UNIVERSITY OF PIRAEUS
DIGITAL SYSTEMS DEPARTMENT
M.SC SECURITY OF DIGITAL SYSTEMS

MASTER THESIS

"Penetration Test on Training.eauction.gr"

Georgakopoulos Panagiotis MTE 1609

Piraeus, Greece, February 2018

SUPERVISOR
Prof. Christos Xenakis

Table of Contents

Table of Figures

1 - Introduction to penetration testing

Penetration testing and vulnerability assessment are among the oldest methods of assessing the security of an information system. In the early 1970s, the Department of Defense used penetration testing and vulnerability assessment methods to demonstrate security vulnerabilities in computer systems and initiate the creation of projects aimed at making safer systems. In the early 1990s, with the publication of the dissertation and software called "Internet Security Scanner" by a student of the Institute of Agricultural Technology, the use of penetration tests has seen a great boom in the Internet community. The first penetration testing practices included an IT security technician equipped with few attack tools. When the penetration test was used, repetitive checks were often not performed and the final result was not reliable enough. Over time, however, as control tools and methods evolved, the tests began to be repeated more regularly.

The penetration test is increasingly being used by organizations that want to ensure the integrity of their IT systems and services and prevent exposure to security vulnerabilities. Frequency and severity of intrusions into information networks, data theft and attacks caused by malicious software, hackers and dissatisfied employees are constantly increasing. The number of risks and damages generated by network security breaches and data theft is important. For each new e-business creation, the requirements for secure, remote access to it are constantly increasing. The truth is that even well-managed applications, made up of the latest hardware and software, may be prone to maladjustment and defective software problems, allowing an attacker to access sensitive information. One way to significantly reduce the risk of invasion is to use the penetration test.

The main factors that lead more and more companies to penetration testing are:

- The spread of malicious software (viruses, worms, etc.)
- The complexity of modern networks
- Continuous upgrading of operating systems and software
- The immediate availability of hacking tools
- Image credibility
- Compliance with commonly accepted standards and rules at the security level

2 - Penetration Testing Methodology

2.1 - Overview

The methodology that we adopted for the implementation of the infiltration testing is based on the NIST International Organization for Inspirational Testing Methodology described in the document "Technical Guide to Information Security Testing" with SP 800-115 and is specialized for web applications from the OWASP International Web Application Pattern Testing Methodology described in the OWASP Testing Guide.

According to the NIST standard methodology, penetration tests are divided into four (4) phases:

• Phase 1: Design
• Phase 2: Detection-Valuation
• Phase 3: Attack
• Phase 4: Reporting

The following figure (Figure 1) illustrates the above phases.

Figure 1. Penetration Test Methodology

2.2 - Penetration Testing Phases

Phase 1 - Design

The main actions under this phase are:

• Presentation of the executives that will get involved in the infiltration tests,
• Presentation of the objectives and methodology adopted,
• Identification of the relevant interlocutors of the Customer,
• Structure to be adopted to coordinate work,
• Programming to initiate testing, including adjustment and sequence of phases,
• Defining the framework and strategies for conducting the tests,
• Range of intrusion tests (applications, domain names, ip ranges, etc.)
• Legal and regulatory obligations,
• Determining the type of tests
• Requirements and limitations.

Phase 2 - Detection-Valuation

After the first phase is completed and the penetration test plan is determined, the detection - valuation phase follows. This phase consists of the following stages:

• Reconnaissance
• Scanning and enumeration
• Vulnerability Assessment

Step 1 - Reconnaissance:

The first phase of the project sets the foundations for identifying and analyzing weaknesses. During the first phase, we will collect all information related to the IT infrastructure that is within the control framework.

The following paragraphs describe in detail the types of information to be collected and the approach that we will follow.

• **Public Information:** By public information we mean information available over the Internet (usually at no cost) relating to the networks and customer systems. The audit involves reviewing "Whois" databases (Arin, RIPE, ApNic and InterNic) and other official directories, such as IP and domains, to locate the IP addresses and domain names that correspond to the Customer. As for the DNS service, this is usually a rich source of information, so we will try Zone transfers, trying to get all the recorded information. Additionally, we will search the Internet for any kind of information that would be useful to an attacker. Typical sources of such information are typically dedicated search engines.

• **Normal Known Traffic:** This process involves collecting information through regular information channels provided by the client, such as web and ftp services. Through such channels we will try to extract useful information, such as internal IP addresses, hostnames, e-mail addresses, user accounts and passwords, network services, used protocols etc.

Step 2 - Scanning and Enumeration

Identifying the systems to be audited is one of the crucial steps of the whole process. In this process, special tools such as pinger, fping, hping and nmap are used to create the fullest picture of the network under control. The techniques we follow do not stop in standard network mapping methods such as ICMP and traceroute, but we use every possible protocol (TCP, UDP, ICMP) and in different ways. Another technique that allows us to detect and collect information from mail servers, especially internal ones, is to send emails with non-existing recipients, a technique that usually helps us to discover, through mail headers, IPs and hostnames of internal systems. At the end of this process we will have identified all the systems that will be the subject of our tests. The following techniques are indicated:

• **Full Contact Enumeration:** Once we have detected all the systems under review, we will conduct a complete log of all service services they offer (service scan), ie all TCP, UDP doors that are "open". This process is of particular importance since we will use the network services offered to discover details about the operating system and the services themselves (eg type, version). The various service scan techniques are described as follows:

1. **Standard TCP and UDP scanning:** This technique does not differ from a regular network communication where attempts are made to connect to open system doors.
2. **Stealth scanning:** This technique aims at identifying network services without the attacker being perceived. During this test, we activate specific flags on TCP packets, such as SYN, FIN, ACK, or interrupt the normal TCP session establishment.
3. **Fragmentation scanning:** It is a variation of any kind of service scan, where fragmentation of IP packets is fragmented. The aim of this technique is to prevent the attacker from being perceived by security systems.
4. **TCP reverse ident scanning:** The ident service, as defined in RFC 1413, returns the username from each process using TCP (even if the process did not start the network communication).
5. **Scanning protocol:** Various network services (especially UDP) do not respond to connection attempts unless the attempt is compatible with the expected application-protocol (eg valid DNS query). This technique virtually assimilates these conditions so that the system responds to our efforts, so we can discover the service.
6. **RPC scanning:** RPC services usually do not listen to predefined doors, so they can be traced through the portmapper service, which listens to a predefined door.
7. **Stack Fingerprinting:** Each network system has a TCP / IP stack as part of its functionality. Due to the fact that the implementation of stacks varies from system to system, we can exploit idiosyncrasies in their "behavior" to discover the type of

operating system. The collection of such information is particularly useful as it allows us to focus our tests.

Having identified the network services that are active-accessible to each system, the next step is to collect information through these services. This process includes techniques such as "banner grabbing", where our service gives us information about it immediately after our link. Other services, such as SNMP, SMTP, NetBIOS, are also rich sources of information, which are particularly useful for planning and implementing attacks.

The majority of these tools are specially modified programs freely available from the Internet and others created entirely by us.

Step 3 - Vulnerability Assessment

The analysis of weaknesses is based on known attacks of the last three years available to us, but also on other such publicly available databases. Additionally, the specific tools we used have the capability of automatically analyzing the target, as well as methods of exploiting weaknesses.

We need to mention that commercial tools for tracking weaknesses have several weaknesses. Many of them offer questionable results due to the false indications they are undergoing, a phenomenon stemming from the automated nature of the controls they carry out.

Additionally, the time required from the moment a weakness is disclosed until it is incorporated into these tools is large enough, thus reducing the value of their use. Perhaps, however, the greatest inherent weakness these tools pose are that they are unable to create scenarios of dangerous attacks, where correlates of seemingly minor weaknesses occur. The checks carried out during this phase are:

• General network scan
• Brute force
• Daemons Test - finger, imapd, rlogin, rsh, uucp, etc.
• SNMP
• Router and Switches - UDP, CHAP, PPP, AAA authentication, IOS, etc.
• NFS
• X windows
• RPC Settings and Services
• SMTP settings, services, protocol application relations
• Web - HTTP, PHF, CGI, ASP, HTTPD etc.
• Password - Policy, sources, SSO, cracking, etc.
• Browsers - setting, active content, secure submission etc.
• Anti-Vandals - Java, Active-X, Worms, Viruses, etc.
• Operating system and Services scan
• Operating system and Services Port

• Operating system and service settings, configuration, vulnerabilities, etc.
• Firewall and other security settings, configuration, policy, vulnerabilities, etc.
• Firewall cracking
• Proxy, FTP, socks, DNS - settings, configuration, policy, vulnerabilities etc.
• IP Spoofing - Rsh, Rlogin and many others.
• Windows Server - C2 criteria, authentication, fpnw, registry, LSA and many other settings, configuration, policy, vulnerabilities.
• Operating systems - Groups, users, Networking, NetBIOS, TCP / IP, DCOM, services, file systems, NTFS, NFS, Mounts

For each site to be traced, we will specify the NIST intrusion test methodology according to the OWASP International Web Injection Testing Methodology (Figure 2).

