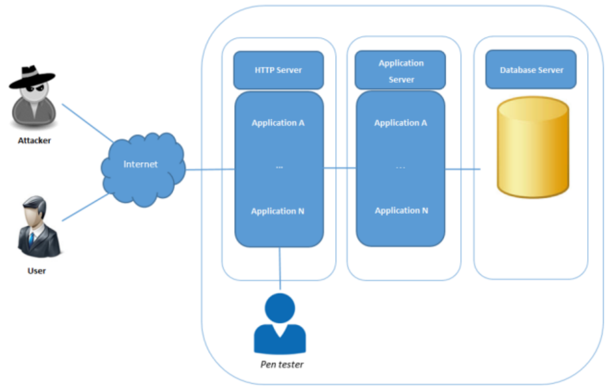

Figure 1. Web applications architecture

The web applications architecture is described as showed in Figure 1. The web applications are organized by layers and hosted in HTTP IIS Server. The access to the data base is made through web services hosted in the application layer. There is no direct access from the web application to the database. Each layer is defended by a firewall. Database access is performed by stored procedures without dynamic SQL and have restricted execution permissions following the principle of "least privilege".

The web scanners used in these tests were OWASP Zed Attack Proxy (ZAP) [17] and Burp Professional [18]. ZAP is a free and no limitation web scanner, Burp is a commercial application and the tests were made in a free trial version with full functionality. Both act as proxies and can perform crawling, create a site tree view, identify and classify vulnerabilities as found with explanations and mitigation suggestions. Both tools can build simple reports with all vulnerabilities and issues found in several formats. An attempt to use also W3AF was made, however W3AF is unable to perform automatic POST fuzzing requests and therefore limited its results.

5. Testing Methodology

This section describes the pen testing per si from the moment the methodology was set and the attacks/testing started. Following a methodology helps a pen tester to prepare its audit, prevents from targets being missed and helps organize the process [8]. In this security audit, the OWASP Testing Guide (version 4) methodology steps were followed but with the help of web scanners.

Pen testing is a try and error endeavor. Every application is different and there are many ways of implementing any functionality. One can only try his best to try to cover all areas in the tests and any the amount of time dedicated to it is short. For the financial web applications security audit, and considering the methodology selected, the major ten tasks to consider were the following:

1. Setting up web scanner configuration - The scanner will register all the pen tester actions on the application acting as a proxy and storing HTTP requests. Setting the web scanner proxy configuration, the technologies used such as programming language, specify which pages are to be considered on the web application, is of key importance to track all the desired targets and avoid unnecessary tests.
2. Navigate through the web application - This is important for the pentester to get acquainted with the application, its purpose, how it is build, and which technologies it supports (such as JavaScript) while the web scanner logs the requests conducted for further analysis.
3. Perform the crawling - Use the scanner web crawler functionality to explore every link it can find in the targeting application. Web scanners are capable of automatically building the entire web application tree structure for analysis and possible attack exploration and vulnerabilities identification.
4. Explore the web application crawled pages - While crawling through the different web application pages, every time it finds new pages, tries to explore them as well.
5. Follow the chosen pen testing methodology steps - Test every aspect of web security as possible keeping notes, test results and report every critical issue found. In this phase, the web scanner can be extremely helpful as it identifies certain security issues just by navigating through the web pages. The web crawler acts as a test accelerator.
6. Perform automatic attacks - Most web scanners have built-in attack capabilities. Using this

functionality, it is possible for the web scanner to test the web application against a series of vulnerabilities in a fast manner. Very useful for quick results and for most common vulnerabilities findings.

7. Perform fuzzing on the web application pages - The attack functionality is great for quick results and a better look and feel of the application but, a manual fuzzing can produce more precise attacks. For instance, focusing SQL injection in the login page. Manual fuzzing allows the pen tester to combine the application inputs with the knowledge from the application behavior's analysis, allowing for targeted or variations on automated attacks, in order to obtain better results from the tests.

8. Explore the application logic parameters - Some parameters have some logic meaning in the application context, and may expect values that the web scanner does not know how to automatically interpret, by opposition to the human pentester. Therefore, it is necessary to combine the scanner fuzzing possibilities with the pen tester knowledge of the web application being tested.

9. Exploitation - In this stage of the methodology it is important to verify if the vulnerability that was identified actually exits and if consists in any danger to the web application security. The level of exploitation should always be agreed with the organization.

