

Unit #10

Application Security

MIS5214

Agenda

- Frameworks for application security assessment
- Best practices for secure application development
- Test areas for auditing applications
- Team Project – guidance and status reporting

PPTM - People, Processes, Tools, and Measures

A brainstorming framework for examining security of an application from the macro-level, based on

People – describes every aspect of the application that deals with a human

- Make sure the right people are involved in planning, design, implementation or operations, and the right stakeholders are involved
- E.g. If the application involves end users, ensure:
 - The application has controls around providing and removing access
 - End users have been involved with the planning and design of components they will (to ensure usability)

Process – Describes every aspect of the application that is involved in a policy, procedure, method, or course of action

- Review the interaction of the application with interfacing systems and verify compliance with security models
 - E.g. Ensure that firewalls are in place to protect the application from external applications, users, business partners, ...
 - Policies and procedures should be written to support how the application is intended to be used
 - Adequate documentation should exist to support technicians who need to maintain the application

Tools – Describe every aspect of the application that deals with concrete technology or product

- Ensure appropriate hardware and environment exist to support the application
- Ensure the application interfaces with recommended technologies appropriate for your intended policies and procedures
- Verify that the application and infrastructure are tested and audited appropriately

Measures – Describe every aspect of the application that is quantifiable conceptually, such as the business purpose or application performance

- E.g. verify that the application meets well-documented and well-thought out acceptance criteria
- E.g. if the application is intended to solve a quantifiable business problem verify that it does indeed solve the problem
- Verify that the logs are meaningful and that you can measure the performance of the application

STRIDE

A “simplified threat-risk model” which is easy to remember

Spooing Identity

- Is a key risk for applications with many users and a single execution context at the application and database tiers
- Users should not be able to become any other user or assume the attributes of another user

Tampering with Data

- Data should be stored in a secure location, with access appropriately controlled
- The application should carefully check data received from the user and validate that it is “sane” (i.e. relevant and valid) and applicable before storing or using it
- Data entered in the client (e.g. browser) should be checked and validated on the server and not in the client where the validation checks might be tampered with
- Application should not send and calculate data in the client where the user can manipulate the data, but in the server-side code

Repudiation

- Determine if the application requires nonrepudiation controls, such as web access logs, audit trails at each tier, or the same user context from top to bottom
- Users may dispute transactions if there is insufficient auditing or record-keeping of their activity

Denial of Service

- Application designers should be aware that their applications are at risk of denial of service attacks
- Use of expensive resources (e.g. large files, heavy-duty searches, long queries) should be reserved for authenticated and authorized users and should not be available to anonymous users.
- Every facet of the application should be engineered to perform as little work as possible, to use fast and few database queries, and to avoid exposing large files or unique links per user to per user to prevent simple denial-of-service attacks

Elevation of Privilege

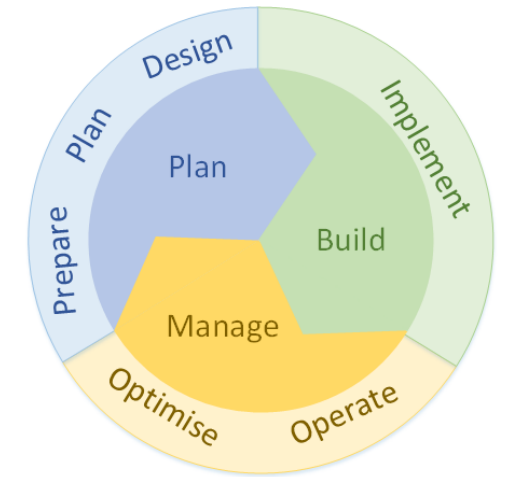
- If an application provides distinct user and administrative roles, ensure that the user cannot elevate his or her role to a more highly privileged one
- All actions should be controlled through an authorization matrix to ensure that only the permitted roles can access privileged functionality. It is not sufficient, for example, to not display privileged-role links

Threat	Desired property
Spooing	Authenticity
Tampering	Integrity
Repudiation	Non-repudiability
Information disclosure	Confidentiality
Denial of Service	Availability
Elevation of Privilege	Authorization

PPDIOO – Prepare, Plan, Design, Implement, Operate, Optimize

from CISCO Systems

- Can help considers potential network challenges for a new application system
- Creates a baseline of how a network should perform and behave
- Uses the baseline to measure project success and monitor network health



Prepare: Establishes organization and business requirements, develops a network strategy, and proposes a high-level architecture

Plan: Identifies the network requirements by characterizing and assessing the network and performing a gap analysis