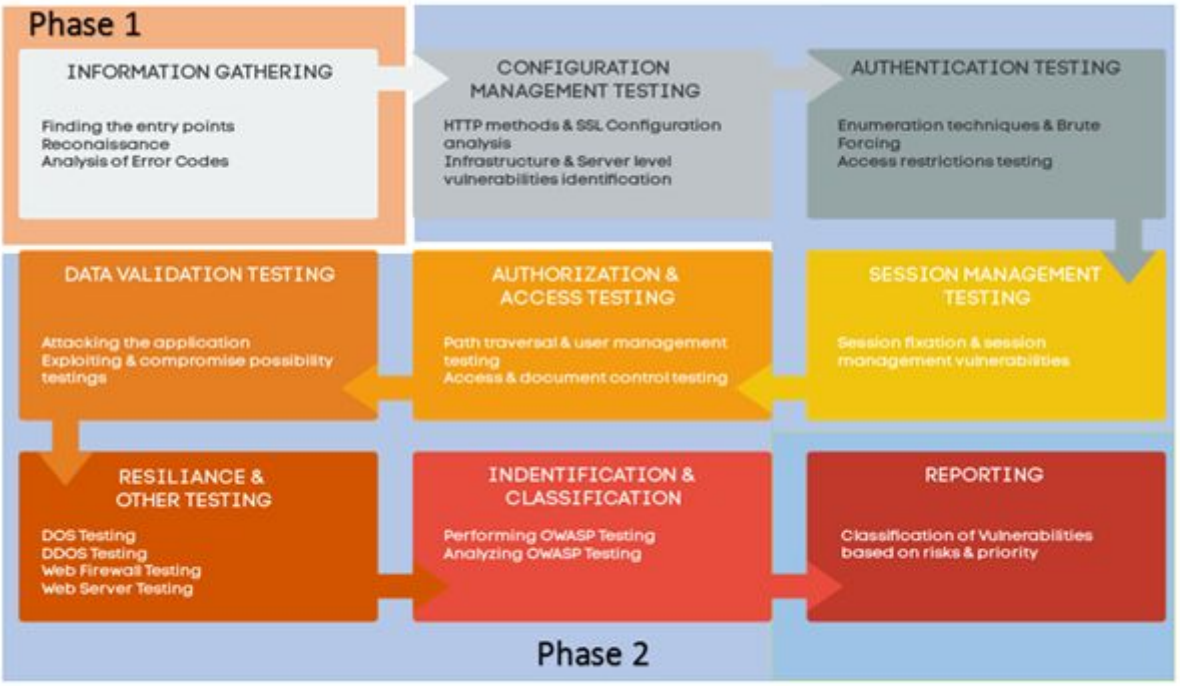

Figure 2. OWASP Penetration Testing Methodology

According to the OWASP methodology the valuation will be divided into two phases:

A. **Passive Way - Collecting information:** We will try to understand the logic of the Customer's applications, ie we will navigate to the applications and use a tool for collecting information, but also an HTTP proxy to observe all requests and HTTP responses. At the end of this phase, we will detect and record all access points (gates) of applications, e.g. HTTP headers, parameters, cookies. For example, we could find the following:

   https://www.customer.gr/login/AuthenticForm.html

   Here is an authentication form in which the application requests a username and

password.

In the next example, we see two (2) parameters that represent two access points (gates) in the application.

https://www.customer.gr/App1.jsp?a=1&b=1

In the above case, the application has two gates, namely parameters a and b. All the gates that are in this phase represent a test point. A calculation with a spreadsheet for the application directory tree and all its access points will be useful for the second phase.

B. **Active Mode:** At this stage, we begin testing using the methodology outlined in the following paragraphs. The set of controls we will implement is divided into nine (9) subcategories:

• Configuration Management Testing
• Authentication Testing
• Session Management Control
• Authorization Testing
• Business Logic Testing
• Data Validation Testing
• Denial of Service and Distributed Denial of Service Testing (Denial of Service)
• Web Services Testing
• AJAX Testing (AJAX Testing)

These checks fully cover all of the following vulnerabilities described in OWASP Top 10:

• A1: Injection
• A2: Cross-Site Scripting (XSS)
• A3: Broken Authentication and Session Management
• A4: Insecure Direct Object References
• A5: Cross-Site Request Forgery -CSRF
• A6: Incorrect Security Misconfiguration
• A7: Insecure Cryptographic Storage
• A8: Failure to Restrict URL Access
• A9: Insufficient Transport Layer Protection
• A10: Unvalidated Redirects and Forwards

Phase 3 - Attack

The invasion tests will include exploiting some of the most critical weaknesses we have identified since the previous phase (Phase 2) in an effort to gain higher levels of access to corporate data, systems and networks, but also to further gather information about from the internal structure of the network, so that we can get more penetration. These tests will be accomplished using our tools and methods, which will implement attack scenarios that have been designed based on a combination of weaknesses. Typical examples of such testing include the use of "trust relationships" between systems, bypassing filters and rules for routers or firewalls, and the creation of Firewall backchannels.

This work is divided into the following stages:

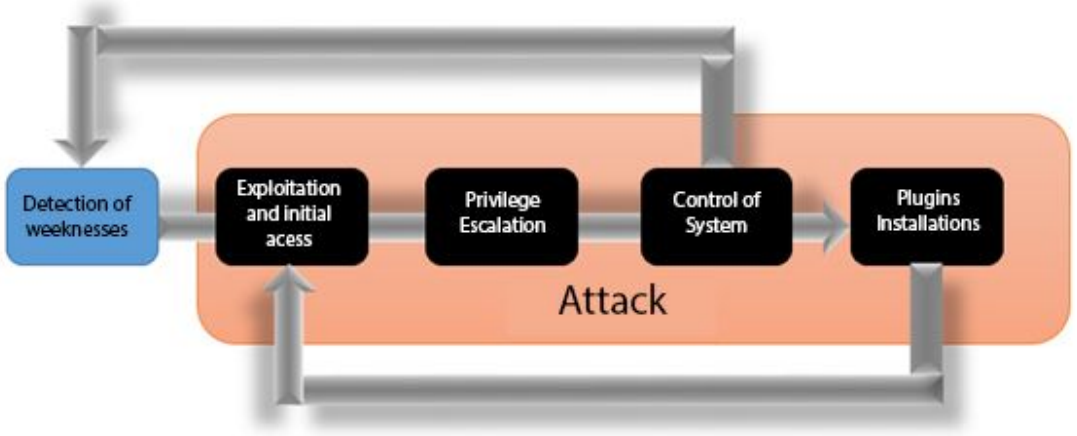

Figure 3. Attack Methodology

• Exploitation of weakness & acquisition of initial access. At this stage we exploit the inability of a computer or communications system to gain initial access even with a low privilege account.

• Get access using a high-privilege account. If the initial access to exploitation of the weakness of the system was on a low privilege account, guest account, then we are trying to gain access to a system account or an administrator account, e.g. administrator, root, etc. The goal of this phase is to fully control the system.

• Controlling the System. After full control is gained in the system, new systems and possible weaknesses as shown by the system being controlled are tested.

• Plugins Installations. Finally, if we need to install additional tools to obtain additional information or access to the system or new systems, we start over again from scratch.

Phase 4 - Reporting

In this final Phase, a complete documentation is written and delivered to the customer in order to fix the security issues of the applications that were scanned. The report must be confidential and it must never be published if it contains informations that can ot be known by external factors.

The final report that must be delivered should contain the below contents:

1. Threats/Risks Summary
2. Threats/Risks Explanations
3. Threats/Risks Severity
4. Fix Recommendations
5. Advisories with Technical Information and Referrences

3 - Main Tools and PenTest Overview

3.1 - Pentest Tools

Category	Tools
Footprinting Tools	Maltego, Nslookup, Whois, ARIN, Neo Trace, VisualRoute, Trace SmartWhois, eMailTrackerPro, Website watcher, Google, HTTrack Web, Googlag, MyIP Suite, BiLe Suite, Alchemy Network Tool, Samspade, SpiderFoot, Web The Ripper, Necrosoft Advanced DIG, DomainKing, Domain Name Analyzer, MSR , Strider URL Tracer, Mozzle Domain Name Pro, Path Analyzer Pro, Maltego, Read Notify και Netcraft Toolbar.
Password Crackers	Cain and Abel, John the Ripper, THC Hydra, Aircrack, Airsnort, SolarWinds, Pwdump, RainbowCrack και το Brutus.
Packet Sniffers	Wireshark, Tcpdump, Dsniff, Ettercap, NetStumbler και Ntop
Vulnerability Scanners	Nessus, OpenVAS, GFI LANguard, Retina, SAINT και MBSA, IBM Appscan
Web Vulnerability Scanners	Nikto, w3af, Paros proxy, WebScarab, Wikto, Acunetix WVS, IBM AppScan και N-Stealth.
Vulnerability Exploitation Tools	Metasploit, w3af, Firefox
Packet Crafting Tools	Nmap, Hping2, Scapy, Nemesis και Yersinia.
Operating Systems	Kali Linux, Samurai

Table 1. Penetration Testing Tools

3.2 - Penetration Test Overview

In this master thesis we will present the penetration test report for the training web application for Electronics Auctions of Greek State - “training.eauction.gr”. The test was made based on the first version of the application, so the results are a little outdated. For security reasons the latests results will not be presented in this thesis.

For our case we used some of the tools mentioned in part 3.1, such as IBM AppScan, W3AF, Acunetix WVS. The tools contain custom modifications that were made with the collaboration of some security companies that have proven experience in the Security field. The tools and the modifications will not be presented in this thesis because they will be part of a new product that will have commercial licence, which does not allow us to make any presentation of how it works or its graphical environment. We can say that all of the tools follow the OWASP guidelines for penetration testing.

IBM AppScan:

We use this tool to find vulnerabilities in a web application. This software delivers a powerful scanning engine that exceeded our expectations. The product can discover a wide range of vulnerabilities and supports a growing range of architectures, including support for Web 2.0, Flash, Javascript, AJAX and more. It has a wide array of options, including replay macros, a mechanism to easily report false positives and a simple but useful dashboard view of remediation tasks. Compliance mapping and reporting features in the product are excellent.

We felt that AppScan’s documentation is outstanding. Included in the remediation sections are several web-based training modules. These modules are automated slide shows with narrative voiceover to help the user understand the vulnerability in greater detail.

Strengths: Powerful scanning engine. Robust set of options. Excellent documentation
Weaknesses: True enterprise management, requires the purchase of addition AppScan Products, commercial licence, lack of scripting interface

W3AF:

It is an extremely popular, powerful and flexible framework for finding and exploiting web application vulnerabilities. It is easy to use and has extended features of web assessment and also exploitation plugins. In some way it is like a web-focuses Metasploit.

Strengths: Open Source tool, Easy to use after proper configuration, great performance, useful scripting interfcace
Weaknesses: Difficult configuration, too many false positives, too many dependencies

Acunetix WVS:

Another web application vulnerability scanner. This tool is perfect choice for SQL Injection testing, Cross-Site scripting (XSS) and OWASP top 10 other Vulnerabilities. It is an

automated web application security testing, founded to combat the rise in attacks at the web application layer. WVS audits a website's security by launching a series of attacks against the site. It provides concise reports of any vulnerabilities it found and will even offer suggestions on how to fix them. Finally, you can config the scan to login in the web-application in order to scan also the after-login webpages for vulnerabilities.

Strengths: Easy to use, straightforward and very robust package, all in one software, great documentation, application authentication
Weaknesses: Few false positives, commercial licence, the scan runs pretty slow, lack of scripting interface

3.3 - Scope of Penetration Test

The penetration test that we run is a web application security test that evaluates the security of the computer system and network by methodically validating and verifying the effectiveness of application security controls. We focus only on evaluating the security of the web application name “training.eauction.gr”. The process involves an active analysis of the application for any weaknesses, technical flaws, or vulnerabilities. The security issues that were found are presented below together with an assessment of the impact, a proposal for mitigation or a technical solution.

