



Web Application Security Policy by Rahul Ghosh

Free Use Disclaimer: *This policy was created by or for the SANS Institute for the Internet community. All or parts of this policy can be freely used for your organization. There is no prior approval required. If you would like to contribute a new policy or updated version of this policy, please send email to policy-resources@sans.org.*

Things to Consider: *Please consult the Things to Consider FAQ for additional guidelines and suggestions for personalizing the SANS policies for your organization.*

Last Update Status: *Updated June 2014*

1. Overview

Web application vulnerabilities account for the largest portion of attack vectors outside of malware. It is crucial that any web application be assessed for vulnerabilities and any vulnerabilities be remediated prior to production deployment.

2. Purpose

The purpose of this policy is to define web application security assessments within **ACME Retail**. Web application assessments are performed to identify potential or realized weaknesses as a result of inadvertent mis-configuration, weak authentication, insufficient error handling, sensitive information leakage, etc. Discovery and subsequent mitigation of these issues will limit the attack surface of **ACME Retail** services available both internally and externally as well as satisfy compliance with any relevant policies in place. Before implementing PCI DSS in your organization, it's very important to determine the scope of the implementation. As a minimum we will need to identify infrastructure that is related to the storing, processing and transmitting of cardholder data, and identify all payments channels, locations and data flows. **To implement PCI DSS Compliance we need to follow these steps below:**

1. Determine the Scope of PCI DSS Compliance
2. Conduct a Gap Analysis
3. Develop Policies and Procedures
4. Train the personnel
5. Encrypt Sensitive Data
6. Ensure Physical Security of Data
7. Scramble Track Data
8. Use secured wireless connections
9. Review Logs Regularly
10. Minimize the scope
11. Regularly Update the Software



12. Implement a Layered Security System
13. Follow Mobile Payment Acceptance Guidelines
14. Adopt best practice for skimming prevention

3. Scope

This policy covers all web application security assessments requested by any individual, group or department for the purposes of maintaining the security posture, compliance, risk management, and change control of technologies in use at **ACME Retail**

All web application security assessments will be performed by delegated security personnel either employed or contracted by **ACME Retail**. All findings are considered confidential and are to be distributed to persons on a “need to know” basis. Distribution of any findings outside of **ACME Retail** is strictly prohibited unless approved by the Chief Information Officer.

Any relationships within multi-tiered applications found during the scoping phase will be included in the assessment unless explicitly limited. Limitations and subsequent justification will be documented prior to the start of the assessment.

4. Policy

4.1 Web applications are subject to security assessments based on the following criteria:

- a) New or Major Application Release – will be subject to a full assessment prior to approval of the change control documentation and/or release into the live environment.
- b) Third Party or Acquired Web Application – will be subject to full assessment after which it will be bound to policy requirements.
- c) Point Releases – will be subject to an appropriate assessment level based on the risk of the changes in the application functionality and/or architecture.
- d) Patch Releases – will be subject to an appropriate assessment level based on the risk of the changes to the application functionality and/or architecture.
- e) Emergency Releases – An emergency release will be allowed to forgo security assessments and carry the assumed risk until such time that a proper assessment can be carried out. Emergency releases will be designated as such by the Chief Information Officer or an appropriate manager who has been delegated this authority.

4.2 All security issues that are discovered during assessments must be mitigated based upon the following risk levels. The Risk Levels are based on the OWASP Risk Rating



Methodology. Remediation validation testing will be required to validate fix and/or mitigation strategies for any discovered issues of Medium risk level or greater.

- a) High – Any high risk issue must be fixed immediately or other mitigation strategies must be put in place to limit exposure before deployment. Applications with high risk issues are subject to being taken off-line or denied release into the live environment.
- b) Medium – Medium risk issues should be reviewed to determine what is required to mitigate and scheduled accordingly. Applications with medium risk issues may be taken off-line or denied release into the live environment based on the number of issues and if multiple issues increase the risk to an unacceptable level. Issues should be fixed in a patch/point release unless other mitigation strategies will limit exposure.
- c) Low – Issue should be reviewed to determine what is required to correct the issue and scheduled accordingly.