Design: Provides high-availability, reliability, security, scalability, and performance

Implement: Installation and configuration of new equipment

Operate: Day-to-day operations

Optimize: Proactive network management; modifications to the design

OWASP (Open Web Application Security Project) Frameworks

- Vulnerabilities

- ▶ API Abuse
- ▶ Authentication Vulnerability
- ▶ Authorization Vulnerability
- ▶ Availability Vulnerability
- ▶ Code Permission Vulnerability
- ▶ Code Quality Vulnerability
- ▶ Configuration Vulnerability
- ▶ Cryptographic Vulnerability
- ▶ Encoding Vulnerability
- ▶ Environmental Vulnerability
- ▶ Error Handling Vulnerability
- ▶ General Logic Error Vulnerability
- ▶ Input Validation Vulnerability
- ▶ Logging and Auditing Vulnerability
- ▶ Password Management Vulnerability
- ▶ Path Vulnerability
- ▶ Sensitive Data Protection Vulnerability
- ▶ Session Management Vulnerability
- ▶ Unsafe Mobile Code
- ▶ Use of Dangerous API

- Principles

- Apply **defense in depth** (complete mediation)
- Use a **positive security model** (fail-safe defaults, minimize attack surface)
- Fail securely
- Run with least privilege
- **Avoid security by obscurity** (open design)
- Keep security simple (verifiable, economy of mechanism)
- Detect intrusions (compromise recording)
- Don't trust infrastructure
- Don't trust services
- Establish secure defaults (psychological acceptability)

- Top 10 Web Application Security Risks

A1:2017 - Injection	7
A2:2017 - Broken Authentication	8
A3:2017 - Sensitive Data Exposure	9
A4:2017 - XML External Entities (XXE)	10
A5:2017 - Broken Access Control	11
A6:2017 - Security Misconfiguration	12
A7:2017 - Cross-Site Scripting (XSS)	13
A8:2017 - Insecure Deserialization	14
A9:2017 - Using Components with Known Vulnerabilities	15
A10:2017 - Insufficient Logging & Monitoring	16

Agenda

- Frameworks for application security assessment
- Best practices for secure application development
- Test areas for auditing applications
- Team Project – guidance and Q&A

Static & Dynamic Application Security Testing

Static application security testing (SAST)

- Can be thought of as testing the application from the inside out
- By examining its source code, byte code or application binaries for conditions indicative of a security vulnerability

Dynamic application security testing (DAST)

- Can be thought of as testing the application from the outside in
- By examining the application in its running state, and trying to poke it and prod it in unexpected ways in order to discover security vulnerabilities

Some vulnerabilities can be found only with SAST testing, others with DAST

Testing in both ways yields the most comprehensive testing

Many web applications that would be traditionally scanned with DAST tools also use a significant amount of client-side code often in the form of Javascript, Python, CSS, PHP, Ruby,...

This code must also be analyzed for security vulnerabilities, typically using static analysis