4 - Penetration Test Report for training.eauction.gr

Table of Contents

Figures and Tables

4.1 - Executive Summary

Training.eauction.gr is a web application created to digitize the auction process of the Greek Public. On this platform, auctions take place on a daily basis following the procedure set out by the consortiums in accordance with Greek law. The training environment was created for the training of notaries before they used the production system

As part of the testing process of its systems we deliver a penetration test that can be used to identify potential security issues and proactively manage risk to the infrastructure. For this project we commissioned a Security Assessment on the infrastructure of training.eauction.gr utilizing both automated and manual methods, namely an External and Internal Security Assessment that consists of the following stages:

- Intelligence Gathering
- Vulnerability Scanning
- Manual Verification
- Analysis and Reporting

The findings of the above steps were judged as high, medium and low risk based on the impact that they have on the web application and its infrastructure.

The report describes in detail the findings, the risk prioritization and recommended remediation regarding the outcomes of the Security Assessment procedure in order to ensure that the infrastructure and the web application is reasonable protected.

5. External Infrastructure Assessment:

5.1 Overview of External Infrastructure Findings

During the security assessment of training.eacution.gr we found three (3) high risk, three (3) Medium and one (1) low risk findings. A graphical representation of the number of findings per severity is presented in the following figure.

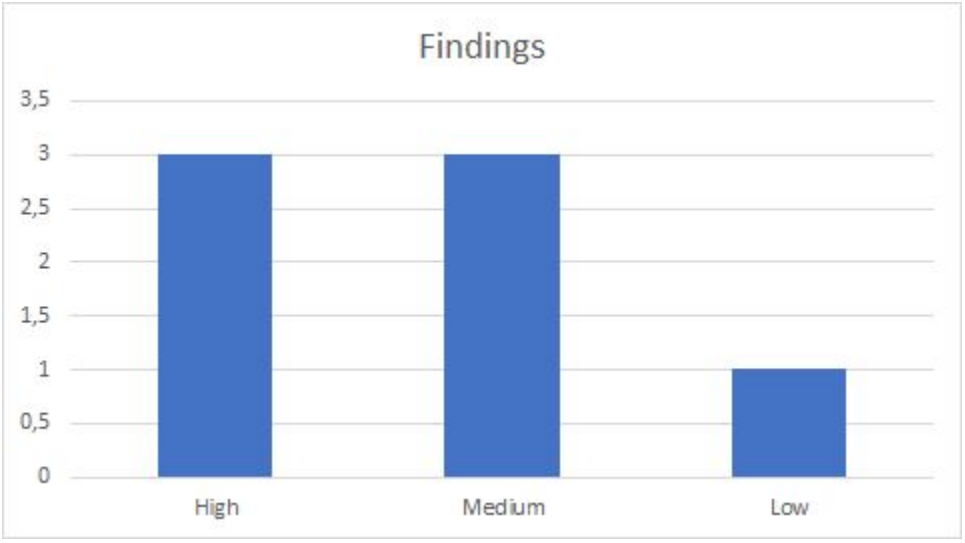

Figure 1: Overview of External Infrastructure Findings

More specifically, as an overview of the testing findings:

- **High risk findings** refer to Microsoft IIS tidle directory enumeration, SSL Certificate Invalid Date, SSL 2.0 Deprecated Protocol
- **Medium risk findings** refer to SSL Weak Ciphers, Microsoft Exchange Client Access Server Information Disclosure and the Poodle attack
- **Low Risk findings** refer to Clickjacking X-Frame-Options Header Missing

5.2 Scope:

Focus	**Scope and Delivery Requirements**
External Infrastructure Security Assessment	**Description:** External Infrastructure Penetration Test for training.eauction.gr
	Scope: 1 subnet ● 192.168.200.0/21 **Testing Scenarios:** ● Black Box Testing
Service Details	We will perform the services at the following geographic locations: Ginfinity IT Solutions HQ (remote) 1. Exploitation of Vulnerabilities: Will be conducted only after the authorization for the company

Table 1: Scope of the external infrastructure security assessment

5.3 Contact Details:

Technical Support	
Full Name:	Panagiotis Georgakopoulos
Title:	Manager, Cunsulting Services
Telephone Numer:	+30213 0445574
Fax	+302109344407
Email:	panos.g.tx@gmail.com
Address:	25, Ag. Ioannou, Athens 173 41, Greece

5.4 External Infrastructure Hosts Identification

The Host Identification activities, as part of the Intelligence Gathering phase, were carried out to provide to the company an overview of the hosts statuses with the network access provided when accessing the on-site VLANs. The table below enlists the running hosts, as well as the majority of the running services:

Host Description	Open Ports	Comments
192.168.200.10	53 (Domain)	
192.168.200.11	25 (SMTP)	Postfix smtpd
192.168.201.30	80 (HTTP) 443 (SSL/HTTP)	Microsoft IIS httpd 6.0
192.168.201.31	443 (SSL/HTTP)	Microsoft IIS httpd 7.5
192.168.203.10	80 (HTTP)	Microsoft IIS httpd 6.0

Table 2: External Infrastructure Hosts Identification

5.5 External Infrastructure Findings Matrix

In the following Section, we provide a comprehensive list of findings that encompass each of the assessment components, including a details description that explains the vulnerabilities discovered by the testing team and a set of recommendations to address each finding.

#	Finding	Vulnerable Point	Overall Rating/CVSS/CW SS Score(Beta)	Recommendation Cost	Ease of Detection	Consequences
1	**Microsoft IIS Tidle Directory Enumeration**	192.168.201.30 192.168.203.10	High / CVSS Base Score: 5.0	Medium	Moderate	Code Execution
2	**SSL Certificate Invalid Date**	192.168.201.30	High / CVSS Base Score: 5.0	Medium	Moderate	Man in the Middle Attack
3	**SSL 2.0 Deprecated Protocol**	192.168.201.30 192.168.201.31	High / CVSS Base Score: 5.5	Medium	Easy	Man In the Middle Attack
4	**SSL Weak Ciphers**	192.168.201.30	Medium / CVSS Base Score: 1.9	Medium	Easy	Man In the Middle Attack
5	**Microsoft Exchange Client Access Server Information Disclosure**	192.168.200.11	Medium / CVSS Base Score: 5.0	Medium	Easy	Information Disclosure
6	**The Poodle Attack**	192.168.201.31	Medium / CVSS Base Score: 5.0	Low	Medium / CVSS Base Score: 5.0	Unautorized Access
7	**Clickjacking: X-Frame-Options Header Missing**	192.168.203.10	Low / CVSS Base Score: 6.8	Low	Easy	Clickjacking Attack

Table 3: External Infrastructure Assessment Findings

5.6. External Infrastructure Findings Details

5.6.1 High - Microsoft IIS tidle directory enumeration

Summary:
It is possible to detect short names of files and directories which have an 8.3 file naming scheme equivalent in Windows by using some vectors in several versions of Microsoft IIS.

Affected Servers:

192.168.201.30
192.168.203.10

Impact:
It is possible to detect all short-names of ".aspx" files as they have 4 letters in their extensions. This can be a major issue especially for the .Net websites which are vulnerable to direct URL access as an attacker can find important files and folders that they are not normally visible. It seems the latest versions of IIS and .Net version 4 have been secured against this attack. Moreover, some of the websites which use special URL-rewrite rules are also safe. Note that the Basic authentication and Windows authentication cannot stop this attack.

How to fix:
Consult the "Prevention Technique(s)" section from Soroush Dalili's paper on this subject. *(A link to this paper is listed in the Web references section below.)*

References:
http://www.acunetix.com/blog/web-security-zone/windows-short-8-3-filenames-web-security-problem/
https://code.google.com/p/iis-shortname-scanner-poc/

5.6.2 High - SSL certificate Invalid date

Summary:
This SSL certificate is either expired or not yet valid. Some browsers will continue connecting to the site after presenting the user with the warning, while others will prompt the user with a dialog box requesting their approval to proceed. These warnings are extremely confusing for the typical web user, and cause most users to question the authenticity of the site they are attempting to view.

Affected Servers:

192.168.201.30
192.168.201.31

Impact:
This SSL certificate is not valid.

How to fix:
Verify Start Date and/or End Dates of your SSL Certificate

5.6.3 High - SSL 2.0 deprecated protocol

Summary:
The remote service encrypts traffic using an old deprecated protocol with known weaknesses. The SSL server (port: 443) encrypts traffic using an old deprecated protocol (SSL 2.0) with known weaknesses. An attacker may be able to exploit these issues to conduct man-in-the-middle attacks or decrypt communications between the affected service and clients.

Affected Servers:

192.168.201.30

How to fix:
Disable SSL 2.0 and use SSL 3.0 or TLS 1.0 instead.

References:
https://www.schneier.com/paper-ssl.pdf

5.6.4 Medium - SSL Weak ciphers

Summary:
The TLS/SSL server supports cipher suites based on weak algorithms. This may enable an attacker to launch man-in-the-middle attacks and monitor or tamper with sensitive data. In general, the following ciphers are considered weak:

- So called "null" ciphers, because they do not encrypt data.
- Export ciphers using secret key lengths restricted to 40 bits. This is usually indicated by the word EXP/EXPORT in the name of the cipher suite.
- Obsolete encryption algorithms with secret key lengths considered short by today's standards, eg. DES or RC4 with 56-bit keys.

Affected Servers:

192.168.201.30

Impact:
Man in the Middle attacks.

How to fix:

Reconfigure the affected application to avoid use of weak ciphers

References:
http://www.openssl.org/docs/apps/ciphers.html

5.6.5 Medium – Microsoft Exchange Client Access Server Information Disclosure

Summary:
The Microsoft Exchange Client Access Server (CAS) is affected by an information disclosure vulnerability. A remote, unauthenticated attacker can exploit this vulnerability to learn the server's internal IP address.

Affected Servers:

192.168.200.11

How to fix:
There is no known fix at this time.

References:
http://foofus.net/?p=758

5.6.6 Medium – The POODLE attack (SSLv3 supported)

Summary:
Websites that support SSLv3 and CBC-mode ciphers are potentially vulnerable to an active MITM (Man-in-the-middle) attack. This attack, called POODLE, is similar to the BEAST attack and also allows a network attacker to extract the plaintext of targeted parts of an SSL connection, usually cookie data. Unlike the BEAST attack, it doesn't require such extensive control of the format of the plaintext and thus is more practical.