4.3 The following security assessment levels shall be established by the InfoSec organization or other designated organization that will be performing the assessments.

- a) Full – A full assessment is comprised of tests for all known web application vulnerabilities using both automated and manual tools based on the OWASP Testing Guide. A full assessment will use manual penetration testing techniques to validate discovered vulnerabilities to determine the overall risk of any and all discovered.
- b) Quick – A quick assessment will consist of a (typically) automated scan of an application for the OWASP Top Ten web application security risks at a minimum.
- c) Targeted – A targeted assessment is performed to verify vulnerability remediation changes or new application functionality.

4.4 **The current approved web application security assessment tools in use which will be used for testing are based on the [PCI DSS Requirements 6](#) :**

6.1 Establish a process to identify security vulnerabilities, using reputable outside sources for security vulnerability information, and assign a risk ranking (for example, as “high,” “medium,” or “low”) to newly discovered security vulnerabilities. A great way to identify security vulnerabilities is to go to the [OWASP Top Ten Project](#) which has the most up to date information and standards.

The top 10 security vulnerability based on risk are as follows:

Vulnerability	Risk
1. Injection	High
2. Broken Authentication and Session Management	High
3. Cross Site Request Forgery	High
4. Cross Site Scripting	High
5. Broken Access Control	High
6. Security Misconfiguration	Medium
7. Sensitive Data Exposure	Medium
8. Insufficient Attack Protection	Medium
9. Using Components with known vulnerabilities	Low
10. Under protected API's	Low

6.2 Ensure that all system components and software are protected from known vulnerabilities by installing applicable vendor supplied security patches. Install critical security patches within one month of release. When a new version of the application is released the **Dev Ops Team** should already have the list of bugs that they can share with their customers to notify them which ones are in the process of getting fixed which ones will take time. Identifying the bugs is very important because that can determine what kind of a flaw it is with the system and what's exploitable. ACME Retail can also perform a Threat Risk Model using Microsoft Threat Modeling Process and the process has five steps:

1. Identify Security Objectives | 2. Survey the Application
3. Decompose it | 4. Identify Threats | 5. Identify Vulnerabilities


Best Practices:

1. Ensure that any web interface in the product has been tested for XSS, SQL Injection and CSRF vulnerabilities
2. Ensure that any web interface has the ability to use HTTPS to protect transmitted information
3. Many deployments are either brownfield (i.e. applied over existing infrastructure) and/or have an extremely long deployment cycle. To maintain the security of devices over time it is critical to plan for patches and updates

6.3 Develop internal and external software applications (including web-based administrative access to applications) securely, as follows: • In accordance with PCI DSS (for example, secure authentication and logging) • Based on industry standards and/or best practices. • Incorporating information security throughout the SDL. In accordance with PCI DSS for eg. Secure authentication and logging, also incorporating information security throughout the software development life cycle.

Vulnerabilities include: 1. Access Control Flaw
 2. Authentication Flaw
 3. Malicious Execution
 4. Session Management Flaws

HIGH RISK (FULL)



Basic Authentication (High Risk): Basic authentication is not secure and should not be used in applications. The username and password are concatenated and sent in an HTTP header on every subsequent request. Compared with session based authentication, this substantially increases the amount of time the credentials are on the wire in plaintext.

Best Practices: Applications should enforce password complexity rules to discourage easy to guess passwords. Password mechanisms should allow virtually any character the user can type to be part of their password, including the space character. Passwords should, obviously, be case sensitive in order to increase their complexity. Occasionally, we find systems where passwords aren't case sensitive, frequently due to legacy system issues like old mainframes that didn't have case sensitive passwords.