Automated application security testing tools

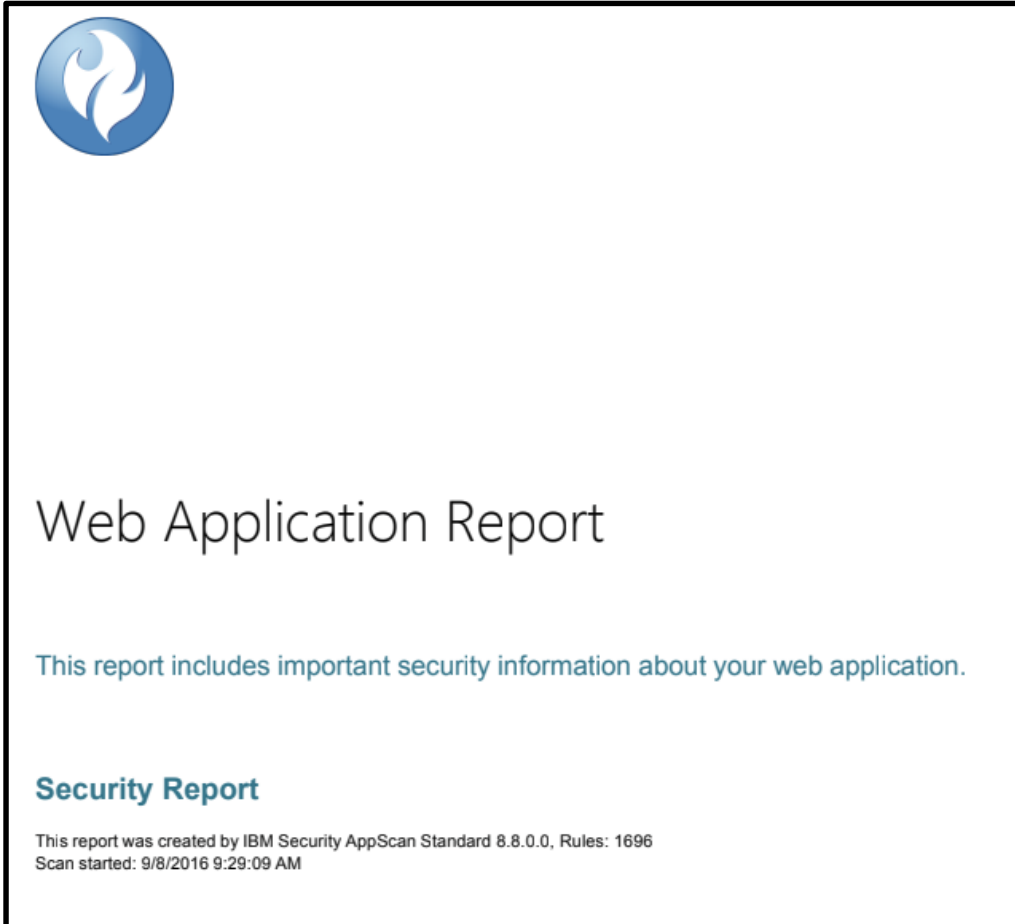
Magic Quadrant

Figure 1. Magic Quadrant for Application Security Testing



Many vendors provide both SAST and DAST tools

Automated application security testing tools often provide vulnerability reports



This report contains the results of a web application security scan performed by IBM Security AppScan Standard.

High severity issues:	79
Medium severity issues:	198
Total security issues included in the report:	277
Total security issues discovered in the scan:	308

Application Security Assessment and Recommendations

Issue Types 21

Issue Type	Number of Issues
H Authentication Bypass Using HTTP Verb Tampering	3
H Cross-Site Request Forgery	23
H Cross-Site Scripting	2
H Microsoft FrontPage Extensions Site Defacement	3
H Missing Secure Attribute in Encrypted Session (SSL) Cookie	5
H RC4 cipher suites were detected	1
M Alternate Version of File Detected	45
M Body Parameters Accepted in Query	9
M Browser Exploit Against SSL/TLS (a.k.a. BEAST)	1
M Cacheable SSL Page Found	67
M Direct Access to Administration Pages	1
M Drupal "keys" Path Disclosure	1
M Insecure "OPTIONS" HTTP Method Enabled	1
M Microsoft FrontPage Server Extensions Vital Information Leakage	2
M Microsoft IIS Missing Host Header Information Leakage	1
M Missing "Content-Security-Policy" header	5
M Missing Cross-Frame Scripting Defence	4
M Query Parameter in SSL Request	185
M Temporary File Download	3
M Unencrypted __VIEWSTATE Parameter	20
M Web Application Source Code Disclosure Pattern Found	1

TOC Fix Recommendations 19

TOC

Remediation Task	Number of Issues
H Review possible solutions for hazardous character injection	2
M Add the 'Secure' attribute to all sensitive cookies	5
M Change server's supported ciphersuites	2
M Configure your server to allow only required HTTP methods	3
M Set proper permissions to the FrontPage extension files	3
M Validate the value of the "Referer" header, and use a one-time-nonce for each submitted form	23
L Always use SSL and POST (body) parameters when sending sensitive information.	185
L Apply configuration changes according to Q218180	1
L Apply proper authorization to administration scripts	1
L Config your server to use the "Content-Security-Policy" header	5
L Config your server to use the "X-Frame-Options" header	4
L Contact the vendor of your product to see if a patch or a fix has been made available recently	1
L Disable WebDAV, or disallow unneeded HTTP methods	1
L Do not accept body parameters that are sent in the query string	9
L Modify FrontPage extension file permissions to avoid information leakage	2
L Modify your Web.Config file to encrypt the VIEWSTATE parameter	20
L Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.	67
L Remove old versions of files from the virtual directory	48
L Remove source code files from your web-server and apply any relevant patches	1

This report contains the results of a web application security scan performed by IBM Security AppScan Standard.