Any website that supports SSLv3 is vulnerable to POODLE, even if it also supports more recent versions of TLS. SSLv3 will be disabled by default in Firefox 34, which will be released on Nov 25 2014. An attacker may be able to exploit this problem to conduct man-in-the-middle attacks and decrypt communications between the affected service and clients. Attackers can perform man-in-the-middle attacks and observe the encryption traffic between your website and its visitors.

Affected Servers:

192.168.201.31

How to fix:
To fix this vulnerability follow the steps:

- It's recommended to disable SSLv3 and replace it with TLSv1.0 as soon as compatibility with legacy clients is no longer required. (The only browser that does not support TLSv1.0 is Internet Explorer 6).
- Disable SSLv2 and SSLv3

References:
http://googleonlinesecurity.blogspot.ro/2014/10/this-poodle-bites-exploiting-ssl-30.html
https://blog.mozilla.org/security/2014/10/14/the-poodle-attack-and-the-end-of-ssl-3-0/
http://blog.rlove.org/2013/12/strong-ssl-crypto.html
https://www.openssl.org/~bodo/ssl-poodle.pdf

5.6.7 Low– Clickjacking: X-Frame-Options header missing

Summary:
Clickjacking (User Interface redress attack, UI redress attack, UI redressing) is a malicious technique of tricking a Web user into clicking on something different from what the user perceives they are clicking on, thus potentially revealing confidential information or taking control of their computer while clicking on seemingly innocuous web pages.

The server didn't return an X-Frame-Options header which means that this website could be at risk of a clickjacking attack. The X-Frame-Options HTTP response header can be used to indicate whether or not a browser should be allowed to render a page in a <frame> or <iframe>. Sites can use this to avoid clickjacking attacks, by ensuring that their content is not embedded into other sites.

Affected Servers:

192.168.203.10

How to fix:
Configure your web server to include an X-Frame-Options header.

References:
https://www.owasp.org/index.php/Clickjacking

6 Internal Infrastructure Assessment

6.1 Overview of Internal Infrastructure Findings

During the security assessment of internal infrastructure of training.eacution.gr we found zero (0) high risk, three (3) Medium and five (5) low risk findings. A graphical representation of the number of findings per severity is presented in the following figure.

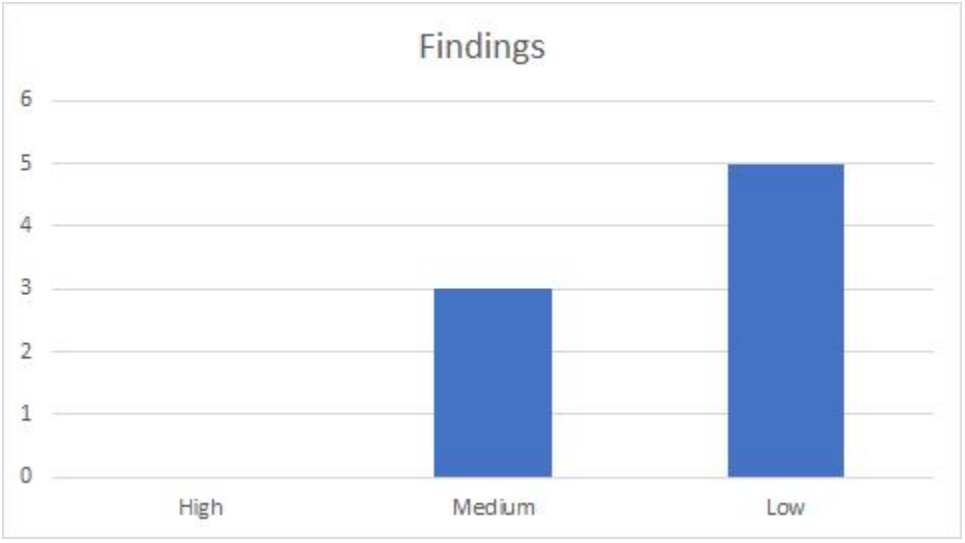

Figure 2: Overview of Internal Findings.

More specifically, as an overview of the testing findings:

- **Medium risk findings** refer to Microsoft Windows RDP Service and SSH Weak Encryption Algorithm
- **Low Risk findings** Oracle Version Detection, Microsoft TCP/IP listener detection, HTTP Server type and version detection and two (2) unknown identity service ports

6.2 Scope:

Focus	Scope and Delivery Requirements
External Infrastructure Security Assessement	**Description:** Internal Infrastructure Penetration Test for training.eauction.gr
	Scope: 1 subnet ● 10.11.100.0/24 **Testing Scenarios:** ● Black Box Testing
Service Details	We will perform the services at the following geographic locations: Ginfinity IT Solutions HQ (remote) 1. Exploitation of Vulnerabilities: Will be conducted only after the authorization for the company

Table 4: Scope of the internal infrastructure security assessment

6.3 External Infrastructure Hosts Identification

The Host Identification activities, as part of the Intelligence Gathering phase, were carried out to provide to the company an overview of the hosts statuses with the network access provided when accessing the on-site VLANs. The table below enlists the running hosts, as well as the majority of the running services:

Host Description	Open Ports	Comments
10.11.100.11	3389,808,443	Microsoft RDP, Unknown Identity Service, HTTPS
10.11.100.21	3389, 1433	Microsoft RDP, Microsoft , SQL
10.11.100.31	22, 1531, 1720	SSH, Oracle DB, Unknown Identity Service

Table 5: Internal Infrastructure Host Identification

6.4 Internal Infrastructure Findings Matrix

#	Finding	Vulnerable Point	Overall Rating/CVSS/CW SS Score(Beta)	Recommendation Cost	Ease of Detection	Consequences
1	**SSL/TLS Weak Cipher Suites**	10.11.100.11 10.11.100.21	Medium / CVSS Base Score: 4.3	Medium	Moderate	Unauthorized access
2	**SSH Weak Encryption Algorithms Supported**	10.11.100.31	Medium / CVSS Base Score: 4.3	Medium	Medium	Unauthorized access
3	**HTTP Server Type and Version**	10.11.100.11	Low / CVSS Base Score: 0.0	Low	Low	Information Leakage
4	**Microsoft SQL TCP/IP listener**	10.11.100.21	Low / CVSS Base Score: 0.0	Low	Low	Unauthorized Access
5	**Oracle DB Versiob**	10.11.100.31	Low / CVSS Base Score: 0.0	Low	Low	Version Detection

Table 6: Internal Infrastructure Assessment Findings

6.5 Internal Infrastructure Findings Details:

6.5.1 Medium - SSL/TLS Weak Cipher Suites

Summary:
The SSL/TLS service uses Die-Hellman groups with insucient strength (key size < 2048).

Affected Servers:
10.11.100.11
10.11.100.21

Impact:
An attacker might be able to decrypt the SSL/TLS communication oine.

How to fix:
Deploy (Ephemeral) Elliptic-Curve Die-Hellman (ECDHE) or use a 2048-bit or stronger Die-Hellman group. (see https://weakdh.org/sysadmin.html)

References:
URL:https://weakdh.org/
URL:https://weakdh.org/sysadmin.html

6.5.2 Medium - SSH Weak Encryption Algorithms

Summary:
The remote SSH server is configured to allow weak encryption algorithms.

Affected Servers:
10.11.100.31

Insight:
The `arcfour` cipher is the Arcfour stream cipher with 128-bit keys. The Arcfour cipher is believed to be compatible with the RC4 cipher [SCHNEIER]. Arcfour (and RC4) has problems with weak keys, and should not be used anymore.
The `none` algorithm species that no encryption is to be done. Note that this method provides no condentiality protection, and it is NOT RECOMMENDED to use it.
A vulnerability exists in SSH messages that employ CBC mode that may allow an attacker to
recover plaintext from a block of ciphertext.

How to fix:
Mitigation
Disable the weak encryption algorithms.

References:
URL:https://tools.ietf.org/html/rfc4253#section-6.3
URL:https://www.kb.cert.org/vuls/id/958563

6.5.3 Low - HTTP Server Type and Version

Summary:
This detects the HTTP Server's type and version.

Affected Servers:
10.11.100.11

Vulnerability Detection Result:
The remote web server type is :
Microsoft-HTTPAPI/2.0

6.5.4 Low - Microsoft SQL TCP/IP listener

Summary:
Microsoft SQL server is running on this port.

Affected Servers:
10.11.100.21

Vulnerability Detection Result:
Detected Microsoft SQL Server
Version: 12.0.5207.0
Location: 1433/tcp
CPE: cpe:/a:microsoft:sql_server:12.0.5207.0

6.5.5 Low - Oracle DB Version

Summary:
Detection of installed version of Oracle.

Affected Servers:
10.11.100.31

Vulnerability Detection Result:
A TNS service is running on this port but it refused to honor an attempt to connect to it. (The TNS reply code was 4)

7 External WEB Application Assessment

7.1 External WEB Application Overview of Findings

During the security assessment of internal infrastructure of training.eacution.gr we found zero one (1) high risk, one (1) Medium, seventy (70) low and eighteen (18) informational risk findings. A graphical representation of the number of findings per severity is presented in the following figure.

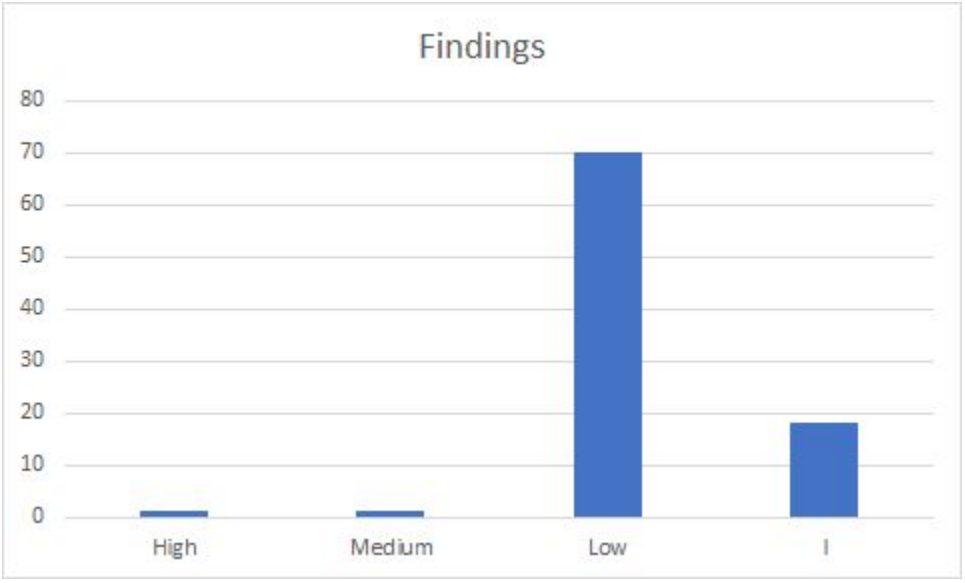

Figure 3: Overview of External WEB Application Findings.

More specifically, as an overview of the testing findings:

- **High risk findings** refer to Blind SQL Injection
- **Medium risk findings** refer to Missing Secure Attribute in Encrypted Session(SSL_ Cookie)

- **Low risk findings** refer to Autocomplete HTML Attribute Not Disabled for Password Filled, Body Parameters Accepted in Query, Cacheable SSL Page Found, Check SRI (Subresource Integrity) support, Hidden Directory Detected, Missing “Content-Security-Policy” header, Missing “X-Content-Type-Options” header, Missing “X-XSS-Protection” header, Missing HTTP Strict-Transport-Security Header, Query parameter in SSL Request, Temporary File Download, Application Error
- **Informational risk findings** refer to Client-Side (Javascript) Cookie References, Email Address Pattern Found, Integer Overflow, Link to unclassified site, Possible Server path Disclosure Pattern Found

7.2 Scope:

Focus	**Scope and Delivery Requirements**
External Infrastructure Security Assessement	**Description:** Penetration Testing for the following site/application
	Scope: ● Training.eauction.gr **Authentication:** ● The test will be conducted without authentication credentials **Scenarios:** ● We will access the application via the Internet
Service Details	We will perform the services at the following geographic locations: Ginfinity IT Solutions HQ (remote) 1. Exploitation of Vulnerabilities: Will be conducted only after the authorization for the company

Table 7: Scope of the external web application security assessment

7.3 External WEB Application Findings Matrix

#	Finding	Vulnerable Point	Overall Rating/CVSS/CW SS Score(Beta)	Recommendation Cost	Ease of Detection	Consequences
1	**Blind SQL Injection**	https://training.eauction.gr/	High / CVSS Base Score: 9.7	High	High	It is possible to view, modify or delete entries and tables
2	**Missing Secure Attribute in Encrypted Session (SSL) Cookie**	r/auction/detailes/102370	Medium / CVSS Base Score: 6.4	Medium	Medium	It may be possible to steal user and session information (cookies) that was sent during an encrypted session
3	**Autocomplete HTML Attribute Not Disabled for Password Field**	/account/login	Low / CVSS Base Score: 5.0	Low	Low	It may be possible to bypass the web application's authentication mechanism
4	**Body Parameters Accepted in Query**	https://training.eauction.gr/	Low / CVSS Base Score: 5.0	Low	Low	It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations

5	**Cacheable SSL Page Found**	/account/login	Low / CVSS Base Score: 5.0	Low	Low	It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations
6	**Check for SRI (Subresource Integrity) support**	/account/login	Low / CVSS Base Score: 5.0	Low	Low	In case the third-party server is compromised, the content/behavior of the site will change
7	**Hidden Directory Detected**	/scripts/	Low / CVSS Base Score: 5.0	Low	Low	It is possible to retrieve information about the site's file system structure, which may help the attacker to map the web site
8	**Missing "Content-Security-Policy" header**	/scripts/auctionslist.js	Low / CVSS Base Score: 5.0	Low	Low	It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations
9	**Missing "X-Content-Type-Options" header**	/scripts/auctionslist.js	Low / CVSS Base Score: 5.0	Low	Low	It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations

10	**Missing "X-XSS-Protection" header**	/scripts/auctionslist.js	Low / CVSS Base Score: 5.0	Low	Low	It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations
11	**Missing HTTP Strict-Transport-Security Header**	/scripts/auctionslist.js	Low / CVSS Base Score: 5.0	Low	Low	It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations
12	**Query Parameter in SSL Request**	/notification/list	Low / CVSS Base Score: 5.0	Low	Low	It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted
13	**Temporary File Download**	/auction/details/102370	Low / CVSS Base Score: 5.0	Low	Low	It is possible to download temporary script files, which can expose the application logic and other sensitive information such as usernames and passwords
14	**Application Error**	https://training.eauction.gr/	Informational / CVSS Base Score: 0.0	Informational	Informational	It is possible to gather sensitive debugging information
15	**Client-Side (JavaScript)**	/scripts/jquery-ui-1.8.24.js	Informational / CVSS Base	Informational	Informational	The worst case scenario for this attack depends on the

	Cookie References		Score: 0.0			context and role of the cookies that are created at the client side
16	**Email Address Pattern Found**	/contact	Informational / CVSS Base Score: 0.0	Informational	Informational	It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations
17	**Integer Overflow**	https://training.eauction.gr/	Informational / CVSS Base Score: 0.0	Informational	Informational	It is possible to gather sensitive debugging information
18	**Link to unclassified site**	/auction/details/102243	Informational / CVSS Base Score: 0.0	Informational	Informational	N/A
19	**Possible Server Path Disclosure Pattern Found**	/scripts/utils.js	Informational / CVSS Base Score: 0.0	Informational	Informational	It is possible to retrieve the absolute path of the web server installation, which might help an attacker to develop further attacks and to gain information about the file system structure of the web application

Table 8: External Web Application Assessment Findings

7.4 External WEB Application Findings Details

7.4.1 High - Blind SQL Injection

Summary:
The test result seems to indicate a vulnerability because it shows that values can be appended to parameter values, indicating that they were embedded in an SQL query. In this test, three (or sometimes four) requests are sent. The last is logically equal to the original, and the next-to-last is different. Any others are for control purposes. A comparison of the last two responses with the first (the last is similar to it, and the next-to-last is different) indicates that the application is vulnerable.

Entity:
debtorName (Parameter)

Impact:
It is possible to view, modify or delete entries and tables

Causes:
Sanitation of hazardous characters was not performed correctly on iser input

How to Fix:
There are several mitigation techniques:
[1] Strategy: Libraries or Frameworks
Use a vetted library or framework that does not allow this weakness or provides constructs that make it easier to avoid.
[2] Strategy: Parameterization
If available, use structured mechanisms that automatically enforce separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this at every point where output is generated.
[3] Strategy: Environment Hardening

Run your code using the lowest privileges that are required to accomplish the necessary tasks.
[4] Strategy: Output Encoding
If you need to use dynamically-generated query strings or commands in spite of the risk, properly quote arguments, and escape any special characters within those arguments.
[5] Strategy: Input Validation
Assume all input is malicious. Use an "accept known good" input validation strategy: a whitelist of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on detecting for malicious or malformed inputs with a blacklist. However, blacklists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright.

.Net
Here are two possible ways to protect your web application against SQL injection attacks:
[1] Use a stored procedure rather than dynamically built SQL query string. The way parameters are passed to SQL Server stored procedures, prevents the use of apostrophes and hyphens.
Here is a simple example of how to use stored procedures in ASP.NET:

```
' Visual Basic example
Dim DS As DataSet
Dim MyConnection As SqlConnection
Dim MyCommand As SqlDataAdapter
Dim SelectCommand As String = "select * from users where username = @username"
...
MyCommand.SelectCommand.Parameters.Add(New SqlParameter("@username", SqlDbType.NVarChar, 20))
MyCommand.SelectCommand.Parameters("@username").Value = UserNameField.Value
// C# example
String selectCmd = "select * from Authors where state = @username";
SqlConnection myConnection = new SqlConnection("server=...");
SqlDataAdapter myCommand = new SqlDataAdapter(selectCmd, myConnection);
myCommand.SelectCommand.Parameters.Add(new SqlParameter("@username", SqlDbType.NVarChar, 20));
myCommand.SelectCommand.Parameters["@username"].Value = UserNameField.Value;
```

2] You can add input validation to Web Forms pages by using validation controls. Validation controls provide an easy-to-use mechanism for all common types of standard validation - for example, testing for valid dates or values within a range - plus ways to provide custom-written validation. In addition, validation controls allow you to completely customize how error information is displayed to the user. Validation controls can be used with any controls that are processed in a Web Forms page's class file, including both HTML and Web server controls.

In order to make sure user input contains only valid values, you can use one of the following validation controls:

a. "RangeValidator": checks that a user's entry (value) is between specified lower and upper boundaries. You can check ranges within pairs of numbers, alphabetic characters, and dates.

b. "RegularExpressionValidator": checks that the entry matches a pattern defined by a regular expression. This type of validation allows you to check for predictable sequences of characters, such as those in social security numbers, e-mail addresses, telephone numbers, postal codes, and so on.

Important note: validation controls do not block user input or change the flow of page processing; they only set an error state, and produce error messages. It is the programmer's responsibility to test the state of the controls in the code before performing further application-specific actions.

There are two ways to check for user input validity:

1. Testing for a general error state:
In your code, test the page's IsValid property. This property rolls up the values of the IsValid properties of all the validation controls on the page (using a logical AND). If one of the validation controls is set to invalid, the page'sproperty will return false.

2. Testing for the error state of individual controls:
Loop through the page's Validators collection, which contains references to all the validation controls. You can then examine the IsValid property of each validation control.

7.4.2 - Medium - Missing Secure Attribute in Encrypted Session (SSL) Cookie

Summary:
AppScan found that an encrypted session (SSL) is using a cookie without the "secure" attribute.

Original Response:

```
...
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US
HTTP/1.1 200 OK
Content-Length: 17305
Server: Microsoft-IIS/8.5
X-AspNet-Version: 4.0.30319
X-Frame-Options: SAMEORIGIN
Set-Cookie:
__RequestVerificationToken=os-1CDlAcIa_6e0zuNmJmNy2wJUY6v4TTuy2zqBzJ6KKzlPPqVoVtzopxAWZHSekrtm8ljJsNKJ53GzvX0Betjm
A44IwYbvlUEEFLVxm0o1;
path=/; HttpOnly
X-Powered-By: ASP.NET
X-AspNetMvc-Version: 5.2
Date: Sat, 07 Oct 2017 16:39:57 GMT
Content-Type: text/html; charset=utf-8
Cache-Control: private
...
```

Entity:
__RequestVerificationToken (Cookie)

Impact:
It may be possible to steal user and session information (cookies) that was sent during an encrypted session

Causes:
The web application sends non-secure cookies over SSL

How to Fix:
Basically the only required attribute for the cookie is the "name" field. Common optional attributes are: "comment", "domain", "path", etc.
The "secure" attribute must be set accordingly in order to prevent to cookie from being sent unencrypted.