10. Mitigations and Reporting - Web scanners can provide useful information to mitigate a vulnerability which should be given to the organization along with the pen tester complementary information. This information is an output from many web scanners such as ZAP and Burp.

As a recommended procedure, during the tests, it is important to report immediately any critical or strange situation found to the security responsible. In order to have better and more complete results from the pen testing audit, it is recommended to repeat the entire process using one or more web scanners as a failsafe.

The following section on this article presents and discusses the results of the tests that were conducted on the different financial web applications, while using the automated web scanners and methodology previously identified.

6. Results and Discussion

This section handles the results obtained during tests. All tests were made in controlled environment that replicate the finals user's platform so that the real system is not affected in performance or availability.

The approach described earlier was followed in order to achieve these results. The amount of logging generated during automatic and manual testing was more that 2Gb of information in requests and responses captured by ZAP and Burp. This amount of data expresses well the advantage of a web scanner usage in terms of performing attacks, identifying and annualizing a great amount of results.

In Figure 2 it is possible to observe the different vulnerabilities that were discovered and confirmed in all the tested applications. For security and confidentiality reasons, no details about the system or provable exploits to the identified application vulnerabilities will be given.

Vulnerabilities listed in Figure 2 are grouped by severity (high, medium, low and information) given by the web scanner during test. The first vulnerabilities listed are the critical ones at the beginning of the chart followed by medium, low and information. The high severity vulnerabilities are also a part of OWASP top 10 and are confirmed vulnerabilities which may compromise one of the security vectors: integrity, confidentiality or availability.

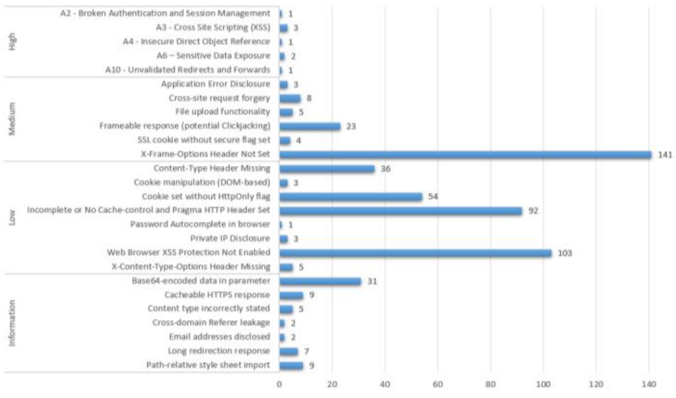

Figure 1. Vulnerabilities Found

As a result, during the pen test auditing were identified and confirmed 26 different vulnerabilities, across 4 applications with several modules each, totaling 554 occurrences.

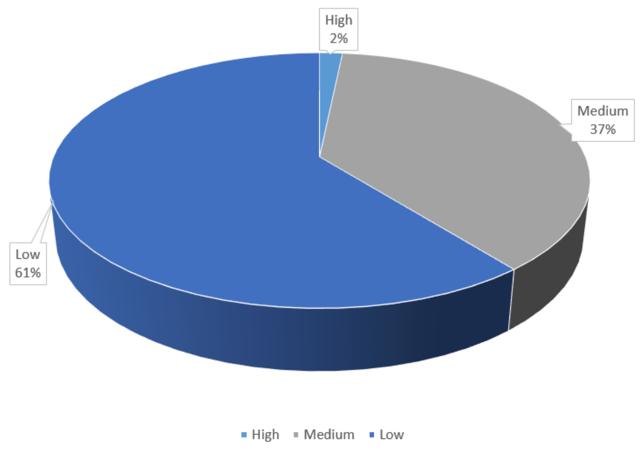

Figure 2. Vulnerabilities severity distribution

The severity distribution of these vulnerabilities is showed in Figure 3. The number of high severity vulnerabilities found was 8, 184 with medium severity and 297 with low severity. This classification so far was calculated by the web scanners by may change if a risk analysis is performed. The information severity vulnerabilities were not taken accountable in this distribution and from here on.