High severity issues:	79
Medium severity issues:	198
Total security issues included in the report:	277
Total security issues discovered in the scan:	308


Application Security Vulnerability Assessment Report

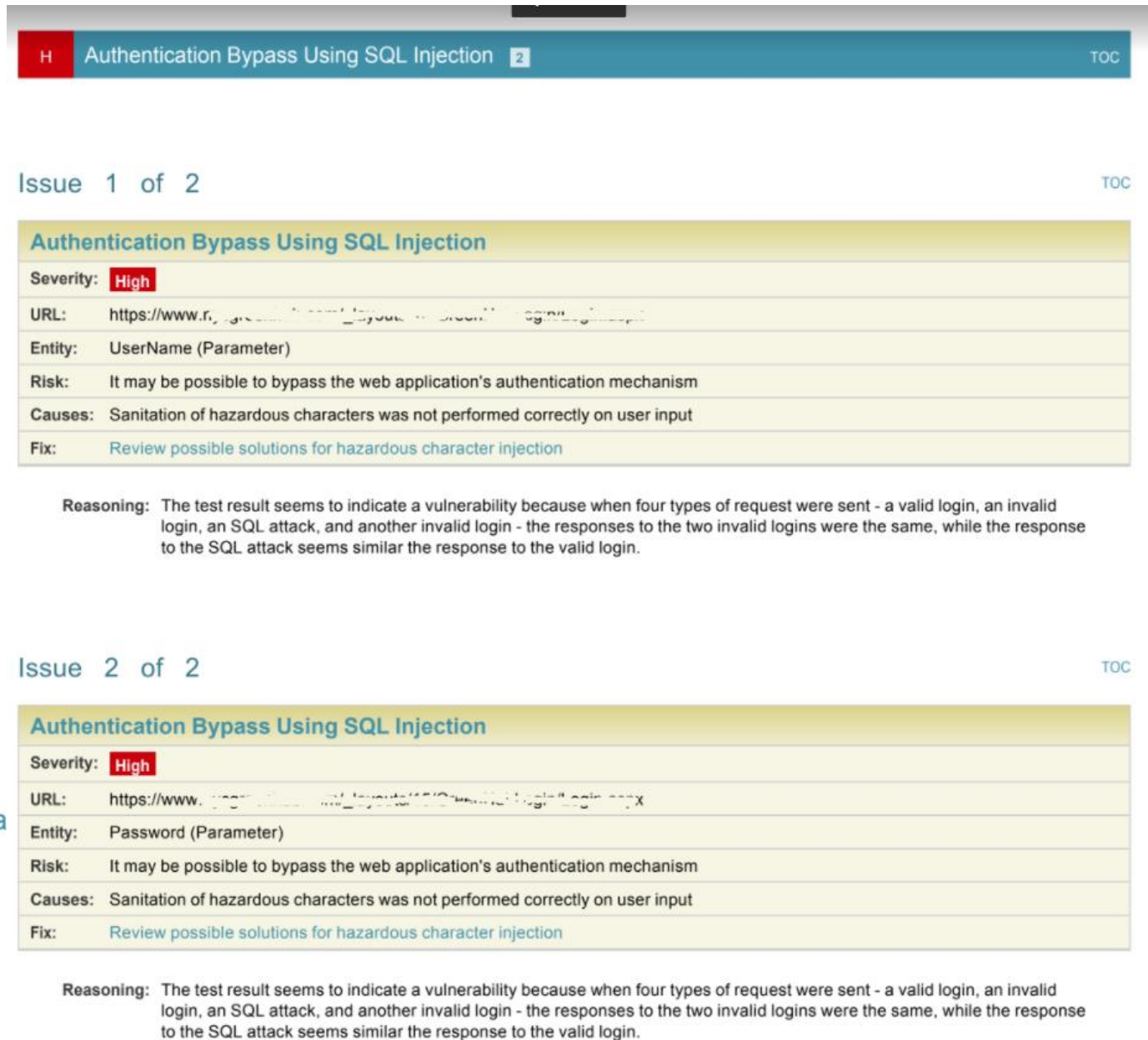
Issues Sorted by Issue Type

- Authentication Bypass Using SQL Injection **2**
- Blind SQL Injection **4**
- Cross-Site Request Forgery **24**
- Cross-Site Scripting **3**
- HTTP PUT Method Site Defacement **20**
- Inadequate Account Lockout **1**
- Microsoft FrontPage Extensions Site Defacement **3**
- Missing Secure Attribute in Encrypted Session (SSL) Cookie **1**
- Phishing Through URL Redirection **1**
- WebDAV MKCOL Method Site Defacement **20**
- Alternate Version of File Detected **50**
- Cacheable SSL Page Found **26**
- Hidden Directory Detected **7**
- Microsoft FrontPage Configuration Information Leakage **1**
- Microsoft FrontPage Server Extensions Vital Information Leakage **2**
- Microsoft IIS Missing Host Header Information Leakage **1**
- Query Parameter in SSL Request **66**
- Temporary File Download **32**
- Unencrypted __VIEWSTATE Parameter **11**
- Web Application Source Code Disclosure Pattern Found **2**

IBM AppScan example

Advisories

- Authentication Bypass Using SQL Injection 
- Blind SQL Injection
- Cross-Site Request Forgery
- Cross-Site Scripting
- HTTP PUT Method Site Defacement
- Inadequate Account Lockout
- Microsoft FrontPage Extensions Site Defacement
- Missing Secure Attribute in Encrypted Session (SSL) Cookie
- Phishing Through URL Redirection
- WebDAV MKCOL Method Site Defacement
- Alternate Version of File Detected
- Cacheable SSL Page Found
- Hidden Directory Detected
- Microsoft FrontPage Configuration Information Leakage
- Microsoft FrontPage Server Extensions Vital Information Leaka
- Microsoft IIS Missing Host Header Information Leakage
- Query Parameter in SSL Request
- Temporary File Download
- Unencrypted __VIEWSTATE Parameter
- Web Application Source Code Disclosure Pattern Found



The screenshot displays two identical vulnerability reports from IBM AppScan. Each report is titled 'Authentication Bypass Using SQL Injection' and is categorized as 'High' severity. The reports include details such as the URL, the affected entity (Username and Password parameters), the risk of bypassing authentication, the cause (improper sanitization of user input), and a fix recommendation to review solutions for hazardous character injection. A reasoning section explains that the vulnerability was identified because the responses to a valid login, an invalid login, and an SQL attack were indistinguishable.

Issue 1 of 2 TOC

Authentication Bypass Using SQL Injection

Severity: **High**

URL: <https://www.r...>

Entity: Username (Parameter)

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

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

Fix: [Review possible solutions for hazardous character injection](#)

Reasoning: The test result seems to indicate a vulnerability because when four types of request were sent - a valid login, an invalid login, an SQL attack, and another invalid login - the responses to the two invalid logins were the same, while the response to the SQL attack seems similar the response to the valid login.

Issue 2 of 2 TOC

Authentication Bypass Using SQL Injection

Severity: **High**

URL: <https://www...>

Entity: Password (Parameter)

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

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

Fix: [Review possible solutions for hazardous character injection](#)

Reasoning: The test result seems to indicate a vulnerability because when four types of request were sent - a valid login, an invalid login, an SQL attack, and another invalid login - the responses to the two invalid logins were the same, while the response to the SQL attack seems similar the response to the valid login.

Test Type:

Application-level test

Threat Classification:

Insufficient Authentication

Causes:

Sanitation of hazardous characters was not performed correctly on user input

Security Risks:

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

Affected Products:

CWE:

566

References:

"Web Application Disassembly with ODBC Error Messages" (By David Litchfield)
SQL Injection Training Module

Technical Description:

The application uses a protection mechanism that relies on the existence or values of an input, but the input can be modified by an untrusted user in a way that bypasses the protection mechanism.

When security decisions such as authentication and authorization are made based on the values of user input, attackers can bypass the security of the software.

Suppose the query in question is:

```
SELECT COUNT(*) FROM accounts WHERE username='$user' AND password='$pass'
```

Where \$user and \$pass are user input (collected from the HTTP request which invoked the script that constructs the query - either from a GET request query parameters, or from a POST request body parameters). A regular usage of this query would be with values \$user=john, \$password=secret123. The query formed would be:

```
SELECT COUNT(*) FROM accounts WHERE username='john' AND password='secret123'
```

The expected query result is 0 if no such user+password pair exists in the database, and >0 if such pair exists (i.e. there is a user named 'john' in the database, whose password is secret123). This would serve as a basic authentication mechanism for the application. But an attacker can bypass this mechanism by submitting the following values: \$user=john, \$password=' OR '1'=1.

Technical Description:

The application uses a protection mechanism that relies on the existence or values of an input, but the input can be modified by an untrusted user in a way that bypasses the protection mechanism.

When security decisions such as authentication and authorization are made based on the values of user input, attackers can bypass the security of the software.

Suppose the query in question is:

```
SELECT COUNT(*) FROM accounts WHERE username='$user' AND password='$pass'
```

Where \$user and \$pass are user input (collected from the HTTP request which invoked the script that constructs the query - either from a GET request query parameters, or from a POST request body parameters). A regular usage of this query would be with values \$user=john, \$password=secret123. The query formed would be:

```
SELECT COUNT(*) FROM accounts WHERE username='john' AND password='secret123'
```

The expected query result is 0 if no such user+password pair exists in the database, and >0 if such pair exists (i.e. there is a user named 'john' in the database, whose password is 'secret123'). This would serve as a basic authentication mechanism for the application. But an attacker can bypass this mechanism by submitting the following values: \$user=john, \$password=' OR '1'='1'.