For more information on how to set the secure flag, see OWASP "SecureFlag" at
https://www.owasp.org/index.php/SecureFlag

RFC 2965 states:
"The Secure attribute (with no value) directs the user agent to use only (unspecified) secure means to contact the origin server whenever it sends back this cookie, to protect the confidentially and authenticity of the information in the cookie."

For further reference please see the HTTP State Management Mechanism RFC 2965 at:
http://www.ietf.org/rfc/rfc2965.txt
and for "Best current practice" for use of HTTP State Management please see
http://tools.ietf.org/html/rfc2964

7.4.3 - Low - Autocomplete HTML Attribute Not Disabled for Password Field

Summary:
AppScan has found that a password field does not enforce the disabling of the autocomplete feature.

Original Response:

```
...
<legend>Είσοδος στο λογαριασμό σας</legend>
<ol>
<li class="email">
<label for="Email">Εισαγωγή E-mail</label>
<input data-val="true" data-val-required="Παρακαλώ συμπληρώστε το πεδίο
&#39;Εισαγωγή E-mail&#39;." id="Email" name="Email" type="text" value="" />
<br /><span class="field-validation-valid" data-valmsg-for="Email" datavalmsg-
replace="true"></span>
</li>
<li class="password">
<label for="Password">Εισαγωγή Κωδικού</label>
<input data-val="true" data-val-required="Παρακαλώ συμπληρώστε το πεδίο
&#39;Εισαγωγή Κωδικού&#39;." id="Password" name="Password" type="password" />
<br /><span class="field-validation-valid" data-valmsg-for="Password" datavalmsg-
replace="true"></span>
</li>
</ol>
<input type="submit" value="Είσοδος" class="buttonblue" />
</fieldset>
</form>
<p>
<a class="forgotPasswordtext" href="/Account/ForgotPassword">Ξέχασα τον κωδικό μου <i
class="forgotPasswordIcon"></i></a>
```

```
</p>
...
```

Entity:
Login (Page)

Impact:
It may be possible to bypass the web application's authentication mechanism

Causes:
Insecure web application programming or configuration

How to Fix:
If the "autocomplete" attribute is missing in the "password" field of the "input" element, add it and set it to "off".
If the "autocomplete" attribute is set to "on", change it to "off".
For example:
Vulnerable site:

```
<form action="AppScan.html" method="get">
Username: <input type="text" name="firstname" /><br />
Password: <input type="password" name="lastname" />
<input type="submit" value="Submit" />
<form>
```

Non-Vulnerable site:

```
<form action="AppScan.html" method="get">
Username: <input type="text" name="firstname" /><br />
Password: <input type="password" name="lastname" autocomplete="off"/>
<input type="submit" value="Submit" />
<form>
```

7.4.4 - Low - Body Parameters Accepted in Query

Summary:
The test result seems to indicate a vulnerability because the Test Response is similar to the Original Response, indicating that the application processed body parameters that were submitted in the query.

Entity:
(Page)

Impact:
It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations
It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.

Causes:
Insecure web application programming or configuration

How to Fix:
Do not accept body parameters that are sent in the query string. Re-program the application to disallow handling of POST parameters that were listed in the Query

7.4.5 - Low - Cacheable SSL Page Found

Summary:
The application has responded with a response that indicates the page should be cached, but not ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma:no-cache" or "Cache-Control: no-cache").

Original Response:

```
HTTP/1.1 200 OK
Content-Length: 14903
Server: Microsoft-IIS/8.5
X-AspNet-Version: 4.0.30319
X-Frame-Options: SAMEORIGIN
Set-Cookie:
__RequestVerificationToken=EVvj2XJ8Q0Bfm3fcXfGs2MAphJvK44MQQwBc9UpC4JdItB6BmJNZt821pl8cd5HopYToWr36Q6ZV1D3QvhsqoTSWWMvIInVKEacpMKuywM1;
path=/; HttpOnly
X-Powered-By: ASP.NET
X-AspNetMvc-Version: 5.2
Date: Sat, 07 Oct 2017 16:38:45 GMT
Content-Type: text/html; charset=utf-8
Cache-Control: private
<!DOCTYPE html>
<html lang="el">
<head>
<meta charset="utf-8" />
<title>Είσοδος - Υπηρεσία διενέργειας online πλειστηριασμών</title>
...
```

Entity:
Login (Page)

Impact:
It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations

Causes:
Sensitive information might have been cached by your browser

How to Fix:
Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. Disable caching on all SSL pages or all pages that contain sensitive data.
This can be achieved by using "Cache-Control: no-store" and either "Pragma: no-cache" or "Cache-Control: no-cache" response directives in your SSL page headers.
Cache-Control: private - This directive instructs proxies that the page contains private information, and therefore should not be cached by a shared cache. However, it does not instruct browsers to refrain from caching the pages.
Cache-Control: no-cache - This directive also instructs proxies that the page contains private information, and therefore should not be cached. It also instructs the browser to revalidate with the server to check if a new version is available. This means that the browser may store sensitive pages or information to be used in the revalidation.
Certain browsers do not necessarily follow the RFC and may treat no-cache as no-store.
Cache-Control: no-store - This is the most secure directive. It instructs both the proxy and the browser not to cache the page or store it in its cache folders.
Pragma: no-cache - This directive is required for older browsers, that do not support the Cache-Control header.

7.4.6 - Low - Check for SRI (Subresource Integrity) support

Summary:
The third-party links/scripts don't have integrity attribute for the browser to confirm they didn't compromised

Raw Test Response:

```
...
<link href="/Content/jquery-ui-timepicker-addon.css" rel="stylesheet" type="text/css" />
<script src="/Scripts/jquery-1.8.2.min.js"></script>
<script src="/Scripts/jquery-ui-1.8.24.js"></script>
<script src="/Scripts/modernizr-2.6.2.js"></script>
<script src="/Scripts/intlTelInput.min.js"></script>
<script src="/Scripts/jquery-ui-timepicker-addon.js"></script>
<link href='//fonts.googleapis.com/css?family=Roboto:400,700&subset=cyrillic,greek'
rel='stylesheet' type='text/css'>
<meta name="viewport" content="width=device-width" />
```

```
<link href="/Content/jquery-confirm.min.css" rel="stylesheet" type="text/css" />
<script src="/Scripts/jquery-confirm.min.js"></script>
<script type="text/javascript">
var sessionLanguage = "el-GR";
var locale = 'el';
</script>
...
```

Entity:
Login (Page)

Impact:
In case the third-party server is compromised, the content/behavior of the site will change

Causes:
There is no support to Subresource Integrity.

How to Fix:
Add Subresource Integrity to every script/link with source not in your domain

W3C Subresource Integrity:
https://www.w3.org/TR/SRI/

SRI Hash Generator:
https://srihash.org

Sample Script Element Not Supporting SRI:

```
<script src="https://example.com/example-framework.js"
crossorigin="anonymous"></script>
```

Sample Script Element Supporting SRI:

```
<script src="https://example.com/example-framework.js"
integrity="sha384-oqVuAfXRKap7fdgcCY5uykM6+R9GqQ8K/uxy9rx7HNQlGYl1kPzQho1wx4JwY8wC"
crossorigin="anonymous"></script>
```

7.4.7 - Low - Hidden Directory Detected

Summary:
The test tried to detect hidden directories on the server. The 403 Forbidden response reveals the existence of the directory, even though access is not allowed.

Raw Test Response:

```
...
GET /scripts/ HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
Referer: https://training.eauction.gr/
Host: training.eauction.gr
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US
HTTP/1.1 403 Forbidden
Server: Microsoft-IIS/8.5
Content-Length: 1233
X-Powered-By: ASP.NET
Date: Sat, 07 Oct 2017 16:55:32 GMT
Content-Type: text/html
...
```

Entity:
scripts/ (Page)

Impact:
It is possible to retrieve information about the site's file system structure, which may help the attacker to map the web site

Causes:
The web server or application server are configured in an insecure way

How to Fix:
If the forbidden resource is not required, remove it from the site.
If possible, issue a "404 - Not Found" response status code instead of "403 - Forbidden". This change will obfuscate the presence of the directory in the site, and will prevent the site structure from being exposed.

7.4.8 - Low - Missing "Content-Security-Policy" header

Summary:
AppScan detected that the Content-Security-Policy response header is missing, which increases exposure to various cross-site injection attacks

Raw Test Response:

```
...
GET /Scripts/AuctionsList.js HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
Referer: https://training.eauction.gr/
Host: training.eauction.gr
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US
HTTP/1.1 200 OK
Last-Modified: Fri, 21 Jul 2017 10:37:44 GMT
...
```

Entity:
AuctionList.js (Page)

Impact:
It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations
It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.
Causes:
Insecure web application programming or configuration

How to Fix:
Configure your server to send the "Content-Security-Policy" header.
For Apache, see:
http://httpd.apache.org/docs/2.2/mod/mod_headers.html
For IIS, see:
https://technet.microsoft.com/pl-pl/library/cc753133%28v=ws.10%29.aspx
For nginx, see:
http://nginx.org/en/docs/http/ngx_http_headers_module.html

7.4.9 - Low - Missing "X-Content-Type-Options" header

Summary:
AppScan detected that the X-Content-Type-Options response header is missing, which increases exposure to drive-by download attacks

Raw Test Response:

```
...
GET /Scripts/AuctionsList.js HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
Referer: https://training.eauction.gr/
Host: training.eauction.gr
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US
HTTP/1.1 200 OK
Last-Modified: Fri, 21 Jul 2017 10:37:44 GMT
...
```

Entity:
AuctionList.js (Page)

Impact:
It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations
It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.

Causes:
Insecure web application programming or configuration

How to Fix:
Configure your server to send the "X-Content-Type-Options" header with value "nosniff" on all outgoing requests.
For Apache, see:
http://httpd.apache.org/docs/2.2/mod/mod_headers.html
For IIS, see:

https://technet.microsoft.com/pl-pl/library/cc753133%28v=ws.10%29.aspx
For nginx, see:
http://nginx.org/en/docs/http/ngx_http_headers_module.html

7.4.10 - Low - Missing "X-XSS-Protection" header

Summary:
AppScan detected that the X-XSS-Protection response header is missing, which may allow Cross-Site Scripting attacks

Raw Test Response:

```
...
GET /Scripts/AuctionsList.js HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
Referer: https://training.eauction.gr/
Host: training.eauction.gr
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US
HTTP/1.1 200 OK
Last-Modified: Fri, 21 Jul 2017 10:37:44 GMT
...
```

Entity:
AuctionList.js (Page)

Impact:
It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations
It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.

Causes:
Insecure web application programming or configuration

How to Fix:
Config your server to use the "X-XSS-Protection" header

7.4.11 - Low - Missing HTTP Strict-Transport-Security Header

Summary:
AppScan detected that the HTTP Strict-Transport-Security response header is missing

Raw Test Response:

```
HTTP/1.1 200 OK
Last-Modified: Fri, 21 Jul 2017 10:37:44 GMT
Server: Microsoft-IIS/8.5
Accept-Ranges: bytes
Content-Length: 825
X-Powered-By: ASP.NET
ETag: "08cd762d2d31:0"
Date: Sat, 07 Oct 2017 16:22:23 GMT
Content-Type: application/javascript
$(function () {
$.datepicker.setDefaults($.datepicker.regional[locale]);
$("#auctionDateFrom").datepicker({ dateFormat: 'dd/mm/yy' });
$("#auctionDateTo").datepicker({ dateFormat: 'dd/mm/yy' });
$(".AList-GridRow").mouseover(function () {
$(this).children().css("background-color", "#234baf").css("color", "#ffffff");
...
```

Entity:
AuctionList.js (Page)

Impact:
It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations
It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.

Causes:
Insecure web application programming or configuration

How to Fix:
Implement the HTTP Strict-Transport-Security policy

7.4.12 - Low - Query Parameter in SSL Request

Summary:
AppScan found parameters in the query part of the HTTP request, which was sent over SSL.

Original Request:

```
...
GET /Notification/List?notificationTypeId=0&itemsNumber=10&lastNotificationId=0&_=1507393200892
HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
Referer: https://training.eauction.gr/Auction/Details/102370
Cookie: __RequestVerificationToken=_ZOx32dRW1TWzATOFGdJQqmMnMpQeIVLFaIz-
GnQEM4V6r4xe6s1QdIDYh6BXYW47d4lROrV--VvhXFwkcfVCbmmVViGfnO7uYMiL4UlTDk1
Host: training.eauction.gr
```

```
X-Requested-With: XMLHttpRequest
Accept: */*
Accept-Language: en-US
...
```

Entity:
_(Parameter)

Impact:
It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted

Causes:
Query parameters were passed over SSL, and may contain sensitive information

How to Fix:
Always use SSL and POST (body) parameters when sending sensitive information.

7.4.13 - Low - Temporary File Download

Summary:

The test tried to retrieve a source code file. The fact that the response did not yield an error, and contained non-HTML contents, indicates that the source code retrieval succeeded.

Raw Test Response:

```
...
GET /Auction/Details/1023701 HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
Referer: https://training.eauction.gr/
Host: training.eauction.gr
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US
HTTP/1.1 200 OK
Content-Length: 16836
Server: Microsoft-IIS/8.5
X-AspNet-Version: 4.0.30319
X-Frame-Options: SAMEORIGIN
```

```
Set-Cookie:
__RequestVerificationToken=VOsbZNUFa8LSco43IJcGwfzqiGxUgV3VxFmNr3aUFeqhA2lBTGGtDf90zOascB4deH8VLD
D2Ew6i0X3UEwA5_KkClxCrQyFmuidDmjrd9Ec1; path=/; HttpOnly
X-Powered-By: ASP.NET
X-AspNetMvc-Version: 5.2
Date: Sat, 07 Oct 2017 16:58:16 GMT
Content-Type: text/html; charset=utf-8
Cache-Control: private
<!DOCTYPE html>
<html lang="el">
<head>
<meta charset="utf-8" />
<title>Στοιχεία Πλειστηριασμού - Υπηρεσία διενέργειας online πλειστηριασμών</title>
<link href="/Content/themes/base/jquery.ui.all.css" rel="stylesheet" type="text/css" />
<link href="/Content/Site.css" rel="stylesheet" type="text/css" />
...
```

Entity:
102370 (Page)

Impact:
It is possible to download temporary script files, which can expose the application logic and other sensitive information such as usernames and passwords

Causes:
Temporary files were left in production environment

How to Fix:
Remove old versions of files from the virtual directory

7.4.14 - Informational- Application Error

Summary:
The application has responded with an error message, indicating an undefined state that may expose sensitive information.

Raw Test Response:

```
...
GET /Auction/Details/1023701 HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
Referer: https://training.eauction.gr/
Host: training.eauction.gr
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US
HTTP/1.1 200 OK
Content-Length: 16836
Server: Microsoft-IIS/8.5
X-AspNet-Version: 4.0.30319
X-Frame-Options: SAMEORIGIN
Set-Cookie:
__RequestVerificationToken=VOsbZNUFa8LSco43IJcGwfzqiGxUgV3VxFmNr3aUFeqhA2lBTGGtDf90zOascB4deH8VLD
D2Ew6i0X3UEwA5_KkClxCrQyFmuidDmjrd9Ec1; path=/; HttpOnly
X-Powered-By: ASP.NET
X-AspNetMvc-Version: 5.2
Date: Sat, 07 Oct 2017 16:58:16 GMT
Content-Type: text/html; charset=utf-8
Cache-Control: private
<!DOCTYPE html>
<html lang="el">
<head>
<meta charset="utf-8" />
<title>Στοιχεία Πλειστηριασμού - Υπηρεσία διενέργειας online πλειστηριασμών</title>
```

```
<link href="/Content/themes/base/jquery.ui.all.css" rel="stylesheet" type="text/css" />
<link href="/Content/Site.css" rel="stylesheet" type="text/css" />
...
```

Entity:
pageNumber (Parameter)

Impact:
It is possible to gather sensitive debugging information

Causes:
Proper bounds checking were not performed on incoming parameter values
No validation was done in order to make sure that user input matches the data type expected

How to Fix:
Application Error

[1] Check incoming requests for the presence of all expected parameters and values. When a parameter is missing, issue a proper error message or use default values.
[2] The application should verify that its input consists of valid characters (after decoding). For example, an input value containing the null byte (encoded as %00), apostrophe, quotes, etc. should be rejected.
[3] Enforce values in their expected ranges and types. If your application expects a certain parameter to have a value from a certain set, then the application should ensure that the value it receives indeed belongs to the set. For example, if your application expects a value in the range 10..99, then it should make sure that the value is indeed numeric, and that its value is in 10..99.
[4] Verify that the data belongs to the set offered to the client.
[5] Do not output debugging error messages and exceptions in a production environment.

Integer Overflow

[1] Check incoming requests for the presence of all expected parameters and values. When a parameter is missing, issue a proper error message or use default values.
[2] The application should verify that its input consists of valid characters (after decoding). For example, an input value containing the null byte (encoded as %00), apostrophe, quotes, etc. should be rejected.
3] Enforce values in their expected ranges and types. If your application expects a certain parameter to have a value from a certain set, then the application should ensure that the value it receives indeed belongs to the set. For example, if your application expects a value in the range 10..99, then it should make sure that the value is indeed numeric, and that its value is in 10..99.
[4] Verify that the data belongs to the set offered to the client.
[5] Do not output debugging error messages and exceptions in a production environment.

.Net

Application Error

In order to disable debugging in ASP.NET, edit your web.config file to contain the following:
<compilation
debug="false"
/>

For more information, see "HOW TO: Disable Debugging for ASP.NET Applications" in:
http://support.microsoft.com/default.aspx?scid=kb;en-us;815157

You can add input validation to Web Forms pages by using validation controls. Validation controls provide an easyto-use mechanism for all common types of standard validation (for example, testing for valid dates or values within a range), plus ways to provide custom-written validation. In addition, validation controls allow you to completely customize how error information is displayed to the user. Validation controls can be used with any controls that are processed in a Web Forms page's class file, including both HTML and Web server controls.

To make sure that all the required parameters exist in a request, use the "RequiredFieldValidator" validation control. This control ensures that the user does not skip an entry in the web form.

To make sure user input contains only valid values, you can use one of the following validation controls:

[1] "RangeValidator": checks that a user's entry (value) is between specified lower and upper boundaries. You can check ranges within pairs of numbers, alphabetic characters, and dates.
[2] "RegularExpressionValidator": checks that the entry matches a pattern defined by a regular expression. This type of validation allows you to check for predictable sequences of characters, such as those in social security numbers, e-mail addresses, telephone numbers, postal codes, and so on.

Important note: validation controls do not block user input or change the flow of page processing; they only set an error state, and produce error messages. It is the programmer's responsibility to test the state of the controls in the code before performing further application-specific actions.

There are two ways to check for user input validity:

1. Test for a general error state:
In your code, test the page's IsValid property. This property rolls up the values of the IsValid properties of all the validation controls on the page (using a logical AND). If one of the validation controls is set to invalid, the page's property will return false.

2. Test for the error state of individual controls:
Loop through the page's Validators collection, which contains references to all the validation controls. You can then examine the IsValid property of each validation control.

Integer Overflow

In order to disable debugging in ASP.NET, edit your web.config file to contain the following:
<compilation
debug="false"
/>

For more information, see "HOW TO: Disable Debugging for ASP.NET Applications" in:
http://support.microsoft.com/default.aspx?scid=kb;en-us;815157

You can add input validation to Web Forms pages by using validation controls. Validation controls provide an easyto-use mechanism for all common types of standard validation (for example, testing for valid dates or values within a range), plus ways to provide custom-written validation. In addition, validation controls allow you to completely customize how error information is displayed to the user. Validation controls can be used with any controls that are processed in a Web Forms page's class file, including both HTML and Web server controls.

To make sure that all the required parameters exist in a request, use the "RequiredFieldValidator" validation control. This control ensures that the user does not skip an entry in the web form.

To make sure user input contains only valid values, you can use one of the following validation controls:

[1] "RangeValidator": checks that a user's entry (value) is between specified lower and upper boundaries. You can check ranges within pairs of numbers, alphabetic characters, and dates.

[2] "RegularExpressionValidator": checks that the entry matches a pattern defined by a regular expression. This type of validation allows you to check for predictable sequences of characters, such as those in social security numbers, e-mail addresses, telephone numbers, postal codes, and so on.

Important note: validation controls do not block user input or change the flow of page processing; they only set an error state, and produce error messages. It is the programmer's responsibility to test the state of the controls in the code before performing further application-specific actions.

