COMP307 Assignment 7

Kevin Ma – 300867968

Ostap Hamarnyk – 300836326

Monday, November 5, 2018

# What are some guidelines for handling credit cards on websites?

## Best Practices

Keeping your customer’s credit card data is essential in current world. Customers hate being force to replace their credit cards. If client data was exposed, it usually means the end of a business as well. However, there are best practices that companies should follow to handle and store credit card payment transaction:

* 1. Process transactions immediately online or hand off the processing to your bank
  2. Do not store any CC numbers, ever. If they must be stored, you must follow the PCI guidelines to the letter. We strongly urge you to not store credit card details.
  3. If you are using a shared host for your site, you cannot comply with the PCI guidelines. You must have your own infrastructure to comply with the PCI guidelines.

## Auth Numbers

After successfully processing a transaction, you are returned an authorization number. This is unique per transaction and has no intrinsic value of its own. It is safe to store this value, write it to logs, present it to staff and e-mail it to the customer.

## Handling Recurring Payments

You should not be storing credit card info unless it’s a recurring payment. However, you need to follow all the merchant agreement rules as well as meet all the PCI guidelines. Also, the payment should not last more than one year and details of credit card should be expunged as soon as the agreement is finished.

## Displaying portions of the credit card

Another rule is to show portions of credit card. PCI allows the presentation of first six or last four digits, but none should be shown if possible. You must not store the CCV, CCV2, and PVV since these values are used to validate information by many payment gateways to protect against imprint fraud as the value is on the reverse of the card.

## Patching and maintenance

Also, PCI requires you to keep your systems up to date. Once the patch has been released you should update your systems within one month. Furthermore, you’re required to have anti-virus software which should be up to date as well.

## Reversals

There are two potential frauds from reversals: an insider pushing money from the organization's account to a third party, and an outsider who has successfully figured out how to use an automated reversal process to "refund" money which is not owing, for example by using negative numbers. Therefore, reversals should be performed by hand and signed off by two distinct employees or groups. It is crucial that all values are within the limit and signing authority is properly assigned.

## Chargeback

There are legitimate reasons for charge backs, and your local consumer laws will tell you what they are. However, most issuers take a dim view of merchants with a high charge back ratio as it costs them a lot of time and money and indicates a lack of fraud controls.

# What are some known data validation strategies?

**There are four strategies for validating data:**

## Accept know good

If you’re expecting a value from user that has a specific format and can contain specific characters, digits etc. you should validate for it.

## Reject known bad

If you don’t expect to see specific characters in input string, reject it. For instance, you’re not expecting % to appear in Canadian postal code. This method is slow and insecure since you must run your field against many regular expressions.

## Sanitize

Eliminate or translate characters to make the input safe. For example, replace ‘ with &rsquo; in HTML source code. This doesn’t work well in practice as there are many exceptions to the rule.

## No Validation

This is inherently unsafe and strongly discouraged. Each example of no validation usually leads to direct obviation of application, host and network security controls.

It is faster, simpler and more secure to validate a single correct positive test rather than try to include complex and slow sanitization routines for all current and future attacks.

Data should be always strong typed, length or range checked, unsigned unless needs to be as well as syntax and grammar should be checked prior to first use or inspection.

# When should the testing process be introduced in the SDL?

Most people today don’t test software until it has already been created and is in the deployment phase of its life cycle (i.e., code has been created and instantiated into a working web application). This is generally a very ineffective and cost-prohibitive practice. One of the best methods to prevent security bugs from appearing in production applications is to improve the Software Development Life Cycle (SDLC) by including security in each of its phases. An SDLC is a structure imposed on the development of software artefacts. Companies should inspect their overall SDLC to ensure that security is an integral part of the development process. SDLCs should include security tests to ensure security is adequately covered and controls are effective throughout the development process.

# What is black-box testing?

It is a testing method, also known as Behavioral Testing, in which the internal structure/design of item being tested is unknown to tester. These can be both functional or non-functional, however they are usually functional. This testing method is used to find incorrect or missing functions, interface errors, errors in data structures or external database access, behavior or performance error as well as initialization and termination errors. This testing method can be applied to the following levels of software testing: **Integration testing, System testing and Acceptance testing.** Meanwhile, you should understand that the higher the level, the bigger and more complex the box. That is when black-box testing method comes in to practice.

## Advantages

* Tests are done from a user’s point of view and will help in exposing discrepancies in the specifications.
* Tester need not know programming languages or how the software has been implemented.
* Tests can be conducted by a body independent from the developers, allowing for an objective perspective and the avoidance of developer-bias.
* Test cases can be designed as soon as the specifications are complete.

## Disadvantages

* Only a small number of possible inputs can be tested, and many program paths will be left untested.
* Without clear specifications, which is the situation in many projects, test cases will be difficult to design.
* Tests can be redundant if the software designer/developer has already run a test case.
* Ever wondered why a soothsayer closes the eyes when foretelling events? So is almost the case in Black Box Testing.

# What are some basic practices for handling authentication in web apps?

Authentication is in the critical pathway for securing web applications and would have a heavy impact in all environments pertaining to the web app. Ergo, it is in the best interest of a business to provide secure authentication services to their web applications. This may be provided through the means of:

* Tying a system identity to an individual user using credentials
* Providing reasonable authentication controls as per the application’s risk
* Denying access to attackers who use various methods to attack the authentication system

Some best practices for handling authentication in web apps are as follows:

* Ensure all internal and external connections (user and entity) go through an appropriate and adequate form of authentication. Be assured that this control cannot be bypassed.
* Ensure all pages enforce the requirement for authentication.
* Ensure that whenever authentication credentials or any other sensitive information is passed, only accept the information via the HTTP “POST” method and will not accept it via the HTTP “GET” method.
* Any page deemed by the business or the development team as being outside the scope of authentication should be reviewed in order to assess any possibility of security breach.
* Ensure that authentication credentials do not traverse the wire in clear text form.
* Ensure development/debug backdoors are not present in production code.
* Authentication is only as strong as your user management processes
  + This pertains to the user issuance and evidence of identity policies (i.e. non-repudiation)
  + However, one must take into consideration, the stronger the requirements for non-repudiation employed, the more expensive (more overhead and resources consumed) the process becomes
    - Must find a balance depending on the requirements of the app (as dictated by the business)
* Use the most appropriate form of authentication suitable for your asset classification
  + For example, it is suitable to use username and password as a means for authenticating low value systems such as blogs and forums. However, for systems which are more critical (or contains sensitive data), more secure forms of authentication should be used such as transaction signing.
* Re-authenticate the user for high value transactions and access to protected areas
  + What this means is that even if a user is authenticated to the web application, they might not necessarily have the authorization to execute the high-level transactions or access protected areas. Alternatively, users may have left their previously authenticated sessions open and a malicious user may have acquired access to the session. Thus, it is imperative for users to be required to re-authenticated when trying to access protected areas or execute high value transactions.
* Authenticate the transaction, not the user
  + Malicious users, such as phishers, rely on poorly implemented user authentication schemas in order to cause harm to a system. Users should be required to authenticate each of their transactions (in addition to authenticating their user) as opposed to simply being able to execute transactions upon authenticating their user. This extra layer of security can be what potentially saves a customer or the business much anguish, time and money. An example of this can be seen in the banking industry. In internet banking, customers are required to digitally “sign” transactions in order to preserve the authenticity an integrity of the online transaction.
* Passwords are trivially broken and are unsuitable for high value systems
  + This means that while passwords may be suitable to use in low value systems, they are not suitable and should not be used in high value systems. Even for low value systems, non-trivial passwords should be enforced so that they are not easily cracked. For example, any passwords less than 16 characters in length can be brute forced in less than two weeks. Password policies should be enforced such that passwords are non-trivial:
    - * Train the users on how to construct suitable passwords
      * Allow users to write down their passwords if they keep them safe
      * Encourage users to use pass phrases instead of passwords
      * Relax password expiry requirements upon the strength of the password chosen
        + For example, passwords between 8 and 16 characters that cannot be easily cracked should have an expiry of no less than 30 days, and pass phrases above 16 characters probably do not need a hard expiry limit, but a gentle reminder after 90 days instead

# Identify and describe the four maturity levels for security practice of OpenSAMM.

The OWASP (Open Web Application Security Project) SAMM (Software Assurance Maturity Model) uses a 0 to 3 maturity grading structure across 4 critical business functions, each containing 3 security practices to rate software development activities.

## Maturity Levels

Each of the twelve security practices of the OWASP SAMM has three defined maturity levels and an implicit starting point at zero. The details for the levels differ between the practices but they are generally as follows:

### Level 0

Implicit starting point representing the activities in the practice being unfulfilled

### Level 1

Initial understanding and adhoc provision of security practice

### Level 2

Increase efficiency and/or effectiveness of the security practice

### Level 3

Comprehensive mastery of the security practice at scale

### Governance

|  |  |  |  |
| --- | --- | --- | --- |
| Security Practice | Level 1 | Level 2 | Level 3 |
| Strategy & Metrics | Establish a unified strategic roadmap for software security within the organization by:  (1) estimate overall business risk profile  (2) build and maintain assurance roadmap | Measure relative value of data and software assets and choose risk tolerance by:  (1) Classify data and applications based on business risk  (2) Establish and measure per-classification security goals | Align security expenditure with relevant business indicators and asset value by:  (1) Conduct periodic industry-wide cost comparisons  (2) Collect metrics for historic security expenditure |
| Policy & Compliance | Understand relevant governance and compliance drivers to the organization by:  (1) Identify and monitor external compliance drivers  (2) Build and maintain compliance guidelines | Establish security and compliance baseline and understand per-project risks by:  (1) Build policies and standards for security and compliance  (2) Establish project audit practice | Require compliance and measure projects against organization-wide policies and standards by:  (1) Create compliance gates for projects  (2) Adopt solution for audit data collection |
| Education & Guidance | Offer development staff access to resources around the topics of secure programming and deployment by:  (1) Conduct technical security awareness training  (2) Build and maintain technical guidelines | Educate all personnel in the software lifecycle with role-specific guidance on secure development by:  (1) Conduct role-specific applications security training  (2) Utilize security coaches to enhance project teams | Mandate comprehensive security training and certify personnel for baseline knowledge by:  (1) Create formal application security support portal  (2) Establish role-based examination/certification |

### Construction

|  |  |  |  |
| --- | --- | --- | --- |
| Security Practice | Level 1 | Level 2 | Level 3 |
| Threat Assessment | Identify and understand high-level threats to the organization and individual projects by:  (1) Building and maintaining application-specific threat models  (2) Developing attacker profile from software architecture | Increase accuracy of threat assessment and improve granularity of per-project understanding by:  (1) Building and maintain abuse-case models per project  (2) Adopt a weighting system for measurement of threats | Concretely align compensating controls to each threat against internal and third-party software by:  (1) Explicitly evaluating risk from third-party components  (2) Elaborating threat models with compensating controls |
| Security Requirements | Consider security explicitly during the software requirements process by:  (1) Deriving security requirements from business functionality  (2) Evaluating security and compliance guidance for requirements | Increase granularity of security requirements derived from business logic and known risks by:  (1) Building an access control matrix for resources and capabilities  (2) Specify security requirements based on known risks | Mandate security requirements process for all software projects and third-party dependencies by:  (1) Building security requirements into supplier agreements  (2) Expanding audit program for security requirements |
| Secure Architecture | Insert consideration of proactive security guidance into the software design process by:  (1) Maintaining list of recommended software frameworks  (2) Explicitly applying security principles to design | Direct the software design process toward known-secure services and secure-by-default designs by:  (1) Identify and promote security services and infrastructure  (2) Identify security design patterns from architecture | Formally control the software design process and validate utilization of secure components by:  (1) Establishing formal reference architectures and platforms  (2) Validating usage of frameworks, patterns, and platforms |