The resulting query is:

```
SELECT COUNT(*) FROM accounts WHERE username='john' AND password='' OR '1'='1'
```

This means that the query (in the SQL database) will return TRUE for the user 'john', since the expression 1=1 is always true. Therefore, the query will return a positive number, and thus the user (attacker) will be considered valid without having to know the password.

Additional best practices for secure application development

1. Defense-in-Depth
2. Positive Security Model
3. Fail Safely
4. Run with Least Privilege
5. Avoid Security by Obscurity
6. Keep Security Simple
7. Use Open Standards
8. Keep, manage and analyze logs to detect Intrusions
9. Never Trust External Infrastructure and Services
10. Establish Secure Defaults

Characteristics which can help in quickly spotting common weaknesses and poor controls

Defense In Depth

Layered approaches provide more security over the long term than one complicated mass of security architecture

- **Sequences of routers, firewalls and intrusion detection/protection monitoring devices used to examine data packets, reduce undesired traffic and protect the inner information systems**
- **Access Control Lists (ACLs)**, for example, on the networking routers and firewall equipment to allow only necessary traffic to reach the application
 - *Quickly eliminating access to services, ports, and protocols significantly lowers the overall risk of compromise to the system on which the application is running*

Positive Security Model

- Positive security models use “whitelist” to allow only what is on the list, excluding everything else by default
 - “Deny by default”
 - A challenge for antivirus programs
- In contrast with negative (blacklist) security models that allow everything by default, eliminating only the items known to be bad
 - Problems:
 - Blacklist must be kept up to date
 - Even if blacklist is updated, an unknown vulnerability can still exist
 - Attack surface is much larger than with a positive security model

Fail Safely

- An application failure can be dealt with in one of 3 ways:
 - Allow
 - Block
 - Error
- In general, application errors should all fail in the same way:
 - Disallow the operation (as viewed by the user) and provide no or minimal information on the failure
 - Do not provide the end user with additional information that may help in compromising the system
 - Put the error information in the logs, but do not provide to the user to use in compromising the system

Run with Least Privilege

- Principle of Least Privilege mandates that accounts have the least amount of privilege possible to perform their activity
- This includes:
 - User rights
 - Resource permissions such as CPU limits, memory capacity, network bandwidth, file system permissions, and database permissions

Avoid Security by Obscurity

- Obfuscating data (hiding it) instead of encrypting it is a very weak security mechanism
 - If a human can figure out how to hide the data a human can learn how to recover the data
- Never obfuscate critical data that can be encrypted or never stored in the first place

Keep Security Simple

- Simple security mechanisms are easy to verify and easy to implement correctly
- Avoid complex security mechanisms if possible
 - *“The quickest method to break a cryptographic algorithm is to go around it”*
- Do not confuse complexity with layers: Layers are good; complexity isn't

Use Open Standards

- Open security standards provide increased portability and interoperability
- IT infrastructure is often a heterogeneous mix of platforms, open standards helps ensure compatibility between systems as the application grows
- Open standards are often well known and scrutinized by peers in the security industry to ensure they remain secure

Keep, manage and analyze logs to help detect intrusions

- Applications should have built-in logging that is protected and easily read
- Logs help you troubleshoot issues, and just as important – help you to track down when or how an application might have been compromised

Never Trust External Infrastructure and Services

- Many organizations use the processing capabilities of third-party partners that more than likely have differing security policies and postures than your organization
- It is unlikely that you can influence or control an external third party
- Implicitly trusting externally run systems is dangerous!

Establish Secure Defaults

- New applications should arrive or be presented to users with the most secure default settings possible that still allow business to function
- This may require training end users or communications messages
- End result is a significantly reduced attack surface
 - *Especially when application is pushed out across a large population*

Agenda

- ✓ Frameworks for application security assessment
- ✓ Best practices for secure application development
- Test areas for auditing applications
- Team Project – guidance and Q&A

Test Areas for Auditing Applications

1. Input Controls

- Review and evaluate controls built into system transactions for input data
- Determine the need for error/exception reports related to data integrity and evaluate whether this need has been filled

2. Interface Controls

- Review and evaluate the controls in place over data feeds to and from interfacing systems
- If the same data is kept in multiple databases and/or systems, ensure that periodic sync processes are executed to detect any inconsistencies in the data