Insufficient Account Lockout: Account lockout mechanisms are used to mitigate brute force password guessing attacks. Accounts are typically locked after 3 to 5 unsuccessful login attempts and can only be unlocked after a predetermined period of time, via a self-service unlocks mechanism, or intervention by an administrator. Account lockout mechanisms require a balance between protecting accounts from unauthorized access and protecting users from being denied authorized access.

Best Practices: Time-based lockout and unlock. Self-service unlock sends unlock email to registered email address. Manual administrator unlocks. Manual administrator unlock with positive user identification. Consider employing network segmentation technologies such as firewalls to isolate IoT systems from critical IT systems

6.4 Follow change control processes: Change control is a systematic approach to managing all changes made to a product or system. The purpose is to ensure that no unnecessary changes are made, that all changes are documented, that services are not unnecessarily disrupted and that resources are used efficiently. There are also ways to ensure the changes are properly managed by defining the change request, submit and review the change request, define options and create response document and final decision and approval. According to PCI DSS, development and test environments are never secure enough for holding cardholder data. These environments should thus be kept separate from production environments. A good way to keep them separate is through network segmentation. To achieve this segmentation, an Access Control List (ACL) on a switch or a firewall can be used. Similarly, to achieve compliance, personnel involved in development and test environment should not be in touch with that of production environment. This can go hand in hand with these vulnerabilities like **Error Handling and Logging**.

Vulnerabilities Include: 1. Improper Error Handling

2. Unhandled Exceptions

3. Information Exposure through Server Log Files

4. Username Harvesting / Enumeration

5. Insufficient Logging

6. Verbose Error Messages

Medium Risk

Low Risk

Best Practices:

1. Development/test environments are separate from production environments with access control in place to enforce separation.
2. A separation of duties between personnel that are assigned to the development/test environments and those persons assigned to the production environment.
3. Production data (live PANs) is not used for testing or development.
4. Test data and accounts are removed before a production system becomes active.
5. Change control procedures that are related to implementing security patches and software modifications are documented.

6.5 Address common coding vulnerabilities in software-development processes as follows:

Train developers at least annually in up to-date secure coding techniques, including how to avoid common coding vulnerabilities. Develop applications based on secure coding guidelines.

It is important for web development teams to understand that client side controls like client based **input-validation, hidden fields and interface controls** (e.g., pull downs and radio buttons), provide little if any Security benefit. An attacker can use tools like client side web proxies (e.g. **OWASP Web Scarab, Burp**) or **network packet capture tools** (e.g., **Wireshark**) to analyze application traffic and submit custom built requests, bypassing the interface all together.

Additionally, Flash, Java Applets and other client side objects can be decompiled and analyzed for flaws. Software security flaws can be introduced at any stage of the software development lifecycle, including:

- Not identifying security requirements up front
- Creating conceptual designs that have logic errors
- Using poor coding practices that introduce technical vulnerabilities
- Deploying the software improperly
- Introducing flaws during maintenance or updating
- Furthermore, it is important to understand that software vulnerabilities can have a scope beyond the software itself. Depending on the nature of the software, the vulnerability and the supporting infrastructure, the Impacts of a successful exploitation can include compromises to any or all of the following:
 - The software and its associated information
 - The operating systems of the associated servers
 - The backend database
 - Other applications in a shared environment
 - The user's system
 - Other software that the user interacts with
- Use tested and approved managed code rather than creating new unmanaged code for common tasks



The goal of software security is to maintain the **confidentiality, integrity, and availability of information resources in order to enable successful business operations.** This goal is accomplished through the implementation of security controls. This focuses on the technical controls specific to mitigating the occurrence of common software vulnerabilities. While the primary focus is web applications and their supporting infrastructure, most of the guidance can be applied to any software deployment platform.