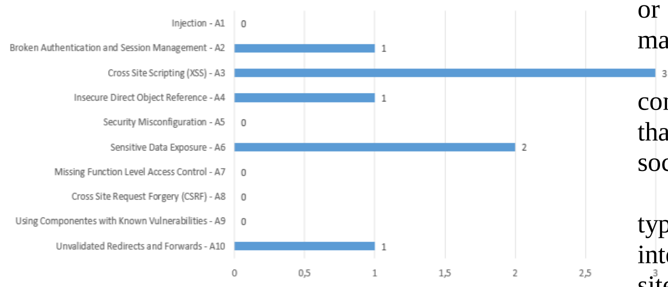

Figure 3. OWASP top 10 findings

The most critical vulnerabilities found within the OWASP Top 10 are identified in Figure 4. These vulnerabilities were classified has having high severity and therefore should be addressed by the web applications security teams as soon as found, due to the exploitability level of the vulnerabilities discovered.

A2 - Broken authentication and session management: this vulnerability deals with the aspects of handling and maintaining sessions in web applications. A vulnerability of this type may allow an attacker to take the user session and access to his data and his profile in the application but local access to the computer is required.

This type of vulnerability may simply rose from an implementation error. For this reason, not only functionalities should be tested but also security oriented testing.

A3 - Cross site scripting (XSS): XSS allows an attacker to send malicious code through the web application, usually as client side code. A successful script execution is a XSS vulnerability. The malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site.

There are several ways to combat this type of vulnerability. First, all input to the server provided from the client should be validated and sanitized. Configurations to prevent this vulnerability can be made at the application level or at the HTTP server. Configuring the HTTP header requests can also prevent the detection of XSS in pages. Current browsers also come with XSS validations but, an application provider cannot guarantee an updated and trusted browser at the client side.

A4 - Insecure direct object reference: is a reference to an object, file, directory or database without access control. Data manipulation can be achieved by exploiting this vulnerability.

This vulnerability can be detected with fuzzing. Every access in the applications should guarantee and verify the user accessing the information has access privileges and clearance to the data he is getting or sending.

A6 – Sensitive data exposure: this is a confidentiality vulnerability where an attacker gains access to private information like credit card number or other information that can be used to other malicious purpose.

A10 – Unvalidated redirects and forwards: consists in allowing page redirect without validation that can lead to phishing or malware sites allowing social engineering.

Unvalidated redirects and forwards is another type of vulnerability that in most cases, requires user interaction. A user can be redirected to a malicious site and through social engineering be misled to disclosure information or credentials.

A8 – Cross Site Request Forgery (CSRF): Although no vulnerabilities were confirmed, this can be explored by XSS. Forged requests can be created and send to users with opened sessions to applications in order to perform an attack.

The mitigation for this vulnerability, at least in the .Net environment is very simple. Using an auto generated token associated in each GET and POST request working as a temporary session for each action can guaranteed no false requests are made.

Cross-site scripting (XSS) was the most recurrent of the OWASP Top 10 vulnerabilities that was found. This kind of vulnerability is ratter dangerous not only because of what mentioned above but also because it can allow to explore social engineering against a user, a conscious user in security matters can avoid many exploits. An updated and modern browser is also an important security measure. In many situations, one institution cannot control what browser the client uses nor keep the web application compliant with new browser versions. More recent browsers already prevent some exploits such as XSS.

In the finance sector, these vulnerabilities may compromise not only systems but also, at a higher scale, compromise businesses. The access to confidential data may leverage competitors in decisions making or attackers to perform fraud and identity theft. The damage in the finance sector ban be monetary or reputational and are hard to calculate [22].

Many kinds of attacks can emerge based on one vulnerability like social engineering can start from an invalidated redirect and forward.

Table 1. Vulnerabilities Found and False Positives

Vulnerabilities	Total	Severity	FP
A2 - Broken Authentication and Session Management	1	High	
A3 - Cross Site Scripting (XSS)	3		
A4 - Insecure Direct Object Reference	1		
A6 – Sensitive Data Exposure	2		
A10 - Unvalidated Redirects and Forwards	1		
Application Error Disclosure	3	Medium	
Cross-site request forgery	8		
File upload functionality	5		1%
Frameable response (potential Clickjacking)	23		
SSL cookie without secure flag set	4		
X-Frame-Options Header Not Set	141		
Content-Type Header Missing	36	Low	
Cookie manipulation (DOM-based)	3		
Cookie set without HttpOnly flag	54		
Incomplete/ No Cache-control & Pragma HTTP Header Set	92		
Password Autocomplete in browser	1		
Private IP Disclosure	3		1%
Web Browser XSS Protection Not Enabled	103		
X-Content-Type-Options Header Missing	5		
	489		

In Table 1 all vulnerabilities are listed as their false positive occurrence. No precision rate can be calculated because we do not know if there are any other exploitable vulnerabilities but the false positives found are only 2% of the results.