There are two ways to check for user input validity:
1. Test for a general error state:
In your code, test the page's IsValid property. This property rolls up the values of the IsValid properties of all the validation controls on the page (using a logical AND). If one of the validation controls is set to invalid, the page's property will return false.

2. Test for the error state of individual controls:
Loop through the page's Validators collection, which contains references to all the validation controls. You can then examine the IsValid property of each validation control.

7.4.15 - Informational- Client-Side (JavaScript) Cookie References

Summary:
AppScan found a reference to cookies in the JavaScript.

Raw Test Response;

```
...
_cookie: function() {
var cookie = this.cookie ||
( this.cookie = this.options.cookie.name || "ui-tabs-" + getNextListId() );
return $.cookie.apply( null, [ cookie ].concat( $.makeArray( arguments ) ) );
},
...
```

Entity:
/* NUGET: BEGIN LICENSE TEXT (Page)

Impact:
The worst case scenario for this attack depends on the context and role of the cookies that are created at the client side

Causes:
Cookies are created at the client side

How to Fix:
[1] Avoid placing business/security logic at the client side.
[2] Find and remove insecure client-side Javascript code which may pose a security threat to the site.

7.4.16 - Informational- Email Address Pattern Found

Summary:
The response contains an e-mail address that may be private.

Raw Test Response:

```
...
<div id="body">
<section class="content-wrapper main-content clear-fix">
<h2>Επικοινωνία</h2>
<h3>Τηλέφωνο: 210 3307450</h3>
<h3>Email: info@eauction.gr</h3>
</section>
</div>
<footer>
<div class="content-wrapper">
<div class="footer-content">
<span><a class="footer-link" href="/Contact">Επικοινωνία</a></span><span
class="footer-sep">|</span>
...
```

Entity:
Contact (Page)

Impact:
It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations

Causes:
Insecure web application programming or configuration

How to Fix:
Remove any e-mail addresses from the website so that they won't be exploited by malicious users.

7.4.17 - Informational- Integer Overflow

Summary:
The application has responded with an error message, indicating an undefined state that may expose sensitive information.

Raw Test Response:

```
...
Host: training.eauction.gr
Content-Length: 225
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US
Content-Type: application/x-www-form-urlencoded
SelectedCountryNumericCode=99999999999999999999&PostFromHomePage=true&isEAuction=True&auctionDate
From=1234&auctionDateTo=1234&offerValueFrom=1234&offerValueTo=1234&debtorName=&sortingFieldId=0&s
ortAscending=False&pageNumber=1
HTTP/1.1 500 Internal Server Error
Server: Microsoft-IIS/8.5
Content-Length: 13532
X-AspNet-Version: 4.0.30319
Cache-Control: private
X-Powered-By: ASP.NET
Date: Sat, 07 Oct 2017 16:40:29 GMT
Content-Type: text/html; charset=utf-8
<!DOCTYPE html>
```

...

Entity:
SelectedCountryNumericCode (Parameter)

Impact:
It is possible to gather sensitive debugging information

Causes:
Proper bounds checking were not performed on incoming parameter values
No validation was done in order to make sure that user input matches the data type expected

How to Fix:
Application Error
[1] Check incoming requests for the presence of all expected parameters and values. When a parameter is missing, issue a proper error message or use default values.
[2] The application should verify that its input consists of valid characters (after decoding). For example, an input value containing the null byte (encoded as %00), apostrophe, quotes, etc. should be rejected.
[3] Enforce values in their expected ranges and types. If your application expects a certain parameter to have a value from a certain set, then the application should ensure that the value it receives indeed belongs to the set. For example, if your application expects a value in the range 10..99, then it should make sure that the value is indeed numeric, and that its value is in 10..99.
[4] Verify that the data belongs to the set offered to the client.
[5] Do not output debugging error messages and exceptions in a production environment.

Integer Overflow

[1] Check incoming requests for the presence of all expected parameters and values. When a parameter is missing, issue a proper error message or use default values.

[2] The application should verify that its input consists of valid characters (after decoding). For example, an input value containing the null byte (encoded as %00), apostrophe, quotes, etc. should be rejected.
[3] Enforce values in their expected ranges and types. If your application expects a certain parameter to have a value from a certain set, then the application should ensure that the value it receives indeed belongs to the set. For example, if your application expects a value in the range 10..99, then it should make sure that the value is indeed numeric, and that its value is in 10..99.
[4] Verify that the data belongs to the set offered to the client.
[5] Do not output debugging error messages and exceptions in a production environment.

.Net
Application Error

In order to disable debugging in ASP.NET, edit your web.config file to contain the following:
<compilation
debug="false"
/>

For more information, see "HOW TO: Disable Debugging for ASP.NET Applications" in:
http://support.microsoft.com/default.aspx?scid=kb;en-us;815157

You can add input validation to Web Forms pages by using validation controls. Validation controls provide an easyto-use mechanism for all common types of standard validation (for example, testing for valid dates or values within a range), plus ways to provide custom-written validation. In addition, validation controls allow you to completely
customize how error information is displayed to the user. Validation controls can be used with any controls that are processed in a Web Forms page's class file, including both HTML and Web server controls.

To make sure that all the required parameters exist in a request, use the "RequiredFieldValidator" validation control. This control ensures that the user does not skip an entry in the web form.

To make sure user input contains only valid values, you can use one of the following validation controls:
[1] "RangeValidator": checks that a user's entry (value) is between specified lower and upper boundaries. You can

check ranges within pairs of numbers, alphabetic characters, and dates.
[2] "RegularExpressionValidator": checks that the entry matches a pattern defined by a regular expression. This type of validation allows you to check for predictable sequences of characters, such as those in social security numbers, e-mail addresses, telephone numbers, postal codes, and so on.

Important note: validation controls do not block user input or change the flow of page processing; they only set an error state, and produce error messages. It is the programmer's responsibility to test the state of the controls in the code before performing further application-specific actions.

There are two ways to check for user input validity:

1. Test for a general error state:

In your code, test the page's IsValid property. This property rolls up the values of the IsValid properties of all the validation controls on the page (using a logical AND). If one of the validation controls is set to invalid, the page's property will return false.

2. Test for the error state of individual controls:

Loop through the page's Validators collection, which contains references to all the validation controls. You can then examine the IsValid property of each validation control.

Integer Overflow

In order to disable debugging in ASP.NET, edit your web.config file to contain the following:
<compilation
debug="false"
/>

For more information, see "HOW TO: Disable Debugging for ASP.NET Applications" in:
http://support.microsoft.com/default.aspx?scid=kb;en-us;815157

You can add input validation to Web Forms pages by using validation controls. Validation controls provide an easyto- use mechanism for all common types of standard validation (for example, testing for valid dates or values within a range), plus ways to provide custom-written validation. In addition, validation controls allow you to completely customize how error information is displayed to the user. Validation controls can be used with any controls that are processed in a Web Forms page's class file, including both HTML and Web server controls.

To make sure that all the required parameters exist in a request, use the "RequiredFieldValidator" validation control. This control ensures that the user does not skip an entry in the web form.

To make sure user input contains only valid values, you can use one of the following validation controls:

[1] "RangeValidator": checks that a user's entry (value) is between specified lower and upper boundaries. You can check ranges within pairs of numbers, alphabetic characters, and dates.

[2] "RegularExpressionValidator": checks that the entry matches a pattern defined by a regular expression. This type of validation allows you to check for predictable sequences of characters, such as those in social security numbers, e-mail addresses, telephone numbers, postal codes, and so on.

Important note: validation controls do not block user input or change the flow of page processing; they only set an error state, and produce error messages. It is the programmer's responsibility to test the state of the controls in the code before performing further application-specific actions.

There are two ways to check for user input validity:

1. Test for a general error state:

In your code, test the page's IsValid property. This property rolls up the values of the IsValid properties of all the validation controls on the page (using a logical AND). If one of the validation controls is set to invalid, the page's property will return false.

2. Test for the error state of individual controls:

Loop through the page's Validators collection, which contains references to all the validation controls. You can then examine the IsValid property of each validation control.

7.4.18 - Informational- Link to unclassified site

Summary:
The link is not listed in the IBM X-Force Exchange URL filter database as either safe or unsafe.

Entity:
http://msf-bi/AuctionDocuments/1234567890/Example.txt (Link)

Impact:
N/A

Causes:
N/A

How to Fix:
Verify that this link is indeed intended to be included in the Web application. If it is not, you may wish to determine how it came to be included.

7.4.19 - Informational- Possible Server Path Disclosure Pattern Found

Summary:
The response contains the absolute paths and/or filenames of files on the server.

Raw Test Response:

```
...
...60|72|8[2-5]|9[2-9]","(?:2[349]|45|54|60|72|8[2-5]|9[2-9])\\d1","(?:2[349]|45|54|60|72|8[2-
5]|9[2-9])\\d11","(?:2[349]|45|54|60|72|8[2-5]|9[2-9])\\d111"],"(0$1)"]...
...
```

Entity:
utils.js (Page)

Impact:
It is possible to retrieve the absolute path of the web server installation, which might help an attacker to develop further attacks and to gain information about the file system structure of the web application

Causes:
Latest patches or hotfixes for 3rd. party products were not installed

How to Fix:
Download the relevant security patch depending on the issue existing on your web server or web application.

Conclusions

The penetration test is a procedure that it must be a process that will necessarily run for every application created, whether it is Web based or is for local use within an organization. Security vulnerabilities, no matter how good the design and development phases are, can only be discovered during such a procedure. Even the smallest pathogenesis of one of the systems that make up the application can cause total destruction by a malicious user or malicious software.

The success thought of any penetration test depends on the underlying methodology. In order to perform a successful penetration test, the underlying methodology should also make use of different security tools. The selection of the tools were based on its versatility, usability and effectiveness. With all the tools in hand, each phase of the methodology were carried out in a systematic and methodological manner covering a complete penetration testing framework. A four phase methodology was proposed and implemented to test the web application environment, from its system infrastructure to its core application files. Such penetration test if performed in an orderly manner, can save extra money to protect the web application evaluating the effectiveness of the security services and safeguard the system form the potential threats, vulnerabilities and exploits.

In conclusion, tools and methodology, if properly utilized, can prove their usefulness for understanding the weaknesses of the network or systems and how they might be exploited. Penetration testing is not an alternative to other security measures. In fact, it should be used to complement the Defense in Depth principle. In today’s world of information security, where threats and vulnerabilities are changing and evolving, penetration testing tools and methods used to combat against such threats and vulnerabilities should also change and evolve along with technological advancement.