Data Validation Vulnerabilities:

1. Buffer Overflow
2. Cross Site Scripting
3. Cross Site Request Forgery
4. HTTP Splitting
5. Injection Flaws
6. Command Injection
7. SQL Injection
8. XPATH Injection
9. Parameter Tampering

HIGH RISK

Best Practices for Input Validation:

- Conduct all data validation on a trusted system (e.g., The server)
- Identify all data sources and classify them into trusted and untrusted. Validate all data from untrusted sources (e.g., Databases, file streams, etc.)
- There should be a centralized input validation routine for the application
- Specify proper character sets, such as UTF-8, for all sources of input
- Encode data to a common character set before validating

Best Practices for Output Encoding:

- Conduct all encoding on a trusted system eg. Server
- Utilize a standard, tested routine for each type of outbound encoding
- Encode all characters unless they are known to be safe for the intended interpreter

Best Practices for Personnel:

- Train developers in secure coding techniques, including how to avoid common coding vulnerabilities, and understanding how sensitive data is handled in memory.

6.6 For public-facing web applications, address new threats and vulnerabilities on an ongoing basis and ensure these applications are protected against known attacks by either of the following methods: Reviewing public-facing web applications via manual or automated application vulnerability security assessment tools or methods, at least annually and after any changes. The OWASP Automated Threats to Web Applications Project has completed a review of reports, academic and other papers, news stories and vulnerability taxonomies/listings to identify, name and classify these scenarios – automated by software causing a divergence from accepted behavior producing one or more undesirable effects on a web application, but

excluding tool-based exploitation of single-issue vulnerabilities. The initial objective was to produce an ontology providing a common language for developers, architects, operators, business owners, security engineers, purchasers and suppliers/ vendors, to facilitate clear communication and help tackle the issues.

Web applications are subjected to unwanted automated usage – day in, day out. Often these events relate to misuse of inherent valid functionality, rather than the attempted exploitation of unmitigated vulnerabilities. Also, excessive misuse is commonly mistakenly reported as **application denial-of-service (DoS) like HTTP-flooding, when in fact the DoS is a side-effect instead of the primary intent.** Frequently these have sector-specific names. Most of these problems seen regularly by web application owners are not listed in any **OWASP Top Ten or other top issue list.** Furthermore, they are not enumerated or defined adequately in existing dictionaries. These factors have contributed to inadequate visibility, and an inconsistency in naming such threats, with a consequent lack of clarity in attempts to address the issues.

Without sharing a common language between devops, architects, business owners, security engineers, purchasers and suppliers/vendors, everyone has to make extra effort to communicate clearly. Misunderstandings can be costly. The adverse impacts affect the privacy and security of individuals as well as the security of the applications and related system components.

Vulnerabilities:

- 1. Account Enumeration:** Use by an intermediary application that collects together multiple accounts and interacts on their behalf
- 2. Aggregation:** Use by an intermediary application that collects together multiple accounts and interacts on their behalf
- 3. Credential Cracking:** Identify valid login credentials by trying different values for usernames and/or passwords
- 4. Denial of Service:** Target resources of the application and database servers, or individual user accounts, to achieve denial of service (DoS)
- 5. Scraping:** Collect application content and/or other data for use elsewhere
- 6. Spamming:** Malicious or questionable information addition that appears in public or private content, databases or user messages

Best Practices: Value: Removing or limiting the value of assets accessed using the application can reduce the benefits of an automated attack. This includes reviewing whether the data and/or functionality are necessary, or whether it can be changed to reduce its value to an attacker.

Requirements: Identify relevant automated threats in security risk assessment, and assess effects of alternative countermeasures on functionality usability and accessibility. Use this to then define additional application development and deployment requirements

Testing: Create abuse and misuse test cases that simulate automated web attacks.

Fingerprinting: Consider identifying and restricting automated usage by automation identification techniques. Utilize user agent string, and/or HTTP request format (e.g. header

ordering), and/or HTTP header anomalies (e.g. HTTP protocol, header inconsistencies), and/or device fingerprint content to determine whether a user is likely to be a human or not.