7. Risk Analysis

Risk analysis is the process of identifying risks to organizations, in their work and in how they are seen by the world. Part of risk analysis incorporates threats and vulnerability analysis but in order to quantify, a measure is required [23]. You cannot control what you cannot measure [8]. A correct risk analysis allows an organization to evaluate their security maturity and prioritize controls and mitigations to invest in [24].

A risk can be defined by a measure of the extent to which an entity is threatened by a potential circumstance or event, the adverse impacts that would arise if the circumstance occurs and the likelihood of the occurrence [23]. In short, impact versus likelihood. The risk describes "what" consequences the business will experience while the vulnerability explains the "why". It is safe to say, risk analysis should be a part of the software development life cycle [24].

Penetration testing is one of many possible lines of defense in software security. Even in the event of vulnerability finding and mitigation are applied, that may only be false sensation of security. The adoption of a methodology and security controls in all SDLC and organization processes is recommended [8].

Here are some examples of the most recent updated methodologies:

• Building Security in Maturity Model (BSIMM)
• OWASP Proactive Controls
• OWASP Application Security Verification Standard (ASVS)
• OWASP Software Assurance Maturity Model (SAMM)
• CVSS v3

BSIMM: A starting point in security can be achieved by BSIMM, this framework is built by analysis of the state of the art in 78 renamed organizations in matter of security. BSIMM is defined as a measure and not as a guide or checklist.

It simply reflects security practices in different organizations [8]. BSIMM acts as a scorecard evaluation by comparison. The goal of BSIMM as any standard is to benefit organizations with a single measure. BSIMM framework is organized in four domains (governance, intelligence, SSDL touchpoints, deployment), each with three different groups of activities and a total of hundred and twenty activities scattered by the twelve activities. In theory, the more activities are included, the more secure will be.

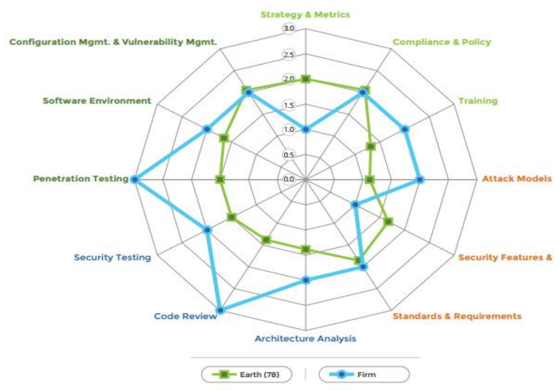

Figure 4. BSIMM deployment domain comparison

Figure 5 shows a comparison between and organization in the deployment domain BSIMM baseline. BSIMM also defines critical vulnerabilities, as those who most organizations evaluated do not expense and are present in successful programs. They are:

Table 2. BSIMM Most Common Activities

Domain	Activity
Governance	Identify gate locations and gather necessary artifacts.
Governance	Identify PII obligations.
Governance	Provide awareness training.

Intelligence	Create a data classification scheme and inventory.
Intelligence	Build and publish security features.
Intelligence	Create security standards.
SSDL	Perform security feature review.
SSDL	Use automated tools along with manual review.
SSDL	Drive tests with security requirements and security features.
Deployment	Use external penetration testers to find problems.
Deployment	Ensure host and network security basics are in place.
Deployment	Identify software bugs found in operations monitoring and feed them back to development.

Penetration testing and code review, are two examples of activities part of the deployment domain and part of the considered successful programs. This concludes that in new initiatives not only should the twelve critical activities be considered but also the other hundred, always according to each organization needs and business.

OWASP Proactive Controls (2016): Similar to the top 10 Vulnerabilities list, the proactive controls are a set of controls to be considered in all software development projects ordered according to its importance. They fill the gap in secure development school teaching and languages and frameworks lack of critical security controls. It is made by programmers for programmers. The document itself is brief and very technical but, none the less very important. The proactive controls are the following, ordered by importance:

	Control
1	Verify for Security Early and Often
2	Parameterize Queries
3	Encode Data
4	Validate All Inputs
5	Implement Identity and Authentication Controls
6	Implement Appropriate Access Controls
7	Protect Data
8	Implement Logging and Intrusion Detection
9	Leverage Security Frameworks and Libraries
10	Error and Exception Handling

These controls may seem covered in the OWASP Testing Guide but, penetration testing occurs late in the SDLC. It is best to have security concerns set from the beginning of the SLDC.

OWASP Application Security Verification Standard introduces a tests requirements checklist for architects, programmers and testers should use to define a secure application. As any other frameworks or standards, it allows vendors to line their necessities by the same measure. ASVS is divided in three levels of security: opportunistic, standard and advanced. Each of them has its own level of security and associated cost. Only through a necessities analysis and its assets valuation can an organization identify which verifications and investment should be implemented [21].

OWASP ASVS first level (Opportunistic) takes in consideration the most frequent vulnerabilities found such as OWASP Top 10 which should be present in any development. The security needs will define what other levels are to be implemented. Standard level was developed for applications with sensitive information. The third and most critical level of security, applies to applications with money transactions or high sensitive information.

OWASP Software Assurance Maturity Model purpose is to help organizations to formulate a strategy for software security [8]. It is built by two layers, the first is Business Functions (BF) and the second, Security Practices (SP). Business Functions are the domains were to apply in the SDLC (Governance, Build, Verification and Deployment), the second layer are validations for each domain.

CVSS evaluation consists in capturing the vulnerability main characteristics and compile a score which reflects the risk severity. The calculated score can be translated to a quantitative scale (low, medium and high) [19]. CVSS is set by three groups, the base group, and two optional, temporal and environmental. The base group represents vulnerabilities that don't change in time, the temporal group categorizes vulnerabilities that change over time and the environmental group considers variables specific to the user's environment.

CVSS is a multi-vector vulnerability analysis that can define vulnerability in such way an institution can understand and prioritize its resolution. Is provides both a qualitative and quantitative risk analysis [20]. CVSS v3 brings a new metric, score. It allows to define what component is compromised by exploiting the vulnerability. Another new important metric is the definition of user interaction needed to explore vulnerability. Attacks like social engineering are linked to this metric.

The final contribution for this study is a classification of the findings. The chosen framework for risk analysis was Common Vulnerability Scoring System (CVSS) v3. This is the latest version of this industry standard released in the end of 2014. Although it is recent, some studies have concluded that this version can provide better risk analysis that its previous version due to new metrics [20].

There can be no vulnerabilities information disclosure but its conclusions can. The most severe vulnerabilities found are the following:

- Cross site scripting (8.0 severity);
- Broken authentication and management (7.3 severity);
- Insecure direct object reference (8.5 severity);
- Cross site request forgery (8.8 severity);
- Invalidated redirect and forward (7.1 severity);

These vulnerabilities share most of these metrics values showed in Table 3.

Table 3. CVSS v3 Evaluation example

Base Metrics	
Attack Vector	Network
Attack Complexity	Low
Privileges Required	None
User Interaction	Required
Scope	Changed
Confidentiality	High
Integrity	Low
Availability	Not affected
Temporal Metrics	
Exploit Maturity Code	Functional
Remediation Level	Workaround
Report Confidence	Confirmed

Each vulnerability access is performed through network with little complexity to perform. The required privileges are none but they require user interaction. The main security vector affected is confidentiality. An analysis based on metrics rather than on numeric values or a high, medium and low scale allows a better understanding with a more detailed analysis and prioritization of mitigations. Broken authentication and session management and insecure direct object reference do not share the user interaction metric since they do not require user interaction.

With limited resources such as time, in order to choose between two risk for resolution, a score is not enough to understand the consequences for management. That is why the ability to describe and articulate the risk exposure is of great importance. It allows risk exploit understanding and what kind of action and time requires [21].

Applying any of these methodologies or any security framework is a time consuming endeavor and for this reason, although recognized these methodologies are strongly criticized [8] but is certain that security should be applied in every moment of the SDLC [4].

8. Conclusion

Security is critical in the finance sector, each vulnerability can be exploited in many ways and compromise monetary or financially the parties involved. Pen testing and important and effective security defense mechanism but the results of a security audit are useless unless mitigations of vulnerability are performed.

Analyzing the results in the web context, even with security considerations in their development, critical vulnerabilities were found. With time and motivation, perhaps even more critical vulnerabilities or with critical consequences could be found. Is most cases, vulnerabilities can only be explored with user interaction. This enhances the awareness that end users should have in security. Both institutions and works must work together to fill this gap. Even aware users, in a demanding organizational world, overwhelmed by large amounts of work and deadlines need to have security well grasped no to miss it even in stressful environments.

The results obtained by both scanners complete each other. This supports the usage of more than one tool. The false positives rate was very low and there is no conclusion to exclude any tool used in this matter. The low number of false positives can also be explained by the web scanners configuration. Since this test used a grey box environment and there was knowledge from the web applications and support structure, the web scanner can better direct the kind of attacks to perform.

This work presents real application pen testing results in the finance sector with .Net technologies and assembles a pen testing methodology since the starting point to mitigations and reporting. A superficial comparison can be made in the finance sector where web applications services based on .Net technologies are developed. Although the .Net Framework has defense mechanisms like injection defense, other vulnerabilities may exist their exploit can be dire to the parties involved.

9. References

[1] M. Walker, CEH Certified Ethical Hacking All-in-one Exam Guide, 2nd ed., McGraw Hill.

[2] A. Razzaq, A. Hur, N. Haider, and F. Ahmad, Multi-Layered Defense against Web Application Attacks, 2009.

[3] M. Buchler, J. Oudinet, A. Pretschner, Semi-automatic security testing of web applications from a secure model.

[4] OWASP Testing Guide v4, https://www.owasp.org/index.php/OWASP_Testing_Guide_v4_Table_of_Contents.

[5] The CIS Critical Security Controlls for Effective Cyber Defense v6.0.

[6] A. Austin, L. Williams, One Technique is not Enough: A comparison of Vulnerability Discovery Techniques. pp

[7] EC Council, http://eccouncil.org/. (Access date: 3 March 2015).

[8] N. Teodoro, C. Serrão, Web application security, 2011.

[9] Y. H. Tung, S. S. Tseng, J. F. Shid, H. L. Shan, W-VST: A testbed for evaluating web vulnerability scanner, 2014.

[10] Choose the best penetration testing method for your company, http://www.techrepublic.com/article/choose-the-best-penetration-testing-method-for-your-company/5755555/. (Access date: 4 April 2015).

[11] OWASP Top 10,http://www.owasp.org/index.php/Category:Owasp_Top_TenProject. (Access date: 12 May 2015).

[12] Web Application Security Consortium, http//www.webappsec.org/. (Access date: 11 March 2015).

[13] OWASP Foundation, https://owasp.org/index.php/About_OWASP#The_OWASP_Foundation. (Access date: 12 March 2015).

[14] A. Doupé, X. Cova, F. Akowuah, A case study on web application security testing with tools and manual testing, 2014.

[15] WASC Thread classification, http://projects.webappsec.org/f/WASC-TC-v2_0.pdf, 2010. (Access date: 26 March 2015).

[16] How to choose a web vulnerability scanner, https://www.acunetix.com/blog/articles/hot-to-choose-web-vulnerability-scanner/, 2010. (Access date: 3 April 2015).

[17] OWASP zed atttack proxy web scanner, https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project. (Access date: 27 April 2015).

[18] Burp web scanner, https://portswigger.net/burp/. (Access date: 4 March 2015).

[19] K. Scarfone, P. Mell, An analysis of cvss version 2 vulnerability scoring, 2019.

[20] A. Younis, Y. Malaiya, Comparing and Evaluating CVSS Base Metrics and Microsoft Rating System, 2015.

[21] E. Wheeler, Security risk management, building an information security risk management program from the gound up, Syngress, pp. 87-103, 2011.

[22] The Damage of a Security Breach: Financial Institutions Face Monetary, Reputational Losses, https://securityintelligence.com/the-damage-of-a-security-breach-financial-institutions-face-monetary-reputational-losses/, 2015.
[23] J. Broad, Risk management framework, a lab-based approach to security information systems, Syngress, pp. 284, 2013.

[24] G. McGraw, S. Migues, & J. West, BSIMM6. Retrieved from BSIMM: https://www.bsimm.com/wp-content/uploads/2015/10/BSIMM6.pdf, 2015.