3. Audit Trails

- Review and evaluate the audit trails present in the system and the controls over those audit trails
- Ensure that the system provides a means of tracing a transaction or piece of data from the beginning to the end of the process enabled by the system

Test Areas for Auditing Applications

4. Software Change Controls

- Ensure that the application software cannot be changed without going through a standard checkout/staging/testing/approval process after it is placed into production
- Evaluate controls regarding code checkout and versioning
- Evaluate controls regarding the testing of application code before it is placed into a production environment
- Evaluate controls regarding batch scheduling

5. Backup and Recovery

- Determine whether a Business Impact Analysis (BIA) has been performed on the application to establish backup and recovery needs
- Ensure that appropriate backup and recovery controls are in place
- Ensure appropriate recovery controls are in place

Test Areas for Auditing Applications

6. Data Retention and User Involvement

- Evaluate controls regarding the application's data retention
- Evaluate overall user involvement and support for the Application

7. Identity, Authentication, and Access Controls...

8. Host Hardening...

Agenda

- ✓ Frameworks for application security assessment
- ✓ Best practices for secure application development
- ✓ Test areas for auditing applications
- **Team Project – guidance and Q&A**

TABLE OF CONTENTS

1.	INFORMATION SYSTEM NAME/TITLE	1
2.	INFORMATION SYSTEM CATEGORIZATION	1
2.1.	Information Types	1
2.2.	Security Objectives Categorization (FIPS 199)	3
2.3.	Digital Identity Determination	4
3.	INFORMATION SYSTEM OWNER	4
4.	AUTHORIZING OFFICIAL	4
5.	OTHER DESIGNATED CONTACTS	5
6.	ASSIGNMENT OF SECURITY RESPONSIBILITY	6
7.	INFORMATION SYSTEM OPERATIONAL STATUS	7
8.	INFORMATION SYSTEM TYPE	7
8.1.	Cloud Service Models	7
8.2.	Cloud Deployment Models	8
8.3.	Leveraged Authorizations	9
9.	GENERAL SYSTEM DESCRIPTION	9
9.1.	System Function or Purpose	9
9.2.	Information System Components and Boundaries	9
9.3.	Types of Users	10
9.4.	Network Architecture	11
10.	SYSTEM ENVIRONMENT AND INVENTORY	12
10.1.	Data Flow	13
10.2.	Ports, Protocols and Services	14
11.	SYSTEM INTERCONNECTIONS	15
12.	LAWS, REGULATIONS, STANDARDS AND GUIDANCE	17
12.1.	Applicable Laws and Regulations	17
12.2.	Applicable Standards and Guidance	17
13.	MINIMUM SECURITY CONTROLS	18

Team Project SSP Deliverable (See [Guidance Notes that follow](#))

15.	ATTACHMENTS	151
Attachment 1	Information Security Policies and Procedures	153
Attachment 2	User Guide	154
Attachment 3	Digital Identity Worksheet	155
	Introduction and Purpose	155
	Information System Name/Title	155
	Digital Identity Level Definitions	155
	Review Maximum Potential Impact Levels	156
	Digital Identity Level Selection	157
Attachment 4	PTA / PIA	158
	Privacy Overview and Point of Contact (POC)	158
	Applicable Laws and Regulations	158
	Applicable Standards and Guidance	159
	Personally Identifiable Information (PII)	159
	Privacy Threshold Analysis	160
	Qualifying Questions	160
	Designation	160
Attachment 5	Rules of Behavior	161
Attachment 6	Information System Contingency Plan	162
Attachment 7	Configuration Management Plan	163
Attachment 8	Incident Response Plan	164
Attachment 9	CIS Workbook	165
Attachment 10	FIPS 199	166

Guidance notes

Instructions for diagrams for sections:

- 9.2 (Information System Components and Boundaries),
- 9.4 (Network Architecture)
- 10.1 (Data Flow)

These can all be based on the high-level logical network diagram you create for section 9.4 (Network Architecture)

Be sure to include the locations of the users in your diagram

In addition to including them in your SSP, you should include and label this diagrams in a separate PDF file document that you deliver along with your SSP

Guidance notes

Instructions for Section 11's Table 11-1, only identify:

- External System (column 2)
- Connection Security (column 4)
- Data Direction (column 5)
- Information Being Transmitted (column 6)

11 SYSTEM INTERCONNECTIONS

Instruction: List all interconnected systems. Provide the IP address and interface identifier (eth0, eth1, eth2) for the CSP system that provides the connection. Name the external organization and the IP address of the external system. Indicate how the connection is being secured. For Connection Security indicate how the connection is being secured. For Data Direction, indicate which direction the packets are flowing. For Information Being Transmitted, describe what type of data is being transmitted. If a dedicated telecom line is used, indicate the circuit number. Add additional rows as needed. This table must be consistent with Table 13-3 CA-3 Authorized Connections.