System Configuration: Examine the system configuration settings and interview responsible personnel to verify that an automated technical solution that detects and prevents web-based attacks (for example, a web-application firewall) is in place as follows: - Is situated in front of public-facing web applications to detect and prevent web-based attacks. - Is actively running and up to date as applicable. - Is generating audit logs. - Is configured to either block web-based attacks, or generate an alert that is immediately investigated.

Firewall: Installing an automated technical solution that detects and prevents web based attacks for example, a web application firewall in front of public facing web applications, to continually check all traffic.

Automated Tools: Reviewing public-facing web applications via manual or automated application vulnerability security assessment tools or methods, at least annually and after any changes.

6.7 Ensure that security policies and operational procedures for developing and maintaining secure systems and applications are documented, in use, and known to all affected parties. Confidentiality, integrity and availability of data by the Information Security Triad are the 3 things necessary. As we went underway from requirement 6.1 to 6.6, all compliance measures taken were recommended to be documented. Requirement 6.7 requires that this documentation is properly managed and made available to all the concerned parties. To make sure that all the required documentation stays updated, it is recommended to conduct an annual review and document the review process as well. All documentation must be accessible to all the involved personnel and any changes in policies and procedures should also be communicated to the concerned individuals.

Policies Personnel Needs to be Aware of:

1. Ensure that any web interface in the product has been tested for **XSS, SQL Injection and CSRF** vulnerabilities
2. Ensure that any web interface has the ability to use HTTPS to protect transmitted information
3. Applications should enforce password complexity rules to discourage easy to guess passwords. Password mechanisms should allow virtually any character the user can type to be part of their password, including the space character. Passwords should, obviously, be case sensitive in order to increase their complexity.
4. A separation of duties between personnel that are assigned to the development/test environments and those persons assigned to the production environment.
5. Production data is not used for testing or development
6. Test data and accounts are removed before a production system becomes active.
7. Conduct all data validation on a trusted system (e.g., The server)
8. Conduct all encoding on a trusted system eg. Server
9. Utilize a standard, tested routine for each type of outbound encoding



10. Reviewing public-facing web applications via manual or automated application vulnerability security assessment tools or methods, at least annually and after any changes.

5. Policy Compliance

5.1 Compliance Measurement

The Infosec team will verify compliance to this policy through various methods, including but not limited to, periodic walk-thrus, video monitoring, business tool reports, internal and external audits, and feedback to the policy owner.

5.2 Exceptions

Any exception to the policy must be approved by the Infosec team in advance.

5.3 Non-Compliance

An employee found to have violated this policy may be subject to disciplinary action, up to and including termination of employment.

Web application assessments are a requirement of the change control process and are required to adhere to this policy unless found to be exempt. All application releases must pass through the change control process. Any web applications that do not adhere to this policy may be taken offline until such time that a formal assessment can be performed at the discretion of the Chief Information Officer.

6 Related Standards, Policies and Processes

[PCI DSS Requirements 6](#)

[OWASP Top Ten Project](#)

[OWASP Testing Guide](#)

[OWASP Risk Rating Methodology](#)

7 Definitions and Terms

None.

8 Revision History

Date of Change	Responsible	Summary of Change
August 6 th , 2017	SANS Policy Team	Updated and converted to new format.



10. References

- [1] PCI Security Standards Council. *PCI DSS REQUIREMENTS* https://www.pcisecuritystandards.org/document_library?category=pci_dss&document=pci_dss
- [2] OWASP.org. *OWASP Top Ten Project*. https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
- [3] Youtube.com. *SQLMAP Techniques Error Based Exploitation*. <https://www.youtube.com/watch?v=S9piWY0aAdI>.
- [4] OWASP Testing Guide. (2008, V3). Retrieved August 4, 2017 from https://www.owasp.org/images/5/56/OWASP_Testing_Guide_v3.pdf
- [5] OWASP Rating Methodology. *Risk Rating* https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology