IDS SDLC

Version 1.2

CHANGES HISTORY

Date	Version	Description	Author
Nov 28, 2016	1.0	IDS SDLC description	Denis Koloshko (denis.koloshko@ids-group.co.uk) Chief Technical Officer
Jul 21, 2017	1.1	Reviewed	Denis Koloshko (denis.koloshko@ids-group.co.uk) Chief Technical Officer
Aug 15, 2018	1.2	Software Assurance Maturity Model added into Pre-SDLC phase Reviewed artefacts produced in each phase	Denis Koloshko (denis.koloshko@ids-group.co.uk) Chief Technical Officer

Summary: This document provides an in-depth description of the IDS SDLC methodology

Document Authors:	Denis Koloshko
Reviewed by:	
Version:	1.2
Date:	Aug, 2018

TABLE OF CONTENTS

Changes history **2**

Introduction **5**

Abstract 5

Purpose of the document 5

1. IDS SDLC **6**

1.1. Basic 6

1.2. Lifecycle 6

1.3. Principles 7

1.4. Testing Plan 8

1.5. Agile 9

2. Pre-SDLC **10**

2.1. Building Assurance Model 10

2.2. Trainings 10

2.2.1. Basic Concepts 11

2.2.2. Advanced Concepts 11

2.3. Policy and standards, code conventions 11

2.4. Artefacts 11

3. Phase 1: Define **12**

3.1. Assign Security Manager 12

3.2. Define compliances and standards 12

3.3. Classify Data and define data retention period 12

3.4. Define Use Case Authorization Matrix and collect business security requirements 13

3.5. Artefacts 14

4. Phase 2: Design **16**

4.1. Secure Design Principles 16

4.2. Establish Design Requirements 16

4.2.1. Architecture Design Document 16

4.3. Use Threat Model 17

4.4. Artefacts 18

5. Phase 3: Develop 19

5.1. Source Code Review 19
5.2. Security testing via unit testing 19
5.3. Static Code Analysis 19
5.4. Secured CI/CD process 19
5.5. Artefacts 19

6. Phase 4: Test 21

6.1. Security Testing 21
6.2. Artefacts 21

7. Phase 5: Deploy 22

7.1. Test production environment 22
7.2. Artefacts 22

8. Phase 6: Maintain 23

8.1. Incident Response Plan 23
8.2. Monthly security day 23
8.3. Artefacts 23

9. References 24

INTRODUCTION

Abstract

The Security Development Lifecycle (SDLC) is a software development process that helps developers build more secure software and address security compliance requirements. Combining a holistic and practical approach, the SDLC introduces security and privacy early and throughout all phases of the development process.

IT Band together with ids-group.co.uk (here and after simple IDS) is an international software services and solutions provider. IDS focuses on IT consulting and custom software engineering services for both the U.S. and European markets, leveraging global resources.

Business domain of the company includes projects with high security requirements. Extending development process with security practices, the company developed own SDLC model for such projects. The name of the process is **IDS SDLC**.

Purpose of the document

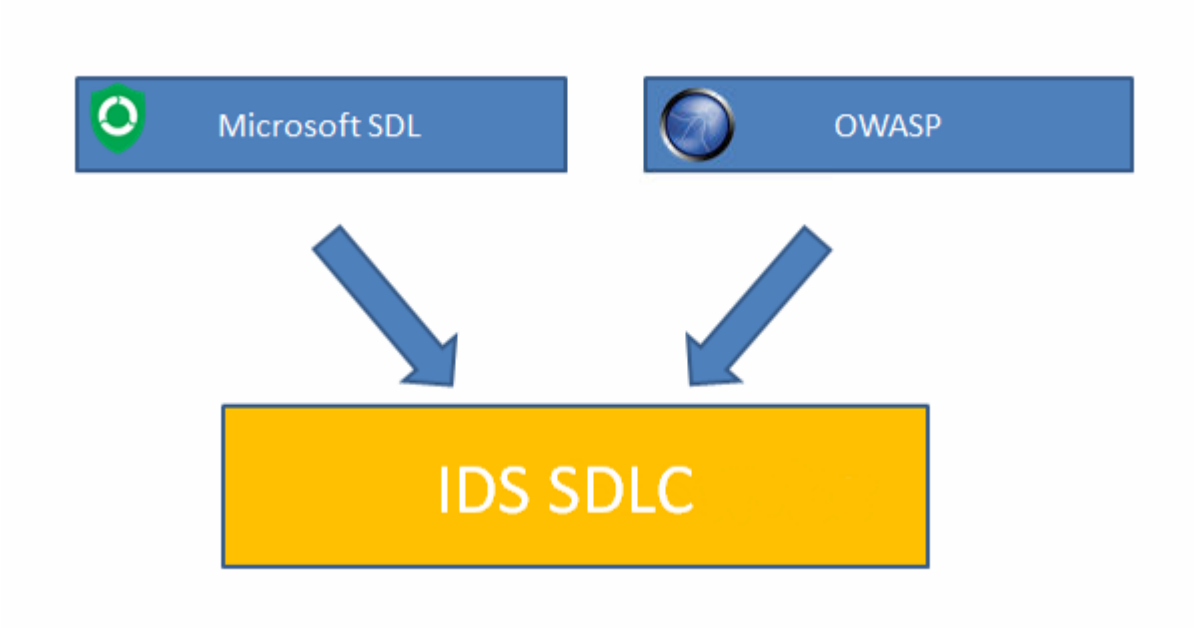
Purpose of this document is to define concept of IDS SDLC and security practices that should be injected into each phase of internal company development process.

The document is intended for project managers, technical leaders, and system architects to help with planning and organizing security practices in development process on projects with high security requirements.

1. IDS SDLC

1.1. Basic

IDS SDLC (here and after IDS SDLC or SDLC) is compiled from well-known projects: [MICROSOFT SDL] and [OWASP]. This compilation allowed adapting the development process closer to IDS-solutions service needs as well as making results of security practices more transparent for final clients.



[MICROSOFT SDL] provides exact step-by-step instructions of organizing SDL process on development projects as well as instrumentation for this process.

[OWASP] presents number of web application security projects with deep knowledge of software security. IDS SDLC is actively using the following owasp projects:

1. [SOFTWARE ASSURANCE MATURITY MODEL]
2. [OWASP CODE REVIEW GUIDE]
3. [OWASP APPLICATION SECURITY VERIFICATION STANDARD]
4. [OWASP TESTING]
5. [OWASP TOP 10]

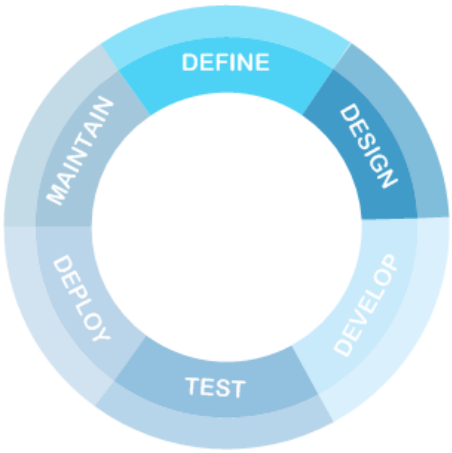
Other more technology specific [OWASP PROJECTS] can also be used by company development teams.

1.2. Lifecycle

IDS SDLC should cover security aspects on all phases of development process:

1) **DEFINE** – gathering requirements
2) **DESIGN** – designing architecture
3) **DEVELOP** – active development

4) **TEST** – quality assurance
5) **DEPLOY** – deploying system into production environment
6) **MAINTAIN** – maintenance



1.3. Principles

IDS SDLC is built on the following principles:

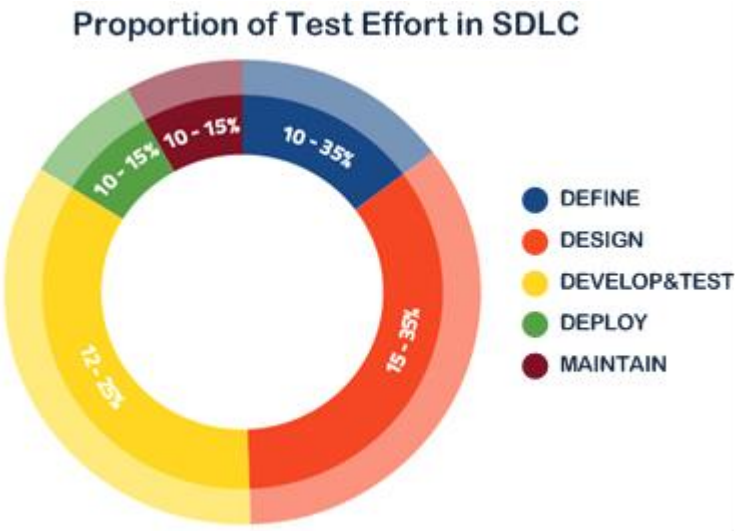
1) Planning security in advance
This principle supposes planning security starting from initial phases of development process. The diagram below shows proportions of Test Effort in SDLC. Most of efforts should be spend at Define and Design phases.

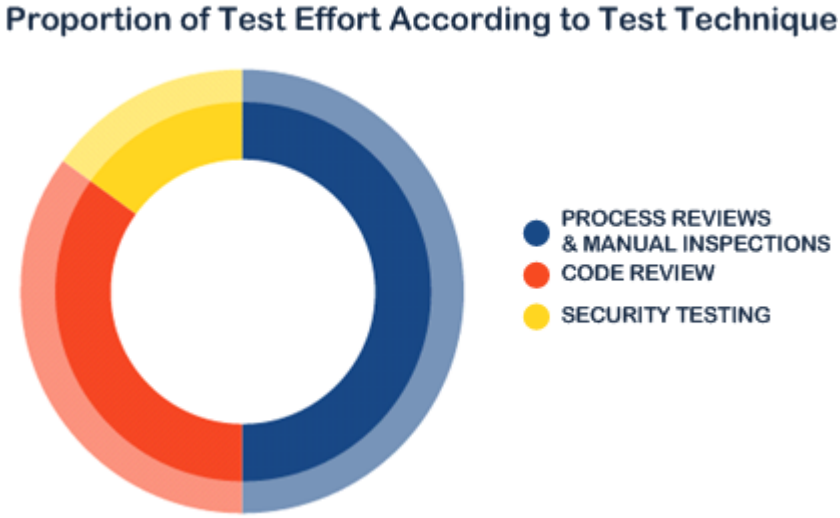
2) Secure by design
Developers consider security issues of the basic architectural design. They review detailed design for possible security issues and develop mitigations for all threats.

3) Capture security requirements
Gathering security requirements during DEFINE phase allows to avoid potential issues and weak places at further system design

4) Test Early
Testing security should be constant process in all phases of SDLC starting from testing security requirements, design and ending with testing production environment with deployed live system

5) Document security
Security documentation and periodic reports should be a bridge between technical staff and non-technical people. Clients (usually non-technical people) should be sure that their systems are protected by particular means and that results of this security process are available for them in understandable view. Transparency is one of the most important factors of IDS SDLC.





1.4. Testing Plan

Each phase produces particular artifacts and **results of each phase should be tested**:

1) DEFINE: test requirements
2) DESIGN: test architecture design and threat model
3) DEVELOP: test code
4) TEST: test application
5) DEPLOY: test production environment
6) MAINTAIN: test live

1.5. Agile

For agile projects planning should include Final Security Review task for each iteration (recommended about 5-10% of development tasks).

This task includes producing, testing, reviewing security artifacts (requirements, design, code, tests, and security documentation).

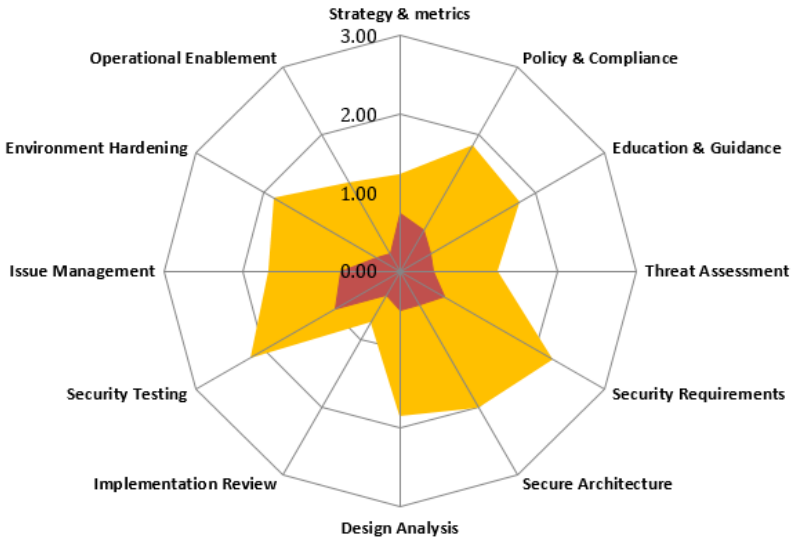
2. PRE-SDLC

2.1. Building Assurance Model

The main purpose of this stage is to formulate and implement a strategy for software security that is tailored to the specific risks facing the organization. This step should be performed on regular basis (at least once a year) on organization, department or team level. This will help to understand weak points in the current security maturity and what exactly in SDLC requires more attention.

Open SAMM ([SOFTWARE ASSURANCE MATURITY MODEL]) framework is recommended to select for defining the current maturity level and build a roadmap. Security maturity should be expressed in figures. Open SAMM has own toolkit with worksheets that contain a set of questions and formulas for calculating maturity in different business functions (Governance, Construction, Verification, Operations).

Below is an example that shows the difference between phases of building Assurance model for a particular company.



The diagram produced by Open SAMM shows that Implementation Review, Threat Assessment, Operational Enablement security functions requires significant improvements.

Note: If some questions are not applicable for the organization, they can be subchanged on other ones with the same level of difficulty.

2.2. Trainings

Developers in the company should receive appropriate training to stay informed about security basics and recent trends in security and privacy. Security training can help ensure software is created with security and privacy principles in mind and can also help development teams to stay up to date about security issues. Project team members are strongly encouraged to seek additional security and privacy education that is suitable to their needs or products.

A number of key knowledge concepts are important for successful software security. These concepts can be broadly categorized as either basic or advanced security knowledge. Each technical member of a project team (developer, tester, program manager) should receive the knowledge of security concepts.

2.2.1. Basic Concepts

The basic concepts suppose minimum level of knowledge that each developer should have working on high-secure systems.

Below is a list of recommended trainings:

1) [OWASP TOP 10]
2) Secure coding
3) Cryptography
4) Attack surface reduction
5) Defense in depth
6) Principle of least privilege
7) Secure defaults
8) Compliances

2.2.2. Advanced Concepts

Team technical leads, system architects, security experts and QA should have more advanced concepts of security testing:

1) [OWASP TESTING]
2) High-secure design and architectures
3) Security concerns in detail
4) Threat Modeling and Mitigations
5) Compliances

2.3. Policy and standards, code conventions

The Company should develop policy and standards of protecting company and client assets, internal correspondence as well as correspondence with clients. General code conventions and coding rules for particular programming languages and frameworks (.NET, Java, HTML, Javascript, CSS and others) should be worked out in advance.

On organizational level the company should develop the process of authentication and authorization to source control, safe build process, protection of sensitive information about production and live systems that developers should have due to their responsibilities.

2.4. Artefacts

This phase should produce and test the following artefacts:

1) Training Plan

2) List of tasks for improving Security Maturity Level

3) Personnel security policy

3. PHASE 1: DEFINE

3.1. Assign Security Manager

This team member should be responsible for integrating IDS SDLC practices into standard company development process, checking that all artifacts are created in each phase, and assign people to these artifacts. The main goal of this role is to make SDLC process manageable, documented and transparent. The following team members can share this role: technical team lead, system architect, security expert. It should be a person with management and technical skills.

3.2. Define compliances and standards

Need to discuss with business what compliances and security standards the System should support. Below is just examples of standards:

1) GDPR
2) HIPAA, HITECH
3) SOX
4) Cryptographic Algorithms and Key Lengths
5) Local government and business standards

3.3. Classify Data and define data retention period

Data classification helps ensure that data is protected in the most cost-effective manner. Each classification should have separate handling requirements and procedures pertaining to how that data is accessed, used, and destroyed.

There are no hard and fast rules on the classification levels that an organization should use. An organization could choose to use any of the classification levels. The following shows the common levels of sensitivity from the highest to the lowest for commercial business:

- Confidential (Trade secrets, Healthcare information)
- PII (Email, Name, Address)
- Private (Medical information, Personal preferences)
- Sensitive (Credit card numbers, business assets)
- Public (disclosure won't cause an adverse impact)

The information lifecycle is shown on the picture below.

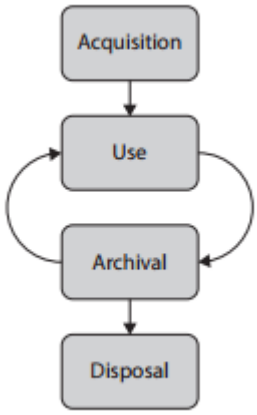


Figure 1 Information Lifecycle

It's important to discuss with business in DEFINE phase data retention period for each type of data.

Below is an example

Type of Data/Record or Document Category	Retention Period		Comments
	Active/Use	Archive	
Pupil PII	1 school year or disposing by request	3 years	Archive: remote encrypted backup
Statistics	1 school year	3 years	Archive: remote encrypted backup
Audit Logs	3 years	-	

3.4. Define Use Case Authorization Matrix and collect business security requirements

An use case authorization control matrix is a table of use cases and subjects indicating what actions individual subjects can take upon individual objects.

Example,

Use case	Admin	Manager	Report & Repair	Repair User	Report User	Read-only User	Client User	Share User
	Web app							
Have access to web app	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
View eviFiles	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes - only shared by other users
Edit eviFiles	Yes	Yes	Yes	No	No	No	No	No

Gathering security requirements is one the most important practices of SDLC.

Security requirements are divided on two parts: business security requirements that definitely should be discussed with business and technical security requirements.

Technical security requirements should cover the following areas:

1) Authentication
2) Session Management
3) Access Control
4) Malicious Input Handling
5) Cryptography at Rest
6) Error Handling and Logging
7) Data Protection
8) Communications
9) HTTP
10) Malicious Controls
11) Business Logic
12) File and Resource
13) Mobile

[OWASP APPLICATION SECURITY VERIFICATION STANDARD] will allow defining security requirements based on system risk level.

Business security requirements should be part of SRS document (System Requirements Specification).

3.5. Artefacts

This phase should produce and test the following artefacts:

1) Software Requirements Specication (SRS)
    a. Includes business security requirements
    b. Data classification, Data retention policy

2) Technical Vision Document

    a. Includes Compliance and Standards

    b. Technical Security Requirements

4. PHASE 2: DESIGN

4.1. Secure Design Principles

IDS SDLC sets up the following main design principles of secure design:

1) Minimize Attack Surface
2) Defense-in-Depth
3) Least Privilege
4) Secure Defaults
5) Keep security simple

For more information, see [SECURE DESIGN PRINCIPLES]

4.2. Establish Design Requirements

The main artifact of architecture phase is Software Development Documents (SDD):

1) Architecture Design Document (ADD) (required)
2) Detailed Design Document (DDD) (optional)

4.2.1. Architecture Design Document

Below is recommended sections for SDD ADD document:

1) Context
2) Functional View
3) Process View
4) Non-functional View
5) Constraints
6) Compliances and Standards
7) Architecture Principles
8) Logical View
9) Design View
10) Infrastructure View
11) Hardware Calculation
12) Deployment View
13) Operational View
14) **Security View**
15) Data View
16) Technology Selection
17) Architecture Justification

Security View section should have the following sections:

Security View
Authentication and Authorization
Minimize Attack Surface

Security Layered Model
Security Controls
Threat modeling/Risk Calculation
Security Requirements
Firewall Rules
Security Solution Justification

The document should also define production environment security configuration:

1) Monitoring and alerting rules in response on security incidents
2) Firewall rules
3) List of accounts that should be registered in system
4) Software installed in production environment
5) Antivirus
6) Backup plans

4.3. Use Threat Model

Threat modeling should be considered as risk assessment process. It enables the architects to develop mitigation strategies for potential vulnerabilities and helps them focus their inevitably limited resources and attention on the parts of the system that require it most.

Developing a threat model, it is recommended taking a simple approach that follows the [NIST 800-30, RISK MANAGEMENT] standard process for risk assessment. This approach involves:

1) *Decomposing the application* – use a process of manual inspection to understand how the application works, its assets, functionality, and connectivity.
2) *Defining and classifying the assets* – classify the assets into tangible and intangible assets and rank them according to business importance.
3) *Exploring potential vulnerabilities* - whether technical, operational, or management.
4) *Exploring potential threats* – develop a realistic view of potential attack vectors from attacker's perspective, by using threat scenarios or attack trees.
5) *Creating mitigation strategies* – develop mitigating controls for each of the threats deemed to be realistic.

The output from a threat model itself can vary but is typically a collection of lists and diagrams. The [OWASP CODE REVIEW GUIDE] outlines an [APPLICATION THREAT MODELING] methodology that can be used as a reference for the testing applications for potential security flaws in the design of the system.

Below is a sample:

#	Threat/Risk	Rate	Mitigations
Physical Tier			
P2	*Threat*: Disks physically copied or stolen *Risk*: extracting sensitive data	9.1 (High)	Full disk encryption
Server Tier			
S1	*Threat*: malicious users access the server using root rights *Risks*: full system compromise and sensitive data leakage	7.8 (High)	Restrict access to the servers only for support team office on firewall level Track each login on the servers (Dhound Intrusion Detection System) Use strong password policies
S2	*Threat*: operating system infected by malicious code that sends data from the servers on malicious users side *Risk*: sensitive data compromise	5.4 (Medium)	Use Dhound Intrusion detection system to track any OUTPUT activities from the servers Set up Antivirus on server level
System Tier			
M3	*Threat*: access other regions by compromising one of them *Risks*: all regions are compromised	4.9 (Medium)	Use dedicated docker container with web site for each region
Data Tier			
D4	*Threat*: malicious user could access the servers and downloaded live database snapshot *Risk*: sensitive data compromise for all regions	8.1 (High)	Use MySql Enterprise engine with enabled encryption mode for live data.
Availability tier			

4.4. Artefacts

This phase should produce and test the following artefacts:

1) Architecture Design Document

2) Security Test Plan

5. PHASE 3: DEVELOP

5.1. Source Code Review

Security Manager should organize source code review on regular basis by a group of the most experienced technical staff in the team.

According to IDS SDLC each iteration has Final Security review phase. Part of this phase should be spent on Code Review process. Code Review should cover particular developer check-ins as well as common source code review. Common source code review usually should be performed by technical team leader.

5.2. Security testing via unit testing

By using unit tests developers should validate the security functionality of components as well as verify that the countermeasures being developed mitigate any security risks previously identified through threat modeling and source code analysis.

A generic set of security unit tests includes security test cases to validate both positive and negative requirements for security controls such as:

- Identity, Authentication & Access Control
- Input Validation & Encoding
- Encryption
- User and Session Management
- Error and Exception Handling
- Auditing and Logging

5.3. Static Code Analysis

Automatic Source Code Analysis is necessary part of IDS SDLC. It allows identifying common developer mistakes connected with security in design time.

The results of automated secure code analysis can also be used as automatic check-in gates for version control, for example software artifacts cannot be checked in the build with high or medium severity coding issues.

The following owasp link [SOURCE CODE ANALYSIS TOOLS] recommends different tools for different languages.

5.4. Secured CI/CD process

Configured continuous integration and continuous delivery is necessary step for SDLC. All created environment and build machine should be protected by particular security controls. All created unit test should be run smoothly and automatically on regular basis.

5.5. Artefacts

This phase should produce and test the following artefacts:

1) Iteration security review reports
2) Code Review Policy

6. PHASE 4: TEST

6.1. Security Testing

Security Testing should be part of quality assurance process.

[OWASP TESTING] project should be taken as basis for Security Testing process. The list of test cases that OWASP Testing defines should be reviewed and extended by project specific security test-cases.

The result of security testing should be documented in a report. The report should contain information available for understanding for non-technical people as well as all containing technical details and findings.

IDS SDLC delivers a template of security testing report.

Below is a sample of a found vulnerability with estimated risk.

Found Most Critical Security Issues

Refer to RISK DETERMINATION (CALCULATION) section for understanding Risk Calculation methods.

Issue #VL-LM3-01 'TBD' Field is XSS Vulnerable

Vulnerability description: TBD Attribute on TBD Field is XSS Vulnerable; malicious code can be injected just using browser

Worst-case scenario: steal identity of another user by catching his auth token

Overall Risk Severity: High (8.8 of 10)

Step to reproduce:

1) User#1. Open doc TBD, add TBD field, enter the following line into TBD:
   vulnerable tip
2) User#1. Send the doc to User#2.
3) User#2. Mouseover on tooltip of TBD field, malicious code is executable with extracting auth token of current user.

Screenshots: STEAL AUTH TOKEN USING XSS VULNERABILITY

Recommended Safeguards:

1) Use Sanitization libraries on servers side to exclude any executable code in HTML
2) Don't allow to use any HTML for this attribute for users

6.2. Artefacts

This phase should produce and test the following artefacts:

1) Misuse test cases list
2) Security Testing Report
3) Automatic Tools Scanning Report

7. PHASE 5: DEPLOY

7.1. Test production environment

Production environment should be built according to defined rules in Architecture Design document and fully documented. The document should include information about Firewalls rules; install software, created accounts, backup plans, paths to application and log files, and others.

The document as well as active protection of the system should be re-tested by the most experienced technical persons (preferable by security experts). A final report should contain information about passed security test-cases, finding, and instrumentation. In comparison with application testing from TEST phase, this testing is more directed to testing protection of production environment (ports, servers, services, alerting systems and so on).

7.2. Artefacts

This phase should produce and test the following artefacts:

1) Deployment Guid or read.me
2) Updated ADD
3) Security Solution Document

8. PHASE 6: MAINTAIN

8.1. Incident Response Plan

Project Manager should coordinate the process of team response on any security finding or production incident including assigning priority, involving particular roles and persons into resolving security issues, security retesting.

8.2. Monthly security day

The seventh day of each month is a security day

GOAL: perform Live systems security checks to prevent them from having easy noticeable critical security issues. Should be achieved with the following steps:

STEPS:

1) Security Administrator will be assigned by Head of Dev Dept to check security on all the projects
2) Security Administrator should spent a day to:
    - to perform a check based on checklists created
    - create a new checklist if some project still not having it (should be done by Team Lead of the project)
    - do deeper check on a chosen project (if time is left)
3) Security Administrator should SIGN that all specified rules for a particular project were checked and upload a new version of document (project security review document should contain date and name of administrator after review a particular project)
4) A short list of recommendations should be shared with a team
5) PM of the project should create issues in JIRA and initiate discussion of priorities of these issues

NOTE for PMs: Security Day itself is not a billable actitivity, whereas all fixes should be billable (added to PPs either as a separate task or hours should be spread among dev tasks/maintenance)

8.3. Artefacts

This phase should produce and test the following artefacts:

1) Regular Security Review Report
2) Incident Response Policy
3) Access Control Policy

9. REFERENCES

[Microsoft SDL] http://www.microsoft.com/sdl

[OWASP] http://owasp.org/

[OWASP Testing] https://www.owasp.org/index.php/Category:OWASP_Testing_Project

[OWASP Top 10] https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

[OWASP Application Security Verification Standard]
https://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project

[OWASP Code Review Guide] https://www.owasp.org/index.php/Category:OWASP_Code_Review_Project

[Software Assurance Maturity Model]
https://www.owasp.org/index.php/Category:Software_Assurance_Maturity_Model

[OWASP Projects] https://www.owasp.org/index.php/Category:OWASP_Project

[BSIMM-V] http://www.bsimm.com/

[Secure Design Principles] https://www.owasp.org/index.php/Secure_Coding_Principles

[Application Threat Modeling] https://www.owasp.org/index.php/Application_Threat_Modeling

[Microsoft Threat Modeling Tool] http://www.microsoft.com/en-us/download/details.aspx?id=42518

[NIST 800-30, Risk Management] http://csrc.nist.gov/publications/nistpubs/800-30-rev1/sp800_30_r1.pdf

[Source Code Analysis Tools] https://www.owasp.org/index.php/Source_Code_Analysis_Tools