Delete this and all other instructions from your final version of this document.

The Table 11-1 System Interconnections below is consistent with Table 13-3 CA-3 Authorized Connections.

Table 11-1 System Interconnections



SP* IP Address and Interface	External Organization Name and IP Address of System	External Point of Contact and Phone Number	Connection Security (IPSec VPN, SSL, Certificates, Secure File Transfer, etc.)**	Data Direction (incoming, outgoing, or both)	Information Being Transmitted	Port or Circuit Numbers
<SP IP Address/Interface>	<External Org/IP>	<External Org POC> <Phone 555-555-5555>	<Enter Connection Security>	Choose an item.	<Information Transmitted>	<Port/Circuit Numbers>
<SP IP Address/Interface>	<External Org/IP>	<External Org POC> <Phone 555-555-5555>	<Enter Connection Security>	Choose an item.	<Information Transmitted>	<Port/Circuit Numbers>
<SP IP Address/Interface>	<External Org/IP>	<External Org POC> <Phone 555-555-5555>	<Enter Connection Security>	Choose an item.	<Information Transmitted>	<Port/Circuit Numbers>
<SP IP Address/Interface>	<External Org/IP>	<External Org POC> <Phone 555-555-5555>	<Enter Connection Security>	Choose an item.	<Information Transmitted>	<Port/Circuit Numbers>
<SP IP Address/Interface>	<External Org/IP>	<External Org POC> <Phone 555-555-5555>	<Enter Connection Security>	Choose an item.	<Information Transmitted>	<Port/Circuit Numbers>
<SP IP Address/Interface>	<External Org/IP>	<External Org POC> <Phone 555-555-5555>	<Enter Connection Security>	Choose an item.	<Information Transmitted>	<Port/Circuit Numbers>

Guidance notes

Instructions for Section 13: Only select and complete one **technical** control family

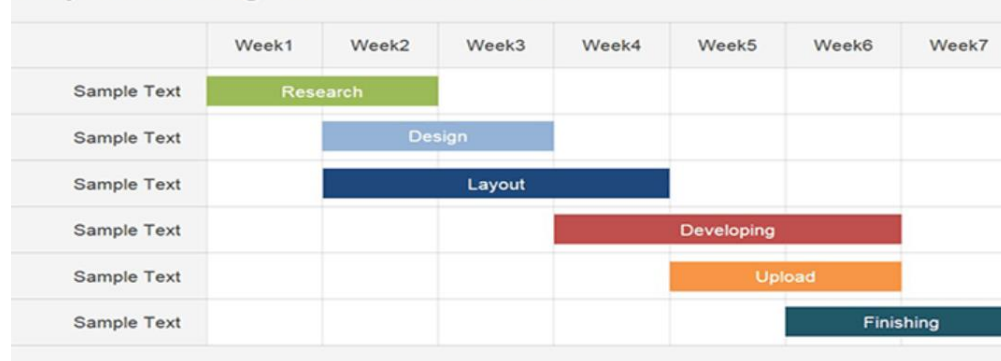
From NIST SP 800-18r1 Guide for Developing Security Plans for Federal Information Systems

CLASS	FAMILY	IDENTIFIER
Management	Risk Assessment	RA
Management	Planning	PL
Management	System and Services Acquisition	SA
Management	Certification, Accreditation, and Security Assessments	CA
Operational	Personnel Security	PS
Operational	Physical and Environmental Protection	PE
Operational	Contingency Planning	CP
Operational	Configuration Management	CM
Operational	Maintenance	MA
Operational	System and Information Integrity	SI
Operational	Media Protection	MP
Operational	Incident Response	IR
Operational	Awareness and Training	AT
Technical	Identification and Authentication	IA
Technical	Access Control	AC
Technical	Audit and Accountability	AU
Technical	System and Communications Protection	SC

Guidance notes

- Attachment 6 - Information System Contingency Plan: Only provide a plan (include a schedule tasks with labor estimate in person-hours) for completing Attachment 6 which is a Information System Contingency Plan (ISCP) based on [FedRAMP ISCP Template](#)

Project Management Gantt Chart



Team status reporting and security diagram reviews...

Agenda

- **Frameworks for application security assessment**
- **Best practices for secure application development**
- **Test areas for auditing applications**
- **Team Project – guidance and Q&A**