### Verification

|  |  |  |  |
| --- | --- | --- | --- |
| Security Practice | Level 1 | Level 2 | Level 3 |
| Design Review | Support ad-hoc reviews of software design to ensure baseline mitigations for known risks by:  (1) Identifying software attack surfaces  (2) Analyzing design against known security requirements | Offer assessment services to review software design against comprehensive best practices for security by:  (1) Inspecting for complete provision of security mechanisms  (2) Deploying design review service for project teams | Require assessments and validate artifacts to develop detailed understanding of protection mechanisms by:  (1) Developing data-flow diagrams for sensitive resources  (2) Establishing release gates for design review |
| Implementation Review | Opportunistically find basic code-level vulnerabilities and  other high-risk security issues by:  (1) Creating review checklists from known security requirements  (2) performing point-review of high-risk code | Make implementation review during development more accurate and efficient through automation by:  (1) Utilizing automated code analysis tools  (2) Integrating code analysis into development process | Mandate comprehensive implementation review process to discover language-level and application-specific risks by:  (1) Customizing code analysis for application-specific concerns  (2) Establishing release gates for code review |
| Security Testing | Establish process to perform basic security tests based on implementation and software requirements by:  (1) Deriving test cases from known security requirements  (2) Conducting penetration testing on software releases | Make security testing during development completer and more efficient through automation by:  (1) Utilizing automated security testing tools  (2) Integrating security testing into development process | Require application-specific security testing to ensure baseline security before deployment by:  (1) Employing application-specific security testing automation  (2) Establishing release gates for security testing |

### Operations

|  |  |  |  |
| --- | --- | --- | --- |
| Security Practice | Level 1 | Level 2 | Level 3 |
| Issue Management | Understand high-level plan for responding to issue reports or incidents by:  (1) Identifying point of contact for security issues  (2) Creating informal security response team(s) | Elaborate expectations for responsive process to improve consistency and communications by:  (1) Establishing consistent issue response process  (2) Adopting a security issue disclosure process | Improve analysis and data gathering within response process for feedback into proactive planning by:  (1) Conducting root cause analysis for issues  (2) Collecting per-issue metrics |
| Environment Hardening | Understand baseline operational environment for applications and software components by:  (1) Maintaining operational environment specifications  (2) Identifying and installing critical security upgrades and patches | Improve confidence in application operations by hardening the operating environment by:  (1) establishing routine patch management processes  (2) monitoring baseline environment configuration statuses | Validate application health and status of operational environment against known best practices by:  (1) Identifying and deploying relevant operations protection tools  (2) Expanding audit program for environment configuration |
| Operational Enablement | Enable communications between development teams and operators for critical security-relevant data by:  (1) capturing critical security information for deployment  (2) documenting procedures for typical application alerts | Improve expectations for continuous secure operations through provision of detailed procedures by:  (1) creating per-release change management procedures  (2) maintaining formal operational security guides | Mandate communication of security information and validate artifacts for completeness by:  (1) Expanding audit program for operational information  (2) Performing code signing for application components |

# What are some activities an organization could perform for the security practice of “Threat Assessment”?

## Construction

Construction is the critical business function part of SAMM that is concerned with the processes and activities related to how an organization defines goals and creates software within development projects. It focuses on product management, requirements gathering, high-level architecture specification, detailed design and implementation of the product. The three security practices associated with Construction in SAMM are:

* + 1. Threat Assessment
    2. Security Requirements
    3. Secure Architecture

## Threat Assessment

This security practice is focused on identification and understanding the project-level risks based on the business, functionality of the software being developed and characteristics of the runtime environment. Facts are gathered from many sources such as commonly known threats, and likely attacks against each project. The organization uses this understanding to facilitate risk management such that it more effectively responds to risks and makes better decisions about prioritizing initiatives for security. Depending on the level of maturity an organization wants to reach, there are different activities in which they can perform for the security practice of Threat Assessment. They are as follows:

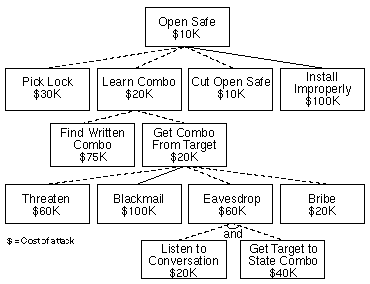
### Level 0:

Do nothing.

### Level 1:

The goal here is to identify and understand high-level threats to the organization and individual projects.

Activities that an organization can perform:

1. Build and maintain application-specific threat models
   * 1. Based purely on the business purpose of each software project and business risk profile
     2. Identifies worst-case scenarios for the software under development in each project team which are likely to occur
        1. Using **attack trees** or more formal threat modeling process such as **Microsoft’s STRIDE, Trike, etc.**
           1. To build attack trees, identify each worst-case scenario in one sentence and label these as the high-level goals of an attacker. From each attacker goal identified, identify preconditions that must hold for each goal to be realized.
           2. For example:
2. Develop attacker profile from software architecture
3. Conduct assessment to identify all likely threats to the organization regarding the software project
4. Focus only on threats from users with malicious intents, omitting known vulnerabilities, potential weaknesses, etc.

### Level 2:

The goal here is to increase accuracy of threat assessment and improve granularity of per-project understanding.

Activities that an organization can perform:

1. Build and maintain abuse-case models per project
2. Conduct more formal analysis to determine potential misuse or abuse of functionality
   * + 1. Begins with identification of normal usage scenarios e.g. use-case diagram
3. End goal is to have a completed set of abuse statements which can form a model for usage patterns that should be disallowed by software
4. Adopt a weighting system for measurement of threats
5. Using the established attacker profiles, identify a rating system that allows relative comparison between the threats
6. Using the new ratings for each threat, prioritize risk mitigation activities within the development lifecycle

### Level 3:

The goal here is to concretely tie compensating controls to each threat against internal and third-party software.

Activities that an organization can perform:

1. Explicitly evaluate risk from third-party components
2. Open-source projects, purchased consumer off the shelf software, online services, etc.
3. Elaborate attacker profiles for the software project for each third-party component the project is using
   * + 1. Update software to mitigate new risks discovered
4. Elaborate threat models with compensating controls
5. Identify factors which directly prevent preconditions from the threat models from being met
   * + 1. These mitigation factors are the compensating controls that formally address the direct risks from the software
6. End goal is to maximize coverage in terms of controls that mark parts of the threat model as mitigated

# What are the two recommended assessment styles for SAMM and how are they used?

An organization can measure itself against the defined Security Practices presented in SAMM to create a future roadmap for iterative improvement. To begin using SAMM, an organization must first conduct an assessment to determine the Maturity Level at which it is currently performing. In general, there are two recommended assessment styles for SAMM:

## 1. Lightweight

* Assessment worksheets for each Security Practice are evaluated and scores are assigned based on answers
* This is usually enough for an organization that is simply trying to map their existing assurance program into SAMM and understand where they currently stand



## 2. Detailed

* Performed after the assessment worksheets are evaluated and scores have been derived
* Additional audit work is performed to ensure that the Activities prescribed by each Security Practice are in place
  + The data from the Success Metrics for each Security Practice are collected to check that the organization’s performance is meeting the expected levels of compliance

